INTERNATIONAL STANDARD

IEC 61926-1

First edition
1999-10

Design automation –

**Part 1:
Standard test language for all systems –
Common abbreviated test language
for all systems (C/ATLAS)**

Automatisation de la conception –

*Partie 1:
Langage de test normalisé pour tout système –
Langage de test commun/abrégé pour tout système
(C/ATLAS)*

Reference number
IEC 61926-1:1999(E)

Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

Consolidated publications

Consolidated versions of some IEC publications including amendments are available. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available in the IEC catalogue.

Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
  Published yearly with regular updates
  (On-line catalogue)*
- **IEC Bulletin**
  Available both at the IEC web site* and as a printed periodical

Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV).

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams.*

* See web site address on title page.

INTERNATIONAL STANDARD

IEC 61926-1

First edition
1999-10

Design automation –

Part 1:
Standard test language for all systems –
Common abbreviated test language
for all systems (C/ATLAS)

Automatisation de la conception –

Partie 1:
Langage de test normalisé pour tout système –
Langage de test commun/abrégé pour tout système
(C/ATLAS)

International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland
Telefax: +41 22 919 0300 e-mail: inmail@iec.ch IEC web site http://www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XS

For price, see current catalogue

Contents

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

DESIGN AUTOMATION –

Part 1: Standard test language for all systems – Common abbreviated test language for all systems (C/ATLAS)

FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.
3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.
6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61926-1 has been prepared by IEC technical committee 93: Design automation.

This standard is based on IEEE Std 716-1995.

IEC 61926 consists of the following parts, under the general title *Design automation:*

– Part 1:1999, Standard test language for all systems – Common abbreviated test language for all systems (C/ATLAS)
– Part 1-1:1999, Harmonization of ATLAS test languages

This standard does not follow the rules for the structure of international standards given in Part 3 of the ISO/IEC Directives.

The text of this standard is based on the following documents:

FDIS	Report on voting
93/106/FDIS	93/111/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

1.0 Scope and object

This standard defines a high order language for testing. This language is designed to describe tests in terms that are independent of any specific test system. It has been constrained to ensure that it can be implemented on Automatic Test Equipment (ATE).

Language processors conforming to this standard shall support all capabilities as specified within C/ATLAS down to the level of nouns, modifiers, dimensional units and pin descriptors. In addition, this support shall include all nouns, modifiers, dimensional units and pin descriptors that are necessary to support the target ATE.

2.0 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

2.1 References

ANSI X3.4 1986 (R1992), Coded Character Set 7-Bit American National Standard Code for Information Interchange (ASCII).

ARINC 570, Automatic Direction Finder.

ARINC 572 and 711, Transponder and VOR Receiver.

ARINC 573, Aircraft Integrated Data Systems Mark 2 (AIDS) for the NRZ Code.

ARINC 575, Sub Sonic Air Data Systems, Bi-Polar RZ Pulse Class.

ARINC 578, Instrument Landing System.

ARINC 579, VHF Omnidirectional, Radio Range.

IEEE Std 100-1992, The New IEEE Dictionary of Electrical and Electronics Terms (ANSI).

IEEE Std 771-1989, IEEE Guide to the Use of the ATLAS Specification (ANSI).

ISO 1000:1992, SI units and recommendations for the use of their multiples and of certain other units.

ISO/IEC 646:1991, Information technology – ISO 7-bit coded character set for information interchange.

ITU-T Recommendation G.702:1988, Digital hierarchy bit rates. Entry No. 8003, AMI Pulse Class, and Entry No. 8005, HDB Pulse Class.

MIL-A-28826, Antenna System, Broadband Identification Friend or Foe, 3 May 1976.

2.2 Document precedence

In case of any conflict in the definition of C/ATLAS within this standard, the definition follows the precedence in the order listed:

a) Formal syntax
b) Rules and descriptions
c) Syntax diagrams
d) Notes and examples

2.3 Document organization and conventions

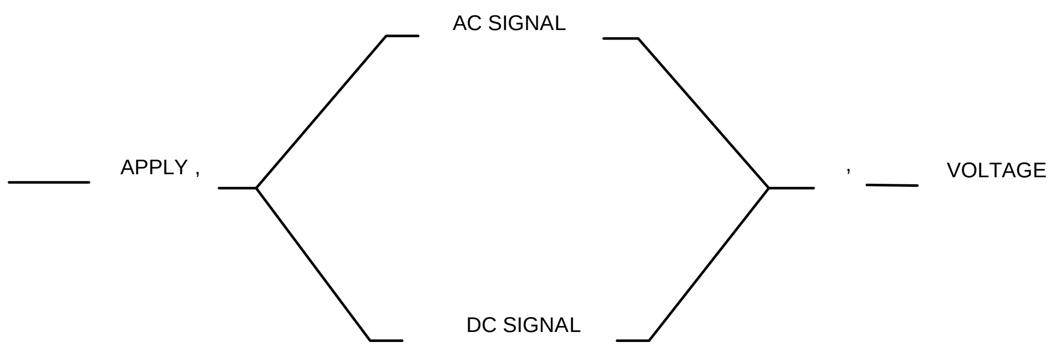
2.3.1 Extensibility

The use of the extend capability for adding nouns, noun modifiers, and pin descriptors shall be in accordance with the dictates of the management agency responsible for procurement of C/ATLAS capability, which incorporates the extensibility mechanism. This is applicable to test requirements which are peculiar to a restricted test discipline.

2.3.2 Organization of syntax specification

The syntax of C/ATLAS is described in syntax diagrams with associated rules and in formal syntax specifications. The syntax diagrams are organized for human readability as described below. The formal syntax is organized for machine analysis use in C/ATLAS compilers and syntax checkers. The formal syntax is fully described in clause 18. Names in subsections entitled "Formal syntax" refer to corresponding names in clause 18.

Syntax diagrams consist of C/ATLAS elements separated by flow lines. Statements are constructed by selecting a path of flow lines from the left of the syntax diagram to the right side and writing in sequence each C/ATLAS element which appears in that path. For example,

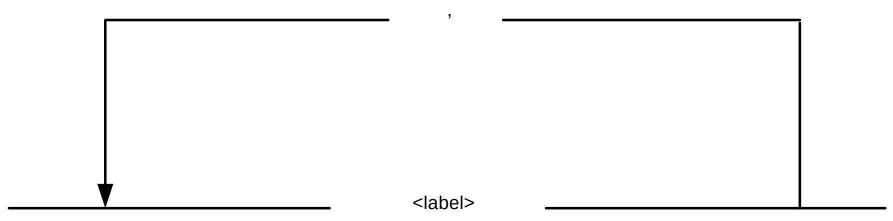


would have the following valid sequences:

APPLY, AC SIGNAL, VOLTAGE

APPLY, DC SIGNAL, VOLTAGE

All syntax diagram flow lines shall be traversed from left to right unless an arrowhead indicates differently. Such arrowheads are commonly used to allow repetition of an element as in the following syntax diagram:



If 'HI_VAL', 'LO_VAL', and 'AVRG_VAL' are names that satisfy the variable <label>, then the following are valid sequences:

'HI_VAL'

'HI_VAL', 'LO_VAL'

'HI_VAL', 'LO_VAL', 'AVRG_VAL'

Three types of C/ATLAS elements are included in C/ATLAS syntax diagrams.

1) Literals. Upper-case and special characters that are included in ATLAS sequences exactly as they appear.

2) Components. Lower-case letters enclosed by the characters "<" and ">" that are replaced by valid symbols and/or numbers as described in clause 15.

3) Subdiagrams. Broken bordered boxes containing a component like the name and the clause/subclause number in which the subdiagram is specified. The box is replaced by sequences valid according to that specified paragraph.

   Subdiagrams being defined by a syntax diagram consist of a solid bordered box to the left of the defining syntax diagram and contain a component like the name and the subclause number in which the definition appears.

2.3.3 Guide to the use of the C/ATLAS language

For guidance on the proper use of C/ATLAS vocabulary, syntax, and structure for the generation of unambiguous test procedures, refer to IEEE Std 771-1989.

3.0 Complete C/ATLAS test program

3.1 <complete atlas test program structure>

A <complete atlas test program structure> (see figure 3-1) consists of an <atlas program structure> that provides control of the overall execution sequence. It may contain one or more <atlas module structure>s that are independently compiled and which provide additional procedures that are invoked at the appropriate points in the test process.

Execution of a <complete atlas test program structure> begins at an entry point (see 15.3.2) and proceeds sequentially by statement except where the sequence is modified by control structure statements.

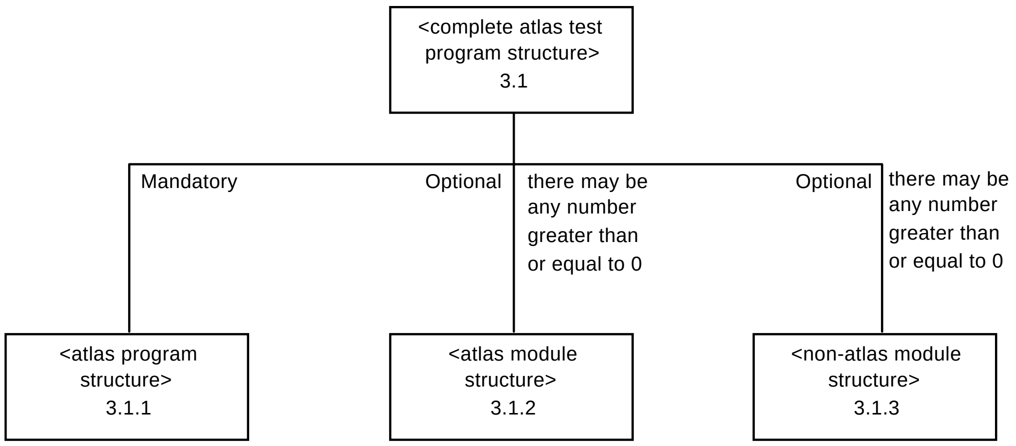

<complete atlas test program structure>
3.1
Mandatory
Optional
there may be any number greater than or equal to 0
Optional
there may be any number greater than or equal to 0
<atlas program structure>
3.1.1
<atlas module structure>
3.1.2
<non-atlas module structure>
3.1.3

Figure 3-1. Complete C/ATLAS test program family tree

3.1.1 <atlas program structure>

An <atlas program structure> consists of four distinct elements: the <begin atlas program statement> which delimits the start of the structure, a <program preamble structure>, the <main procedural structure>, and the <terminate atlas program statement> which delimits the end of the structure.

The <program preamble structure> precedes the <main procedural structure>. Program preamble statements do not cause any tests to be executed. Information contained in program preamble statements may be referenced later in the <program preamble structure>, the <main procedural structure>, an <atlas module structure>, or a <non-atlas module structure> which have been associated with the <atlas program structure> by <include statement>s.

The <main procedural structure> is where the execution of the test procedure starts.

Figure 3-2 illustrates the structure of a C/ATLAS program and identifies the subclause where each structure or statement is located in this standard.

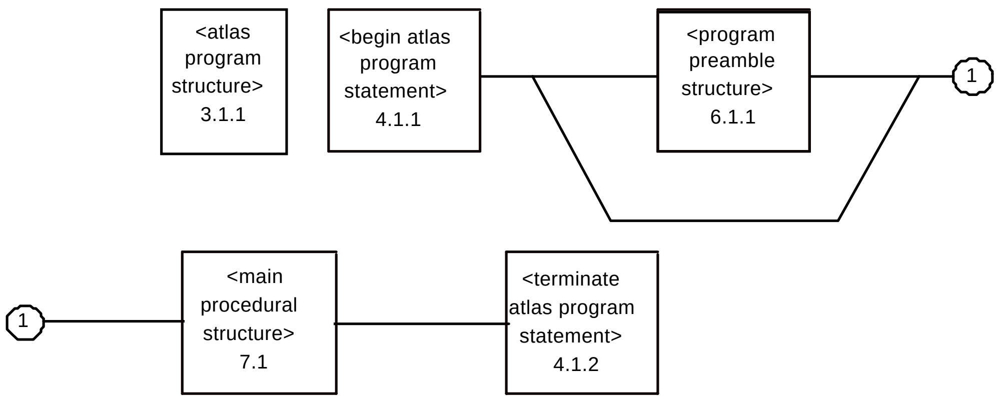

<atlas program structure> 3.1.1
<begin atlas program statement> 4.1.1
<program preamble structure> 6.1.1
1
1
<main procedural structure> 7.1
<terminate atlas program statement> 4.1.2

Figure 3-2. C/ATLAS program structure

3.1.2 <atlas module structure>

An <atlas module structure> consists of three distinct elements: the <begin atlas module statement> which delimits the start of the structure, the <module preamble structure>, and the <terminate atlas module statement> which delimits the end of the structure.

Statements within a <module preamble structure> do not cause any tests to be executed. Information contained in module preamble statements may be referenced later in the same <module preamble structure>, in the <atlas program structure>, in another <atlas module structure>, or in a <non-atlas module structure> when they have been associated by <include statement>s. The <atlas module structure> is referenced in the <program preamble structure> via an INCLUDE statement.

Figure 3-3 illustrates the structure of a C/ATLAS module and identifies the subclause where each structure or statement is located in this standard.

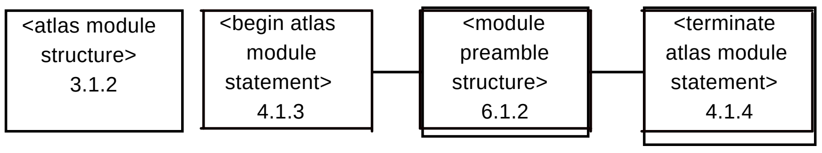

<atlas module structure> 3.1.2
<begin atlas module statement> 4.1.3
<module preamble structure> 6.1.2
<terminate atlas module statement> 4.1.4

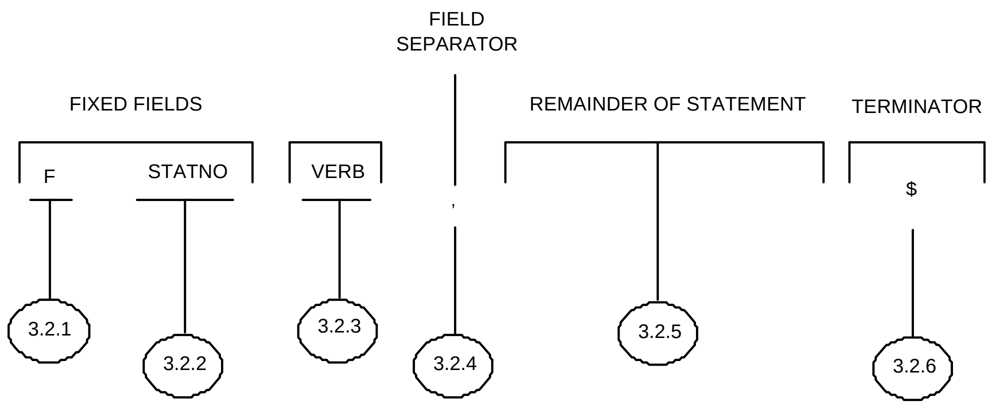
Figure 3-3. C/ATLAS module structure

3.1.3 <non-atlas module structure>

A <non-atlas module structure> is similar to an <atlas module structure> except that it is written in a language other than C/ATLAS. The <non-atlas module structure> is an independent entity and its implementation is dependent upon the run-time system. Information contained in a <non-atlas module structure> may be referenced in the <atlas program structure>, in an <atlas module structure> or another <non-atlas module structure> when they have been associated by <include statement>s.

3.2 Basic statement elements

The structure of a general C/ATLAS statement is shown in figure 3-4. The detailed format for each verb type is specified in the appropriate clause of this document. There is no limit to the length of a C/ATLAS statement. It is not necessary to complete the statement in one line. By inserting spaces between statement elements (as allowed by 15.4), the writer can begin a new line at any point in the statement with subsequent lines indented, if desired. This flexibility allows the writer to highlight some portion of the statement (such as the label) by its location on the page. The first two fields (flag and statement number) shall be written on the first line of the statement.



NOTEùNumbers in circles refer to subclauses in this specification.

Figure 3-4. General C/ATLAS statement structure

3.2.1 Flag field

The flag field can be used to associate special meaning to a C/ATLAS statement or to designate the beginning of a line of text which is commentary (see 15.3.2).

3.2.2 Statement number field

The statement number field is used to uniquely designate C/ATLAS statements (see 15.3.3).

3.2.3 Verb field

The verb field identifies the action for each type of statement. The C/ATLAS verbs are reserved words in that they have unique and special meanings in the C/ATLAS language and shall not be used as variable names.

3.2.4 Field separator

Since the fields following the verb are variable, a separator is required to identify the start of each new major field. A comma followed by zero or more spaces is designated for this purpose. Within some major fields, blank spaces are used as subfield separators (see 15.4).

3.2.5 Remainder of statement

This part of each particular type of C/ATLAS statement varies considerably. The detailed format following each verb-type is specified in the appropriate clause/subclause of this document.

3.2.6 Statement terminator ($)

The last character of every C/ATLAS statement shall be the currency symbol ($).

4.0 Structure delimiter statements

4.1 BEGIN/TERMINATE statements

The BEGIN and TERMINATE statements indicate the first and last statement of an <atlas program structure> or <atlas module structure>.

4.1.1 BEGIN, ATLAS PROGRAM statement

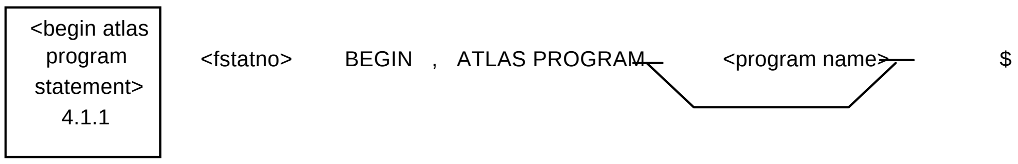
4.1.1.1 Function

To indicate the start of the text of an <atlas program structure>.

4.1.1.2 Formal syntax

Reference begin–atlas–program–statement

4.1.1.3 Syntax diagram



4.1.1.4 Rules

1. The first statement of every <atlas program structure> is a <begin atlas program statement>. The statement can be used only once within a single <atlas program structure>.

2. The optional <program name> is used to identify the <atlas program structure>.

4.1.1.5 Notes and examples

110000 BEGIN, ATLAS PROGRAM 'SERVO AMPLIFIER' $

4.1.2 TERMINATE, ATLAS PROGRAM statement

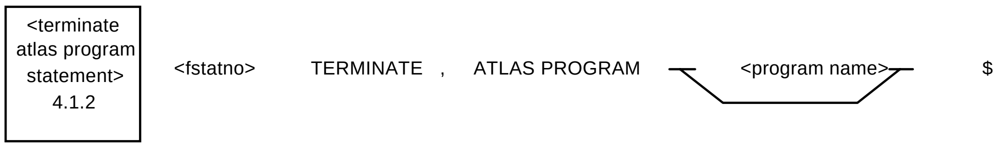
4.1.2.1 Function

To indicate the end of the text of an <atlas program structure>.

4.1.2.2 Formal syntax

Reference terminate–atlas–program–statement

4.1.2.3 Syntax diagram



4.1.2.4 Rules

1. The last statement in every <atlas program structure> is a <terminate atlas program statement>. The statement can be used only once within a single <atlas program structure>.

2. The optional <program name> shall be identical to the <program name> in the <begin atlas program statement>.

4.1.2.5 Notes and examples

999999 TERMINATE, ATLAS PROGRAM 'SERVO AMPLIFIER' $

4.1.3 BEGIN, ATLAS MODULE statement

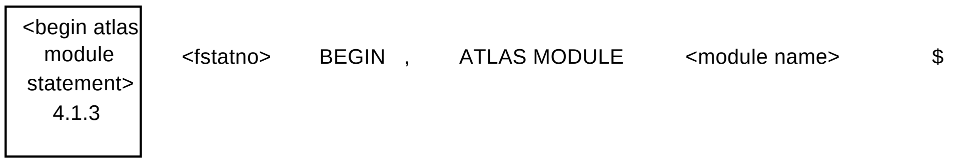
4.1.3.1 Function

To indicate the start of the text of an <atlas module structure>.

4.1.3.2 Formal syntax

Reference begin–atlas–module–statement

4.1.3.3 Syntax diagram



4.1.3.4 Rules

1. The first statement of every <atlas module structure> is a <begin atlas module statement>. The statement can be used only once within a single <atlas module structure>.

2. The <module name> is used to identify the <atlas module structure>, and is referenced by an INCLUDE statement within the <program preamble structure>.

3. The <module name> shall be unique within one <complete atlas test program structure>.

4.1.3.5 Notes and examples

```
120000 BEGIN, ATLAS MODULE 'CALIBRATION PROC'  $
```

4.1.4 TERMINATE, ATLAS MODULE statement

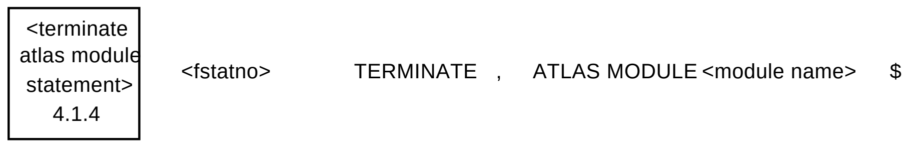
4.1.4.1 Function

To indicate the end of the text of an <atlas module structure>.

4.1.4.2 Formal syntax

Reference terminate–atlas–module–statement

4.1.4.3 Syntax diagram



4.1.4.4 Rules

1. The last statement in every <atlas module structure> is a <terminate atlas module statement>. The statement can be used only once within a single <atlas module structure>.

2. The <module name> shall be identical with the <module name> in the <begin atlas module statement> of the same <atlas module structure>.

5.0 This clause is reserved for future use.

6.0 Preamble statements

An <atlas program structure> may contain a single <program preamble structure>, and each <define procedure structure> may contain a single <local preamble structure>. In each case the preamble serves to identify the names and attributes of each <label> which is subsequently referenced within the appropriate structure.

Within the atlas preamble structures, labels may be established for use in the <atlas program structure>, an <atlas module structure> or a <non-atlas module structure>. These labels are generally classified as follows:

1. Global. A label is global in scope if it has been established with the GLOBAL attribute in one of the structures mentioned above and with the EXTERNAL attribute in the others in which it is to be referenced.

2. Local. A label is local in scope if it is established with neither the GLOBAL nor EXTERNAL attribute. It may then only be accessed within the structure in which it was established.

A full description of global and local attributes is located in 15.6.1.

6.1 Main preamble structure

6.1.1 <program preamble structure>

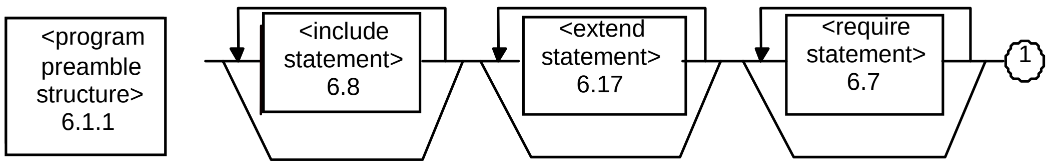
6.1.1.1 Function

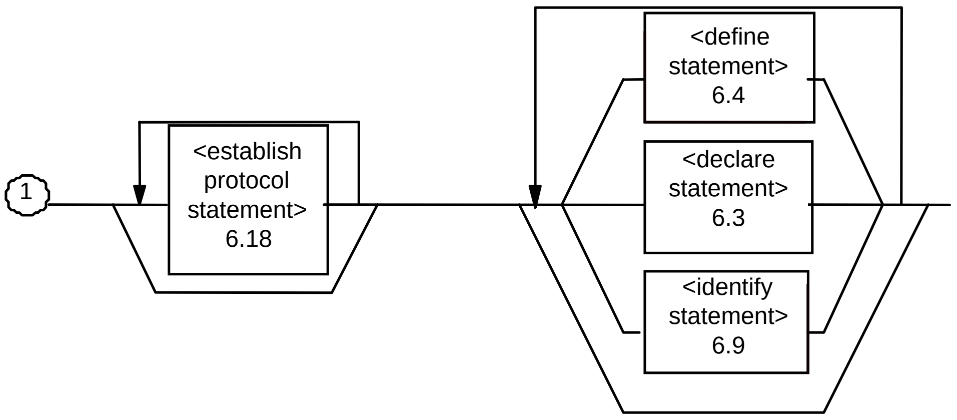
The <program preamble structure>, which appears immediately following the <begin atlas program statement>, establishes the name and attributes for labels whose scope includes the whole of the <atlas program structure>.

6.1.1.2 Formal syntax

Reference program preamble structure

6.1.1.3 Syntax diagram





6.1.1.4 Rules

1. There may be only one <program preamble structure> within an entire <atlas program structure>.

2. The <program preamble structure> shall appear immediately following the BEGIN, ATLAS PROGRAM statement.

3. Any <label> identified within the <program preamble structure> can be referenced within the entire <atlas program structure> unless an identical <label> is established at a lower level structure within a <local preamble structure>.

6.1.2 <module preamble structure>

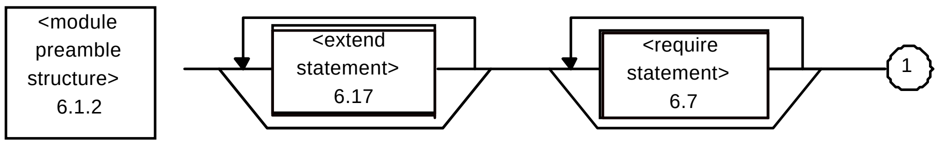
6.1.2.1 Function

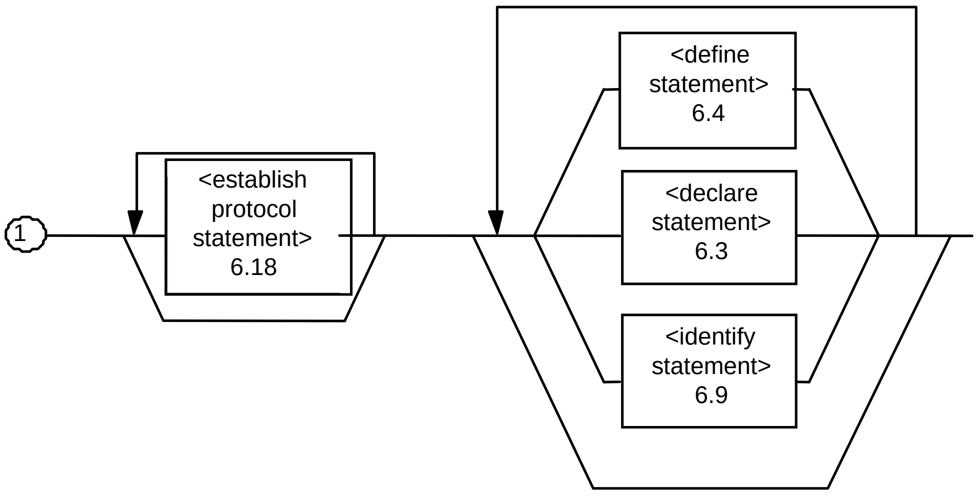
The <module preamble structure> that appears immediately following the BEGIN, ATLAS MODULE statement establishes the name and attributes for labels whose scope includes the whole of the <atlas module structure>.

6.1.2.2 Formal syntax

Reference module preamble structure

6.1.2.3 Syntax diagram





6.1.2.4 Rules

1. There may be only one <module preamble structure> within an entire <atlas module structure>.

2. Any <label> identified within the <module preamble structure> can be referenced within the entire <atlas module structure> unless an identical <label> is established at a lower level structure within a <local preamble structure>. Each label identified as being EXTERNAL shall have a corresponding statement in either the <program preamble structure> which identifies it, or another <module preamble structure> which identifies it as being GLOBAL. All references to the label shall be to the same object, except where an identical label is established at a lower structure within a <local preamble structure>.

6.2 <local preamble structure>

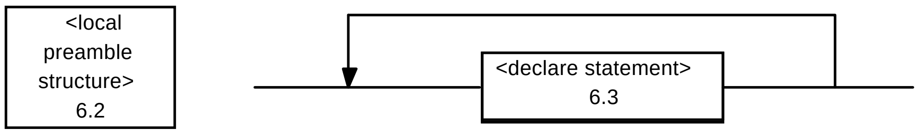
6.2.1 Function

A <local preamble structure> identifies the name and attributes of each <label> that is local to the <define procedure structure>.

6.2.2 Formal syntax

Reference local preamble structure

6.2.3 Syntax diagram



6.2.4 Rules

1. There may only be one <local preamble structure> within each <define procedure structure>.

2. The <local preamble structure> shall appear immediately following a DEFINE PROCEDURE statement within the appropriate structure.

3. A <local preamble structure> only contains DECLARE statements.

4. A <declare statement> within a <local preamble structure> is always local and consequently may not specify either a GLOBAL or EXTERNAL attribute.

6.3 DECLARE statement

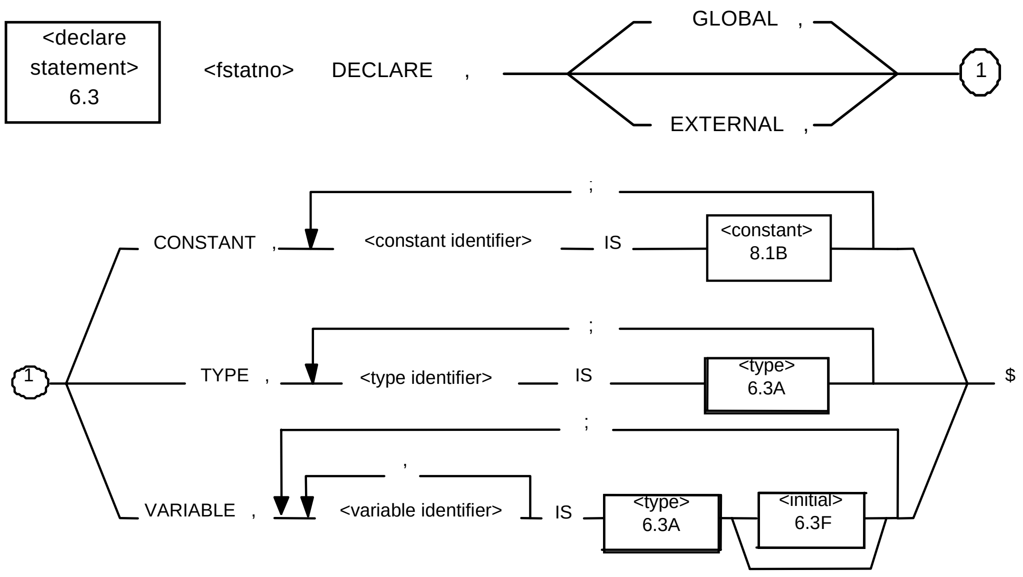
6.3.1 Function

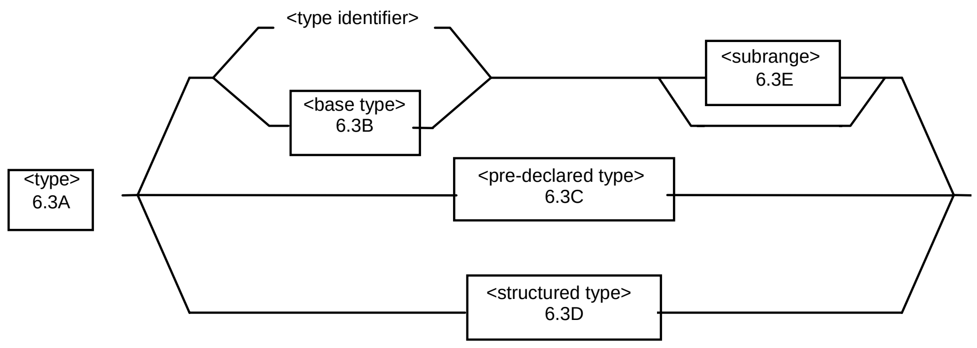
To establish, label, and identify data types, variables, and constants.

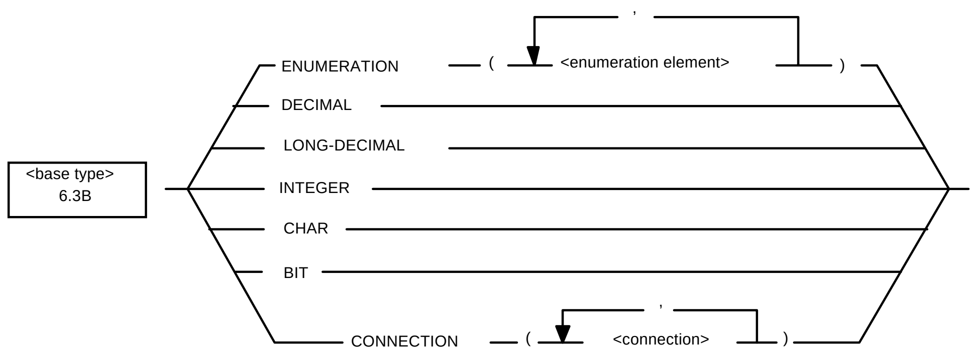
6.3.2 Formal syntax

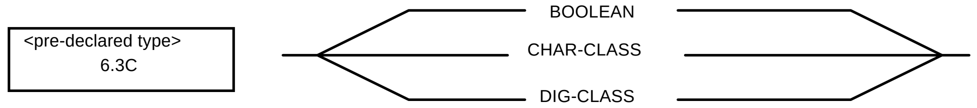
Reference declare statement

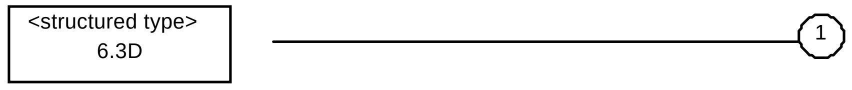
6.3.3 Syntax diagram

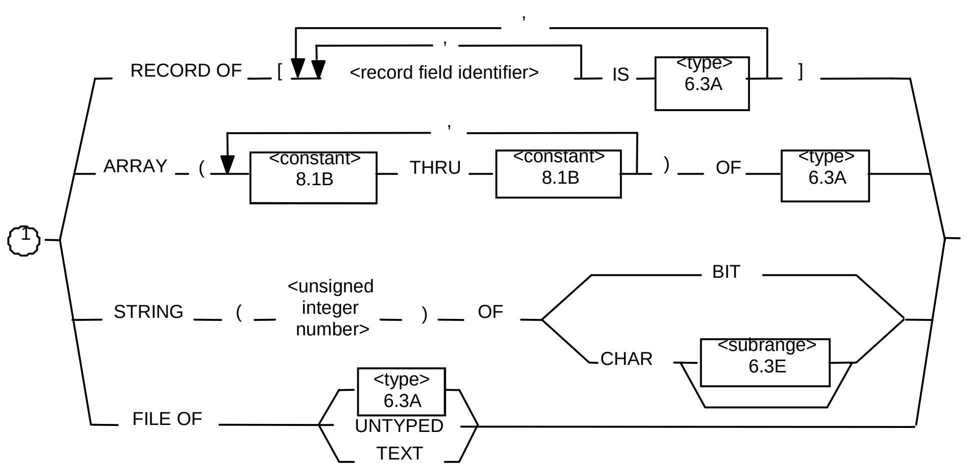


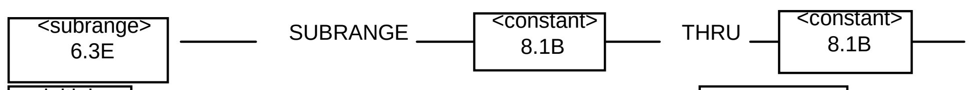

<type>
6.3A
<type identifier>
<base type>
6.3B
<subrange>
6.3E
<pre-declared type>
6.3C
<structured type>
6.3D

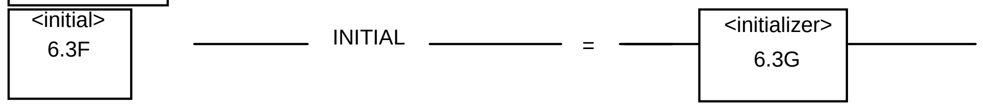

<base type>
6.3B
ENUMERATION
(
<enumeration element>
,
)
DECIMAL
LONG-DECIMAL
INTEGER
CHAR
BIT
CONNECTION
(
<connection>
,
)

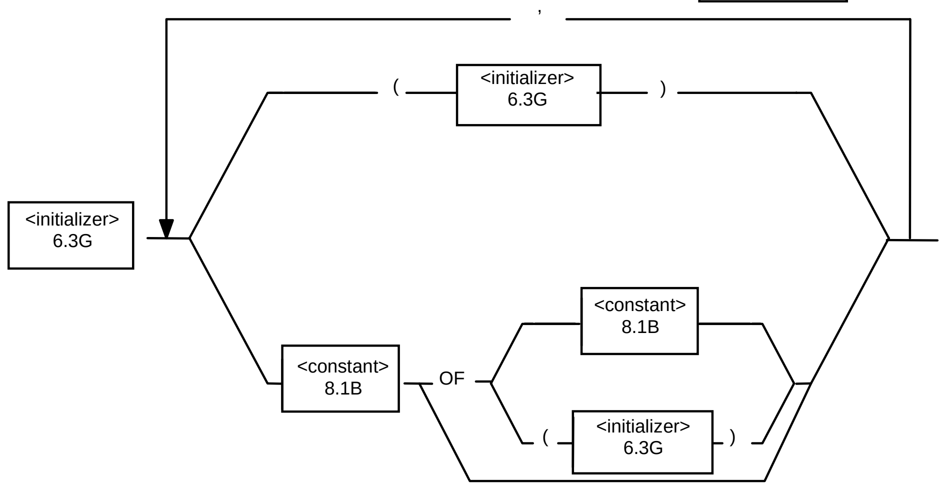

<pre-declared type>
6.3C
BOOLEAN
CHAR-CLASS
DIG-CLASS


<structured type>
6.3D
1


1
RECORD OF
[
<record field identifier>
,
,
IS
<type> 6.3A
]
ARRAY
(
<constant> 8.1B
THRU
<constant> 8.1B
,
)
OF
<type> 6.3A
STRING
(
<unsigned integer number>
)
OF
BIT
CHAR
<subrange> 6.3E
FILE OF
<type> 6.3A
UNTYPED
TEXT


<subrange> 6.3E
SUBRANGE
<constant> 8.1B
THRU
<constant> 8.1B


<initial> 6.3F
INITIAL
=
<initializer> 6.3G


<initializer> 6.3G
,
(
<initializer> 6.3G
)
<constant> 8.1B
OF
<constant> 8.1B
(
<initializer> 6.3G
)

6.3.4 Rules

6.3.4.1 Rules for CONSTANT, TYPE and VARIABLE forms of <declare statement>

1. The CONSTANT, TYPE, or VARIABLE <label> created is accessible at the structural level where it is declared and at any inner nested structural level.

2. Repetition via the semicolon, ";", allows for the creation of multiple CONSTANT, TYPE, or VARIABLE <label>s in one DECLARE statement without keyword repetition.

3. References to CONSTANT, TYPE, and VARIABLE <label>s can only be made after they have been properly declared in a DECLARE statement.

4. A <label> that is declared as GLOBAL shall be unique within the <complete atlas test program structure>. A GLOBAL <label> declared in an <atlas program structure> may be accessed in an <atlas module structure> if that <atlas module structure> contains a DECLARE for that <label> as EXTERNAL. A GLOBAL <label> declared in an <atlas module structure> may be accessed in another <atlas module structure> or in the <atlas program structure> if that <label> is declared as EXTERNAL in those structures.

5. The attributes GLOBAL and EXTERNAL are not allowed in a <local preamble structure>.

6. Within any main preamble structure or any <local preamble structure> all declared labels shall be unique. However, <record field identifier> labels in different record types may be the same. Where nested preambles use the same labels, the innermost sense shall prevail following the redefinition until the structure in which it is redefined is exited.

7. If a <label> is declared as GLOBAL in one structure and as EXTERNAL in another then the two shall be of the same <type> and:

a) If they are ARRAYs, the corresponding dimensions shall have the same number of components with the same index number.

b) If they are STRINGs, they shall have the same number of components.

8. INITIAL cannot be used to initialize an EXTERNAL variable.

6.3.4.2 Rules for CONSTANT form of <declare statement>

The type of the <constant identifier> label created is determined by and is the same as that of the constant. This type depends on the form of the constant in accordance with 8.1.4.5.

6.3.4.3 Rules for TYPE form of <declare statement>

TYPE declarations are used to avoid repetition of identical descriptions of data structures in VARIABLE declarations.

6.3.4.4 Rules for VARIABLE form of <declare statement>

1. A VARIABLE declaration creates storage space for that variable according to its <type>. The <type> is expressed either explicitly in the VARIABLE declaration or by reference to a previously declared <type identifier>. The VARIABLE will be initialized if <initial> is invoked.

2. Any number of VARIABLEs of the same <type> can be created using the path containing the comma.

3. VARIABLEs declared in a <local preamble structure> cannot be relied on to retain their values between invocations of the structure containing that <local preamble structure>.

4. Initialization of VARIABLEs is discussed in 6.3.4.10.

6.3.4.5 Rules for <type>

1. A DECLARE, TYPE statement establishes a <type identifier> label which is associated with the described <type>.

2. When used in a DECLARE, VARIABLE statement or in a DEFINE, PROCEDURE statement within a <parametre> subfield, the description of the <type> can be stated explicitly without reference to a <type identifier> label.

3. A value is assignment compatible with a <data store> provided that one of the following conditions is true:

a) They are not <structured type>s, are of the same <base type>, and if the <data store> has been declared with a sub-range, then the value lies within that sub-range.

b) Their <structured type> is STRING OF BIT and the current length of the value is less than or equal to the declared length of the <data store>.

c) Their <structured type> is STRING OF CHAR, the current length of the value is less than or equal to the declared length of the <data store>, and if the <data store> has been declared with a sub-range of CHAR, then the value of each character lies within that sub-range.

d) Their <structured type> is ARRAY, they have the same dimensions with the same number of components, and each component is assignment compatible with the corresponding component of the <data store>.

e) Their <structured type> is RECORD and their corresponding fields are assignment compatible.

f) The <data store> is of <base type> LONG DECIMAL, the value is of <base type> DECIMAL or INTEGER, and if the <data store> has been declared with a sub-range, then the value, when coerced to type LONG DECIMAL, lies within that sub-range.

g) The <data store> is of <base type> DECIMAL, the value is of <base type> INTEGER, and if the <data store> has been declared with a sub-range, then the value, when coerced to type DECIMAL, lies within that sub-range.

h) The <data store> is of <structured type> STRING OF CHAR, the value is of <base type> CHAR, and if the <data store> has been declared with a sub-range, then the value of the CHAR lies within that sub-range. Assignments according to this rule set the dynamic length of the STRING OF CHAR to one.

i) The <data store> is of <structured type> STRING OF BIT and the value is of <base type> BIT. Assignments according to this rule set the dynamic length of the STRING OF BIT to one.

6.3.4.6 Rules for <base type>

1. A variable of <base type> shall assume a value whose type conforms to the declared <base type>. Thus, for example, a <variable identifier> of type DECIMAL may only assume a <decimal number>.

2. The <enumeration element> labels of a particular declared ENUMERATION type are <constant>s ordered by the sequence in which they are written into the declaration. They have the same format as other labels, including the requirement for enclosing them in apostrophes.

3. The <enumeration element> labels shall be unique within the <atlas module structure> or <atlas program structure> in which they are defined.

4. A CONNECTION variable identifies a UUT <connection> as defined in 14.12 <conn> (Connection Field).

5. A CONNECTION variable can only assume values taken from the given <connection> list. The <connection> elements in all CONNECTION type declarations shall be unique within the <atlas module structure> or <atlas program structure> in which they are defined.

6. LONG DECIMAL <type> provides for an increased number of significant digits in comparison with DECIMAL <type> in order to gain high accuracy. See 15.2, item g.

6.3.4.7 Rules for <pre-declared type>

1. The <pre-declared type>s are commonly used ENUMERATIONs. The <pre-declared type>s exist as though each C/ATLAS program contained the following declaration:

```
DECLARE, GLOBAL, TYPE,
    'BOOLEAN' IS ENUMERATION ('FALSE', 'TRUE');
    'DIG_CLASS' IS ENUMERATION ('BNR', 'B1C', 'B2C', 'BSM', 'BCD',
'SBCD');
    'CHAR_CLASS' IS ENUMERATION ('ASCII7', 'ISO7') $
```

2. The <type identifier>s and <enumeration element>s that are enclosed by apostrophes in the above declaration defining <pre-declared type>s are written without apostrophes in an <atlas program structure> or <atlas module structure> in accordance with 15.5.

3. DIG_CLASS variables have values that are associated with digital data formats as follows:

BNR	for unsigned positive binary numbers (no sign bit)
B1C	for ones-complement binary numbers
B2C	for twos-complement binary numbers

BSM	for sign-plus-magnitude binary numbers
BCD	for binary-coded-decimal numbers (4 bits per digit)
SBCD	for sign-plus-magnitude BCD numbers (4 bits per digit plus a sign bit)

BCD and SBCD numbers have an 8-4-2-1 coding.

Signed numbers have a sign bit in the most significant bit position with a binary zero representing a positive sign.

Other types of coding and sign conventions can be represented by BNR, with the value of each bit of a number being translated into and out of C/ATLAS number types by conversion routines defined by a <define procedure structure>.

4. CHAR_CLASS variables have values that are associated with character codes as follows:

ASCII7 for ASCII characters transmitted across the UUT/test equipment interface (7 bits per character) (see 15.9.1).

ISO7 for ISO characters transmitted across the UUT/test equipment interface (7 bits per character) (see 15.9.2).

6.3.4.7.1 Notes and examples

1. Values of DIG_CLASS are arguments to the <pre-defined function>s DIG and DEC.

2. Values of CHAR_CLASS are arguments to the <pre-defined function>s CHAR and BIT.

6.3.4.8 Rules for <structured type>

1. An ARRAY is a collection of data items of the same <type>. Any <type> except FILE may be specified. An ARRAY may have any number of dimensions. Each dimension has an associated index that shall be bounded by upper and lower limits. A multi-dimensional ARRAY declaration is structurally identical with and is shorthand for a declaration of an ARRAY of an ARRAY. For example:

```
ARRAY (1 THRU 7, 1 THRU 9)
```

is shorthand for:

```
ARRAY (1 THRU 7) OF ARRAY (1 THRU 9)
```

The <constant>s that specify the boundary limits of a dimension of an ARRAY shall be of the same <type>. The <type> shall be either INTEGER, CHAR, or <enumeration element> of the same ENUMERATION type. The lower boundary limit is written first. Any element in an ARRAY can be addressed by its indices.

2. Array elements are assigned in row major order, such that the last written index varies most rapidly and the first written index varies more slowly in the conventional manner.

3. Operations between entire arrays require that the arrays involved are assignment compatible and have an equal number of elements. The only operations permitted between entire arrays are assignment and test for equality and inequality of all corresponding elements in both arrays.

4. A RECORD is a collection of data items that need not be of the same <type> or structure. Any type except FILE may be specified. Individual fields within a RECORD are given field names. Items of a RECORD are addressed by appending a period followed immediately by the <record field identifier> label to the RECORD label. The declaration of fields of records is the same as for variables and the rules of 6.3.4.4 apply.

5. The only operations permitted between entire records are assignment and test for equality and inequality of all corresponding fields in both records. Records shall be assignment compatible and have equal numbers of elements.

6. The constant specifying the length of a string shall be greater than zero and of <base type> INTEGER.

7. The <base type> of a STRING shall be either BIT or CHAR, or a subrange of CHAR.

8. A STRING has an associated dynamic length attribute that holds the current length of the STRING.

a) Initialization, input, parametre passing, and assignment (whether explicit as in CALCULATE or implicit as in such verbs as SENSE) set the dynamic length.

b) References to an individual BIT or CHAR of a STRING are allowed within the current dynamic length.

c) Operations between STRINGs of the same <base type> and different declared sizes are permitted.

d) It is an error if an operation on a STRING would cause the current dynamic length to exceed the declared length.

9. Conversion of a STRING OF BIT to and from a numerical value is accomplished by using the DEC and DIG functions respectively, in accordance with the DIG_CLASS parametre.

10. Conversion of a STRING OF BIT to and from a character is accomplished by using the CHAR and BIT functions respectively, in accordance with the CHAR_CLASS parametre.

11. Within RECORDs, any number of fields of the same <type> can be created by using the path with a comma.

12. Within RECORDs, use of the ";" repetition operator allows for the creation of any number of fields, which may be of different types.

13. A FILE is a collection of data items that may not contain another FILE. Each data item has a position in the FILE beginning with zero and advancing by one with each data item. Three classes of FILE are specified:

a) A known type file consists of a sequence of data items all of which have the same type.

Input and output shall be to and from a variable of identical type.

b) An untyped file consists of bytes of data that may be representative of anything. Input and output for this type of file requires a detailed knowledge of the data format, which may be very machine dependent. For graphic applications, a graphic file containing all the needed graphic data should be declared as an untyped file.

c) A text file consists of a stream of characters that equate to declared data types, after possible automatic I/O conversion.

6.3.4.9 Rules for <subrange>

1. A <subrange> is used to restrict the range of the values that may be assigned to a variable to lie within a sub-division of the range of values that may be assigned to its <base type>.

2. The type of <constant> shall conform to the <base type>.

3. The value of the <constant>s shall lie within the bounds of the <type> being ranged. The value of the first constant shall be less than or equal to the value of the second constant.

4. The <type> being ranged shall be ENUMERATION, DECIMAL, LONG_DECIMAL, INTEGER, or CHAR and shall not be any structure.

5. A <subrange> variable is of the same type as a variable of the associated <base type>.

6. Initialization, input, assignment, and parametre passing to subrange variables shall lie within the subrange bounds.

6.3.4.10 Rules for <initial>

1. The action of an <initial> field is to set the initial value of variables. Initialization of variables always takes place in the order in which the variables are declared.

a) For declarations in a <program preamble structure> or <module preamble structure> and not in a <local preamble structure>, initialization takes place prior to the execution of the first executable statement of the <main procedural structure>.

b) For declarations in a <local preamble structure>, initialization takes place each time the procedure is invoked prior to the execution of the first executable statement following the <local preamble structure> that contains the declaration.

2. Variables cannot be initialized if:

a) They are of type FILE.

b) They are EXTERNAL.

3. INITIAL can be used to initialize a single variable or a series of variables. The <initializer> following INITIAL is a comma delimited list of values having a one-to-one correspondence with the variables. Each value may be one of the following:

a) A <constant> that corresponds to a simple variable.

b) A <constant> number of <constant>s where the first <constant> shall evaluate to a positive integer that specifies the number of instances of the second <constant>. Each instance of the second <constant> corresponds to a simple variable.

c) A parenthesized list of values having a one-to-one correspondence with the elements of an ARRAY or <record field identifier>s of a RECORD.

d) A shorthand notation for a repeated occurrence of a parenthesized list of values defined by item c). This is in the form of "<constant> OF (list of values)", where <constant> shall be a positive integer that specifies the number of times the parenthesized list is to be repeated (e.g.,"2 OF (0,1)" is equivalent to "(0,1), (0,1)").

4. The value of the <constant> supplied by the <initializer> shall be assignment compatible with the corresponding variable.

5. For ARRAYs, initial values are assigned starting with the lowest value of all indices and varying the last written index most rapidly.

6.3.5 Notes and examples

1. This example shows the multiple <variable identifier> use of DECLARE. In this example, three locations are set aside for pin designations.

```
DECLARE, VARIABLE,
   'PIN-1', 'PIN-2', 'PIN-3' IS
   CONNECTION (J1, J2, J3, J4) $
```

2. This example is used to illustrate how storage is set aside for a <variable identifier> of type STRING OF CHAR. The length of the character string stored in this location is equal to or less than 36 characters.

```
DECLARE, VARIABLE,
   'MESSAGE' IS STRING (36) OF CHAR $
```

3. This example illustrates how three storage locations, each 10 bits wide, are specified. The format of the data stored in these locations can be controlled by using the <pre-defined function> DIG for digital numbers in which a bit weight is to be associated with each bit position.

```
DECLARE, VARIABLE,
   'STIM', 'MASK', 'RESP' IS STRING (10) OF BIT $
```

4. This example is a use of <record field identifier> to specify storage for 'PATTERN' consisting of 22 bits composed of three individual parts 'PART1', 'PART2', and 'PART3' consisting of 4, 8, and 10 bits respectively.

```
DECLARE, VARIABLE,
   'PATTERN' IS
   RECORD OF ['PART1' IS STRING (4) OF BIT;
              'PART2' IS STRING (8) OF BIT;
              'PART3' IS STRING (10) OF BIT] $
```

5. This example shows how three one dimensional arrays, each containing five elements, are declared.

```
DECLARE, VARIABLE,
   'INDEX', 'WEIGHT FACTOR', 'COUNTER'
   IS ARRAY (1 THRU 5) OF INTEGER $
```

6. This example shows how storage may be established for an array of connection variables.

```
DECLARE, VARIABLE,
   'PINSET1' IS ARRAY (1 THRU 4) OF
   CONNECTION (J1, J3, J5, J7) $
```

7. This example establishes storage for five bit-strings each of which may be up to 20 bits in length.

```
DECLARE, VARIABLE,
   'COMMAND' IS
   ARRAY (1 THRU 5) OF STRING (20) OF BIT $
```

8. This example illustrates how several structures may be declared and initialized within a single DECLARE statement.

```
DECLARE, VARIABLE,
   'SINGLE' IS DECIMAL INITIAL = 1.234;
   'LEFT', 'RIGHT', 'CENTER' IS
   BOOLEAN INITIAL = 3 OF TRUE;
   'PATTERN' IS ARRAY (1 THRU 1200)
   OF STRING (16) OF BIT
   INITIAL = (1200 OF X'0000');
   'WEEK-DAY' IS RECORD OF
   ['SUN', 'MON', 'TUE', 'WED',
   'THURS', 'FRI', 'SAT' IS BOOLEAN]
   INITIAL = (FALSE, 5 OF TRUE, FALSE) $
```

9. This example illustrates the declaration of connection data elements.

```
DECLARE, GLOBAL, VARIABLE,
   'PINSET1' IS CONNECTION (PIN1, PIN2, PIN3);
   'PINSET2' IS CONNECTION (PIN4, PIN5, PIN6);
   'PINSET3' IS CONNECTION (PIN7, PIN8, PIN9) $
```

10. This example illustrates the declaration and initialization of enumerated data elements and the record structure.

```
DECLARE, VARIABLE,
   'CHESS_BOARD' IS ARRAY (1 THRU 8, C'A' THRU C'H')
      OF ENUMERATION ('EMPTY', 'BLACK', 'WHITE')
   INITIAL = ((2 OF (8 OF 'BLACK')),
      (4 OF (8 OF 'EMPTY')),
      (2 OF (8 OF 'WHITE')));
   'FAILURES' IS RECORD OF
      ['COUNT' IS INTEGER;
      'FAIL_MESS' IS
      ARRAY (1 THRU 2) OF STRING (20) OF CHAR]
   INITIAL = (0, (C'OVER PRESSURE',
      C'OVER VOLTAGE')) $
```

11. This example shows the initialization of an array composed of records.

```
DECLARE, VARIABLE,
   'DATES' IS ARRAY (1 THRU 100) OF RECORD OF
   ['DAY', 'MONTH', 'YEAR' IS INTEGER]
   INITIAL = (100 OF (0,0,0)) $
```

12. This example illustrates the concept of most significant bit (MSB) and least significant bit (LSB) for digital data patterns:

```
DECLARE, VARIABLE,
   'DATA PATTERNS' IS ARRAY (1 THRU 1000)
   OF STRING (32) OF BIT $

DECLARE, VARIABLE,
   'MSB', 'LSB' IS STRING (32) OF  BIT
   INITIAL = (X'80000000', X'00000001') $
```

6.4 DEFINE statements definition

6.4.1 Function

The function of DEFINE statements is to establish and label a portion of the <complete atlas test program structure> that is not executed until the label is invoked by direct or indirect reference in a subsequent <main procedural statement>.

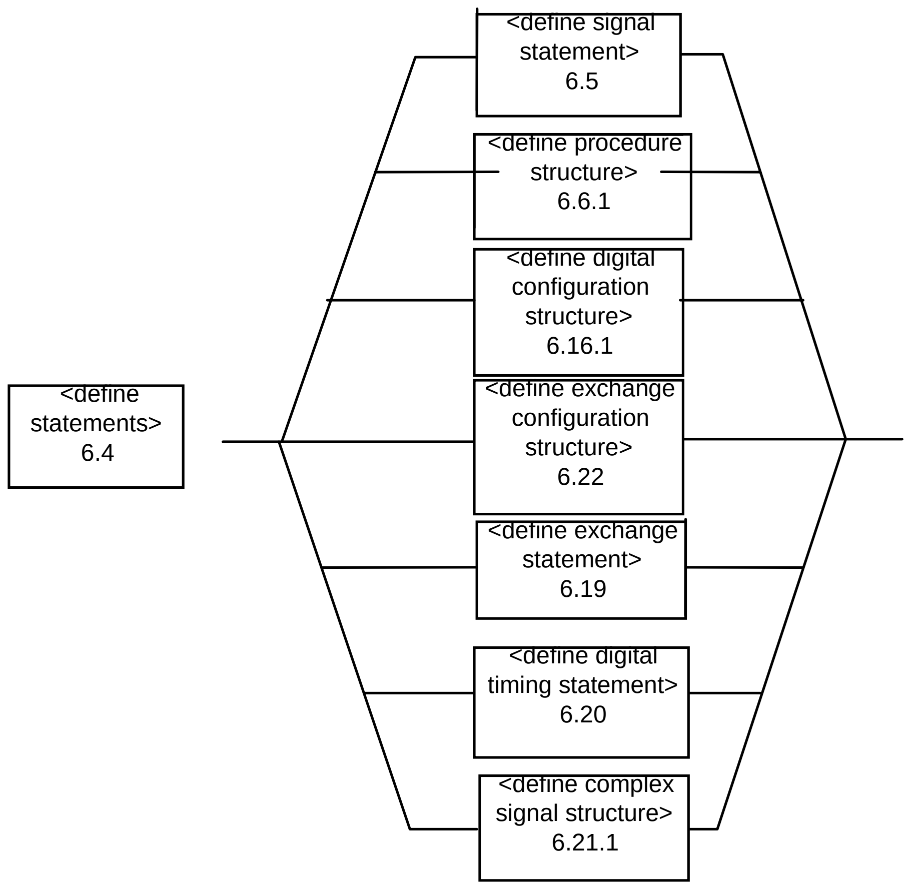
DEFINE statements establish and identify all or part of one statement or a series of statements.

This capability can be employed to avoid rewriting often-used C/ATLAS segments by simply writing the <label> in the appropriate place in a subsequent DEFINE or procedural statement.

6.4.2 Formal syntax

Reference define statements

6.4.3 Syntax diagram


<define statements>
6.4
<define signal statement>
6.5
<define procedure structure>
6.6.1
<define digital configuration structure>
6.16.1
<define exchange configuration structure>
6.22
<define exchange statement>
6.19
<define digital timing statement>
6.20
<define complex signal structure>
6.21.1

6.4.4 Rules applicable to all DEFINE statements

1. DEFINE statements are not allowed in a <local preamble structure>.

2. A DEFINE statement may only refer to previous DEFINE statements by the appropriate <label>.

6.5 DEFINE <signal> statement

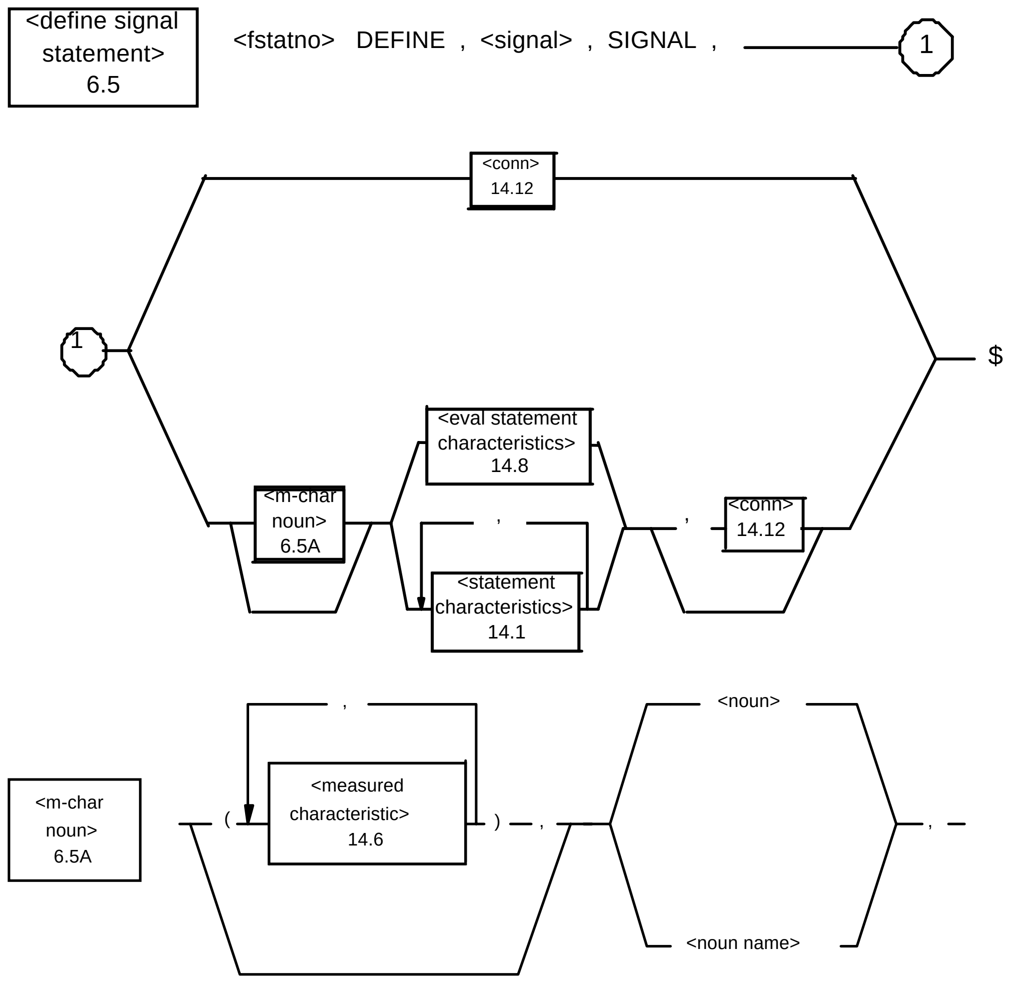
6.5.1 Function

The <define signal statement> assigns a <label> to one or more adjacent fields of a C/ATLAS signal-oriented statement. When the <signal> label is written in a subsequent statement, a strict character substitution takes place to form a complete statement. Quantities in the <define signal statement> may be represented by a parentheses pair ( ) to indicate that the quantity is variable and a specific value will be provided when the label is referred to in the subsequent <define statement> or <main procedural statement>. In a <define signal statement> the number of spaces between the parentheses is not important.

6.5.2 Formal syntax

Reference define signal statement

6.5.3 Syntax diagram



<define signal statement> 6.5
<fstatno> DEFINE , <signal> , SIGNAL ,
1
<conn> 14.12
$
<eval statement characteristics> 14.8
<m-char noun> 6.5A
,
<statement characteristics> 14.1
<conn> 14.12
<m-char noun> 6.5A
(
<measured characteristic> 14.6
)
<noun>
<noun name>

NOTE A <decimal number>, <digital number>, and <connection> may be designated as variable by replacing the item by a pair of parentheses. <label>s defined in a <define signal statement> may be used either in subsequent DEFINE statements or in procedural statements. The value for any parentheses pairs in the definition shall be stated explicitly whenever the <label> is used. The values are listed in sequence immediately following the <label> and will replace the parentheses pairs in the definition in strict sequence, and then the complete character sequence from the definition will replace the <label> and all variables, if any, following it.

A valid C/ATLAS statement shall result when the substituted information is combined with the other information in the statement. Following the parentheses pairs designating a variable quantity, a range field containing non-variable quantities may be given (is optional).

6.5.4 Rules

1. This type of statement is employed to establish and identify one or more fields that follow the verb of a signal-oriented statement for direct substitution in place of the <label> when the <label> is written in a subsequent statement.

2. Only complete comma-to-comma or comma-to-dollar fields that are adjacent in the completed procedural statement may be included in the same DEFINE signal statement.

3. Connections and the numerical values associated with any <real quantity> in a DEFINE statement may be designated variable by inserting a parentheses pair in place of the number or connection. The dimensions of the variables shall immediately follow the parentheses pairs. Only blank spaces, if desired, may be written between the left and right parentheses. When a characteristic is designated variable in a <define signal statement>, a range field may be included to state the lowest and highest values that will be assigned to that characteristic in subsequent statements.

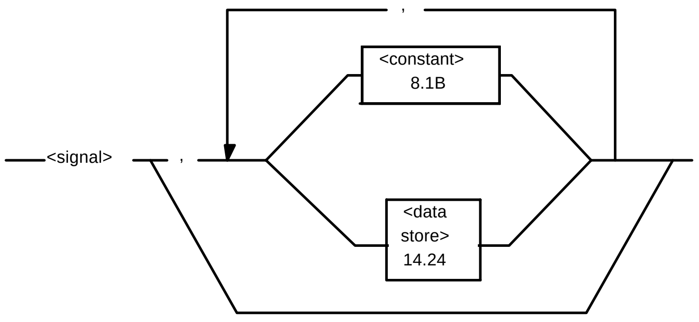
4. In the statement that references the definition, the <label> is inserted in place of the defined fields and is followed by the value(s) of any parenthetical variables from the DEFINE statement. These values may be either constant or DECLAREd variables. Each value is written in the same sequence as defined, and preceded by a comma regardless of the location of commas in the DEFINE statement.

5. If a DEFINE <signal> references the <label> of a previous DEFINE <signal> statement, then any pairs of parentheses in the referenced DEFINE shall be filled explicitly. They may not be filled by "daisy-chaining" through another pair of parentheses in the calling DEFINE statement.

6. A dimension should not be specified as a variable.

7. The DEFINE <signal> statement provides a standard format that may be used to predefine portions of signal-oriented statements. It may also be used to predefine subfields of the REMOVE statement that conform to the standard format and to predefine the connection field in digital statements. The mechanism for using predefined portions of a signal-oriented statement is not included in the syntax diagrams of clause 11 or in the corresponding Formal Syntax Definition.

8. Signals are referenced within individual statements according to the following diagram:



6.5.5 Notes and examples

Each example in this clause is written with a corresponding procedural statement.

```
000011 DEFINE, 'RZ STIM', SIGNAL,  AC SIGNAL,
          VOLTAGE ( ) V, FREQ ( ) HZ,
          CNX HI J2-1 LO J2-2 $
111133 APPLY, 'RZ STIM', 110, 400 $
```

Note that the label is inserted in its appropriate place in the procedural statement. Definition 000011 included the noun; so, in step 111133, 'RZ STIM' is written following the verb.

```
000022 DEFINE, 'GRID', SIGNAL,
          VOLTAGE ( ) RANGE 1V TO 10V ERRLMT + - 10 PC,
          CNX HI J2-3 LO J2-4 $
111144 APPLY, DC SIGNAL, AC_COMP MAX 5MV, 'GRID',  'VDC'  $
111155 APPLY, AC SIGNAL, DC_OFFSET 20V, 'GRID',  'VDC2'  $
```

Step 000022 did not include the noun; so, in steps 111144 and 111155, it is possible to use any noun and provide statement characteristics in addition to VOLTAGE. The only restriction in this regard is that the completed statement, after substitutions, conforms to C/ATLAS format requirements.

```
000033 DEFINE, 'PWR PINS', SIGNAL,
          CNX HI ( ) LO ( ) $
000099 APPLY,  DC SIGNAL, VOLTAGE 500V, AC_COMP 5V,
          'PWR PINS', J1-1, J1-4 $
```

This example illustrates how a connection field can be defined by itself. Note that the pins are separated by commas in the APPLY statement because they are values for separate parametres, even though a comma does not appear in the normal pin format.

```
112244 DEFINE, 'MULTIMETRE', SIGNAL, AC SIGNAL,
          FREQ MAX 410 HZ, VOLTAGE MAX 15V,
          CURRENT MAX 1 MA,
          CNX HI J1-2 LO J1-3 $
    88 MEASURE,(FREQ INTO 'SAVE1'),'MULTIMETRE' $
    98 MEASURE,(VOLTAGE INTO 'SAVE2'),'MULTIMETRE' $
    99 MEASURE,(CURRENT INTO 'SAVE3'),'MULTIMETRE' $
```

This example illustrates how a sensor signal may be defined. In the procedural section, various <measured characteristic> fields can be supplied to make the signal serve for several measurement operations.

6.6 PROCEDURE definition

6.6.1 <define procedure structure>

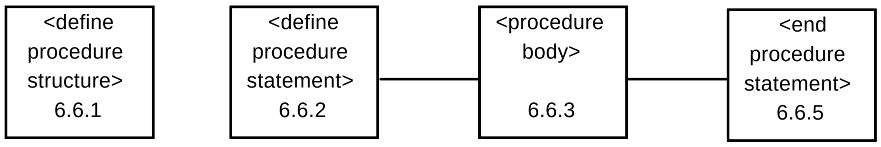
6.6.1.1 Function

The <define procedure structure> identifies one or more complete C/ATLAS statements that may be subsequently invoked by use of the PERFORM statement. The definition consists of three parts: the <define procedure statement>, the <procedure body>, and finally, the <end procedure statement>.

6.6.1.2 Formal syntax

Reference define procedure structure

6.6.1.3 Syntax diagram


<define procedure structure> 6.6.1
<define procedure statement> 6.6.2
<procedure body> 6.6.3
<end procedure statement> 6.6.5

6.6.1.4 Rules

1. The <define procedure structure> is employed to establish and label one or more complete C/ATLAS statements that are executed in the sequence defined whenever the procedure label is called in a subsequent PERFORM statement.

2. The first statement of the series is a <define procedure statement>. The series of C/ATLAS statements is written next. The last statement of the series is an <end procedure statement>.

3. There is no limit to the number of statements that may be included in a defined procedure.

6.6.1.5 Notes and examples

```
000421  DEFINE,'NBD',PROCEDURE ... $  define procedure statement
       DECLARE,...$
       .
       .
       APPLY, ...$                          procedure body
       .
       .
       END, 'NBD' $                  end procedure statement
```

6.6.2 DEFINE PROCEDURE statement

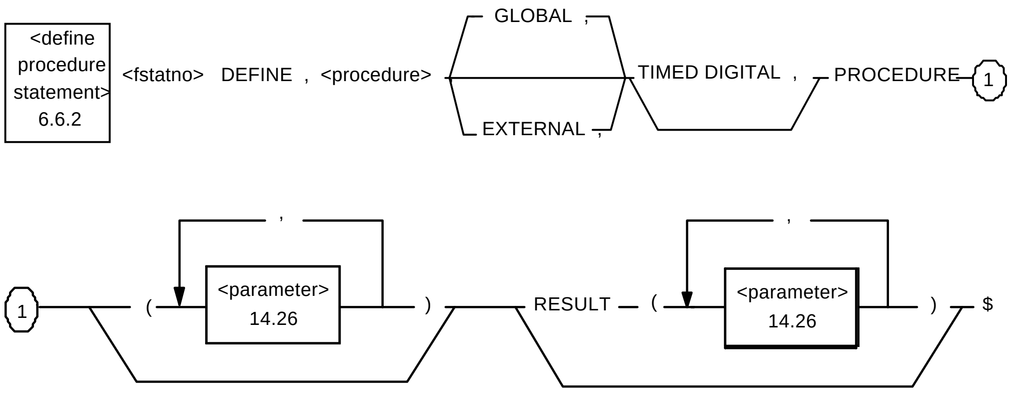
6.6.2.1 Function

The <define procedure statement> identifies the following <procedure body> and assigns labels (known as formal parametres) to data items, if any, which are referenced within the following <procedure body>. The formal parametres are equated to actual parametres at the time of invocation by the PERFORM statement. This vehicle for transferring data to and from the procedure provides the facility to change, if desired, specified data items at each procedure invocation.

6.6.2.2 Formal syntax

Reference define procedure statement

6.6.2.3 Syntax diagram



6.6.2.4 Rules

1. Formal parametre names for a specific <define procedure statement> shall be unique. The names may be the same as labels or formal parametres of other procedures, however, it is recommended that such naming conflicts be avoided.

2. The parametre labels assigned in the formal parametre list will be written in place of the corresponding numbers or connections in the C/ATLAS statements within the <procedure body>.

3. When the procedure is referenced by a PERFORM statement, a value is passed to the <parametre>s, thus making the value available for use in subsequent statements within the <procedure body>.

Note A <label> specified in a DECLARE statement in the <main preamble structure> may be referenced from within procedure bodies without the need to be passed as a formal parametre.

4. Any formal parametres appearing in the first parenthetical pair should only be referenced in the <procedure body> as an independent variable since they are called by value. Any

assignment to this class of parametres will not cause assignment of the equivalent actual parametre during procedure invocation.

5. Formal parametres appearing in the second parenthetical pair may be referenced in the <procedure body> as either dependent or independent variables. RESULT <parametre>s are called by reference, with the restriction that the actual parametres shall be variables.

6. Variables shall be assigned labels in the formal <parametre> list of the DEFINE <procedure> statement. This list follows the word PROCEDURE in the statement and is formed by writing the <parametre> labels of all values to be input to the procedure at execution time. The list is enclosed in parentheses.

7. A <procedure> can be made global by using the GLOBAL attribute. In this case, the <procedure> has to be unique within the entire <complete atlas test program structure>. A global <procedure> can be referenced in modules if it is established in the <atlas program structure> or in another module of the <complete atlas test program structure>. It can be referenced in an <atlas program structure> if it is established in a module of the <complete atlas test program structure>.

8. To reference a global <procedure>, the <define procedure statement> shall be used with the attribute EXTERNAL. The sequence of parametres has to be the same as in the corresponding global definition (i.e., the first parametre of the external definition corresponds to the first parametre of the global definition).

9. RESULT <parametre>s may be included in the formal parametre list. They are enclosed in a pair of parentheses that follows the word RESULT.

10. All formal <parametre>s are local and are followed by their <type>.

11. The TIMED DIGITAL path is used to define a TIMED DIGITAL procedure, which can only be invoked using a PERFORM, TIMED DIGITAL statement from within a <do timed digital body> (reference 12.5.7).

6.6.3 PROCEDURE body

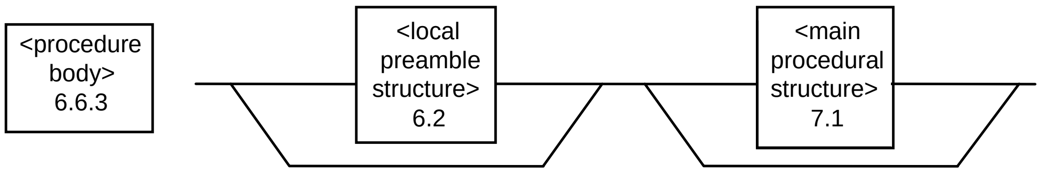
6.6.3.1 Function

The <procedure body> consists of an optional <local preamble> that may include only DECLARE statements. Those DECLARE statements may be utilized to provide local data declarations. The following <main procedural structure> consists of one or more C/ATLAS procedural statements that describe the desired logic.

6.6.3.2 Formal syntax

Reference procedure body

6.6.3.3 Syntax diagram



6.6.3.4 Rules

1. Statement numbers appearing within a <procedure body> may only be referenced from within the bounds of that procedure.

2. Within a single <atlas program structure> or <atlas module structure>, a previously defined procedure may be invoked in a subsequent procedure by including a PERFORM statement within the <procedure body>. Recursion by the use of procedures in different <atlas module structure>s or the <atlas program structure> is not permitted (i.e., a C/ATLAS procedure can never be nested within itself).

3. Procedure bodies may not contain entry points (E-Flags).

4. When the <define procedure statement> is used with the attribute EXTERNAL, the <procedure body> is empty.

5. When the <define procedure statement> is used with the attribute TIMED DIGITAL, the <main procedural structure> will be restricted to the statements defined in the <do timed digital body> (reference 12.5.7).

6.6.4 LEAVE <procedure> statement

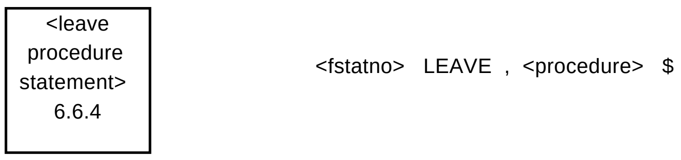
6.6.4.1 Function

Causes execution of a <procedure> to terminate prior to normal sequential execution of the END <procedure> statement.

6.6.4.2 Formal syntax

Reference leave procedure statement

6.6.4.3 Syntax diagram



6.6.4.4 Rules

1. When a LEAVE <procedure> statement is executed, control is transferred to the END <procedure> statement.

2. LEAVE <procedure> statements shall be located within the applicable <define procedure structure>.

3. Any structures in the procedure containing the LEAVE <procedure> statement are also exited upon execution of the LEAVE <procedure> statement.

6.6.5 END PROCEDURE statement

6.6.5.1 Function

To delimit a procedure definition.

6.6.5.2 Formal syntax

Reference end procedure statement

6.6.5.3 Syntax diagram

```
<end
procedure
statement>      <fstatno>   END  ,  <procedure>    $
6.6.5
```

6.6.5.4 Rules

1. The procedure <label> following the END verb shall be identical to that appearing within the <define procedure statement>.

2. Following the execution of the <end procedure statement>, control returns to the first statement following the PERFORM that invoked the procedure.

6.7 REQUIRE statement

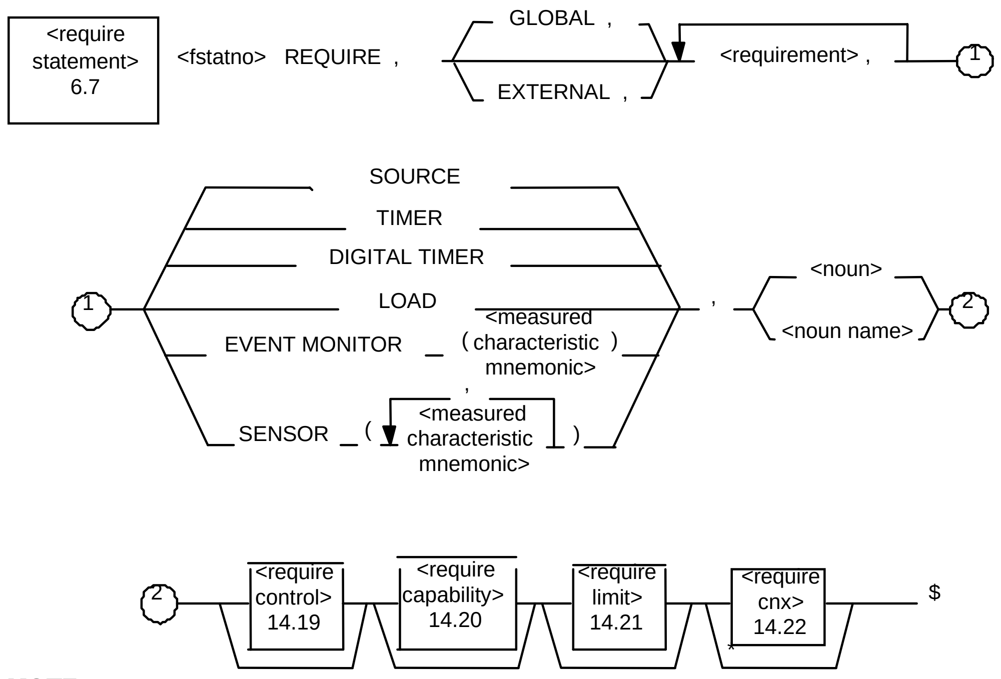
6.7.1 Function

The REQUIRE statement allows a test procedure writer to establish and label a Unit Under Test oriented summary of the characteristics needed by a test resource to implement the <complete atlas test program structure>. The test resources thus established are termed "virtual resources" since they refer to Unit Under Test requirements and not the "real resources" in the hardware used for implementation. The implementation of the <complete atlas test program structure> requires a checking and equating of the virtual resources to the real resources available for implementation. The <requirement> label serves as a linking mechanism that may be utilized in this process.

6.7.2 Formal syntax

Reference require statement

6.7.3 Syntax diagram



NOTE

* This bypass shall be used with the nouns that do not require a CNX field, e.g., TIME INTERVAL.

6.7.4 Rules

6.7.4.1 Rules applicable to all REQUIRE statements

1. A <require statement> may be written only in a <program preamble structure> or a <module preamble structure> immediately following

a) A <begin atlas program statement>, all <include statement>s, <begin atlas module statement>, or an <extend statement>

b) Another <require statement>

2. Each <require statement> shall be a complete summary of the test requirements required for all of the statements that reference the requirement.

3. Each <require statement> can only establish the test requirements for the C/ATLAS noun named.

4. Test requirements that occur simultaneously, but which cannot be fulfilled from a single

virtual resource, shall reference a separate <requirement> label.

5. The test requirement summary shall utilize only modifiers that are valid for the C/ATLAS noun.

6. The same modifier may be utilized only once within a given <require control>, <require capability>, or <require limit> field, but may be used in more than one of these fields.

7. The <require control> field specifies the value or range of values for those test characteristics that shall be controlled by the test resource to execute the <complete atlas test program structure>.

8. The <require limit> field specifies the value or range of values for those test characteristics that shall be limited by some discrete action of the resource to execute the <complete atlas test program structure>.

9. The <require capability> field specifies the required parametre range for those test characteristics that relate to the capacity of the test resource to meet a test requirement, but which are not required to be either controlled or limited to execute the <complete atlas test program structure>.

10. The <require cnx> field lists the pin descriptors associated with the connection points of the resource associated with the <requirement>.

11. A <requirement> label shall be written in the <noun field> following the C/ATLAS word USING in all ATLAS statements referencing the associated test requirement.

12. No <require statement> can include the C/ATLAS word USING.

13. The statement REQUIRE, GLOBAL may only appear in the preamble of an <atlas program structure>. The statement REQUIRE, EXTERNAL is the only statement form that can appear in the preamble of an <atlas module structure>. Each <requirement> label in a <require statement> that contains the word EXTERNAL shall also exist in a <require statement> containing the word GLOBAL.

14. If a <requirement> occurs in the USING subfield of any statement in an <atlas program structure>, the <program preamble structure> shall contain a <require statement> whose characteristics encompass the characteristics of that statement.

15. If a <requirement> occurs in the USING subfield of any statement in an <atlas module structure>, the <module preamble structure> shall contain a REQUIRE, EXTERNAL statement whose characteristics encompass the characteristics of that statement. Its characteristics shall be encompassed by the characteristics in a REQUIRE, GLOBAL statement in the <program preamble structure> that has the same <requirement> label.

6.7.4.2 Rules applicable to SOURCE and LOAD resource types

1. The <require control> field specifies those test characteristics that shall be established or directly controlled by the test resource.

2. The <require capability> field specifies those intrinsic test characteristics that are not required to be either directly controlled or limited by the test resource.

3. The optional subfield beginning with the word "BY" is not used in the <require capability> field for SOURCE or LOAD resource types.

4. The <require limit> field specifies those test characteristics that shall be constrained (limited) by a discrete action of the test resource to provide a valid test or to safeguard the UUT from malfunction of either itself or the test resource.

5. The <require cnx> field specifies the pin descriptors for the connection points to the test resource.

6.7.4.3 Rules applicable to SENSOR and EVENT MONITOR resource types

1. The <require control> field specifies those test characteristics that shall be directly controlled by the test resource.

2. The <require capability> field specifies characteristics to be evaluated by the sensor function and any other intrinsic characteristics that are not required to be either controlled or limited by the test resource.

3. The <require limit> field specifies those test characteristics that shall be constrained (limited) by a discrete action of the test resource to provide a valid test or to safeguard the UUT from malfunction of either itself or the test resource.

4. The <require cnx> field specifies the pin descriptors for the connection points to the test resource.

5. A <measured characteristic mnemonic> subfield is included for SENSOR type resources to specify the characteristic(s) to be evaluated by the sensor function.

6. An appropriate <mnemonic> shall be included in the <require capability> field of SENSOR type resources to specify ranging information of the resource.

7. The accuracy of a SENSOR type resource, where applicable, is written in the ERRLMT subfield of the <require capability> field.

8. A <measured characteristic mnemonic> is included for EVENT MONITOR type resources to specify the characteristics of the signal to be monitored by that resource.

6.7.5 Notes and examples

6.7.5.1 SOURCE examples

A unit being tested non-concurrently requires a DC source having values of 1 V (volt) at 10 A (amperes) max, 5 V at 2 A max, and 9 V at 0.1 A max. It is also required that the current not be allowed to exceed 12 A at any time.

```
000100 REQUIRE, GLOBAL, 'DC SUPPLY', SOURCE, DC SIGNAL,
           CONTROL,
               VOLTAGE RANGE 1V TO 9V BY 4V,
           CAPABILITY,
               CURRENT MAX 10A,
           LIMIT,
               CURRENT MAX 12A,
           CNX HI LO $
```

The unit being tested requires a ramp signal in the frequency range of 75-150 Hz (hertz) with a resolution of 5 Hz and an accuracy of 2 Hz. The amplitude of the signal shall be continuously adjustable over the range of 7.3-7.7 V peak-to-peak.

```
 000200 REQUIRE, EXTERNAL, 'RAMP', SOURCE, RAMP SIGNAL,
           CONTROL,
               FREQ RANGE 75HZ TO 150HZ BY 5HZ ERRLMT +-2HZ,
               VOLTAGE-PP RANGE 7.3V TO 7.7V CONTINUOUS,
           CNX HI LO $
```

The use of this requirement in the procedure of the program could appear as follows:

```
 050000 APPLY, RAMP SIGNAL USING 'RAMP',
           FREQ 105HZ ERRLMT +-2HZ,
           VOLTAGE-PP 7.5V,
           CNX HI J1-7 LO J3-5 $
```

The unit being tested requires a pulsed DC input with a PRF of either 2000 Hz or 6000 Hz (never both) depending on where it is used. The rise time will be specified in the range of 50-55 µs in 1 µs steps. The maximum peak voltage shall not exceed 7.5 V peak.

```
 000300 REQUIRE, GLOBAL, 'DC PULSE', SOURCE, PULSED DC,
           CONTROL,
               PRF 2000HZ, 6000HZ,
               RISE-TIME RANGE 50 USEC TO 55 USEC BY 1 USEC,
           LIMIT,
               VOLTAGE-P RANGE -7.5V TO 7.5V,
           CNX HI LO $
```

6.7.5.2 SENSOR examples

The procedure for the unit being tested requires a sensor that can measure a DC voltage in the range of 15-25 volts with an accuracy of 100 mV.

```
 000400 REQUIRE, EXTERNAL, 'DMM',SENSOR (VOLTAGE), DC SIGNAL,
           CAPABILITY,
                VOLTAGE RANGE 15V TO 25V ERRLMT +-0.1V,
           CNX HI LO $
```

The use of this requirement in the procedural portion of the program could appear as follows:

```
 055000 MEASURE,  (VOLTAGE ERRLMT +-0.1V INTO 'RESULT'),
           DC SIGNAL USING 'DMM',
           VOLTAGE RANGE 15V TO 20V,
           CNX HI J5-9 LO J1-E $
```

The procedure for the unit being tested requires a sensor that can measure the frequency of an AC signal in the voltage range of 4-7 V AC, with a maximum DC level of 75 V. The frequency will be in range of 0-1000 Hz and needs to be measured with an accuracy of 2 Hz.

```
000500 REQUIRE, GLOBAL, 'COUNTER', SENSOR (FREQ), AC SIGNAL,
          CAPABILITY,
               FREQ RANGE 0HZ TO 1000HZ ERRLMT +-2HZ,
               VOLTAGE RANGE 4V TO 7V,
               DC-OFFSET RANGE 0V TO 75V,
          CNX HI LO $
```

6.7.5.3 TIMER example

```
001000 REQUIRE, EXTERNAL, 'T1', TIMER, TIME INTERVAL,
          CONTROL,
               TIME RANGE 0 SEC TO 1000 SEC
               BY 1 SEC ERRLMT +- 0.1 SEC $
```

6.8 INCLUDE statement

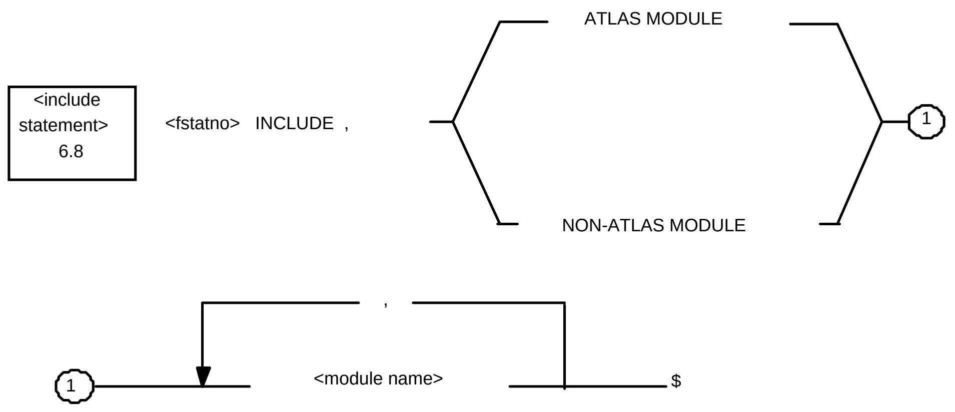
6.8.1 Function

The <include statement> is used in the <program preamble structure> to reference the <module name> of the modules that are part of the <complete atlas test program structure>. Thus, a complete summary of modules used in a <complete atlas test program structure> is established in the <program preamble structure>.

6.8.2 Formal syntax

Reference include-statement

6.8.3 Syntax diagram



6.8.4 Rules

1. An <include statement> can only be written in the <program preamble structure>.

2. Any module that is part of a <complete atlas test program structure> shall be referenced by an INCLUDE statement in the associated <program preamble structure>.

6.9 IDENTIFY statements

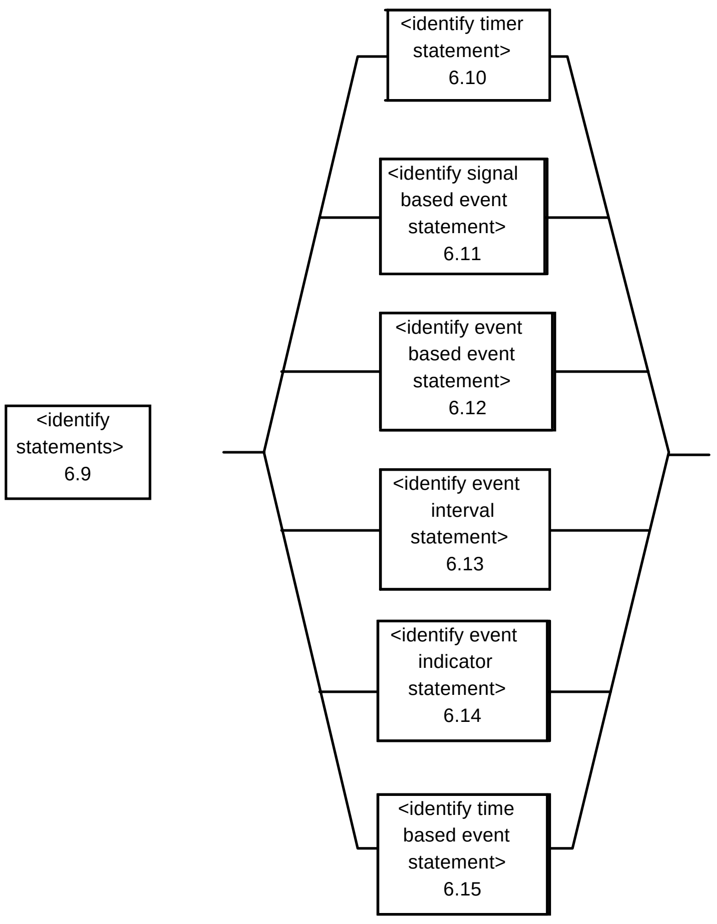
6.9.1 Function

An <identify statements> labels and describes an <event>, <event interval>, <event indicator>, <digital timer>, or <timer> that is used for the purpose of allowing the establishment and implementation of signal and test action timing relationships.

6.9.2 Formal syntax

Reference identify-statements

6.9.3 Syntax diagram


<identify statements> 6.9
<identify timer statement> 6.10
<identify signal based event statement> 6.11
<identify event based event statement> 6.12
<identify event interval statement> 6.13
<identify event indicator statement> 6.14
<identify time based event statement> 6.15

6.10 IDENTIFY TIMER statement

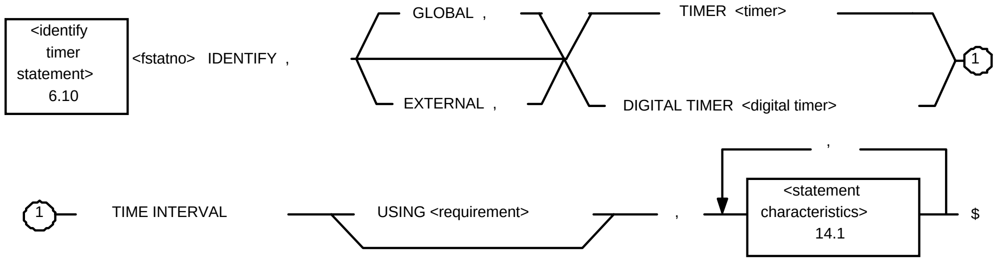
6.10.1 Function

An <identify timer statement> identifies a <timer> or <digital timer>. A <timer> is used for the timing of critical actions. A <digital timer> is used for critical actions within a <do timed digital structure> only. A <timer> or <digital timer> is a constantly running device that allows the definition of a time scale on which critical actions may be precisely and unambiguously positioned.

6.10.2 Formal syntax

Reference identify-timer-statement

6.10.3 Syntax diagram


<identify timer statement> 6.10
<fstatno> IDENTIFY ,
GLOBAL ,
EXTERNAL ,
TIMER <timer>
DIGITAL TIMER <digital timer>
1
1
TIME INTERVAL
USING <requirement>
,
,
<statement characteristics> 14.1
$

6.10.4 Rules

1. A <timer> or <digital timer> starts counting time from zero as soon as it is reset. It has no "idle state".

2. The statement IDENTIFY, GLOBAL may only appear in the preamble of an <atlas program structure>. The statement IDENTIFY, EXTERNAL may only appear in the preamble of an <atlas module structure>. Each label defined by an IDENTIFY, EXTERNAL statement shall also be defined by an IDENTIFY, GLOBAL statement which shall encompass all the information contained in the IDENTIFY, EXTERNAL statement.

3. The allowable <statement characteristics> for <timer> or <digital timer> are defined in clause 16 under TIME INTERVAL.

6.11 IDENTIFY SIGNAL BASED EVENT statement

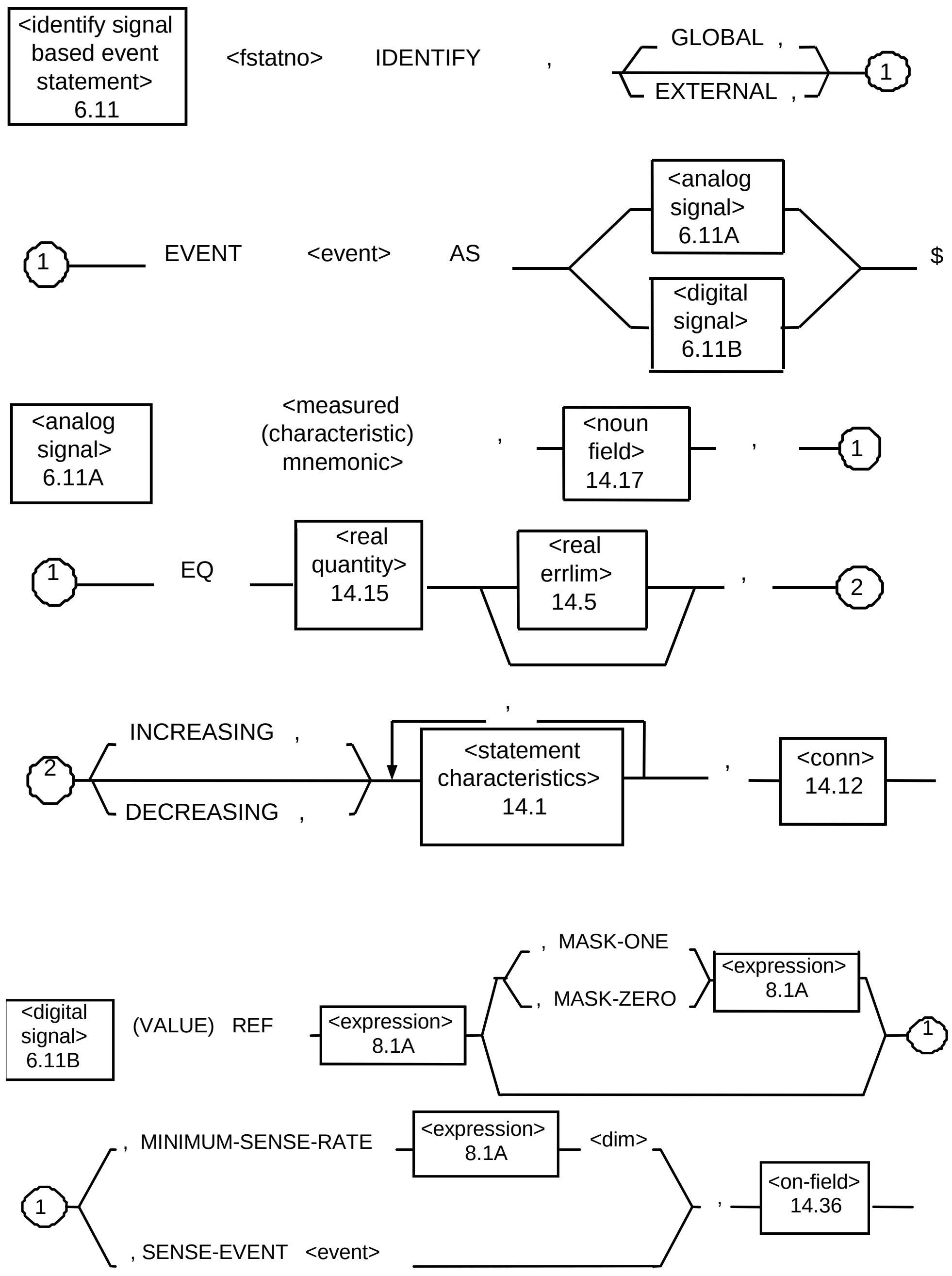
6.11.1 Function

An <identify signal based event statement> describes instances in time that are based on signal conditions occurring at the UUT interface. An event will be generated each time the stated signal conditions are satisfied.

6.11.2 Formal syntax

Reference identify-signal-based-event-statement

6.11.3 Syntax diagram



6.11.4 Rules

1. <event>s are described in terms of certain basic UUT signal characteristics. An event will be generated each time a signal achieves the specified condition. In order for a new <event> to be generated, the signal shall deviate from the specified condition. The optional <real errlim> field indicates the accuracy required of the event monitoring resource.

2. If the INCREASING branch is selected, an <event> will be generated only if the value of the <measured characteristic mnemonic> is increasing when it passes through the specified trigger point. If the DECREASING branch is selected, an <event> will be generated only if the value of the <measured characteristic mnemonic> is decreasing when it passes through the trigger point. If the bypass around DECREASING and INCREASING is selected, an event will be generated each time the <measured characteristic mnemonic> passes through the trigger point regardless of the direction. For the purpose of providing variable control of slope, the INCREASING/DECREASING option may be bypassed and a RANGE field used in the <statement characteristics> to identify the desired slope as well as total signal range. When a RANGE field is used in this manner, the rule above for INCREASING applies when the algebraic value of the final quantity is greater than the initial quantity and the rule for DECREASING applies for the opposite case.

3. VALUE as a <measured characteristic mnemonic> has a different effect for each of the two basic configurations that data and control words may assume.

a) Serial data or control will generate an <event> on the last bit of the word if the VALUE of that word is equal to the specified VALUE.

b) Parallel data or control will generate an <event> if the VALUE of the specified word is equal to the specified VALUE.

4. An IDENTIFY, EVENT based on sensing the <measured characteristic mnemonic> of a signal at the UUT interface requires a resource called an EVENT MONITOR. EVENT MONITOR characteristics are defined by an IDENTIFY EVENT statement.

5. The configuration that contains the event digital sensor definition shall be enabled prior to the explicit or implicit enabling of the appropriate EVENT MONITOR.

6. The optional MINIMUM-SENSE-RATE field, which may occur in digital <event> identification, specifies the minimum rate at which the EVENT MONITOR shall interrogate the <digital sensor> that is explicitly referenced in the <on field>. This is to determine whether it has yet achieved the required pattern as specified in the REF field after any necessary processing arising from the optional MASK field. The operation of REF and MASK subfields is described in the rules for the <prove statement>.

7. Similarly the optional SENSE-EVENT field, which may occur in digital <event> identifications, specifies an <event> that identifies the intervals at which the EVENT MONITOR shall interrogate the digital sensor that is explicitly referenced in the <on field>.

8. The <sync subfield> path through the <statement characteristics> subfield is not allowed for the <identify signal based event statement>.

9. If variables are used for any of the quantities or connections in an <identify statement>, values shall be assigned to these variables prior to the execution of the associated ENABLE EVENT statement. When the enable statement is executed, the current values of these variables are used to complete the <event> specification. Succeeding changes of the variable do not affect the enabled event.

10. The statement IDENTIFY, GLOBAL may only appear in the preamble of an <atlas program structure>. The statement IDENTIFY, EXTERNAL may only appear in the preamble of an <atlas module structure>. Each label defined by an IDENTIFY, EXTERNAL statement shall also be defined by an IDENTIFY, GLOBAL statement in which the requirements of the IDENTIFY, GLOBAL statement encompass those of the IDENTIFY, EXTERNAL statement.

6.12 IDENTIFY EVENT BASED EVENT statement

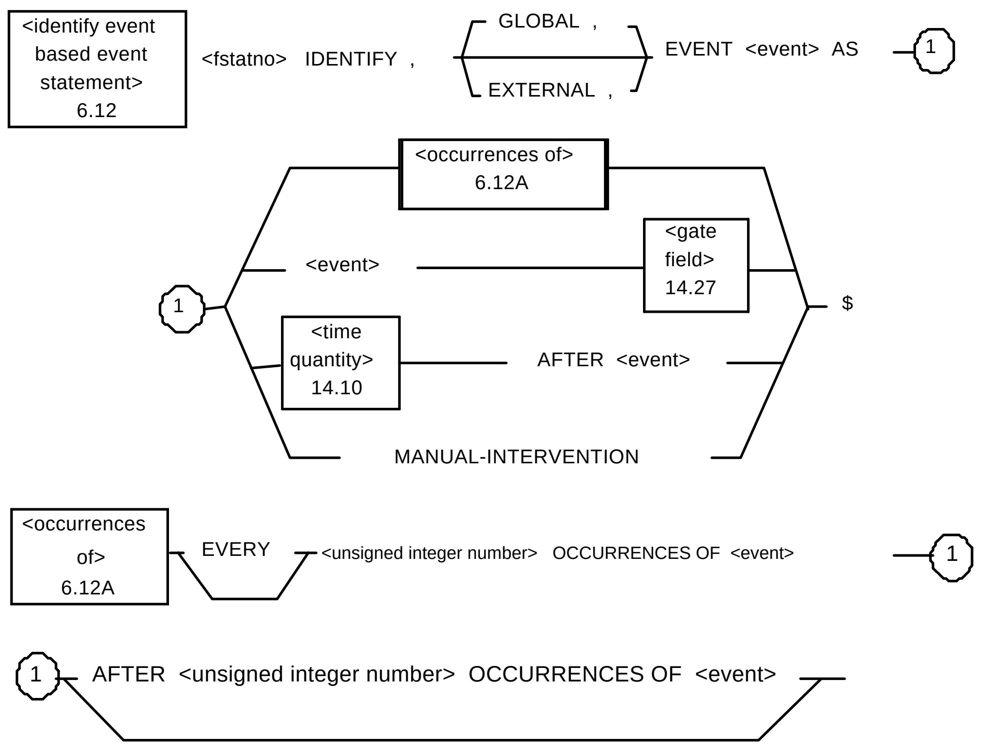
6.12.1 Function

The <identify event based event statement> serves to identify instances in time that are determined by other instances in time.

6.12.2 Formal syntax

Reference identify-event-based-event-statement

6.12.3 Syntax diagram



6.12.4 Rules

1. <event>s may be described in terms of other <event>s as follows:

a) EVERY n OCCURRENCES

   An <event> will be generated every time another referenced <event> occurs n times.

b) EVERY n OCCURRENCES AFTER m OCCURRENCES

   After m occurrences of one specified <event>, an <event> will be generated every n occurrences of another specified <event>. It is allowable to specify the same <event> label for the EVERY n OCCURRENCES OF <event> and for the AFTER m OCCURRENCES OF <event>, but they are not required to be the same.

c) n OCCURRENCES OF

   An <event> will be generated on the nth occurrence of a specified <event>. After this generation of the <event>, no further <event>s will be generated unless the <event> is disabled, re-enabled, and then n more referenced <event>s occur.

d) n OCCURRENCES AFTER m OCCURRENCES

   An <event> will be generated on the nth occurrence of a specified <event> after m occurrences of another specified <event>. After this <event> generation, no further <event> will be generated unless first disabled and re-enabled.

e) GATED BY

   An <event> will be generated each time a referenced <event> occurs during a referenced <event interval>.

f) AFTER

   An <event> will be generated after the specified <event>.

g) MANUAL-INTERVENTION

   An <event> will be generated each time a manual intervention is performed. A manual intervention is a system dependent operator action.

2. Items a thru d of rule 1) will require the counting of other referenced <event>s and are affected by the enabling of the identified <event>. Each time that the identified <event> is enabled, the counting mechanism(s) for that <event> will be set to a zero count.

3. <event>s being identified shall not reference themselves in their definition.

4. If variables are used for any of the quantities in an <identify statement>, values shall be assigned to these variables prior to the execution of the associated ENABLE, EVENT statement. When this ENABLE statement is executed, the current values of these variables are used to complete the <event> specification. Succeeding changes of these variables do not affect the enabled <event>.

5. The statement IDENTIFY, GLOBAL may only appear in the preamble of an <atlas program structure>. The statement IDENTIFY, EXTERNAL may only appear in the preamble of an <atlas module structure>. Each label defined by an IDENTIFY, EXTERNAL statement shall also be defined by an IDENTIFY, GLOBAL statement in which the requirements of the IDENTIFY, GLOBAL statement encompass those of the IDENTIFY, EXTERNAL statement.

6.13 IDENTIFY EVENT INTERVAL statement

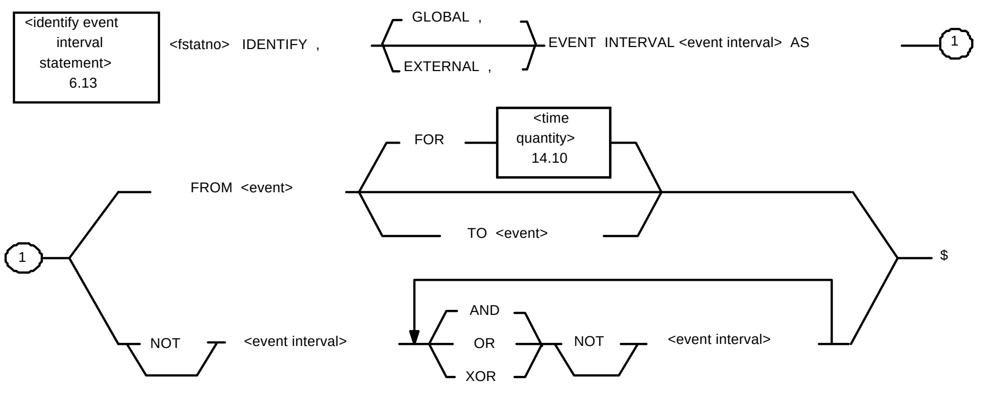
6.13.1 Function

The <identify event interval statement> serves to identify a period of time.

6.13.2 Formal syntax

Reference identify-event-interval-statement

6.13.3 Syntax diagram



6.13.4 Rules

1. <event interval>s identify particular regions of time. The regions may be specified in the following manners:

a) FROM <event> FOR (a specified time)

An <event interval> may be defined to begin at the occurrence of a specified <event> and to end a specified time period after the occurrence of that <event>. If the referenced <event> reoccurs during this time period, the period of the <event interval> will be extended by the specified time period from the point of re-occurrence.

b) FROM <event> TO <event>

An <event interval> may be defined as the time period between the occurrence of one referenced <event> and the next occurrence of another referenced <event>. A different <event> shall be specified to start and end the <event interval>.

c) FROM <event>

An <event interval> may be defined as the time period starting at the first occurrence of a specified <event> and lasting for the duration of the <complete atlas test program structure> execution.

d) Logical combinations of <event interval>s

An <event interval> may be identified in terms of the logical combination of other referenced <event interval>s. The <event interval> being identified may not reference itself in its specification. Operator precedence is in accordance with the rules of 8.1.4.2 and table 8-1.

2. If variables are used for any quantities in an <identify statement>, values shall be assigned to these variables prior to the execution of the associated ENABLE, EVENT statement. When this ENABLE statement is executed, the current values of these variables are used to complete the <event> specification. Succeeding changes of the variable values do not affect the enabled <event>.

3. The statement IDENTIFY, GLOBAL may only appear in the preamble of an <atlas

program structure>. The statement IDENTIFY, EXTERNAL may only appear in the preamble of an <atlas module structure>. Each label defined by an IDENTIFY, EXTERNAL statement shall also be defined by an IDENTIFY, GLOBAL statement in which the requirements of the IDENTIFY, GLOBAL statement encompass those of the IDENTIFY, EXTERNAL statement.

6.14 IDENTIFY EVENT INDICATOR statement

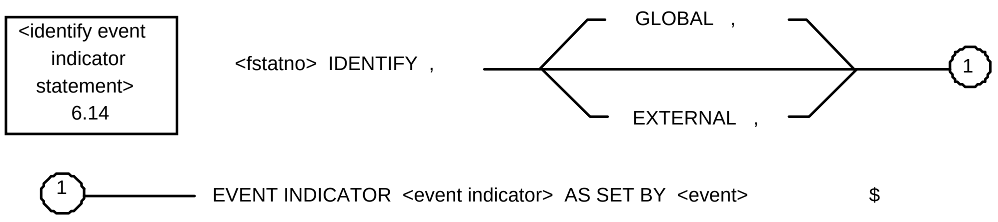
6.14.1 Function

An <identify event indicator statement> identifies that a particular BOOLEAN variable, that had previously been declared, is an <event indicator>.

6.14.2 Formal syntax

Reference identify-event-indicator-statement

6.14.3 Syntax diagram


<identify event indicator statement> 6.14
<fstatno> IDENTIFY ,
GLOBAL ,
EXTERNAL ,
1
1
EVENT INDICATOR <event indicator> AS SET BY <event>
$

6.14.4 Rules

1. <event indicator>s are BOOLEAN variables that are used to indicate the occurrence of an <event>. They have the property of normal BOOLEAN variables with additional features that are automatically set when the <event> with which they are associated occurs while the <event> is ENABLEd.

2. Once an <event indicator> is set to the value TRUE by the occurrence of a reference <event>, it will remain TRUE until specifically reset in a CALCULATE statement.

3. Before an <event indicator> is identified, it shall have been declared as a BOOLEAN variable.

4. The statement IDENTIFY, GLOBAL may only appear in the preamble of an <atlas program structure>. The statement IDENTIFY, EXTERNAL may only appear in the preamble of an <atlas module structure>. Each label defined by an IDENTIFY, EXTERNAL statement shall also be defined by an IDENTIFY, GLOBAL statement in which the requirements of the IDENTIFY, GLOBAL statement encompass those of the IDENTIFY, EXTERNAL statement.

6.15 IDENTIFY TIME BASED EVENT statement

6.15.1 Function

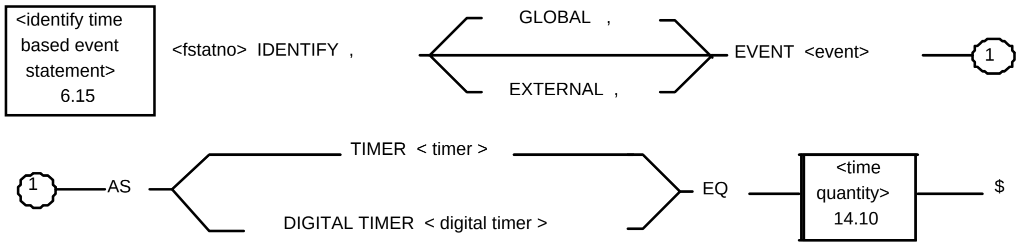
The <identify time based event statement> serves to identify one or more points in time that are directly derived from a <timer> or <digital timer>. An <event> will be generated each time the <timer> or <digital timer> matches the specified time value and the <event> is enabled.

The accuracy and resolution of the timer are defined in the <identify timer statement>.

6.15.2 Formal syntax

Reference identify-time-based-event-statement

6.15.3 Syntax diagram



6.15.4 Rules

1. The statement IDENTIFY, GLOBAL may only appear in the preamble of an <atlas program structure>. The statement IDENTIFY, EXTERNAL may only appear in the preamble of an <atlas module structure>. Each label defined by an IDENTIFY, EXTERNAL statement shall also be defined by an IDENTIFY, GLOBAL statement in which the requirements of the IDENTIFY, GLOBAL statement encompass those of the IDENTIFY, EXTERNAL statement.

2. An <identify time based event statement> deriving an <event> based on one or more points in time requires a resource called an EVENT MONITOR.

6.16 DIGITAL CONFIGURATION definition

6.16.1 <define digital configuration structure>

6.16.1.1 Function

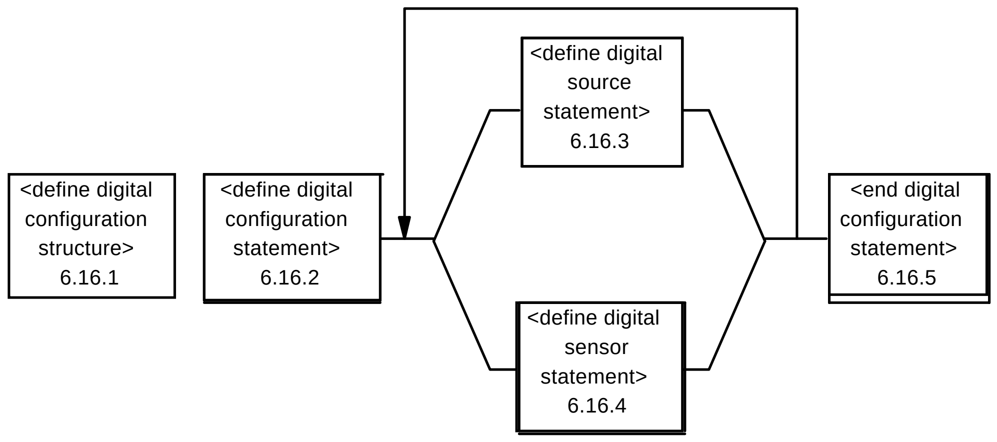
This structure provides a mechanism for defining the relationship between digital data and source and sensor signals.

It defines for the compiler the nature of the simultaneous resource required, but run-time alteration of the values is permitted. The <define digital configuration structure> contains two main statement types: one for source requirements and one for sensor requirements.

6.16.1.2 Formal syntax

Reference define-digital-configuration-structure

6.16.1.3 Syntax diagram


<define digital configuration structure> 6.16.1
<define digital configuration statement> 6.16.2
<define digital source statement> 6.16.3
<define digital sensor statement> 6.16.4
<end digital configuration statement> 6.16.5

6.16.1.4 Rules

1. More than one <define digital configuration structure> may be defined, but only one configuration will be active at any one time.

2. If the same pin occurs in two or more statements in a <define digital configuration structure>, the affected sources and/or sensors shall be either all parallel or all serial.

6.16.2 DEFINE DIGITAL CONFIGURATION statement

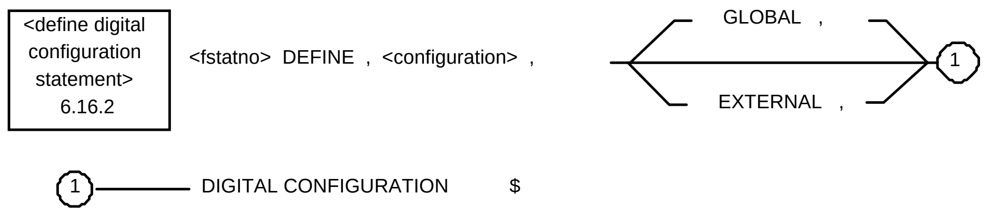
6.16.2.1 Function

This statement names and introduces a <define digital configuration structure>.

6.16.2.2 Formal syntax

Reference define-digital-configuration-statement

6.16.2.3 Syntax diagram


<define digital configuration statement> 6.16.2
<fstatno> DEFINE , <configuration> ,
GLOBAL ,
EXTERNAL ,
1
1
DIGITAL CONFIGURATION $

6.16.2.4 Rules

The statement DEFINE, <configuration>, GLOBAL may only appear in the preamble of an <atlas program structure>. The DEFINE, <configuration>, EXTERNAL may only appear in the preamble of an <atlas module structure>. Each <configuration> label defined by a DEFINE, <configuration>, EXTERNAL statement shall also be defined by a DEFINE, <configuration>, GLOBAL statement in which the requirements of the DEFINE, <configuration>, GLOBAL statement encompass those of the DEFINE, <configuration>, EXTERNAL statement.

6.16.3 DEFINE DIGITAL SOURCE statement

6.16.3.1 Function

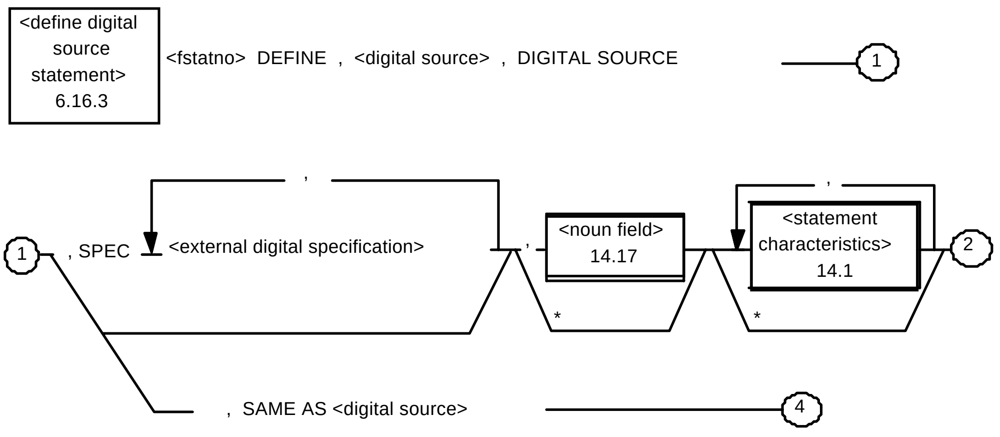
This statement establishes a label <digital source> for a noun, a set of digital source characteristics and a set of connections to be used as a digital source. For each signal connection it allows for the definition of the physical signal corresponding to each logic state. For LEVEL-LOGIC-ONE and LEVEL-LOGIC-ZERO the physical characteristics of signals may be defined. This allows a precise definition of a range of "pulse-classes" such as BIP, AMI, etc.

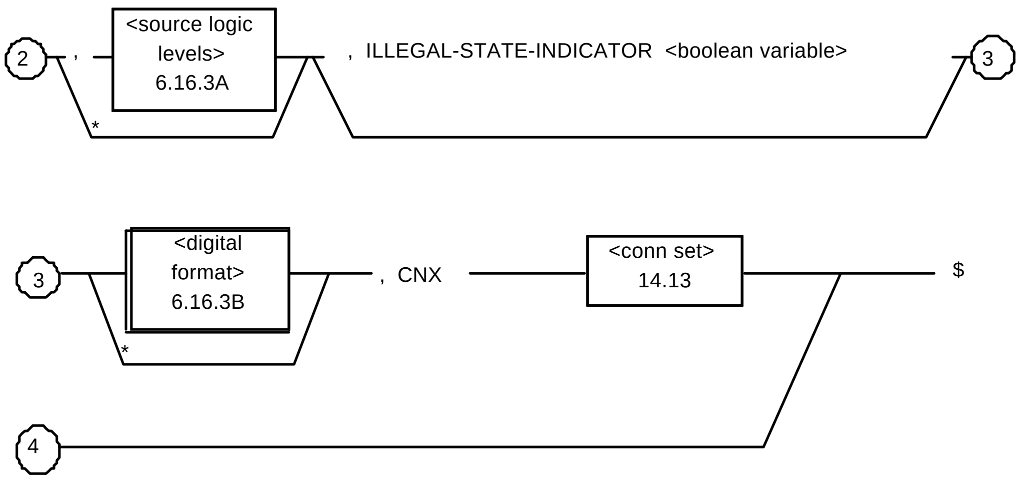
The statement allows secondary as well as primary characteristics of signals to be specified. The primary characteristics, in which the transformation operates, can have temporal qualifications which can be used to synchronize them to other UUT signals or to a test resource time-scale. Serial as well as parallel relationships can be defined.

6.16.3.2 Formal syntax

Reference define-digital-source-statement

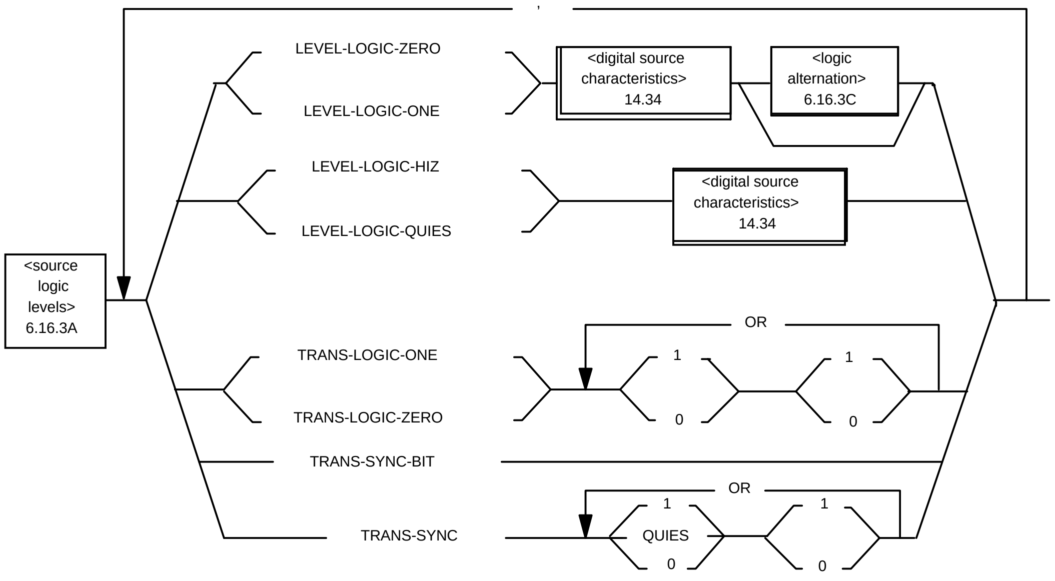
6.16.3.3 Syntax diagram

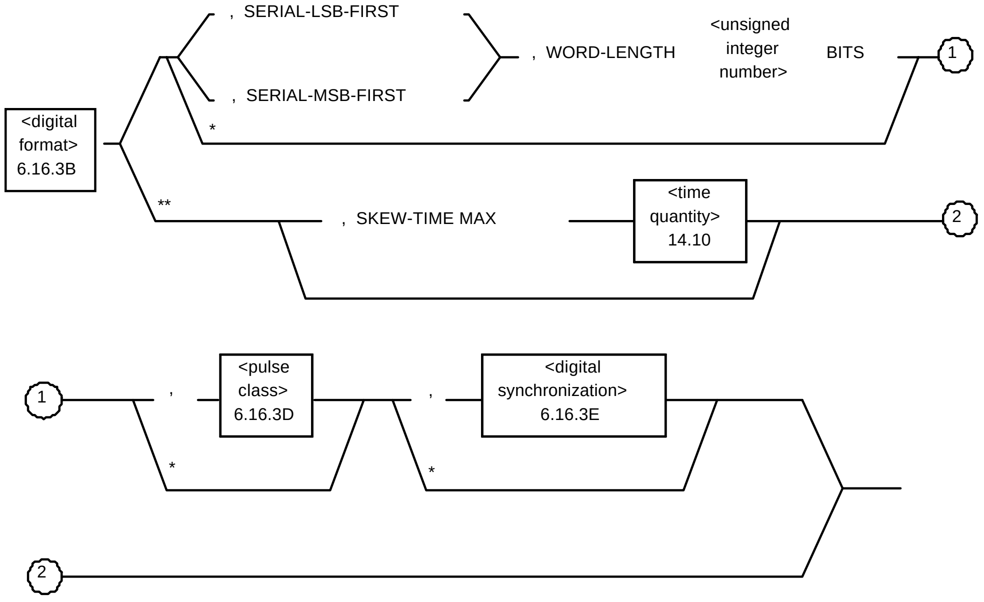

<define digital source statement> 6.16.3
<fstatno> DEFINE , <digital source> , DIGITAL SOURCE
1
,
1
, SPEC
<external digital specification>
,
<noun field> 14.17
*
,
<statement characteristics> 14.1
*
2
, SAME AS <digital source>
4


2
,
<source logic levels> 6.16.3A
*
, ILLEGAL-STATE-INDICATOR <boolean variable>
3
3
<digital format> 6.16.3B
*
, CNX
<conn set> 14.13
$
4

NOTE

* This bypass is used when the relevant information is completely defined in the specification identified by the <external digital specification>.

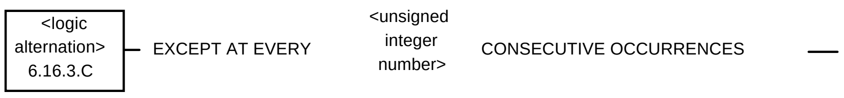


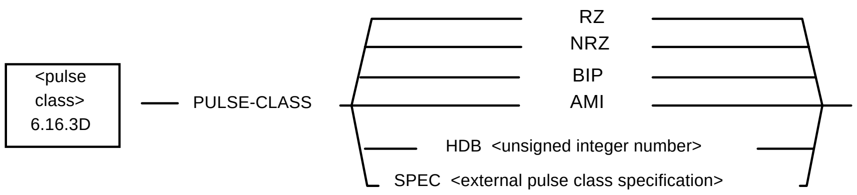


NOTES

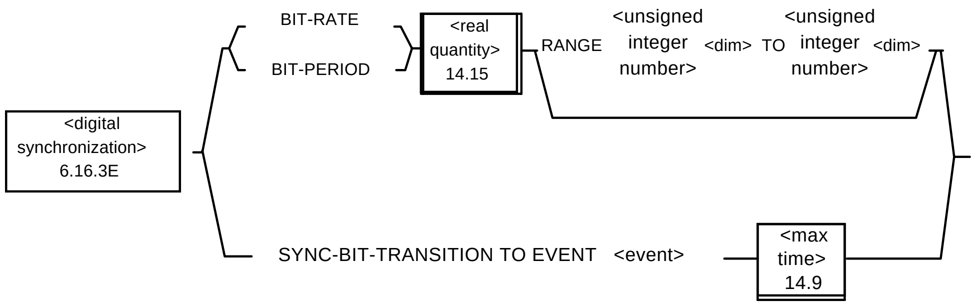
* This bypass is used only if all the information has been defined in the <external digital specification>.

** This path is used only with parallel digital signals.





<pulse class> 6.16.3D
PULSE-CLASS
RZ
NRZ
BIP
AMI
HDB <unsigned integer number>
SPEC <external pulse class specification>



<digital synchronization> 6.16.3E
BIT-RATE
BIT-PERIOD
<real quantity> 14.15
RANGE
<unsigned integer number>
<dim>
TO
<unsigned integer number>
<dim>
SYNC-BIT-TRANSITION TO EVENT <event>
<max time> 14.9

6.16.3.4 Rules for all <define digital source statement>s

1. For parallel sources the order in which signal connections are written after CNX is significant. The first connection corresponds to the most significant bit. If the <digital source> is used in a <stimulate statement> which includes a CNX in its <on field>, the connection list in that CNX supersedes the pin list in the <define digital source statement>.

2. The <external digital specification> option is used when the signal characteristics of a digital word are defined in an existing specification. The label, <external digital specification>, shall identify the specification for the digital source's signal characteristics. If the information identified by <external digital specification> is either incomplete or inconsistent with the requirement, then information shall be supplied explicitly by the appropriate fields in the <define digital source statement> which then take precedence over the information in the <external digital specification>.

Any referenced external specification is a part of the test documentation and shall be available to all users of the C/ATLAS test program.

3. Explicit signal characteristics of LEVEL-LOGIC-ONE and LEVEL-LOGIC-ZERO may be defined in terms of levels, which are steady states that hold true throughout the bit period and for all instances of a given state. If the <pin descriptor>s TRUE and COMPL are employed the value stated is that of the TRUE connection with respect to the COMPL connection.

Provision is made for the exceptional insertion of a different level. This technique is used to break up long chains of a given state in some digital transmission systems such as HDB.

4. When the PULSE-CLASS field is used, it shall be consistent with the information included in the digital logic levels and/or the <external digital specification>. The SPEC subfield of the PULSE- CLASS field is used to identify additional pulse classes.

5. LEVEL-LOGIC-QUIES, when specified, refers to the state that exists between bits or between words. For bus systems, this is the neutral state when no data is present.

6. A named <digital source> can be individually referenced in a <stimulate statement>.

7. The term LEVEL-LOGIC-HIZ applies only to electrical systems where LEVEL-LOGIC-ONE and LEVEL-LOGIC-ZERO are defined in terms of voltages. LEVEL-LOGIC-HIZ is defined in terms of the maximum current flow when the defined LEVEL-LOGIC-ONE and LEVEL-LOGIC-ZERO levels are impressed on test system outputs. If only one LEVEL-LOGIC-HIZ is specified the allowed leakage current applies to both of the impressed logic states. If it is required to specify different leakages for the two impressed logic states, then the LEVEL-LOGIC-HIZ specification shall be repeated and the position, which shall be immediately following the LEVEL-LOGIC-ONE or LEVEL-LOGIC-ZERO field, defines to which state the leakage relates.

8. The initial state that will exist prior to encountering the first STIMULATE statement will be LEVEL-LOGIC-HIZ, if defined, else LEVEL-LOGIC-QUIES, if defined, else LEVEL-LOGIC- ZERO.

9. When an ILLEGAL-STATE-INDICATOR is used, the associated <boolean variable> will be set TRUE if the <digital source> which generates the signal is out of specification as defined for the generated signal level by the <digital source characteristic>s. That is, it is set TRUE if the signal does not satisfy the programmed LEVEL-LOGIC-ONE, LEVEL-LOGIC-ZERO, LEVEL-LOGIC- QUIES or LEVEL-LOGIC-HIZ. When TRANS is used the ILLEGAL-STATE-INDICATOR is set TRUE if the signal is inside neither of the ranges that are specified by the signal characteristics associated with the -ZERO and -ONE suffixes. It is set FALSE only by a <calculate statement> and an <input statement>.

10. The BIT-RATE/BIT-PERIOD sub-field, when present, specifies the speed of bit transmissions for SERIAL logic signals.

11. For signals that utilize an independent clock source, the synchronizing event shall be identified as an <event>. The beginning of every bit period of the serial data stream generated by the <digital source> is then synchronized to this <event> via the SYNC BIT-TRANSITION TO EVENT branch.

12. If an <event> is specified for serial data clocking, the <max time> field indicates the maximum time that the digital configuration is to wait, following a STIMULATE start, for the first occurrence of the <event>. If the <event> does not occur within this time limit, a MAX-TIME condition is set and testing resumes without completion of the <stimulate statement>.

13. The TRANS-LOGIC-ONE and TRANS-LOGIC-ZERO fields are used to define transitions between the defined -ONE and -ZERO signal levels that correspond to a digital one and a digital zero respectively. The OR is available because some transmission systems permit a logic one or zero to be represented in two different ways, depending on the start condition. These transitions occur at the mid-point of a data bit duration in Bi-Phase Signals.

14. The TRANS-SYNC field is used to define that there shall be a transition at the start of each word. A number pair following the key-word indicates the transition required prior to the first bit. If the keyword QUIES occurs after TRANS-SYNC, it indicates that the signal is required to be at the quiescent level prior to the initial transition of the word. The second entry indicates the level it acquires at the start of the word transmission. The TRANS-SYNC-BIT field is used to define that there shall be a transition at the start of each bit period.

15. In statements employing the TRANS-LOGIC-ONE and TRANS-LOGIC-ZERO fields indicating that the logical information is carried by signal transitions, the -ONE and -ZERO suffixes can be used after the mnemonic in the <statement characteristics> field to indicate that the associated signal characteristic carries the digital information and assumes the stated value when the signal is a ONE or a ZERO respectively. If one of these suffixes is used, the same mnemonic shall occur in another field with the other suffix.

16. The SKEW-TIME MAX field defines the maximum acceptable skew between the various output ports employed by a parallel digital pattern.

17. SERIAL-MSB-FIRST and SERIAL-LSB-FIRST identify the digital source as being serial and identify respectively that the left-most bit or the right-most bit is to be transmitted first.

18. A statement containing a SAME AS branch introduces a new <digital source>. This

<digital source> has identical characteristics and connections to the <digital source> placed after SAME AS. The <digital source> after SAME AS shall have been previously defined in another <define digital configuration structure> which is local within the C/ATLAS Test Program.

19. The <noun field> specifies the type of analog signal which conveys the digital data. The <source logic levels> specify the different values for the signal characteristic which conveys the digital data associated with the digital data values. The <statement characteristics> specify the other signal characteristics which do not convey data values. It should be noted that, although the signal value of a characteristic is normally invariant, when it is associated with the <source logic level>s, it varies whenever the data value changes.

20. The RANGE option in <digital synchronization> shall be used when <real quantity> includes a <data store>, but not otherwise. It defines the acceptable range of values that the <data store> may contain.

21. Either TRANS-LOGIC-ONE and TRANS-LOGIC-ZERO, or LEVEL-LOGIC-ONE and LEVEL-LOGIC-ZERO shall be specified.

6.16.3.5 Rules applicable to the pulse-class field

1. The PULSE-CLASS is required to be explicitly or implicitly defined for all serial data statements, for both SOURCE and SENSOR.

2. The subfield after PULSE-CLASS identifies the class of pulse coding used to represent the digital data. The following classes are defined:

RZ	Return to zero
NRZ	Non return to zero
BIP	Bi-phase
AMI	Alternate mark inversion
HDBn	High density bipolar where n is an <unsigned integer number>.

3. The keyword SPEC followed by a reference to an <external pulse class specification> may be used when a pulse class that is not one of the standard classes is required. The referenced specification shall fully describe the pulse coding of the transmission.

6.16.3.6 Rules applicable to the <source logic levels>

1. The <mnemonic> chosen shall be applicable to the <noun> specified in the associated digital source field.

2. No source can employ both LEVEL-LOGIC-... and TRANS-LOGIC-... keywords.

3. When used following LEVEL-LOGIC-HIZ to specify the high impedance condition of the test system, the direction of leakage current flow is specified as follows:

a) Current flowing from the test system as a positive current

b) Current flowing into the test system as a negative current

6.16.3.7 Notes and examples

6.16.3.7.1 Notes

1. For the RZ type, the bit period is subdivided into two sub-periods. The binary data is carried in the first sub-period. If the signal is carried by voltage levels, the noun PULSED DC is used as follows:

a) A binary 1, by a pulse of LEVEL-LOGIC-ONE amplitude

b) A binary 0, by a pulse of LEVEL-LOGIC-ZERO amplitude

The second sub-period contains a "no pulse" condition as a LEVEL-LOGIC-ZERO level. Generally, the two sub-periods are of equal duration; this is the default case in C/ATLAS. For non-equal sub-periods, either a PULSE-WIDTH or DUTY-CYCLE modifier is used to define the duration of the first sub-period.

2. The NRZ type corresponds to ARINC 573 and is also known as the TWG/RCC NRZ-Level (or NRZ Change). If the signal is carried by voltage levels, it employs the noun DC SIGNAL and is characterized by the following:

a) A binary 1 and binary 0, respectively, are represented by voltages of amplitude LEVEL-LOGIC-ONE and LEVEL-LOGIC-ZERO.

b) The width of each pulse is the full bit period.

3. The BIP type represents a family of waveforms that are further defined by the TRANSition modifier. These waveforms are characterized by at least one change in voltage level (and for some BIP types, by two changes) for each data bit.

Note: Telemetry Std, 106-71. See PULSE-CLASS figure 6-1.

4. The AMI type of waveform is similar to the NRZ waveform. Code elements may take the values +1, -1, and 0 where each successive binary '1' value is encoded using alternate inversions; i.e., +1, -1, +1, -1, etc. All binary '0's are encoded as '0'. Three signal levels need to be defined in the LEVEL-LOGIC-... fields. Two -ONE levels and one -ZERO level are defined. The -ONE level defined by the first LEVEL-LOGIC-ONE field is assumed by the initial binary '1' value in the data string and then by the third and subsequent alternate logic ones. The level defined by the second LEVEL-LOGIC-ONE field is assumed by the second, fourth, and subsequent alternate logic ones.

Refer to: ITU-T Recommendation G.702: 1988, entry no. 8003.

5. The HDB type of waveform is similar to the AMI waveform, where each successive binary '1' value is encoded using alternate inversions. 'n' is an integer which specifies the maximum number of consecutive binary '0's, encoded as '0', allowed in the data string. Thus all binary '0's apart from those in sequences of greater than 'n' are encoded as '0'. In any sequence of 'n' consecutive binary '0's the ultimate '0' shall be replaced by a +1 or -1 level, the polarity being the same as that of the preceding '1' value in the data string. The only difference from the definition of an AMI waveform is that the EXCEPT AT EVERY sequence is employed to define the maximum permitted number of successive logic zeros to be represented by the LEVEL-LOGIC-ZERO signal.

Refer to: ITU-T Recommendation G.702: 1988, entry no. 8005.

6.16.3.7.2 Examples

1. Figure 6.1 illustrates several typical serial data waveforms, using voltages to carry digital information, including several BIP types. These waveforms are defined as follows:

a) Basic RZ. Described in C/ATLAS using the noun, PULSED DC, as

```
LEVEL-LOGIC-ONE VOLTAGE 1 V,
LEVEL-LOGIC-ZERO VOLTAGE 0 V, ...
PULSE-CLASS RZ,
```

b) Basic NRZ. Referred to as the NRZ code (ARINC 573) or the TWG/RCC NRZ-Level (or NRZ Change code). Described as in (a) but with the noun DC SIGNAL and the "NRZ" designation.

c) Bi-Phase-Level (or Split Phase, Manchester 11 + 180). Each bit period is divided into two sub-periods. The binary data is represented in the TRANS modifier fields as follows:

1) A binary 1 by a sub 10

2) A binary 0 by a sub 01

The illustrated waveform is described in C/ATLAS as

```
VOLTAGE-ONE 1 V, VOLTAGE-ZERO  0 V, ...
TRANS-LOGIC-ONE 10, TRANS-LOGIC-ZERO 01, ...
PULSE-CLASS BIP,
```

d) Bi-Phase-Space. Also called Harvard Bi-Phase (ARINC 573). A transition occurs at the start of each bit period. Each bit is represented as follows:

1) A binary 1, by a second transition at the mid point of the bit period

2) A binary 0, by no second transition

The waveform is described in C/ATLAS as

```
VOLTAGE-ONE 1 V, VOLTAGE-ZERO  0 V, ...
TRANS-LOGIC-ONE 01 OR 10,
TRANS-LOGIC-ZERO 00 OR 11,
TRANS-SYNC 01, .... , PULSE-CLASS BIP,
```

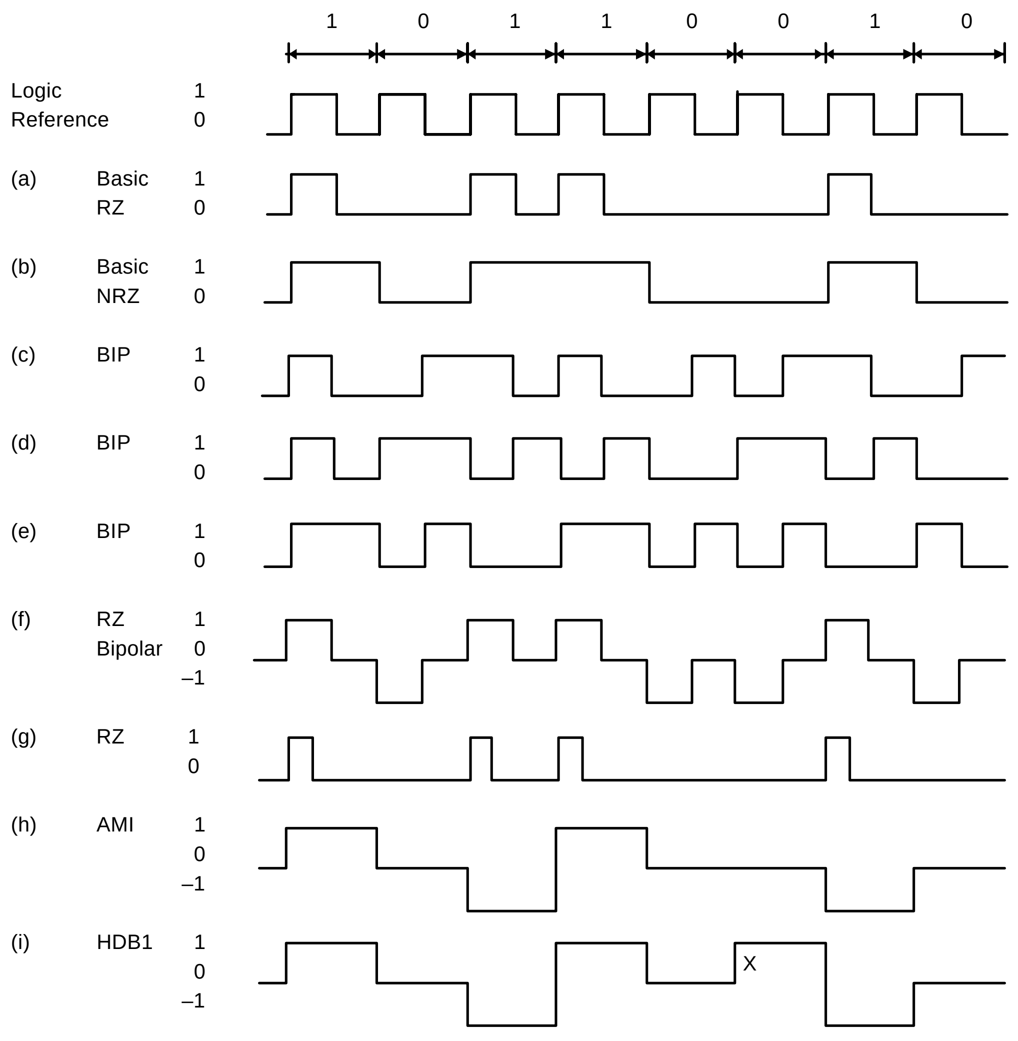


Figure 6-1. Illustration of RZ, NRZ, and BIP pulse types for B '10110010'

e) Bi-Phase-Mark. A transition occurs at the start of each bit period. Each bit is represented as follows:

1) A binary 0 by a second transition at the mid point of the bit period

2) A binary 1 by no second transition

The waveform is described in C/ATLAS as

```
VOLTAGE-ONE 1V, VOLTAGE-ZERO  0V, ...
TRANS-LOGIC-ONE 00 OR 11,
TRANS-LOGIC-ZERO 01 OR 10,
TRANS-SYNC 01, ....
PULSE-CLASS BIP,
```

f) Bi-Polar RZ (ARINC 575). Three voltage levels are defined and each bit period is subdivided into two sub-periods. Each bit is represented during the first sub-period as follows:

1) A binary 1 by the LEVEL-LOGIC-ONE level

2) A binary 0 by the LEVEL-LOGIC-ZERO level

The second sub-period contains the DC-OFFSET level.

```
DC-OFFSET 0 V, ...
LEVEL-LOGIC-ONE VOLTAGE 1 V,
LEVEL-LOGIC-ZERO VOLTAGE -1 V, ...
PULSE-CLASS RZ,
```

g) Non-symmetric RZ. The same as a), the Basic RZ, but with a duty cycle of 25 percent.

The waveform is described in C/ATLAS as

```
DUTY-CYCLE 25PC, ...
LEVEL-LOGIC-ONE VOLTAGE 1 V,
LEVEL-LOGIC-ZERO VOLTAGE 0 V, ...
PULSE-CLASS RZ,
```

h) AMI. Three voltage levels need to be defined.

The waveform is described in C/ATLAS using the noun DC SIGNAL as

```
LEVEL-LOGIC-ONE VOLTAGE 1 V,
LEVEL-LOGIC-ONE VOLTAGE -1 V,
LEVEL-LOGIC-ZERO VOLTAGE 0 V, ...
PULSE-CLASS AMI,
```

i) HDBn. As in AMI, three voltage levels need to be defined. Since n=1, the second consecutive binary '0' value in the data string is replaced by a +1 level [i.e., the polarity is the same as the preceding binary '1' level, designated by 'X' in Figure 6-1 (I)].

The waveform is described in C/ATLAS as

```
LEVEL-LOGIC-ONE VOLTAGE 1 V,
LEVEL-LOGIC-ONE VOLTAGE -1 V,
LEVEL-LOGIC-ZERO VOLTAGE 0 V EXCEPT AT EVERY 2
      CONSECUTIVE OCCURRENCES, ...
PULSE-CLASS HDB2,
```

6.16.4 DEFINE DIGITAL SENSOR statement

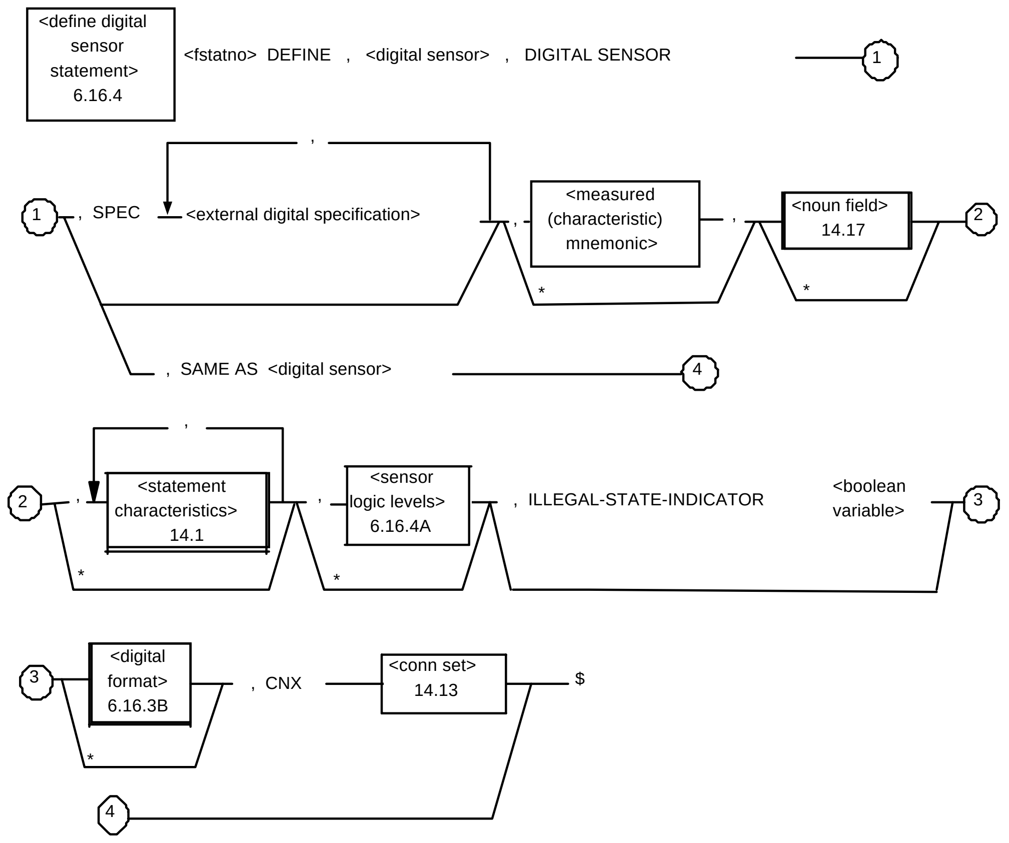
6.16.4.1 Function

This statement establishes a label, <digital sensor>, for a noun associated with a measured characteristic, a set of digital sensor characteristics, and a set of connections to be used as a digital sensor. For each signal connection it allows for the definition of the signal corresponding to each logic state.

6.16.4.2 Formal syntax

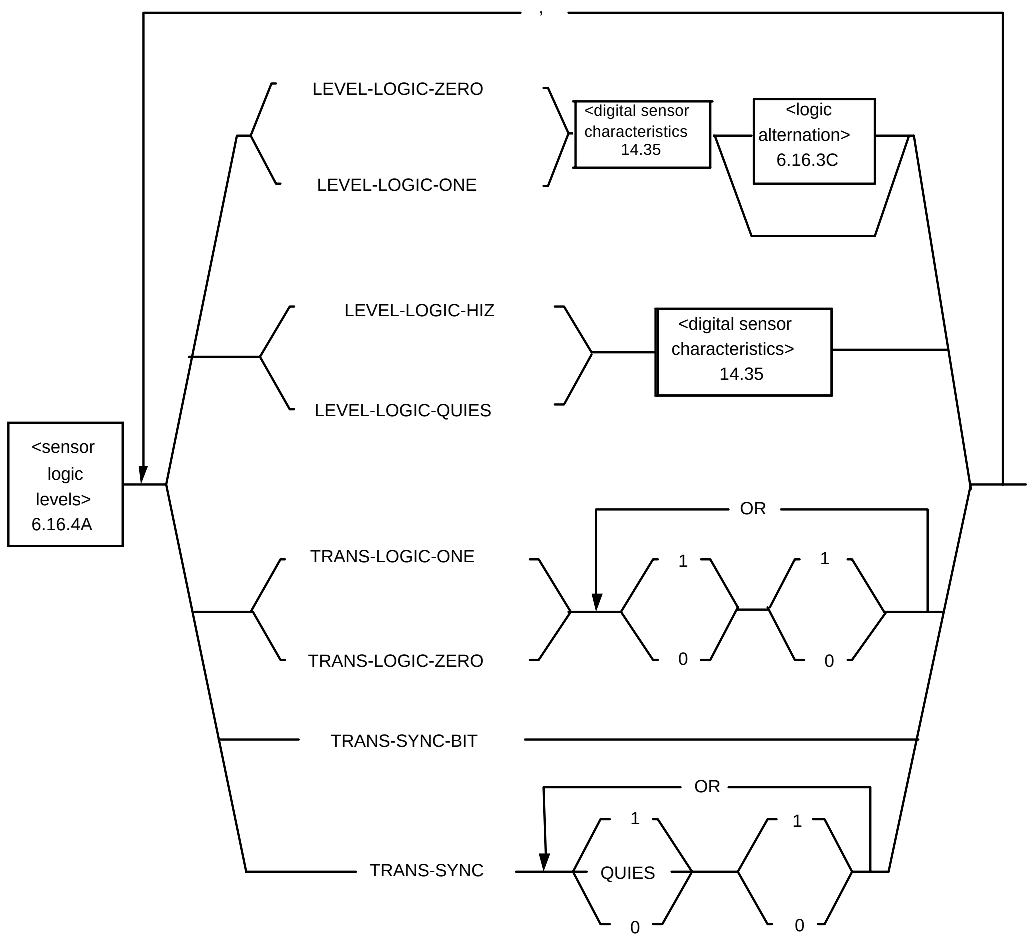
Reference define-digital-sensor-statement

6.16.4.3 Syntax diagram


<define digital sensor statement> 6.16.4
<fstatno> DEFINE , <digital sensor> , DIGITAL SENSOR
1
,
1 , SPEC <external digital specification>
, <measured (characteristic) mnemonic>
, <noun field> 14.17
2
*
*
, SAME AS <digital sensor>
4
,
2 , <statement characteristics> 14.1
, <sensor logic levels> 6.16.4A
, ILLEGAL-STATE-INDICATOR <boolean variable>
3
*
*
3 <digital format> 6.16.3B
, CNX <conn set> 14.13
$
*
4

NOTE

* This bypass is used when the relevant information is completely defined in the specification identified by the <external digital specification>.



6.16.4.4 Rules

1. For parallel sensors, the order in which signal connections are written after CNX is significant. The first connection corresponds to the most significant bit. If the <digital sensor> is used in a <sense statement> or a <prove statement> that includes a CNX in its <on field>, the connection list in that CNX supersedes the pin list in the <define digital sensor statement>.

2. The <external digital specification> option is used when the signal characteristics of a digital word are defined in an existing specification. The label, <external digital specification>, shall identify the specification for the digital sensor's signal characteristics. If the information identified by <external digital specification> is either incomplete or inconsistent with the requirement, then information shall be supplied explicitly by the appropriate fields in the <define digital sensor statement>, which then takes precedence over the information in the <external digital specification>.

Any referenced external specification is a part of the test documentation and shall be available to all users of the C/ATLAS test program.

3. Explicit signal characteristics of LEVEL-LOGIC-ONE and LEVEL-LOGIC-ZERO may be defined in terms of levels, which are steady states that hold true throughout the bit period and for all instances of a given state. If the <pin descriptor>s TRUE and COMPL are employed, the value stated is that of the TRUE connection with respect to the COMPL connection.

Provision is made for the exceptional insertion of a different level. This technique is used to break up long chains of a given state in some digital transmission systems such as HDB.

4. When the PULSE-CLASS field is used, it shall be consistent with the information included in the digital logic levels and/or the <external digital specification>. The SPEC subfield of the PULSE- CLASS field is used to identify additional pulse classes.

5. LEVEL-LOGIC-QUIES, when specified, refers to the state that exists between bits, between words, or outside <event interval>s. For bus systems, this is the neutral state when no data is present.

6. A named <digital sensor> can be individually referenced in a <sense statement> or <prove statement>.

7. The BIT-RATE/BIT-PERIOD sub-field, when present, specifies the speed of bit transmissions for serial logic signals.

8. For signals that utilize an independent clock source, the synchronizing event shall be identified as an <event>. The beginning of every bit period of the serial data stream generated by the UUT is then synchronized to this <event> via the SYNC BIT-TRANSITION TO EVENT branch.

9. The TRANS-LOGIC-ONE and TRANS-LOGIC-ZERO fields are used to define transitions between the defined -ONE and -ZERO signal levels that correspond to a digital one and a digital zero respectively. The OR is available because some transmission systems permit a logic one or zero to be represented in two different ways depending on the start condition. These transitions occur at the mid-point of a data bit duration in Bi-Phase Signals.

10. The TRANS-SYNC field is used to define that there shall be a transition at the start of each word. A number pair following the key-word indicates the transition required prior to the first bit. If the keyword QUIES occurs after TRANS-SYNC, it indicates that the signal is required to be at the quiescent level prior to the initial transition of the word. The second entry indicates the level it acquires at the start of the word transmission.

The TRANS-SYNC-BIT field is used to define that there shall be a transition at the start of each bit period.

11. In statements employing the TRANS-LOGIC-ONE and TRANS-LOGIC-ZERO fields indicating that the logical information is carried by signal transitions, the -ONE and -ZERO suffixes can be used after the mnemonic in the <statement characteristics> field to indicate that the associated signal characteristic carries the digital information and assumes the stated value when the signal is a ONE or a ZERO respectively. If one of these suffixes is used, the same mnemonic shall occur in another field with the other suffix.

12. The SKEW-TIME MAX field defines the maximum skew between the various input ports of a parallel digital pattern that the test equipment shall be able to tolerate.

13. SERIAL-MSB-FIRST and SERIAL-LSB-FIRST identify the digital signal as being serial and identify respectively that the left-most bit or the right-most bit is received first.

14. A statement containing a SAME AS branch introduces a new <digital sensor>. This <digital sensor> has identical characteristics and connections to the <digital sensor> placed after SAME AS. The <digital sensor> after SAME AS shall have been previously defined in another <define digital configuration structure> that is local within the C/ATLAS Test Program.

15. The <noun field> specifies the type of analog signal that conveys the digital data. The <sensor logic level>s specify the different values for the signal characteristic that conveys the digital data associated with the digital data values. The <statement characteristics> specify the other signal characteristics that do not convey data values. It should be noted that, although the signal value of a characteristic is normally invariant when it is associated with the <sensor logic levels>, it varies whenever the data value changes.

16. The RANGE option in <digital synchronization> shall be used when <real quantity> includes a <data store>, but not otherwise. It defines the acceptable range of values that the <data store> may contain.

17. Either TRANS-LOGIC-ONE and TRANS-LOGIC-ZERO or LEVEL-LOGIC-ONE and LEVEL- LOGIC-ZERO shall be specified.

18. The term LEVEL-LOGIC-HIZ applies only to electrical systems, where LEVEL-LOGIC-ONE and LEVEL-LOGIC-ZERO are defined in terms of voltages. LEVEL-LOGIC-HIZ is defined in terms of the maximum current flows when the defined LEVEL-LOGIC-ONE and LEVEL-LOGIC- ZERO levels are impressed on the test subject outputs. If only one LEVEL-LOGIC-HIZ is specified, the allowed leakage current applies to both of the impressed logic states. If it is required to specify different leakages for the two impressed logic states, then the LEVEL-LOGIC-HIZ specification shall be repeated and the position, which shall be immediately following the LEVEL-LOGIC-ONE or LEVEL-LOGIC-ZERO field, defines to which state the leakage relates.

19. When an ILLEGAL-STATE-INDICATOR is used, the associated <boolean variable> will be set TRUE if the signals sensed by the <digital sensor> do not fall within the defined logic states expressed in the <evaluation field> of the <digital sensor characteristic>. It is set FALSE only by a <calculate statement> and an <input statement>.

20. If only LEVEL-LOGIC-ONE and LEVEL-LOGIC-ZERO are defined, the LEVEL-LOGIC-HIZ state is assumed to be illegal and will cause the ILLEGAL-STATE-INDICATOR <boolean variable> to be set TRUE.

21. If an <event> is specified for serial data clocking, the <max time> field indicates the maximum time that the digital configuration is to wait, following a SENSE or PROVE start, for the first occurrence of the <event>. If the <event> does not occur within this time limit, a MAX-TIME condition is set and testing resumes without completion of the <sense statement> or <prove statement>.

6.16.4.5 Rules applicable to the pulse-class field

1. The PULSE-CLASS is required to be explicitly or implicitly defined for all serial data statements, for both SOURCE and SENSOR. It cannot be used as a measured characteristic in SENSOR statements.

2. The subfield after PULSE-CLASS identifies the class of pulse coding used to represent the digital data. The following classes are defined:

RZ	Return to zero
NRZ	Non return to zero
BIP	Bi-phase
AMI	Alternate mark inversion
HDBn	High density bipolar where n is an <unsigned integer number>.

3. The keyword SPEC followed by a reference to an <external pulse class specification> may be used when a pulse class that is not one of the standard classes is required. The referenced specification shall fully describe the pulse coding of the transmission.

6.16.4.6 Rules applicable to the <sensor logic levels>

1. The <mnemonic> chosen shall be applicable to the <noun> specified in the associated <digital source field>.

2. No source can employ both LEVEL-LOGIC-... and TRANS-LOGIC-... keywords.

3. When used following LEVEL-LOGIC-HIZ to specify the high impedance condition of the test system, the direction of leakage current flow is specified as follows:

a) Current flowing from the test system as a positive current

b) Current flowing into the test system as a negative current

6.16.5 END DIGITAL CONFIGURATION statement

6.16.5.1 Function

This statement terminates a <digital configuration structure>.

6.16.5.2 Formal syntax

Reference end-digital-configuration-statement

6.16.5.3 Syntax diagram

<end digital configuration statement> 6.16.5	<fstatno> END, <configuration> $

6.17 EXTEND statement

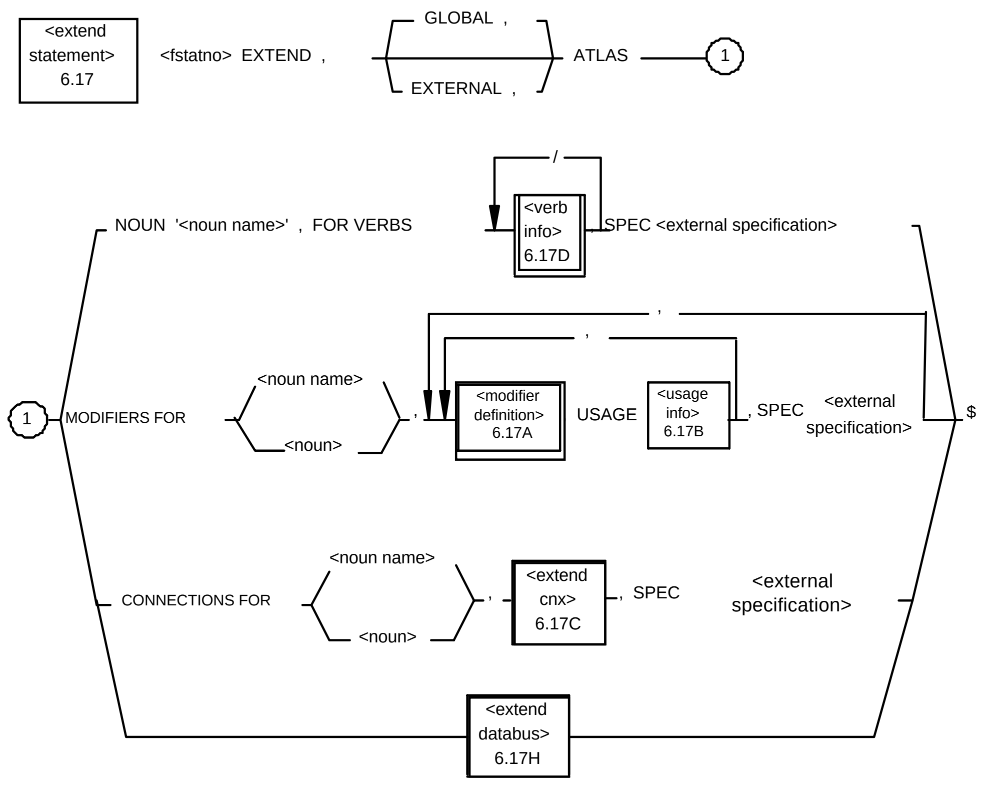
6.17.1 Function

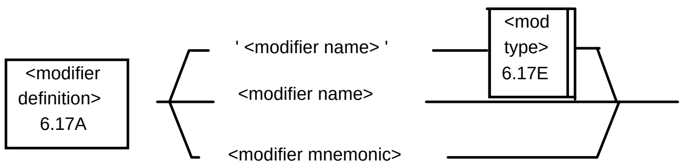
The function is to provide capabilities for adding nouns, noun modifiers, pin-descriptors, and databus extensions to the C/ATLAS language for use in a particular <atlas program structure> or <atlas module structure>. An <extend statement> provides an extensibility mechanism that will allow users to introduce new nouns, noun modifiers, and pin descriptors as required to accommodate new signal types.

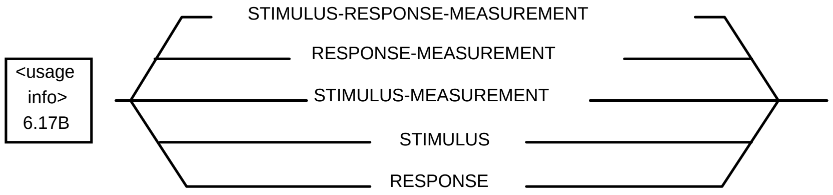
6.17.2 Formal syntax

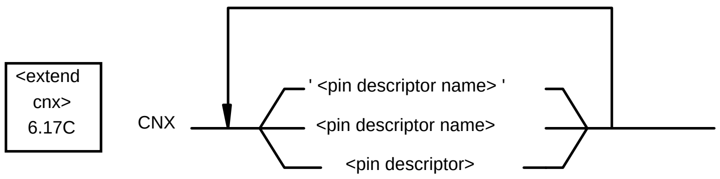
Reference extend-statement

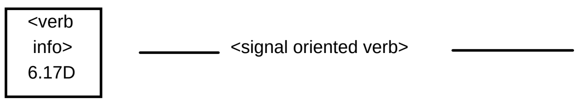
6.17.3 Syntax diagram

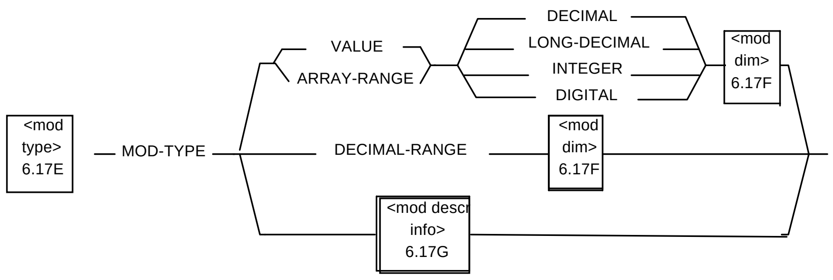


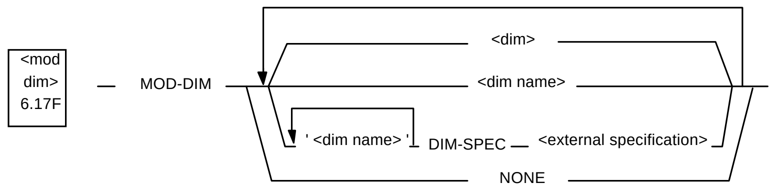

<modifier definition> 6.17A
' <modifier name> '
<mod type> 6.17E
<modifier name>
<modifier mnemonic>

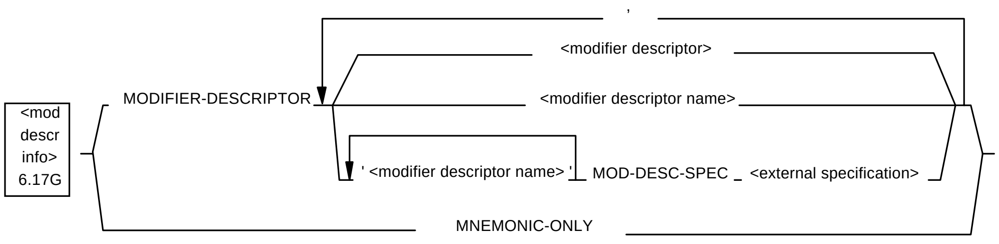

<usage info> 6.17B
STIMULUS-RESPONSE-MEASUREMENT
RESPONSE-MEASUREMENT
STIMULUS-MEASUREMENT
STIMULUS
RESPONSE

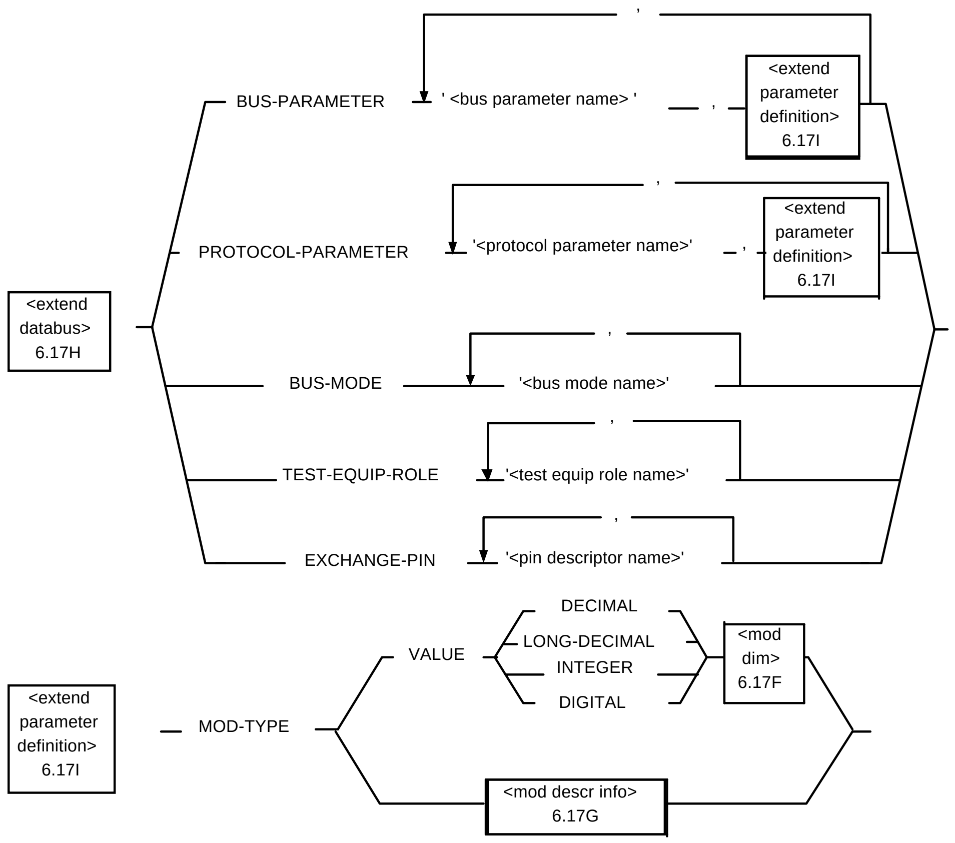

<extend cnx> 6.17C
CNX
' <pin descriptor name> '
<pin descriptor name>
<pin descriptor>


<verb info> 6.17D
<signal oriented verb>


<mod type> 6.17E
MOD-TYPE
VALUE
ARRAY-RANGE
DECIMAL
LONG-DECIMAL
INTEGER
DIGITAL
<mod dim> 6.17F
DECIMAL-RANGE
<mod dim> 6.17F
<mod descr info> 6.17G


<mod dim> 6.17F
MOD-DIM
<dim>
<dim name>
' <dim name> '
DIM-SPEC
<external specification>
NONE


,
<modifier descriptor>
MODIFIER-DESCRIPTOR
<modifier descriptor name>
<mod descr info> 6.17G
' <modifier descriptor name> '
MOD-DESC-SPEC
<external specification>
MNEMONIC-ONLY



6.17.4 Rules

6.17.4.1 Rules for all EXTEND statements

1. All newly introduced <noun name>s, <modifier name>s, <pin descriptor name>s, <modifier descriptor name>s, and <dim name>s are enclosed in single quotes in the statement that defines them. When they are used thereafter, they are written without quotes.

2. The statement EXTEND, GLOBAL may only appear in the preamble of an <atlas program structure>. The statement EXTEND, EXTERNAL may only appear in the preamble of an <atlas module structure>. All EXTEND statements in an <atlas module structure> shall include the word EXTERNAL. Each <noun>, <noun name>, <modifier mnemonic>, <modifier name>, <modifier descriptor>, <modifier descriptor name>, <dim>, <dim name>, <pin descriptor>, <bus parametre name>, <protocol parametre name>, <bus mode name>, <test equipment role name>, or <pin descriptor name> label defined by an EXTEND, EXTERNAL statement shall also be defined by an EXTEND, GLOBAL statement that shall encompass all the information contained in the EXTEND, EXTERNAL statement.

3. It is recommended that any word used in a C/ATLAS extension should be consistent with IEEE Std 100-1992.

6.17.4.2 Rules for EXTEND ATLAS NOUN statements

1. The <noun name> shall be unique in the <atlas module structure> or <atlas program structure> in which it is introduced by an EXTEND statement. It has to be different from all intrinsic or "extended" nouns, noun modifiers, verbs, dimensions, pin-descriptors, and all other literals used in C/ATLAS.

2. <verb info> identifies all verbs with which <noun name> can be used.

3. Every noun modifier or pin-descriptor to be used with a newly defined C/ATLAS noun shall be defined and referenced to the new noun in an EXTEND ATLAS MODIFIERS or EXTEND ATLAS CONNECTIONS statement.

4. Every noun shall be defined in accordance with clause 16. This information shall be included as part of the extension definition. This definition shall be referenced within the SPEC '<external specification>' subfield.

6.17.4.3 Rules for EXTEND ATLAS MODIFIERS statements

1. The <modifier name>, <modifier descriptor name>, or <dim name> that appears in quotes has to be different from all intrinsic and "extended" nouns, noun modifiers, verbs, dimensions, pin descriptors, and all other literals used in C/ATLAS.

2. If new modifiers require the use of prefixes or suffixes, then each of these shall be treated as a separate modifier.

3. USAGE is in accordance with the usage codes of clause 16 (modifier usage).

4. Every noun modifier, prefix, suffix, and modifier definition shall be defined in accordance with clause 17. This definition shall be referenced within the SPEC '<external specification>' subfield. The dimensions should be defined in accordance with 15.8. This definition shall be referenced within the DIM-SPEC '<external specification>' subfield.

6.17.4.4 Rules for EXTEND ATLAS CONNECTIONS statement

1. The <pin descriptor name> has to be different from all intrinsic and "extended" nouns, noun modifiers, verbs, dimensions, pin-descriptors, and all other literals used in C/ATLAS.

2. Every <pin descriptor name> shall be defined in accordance with 14.13. This definition shall be referenced within the SPEC '<external specification>' subfield.

6.17.4.5 Rules for EXTEND BUS-PARAMETRES statement

Every <bus parametre name> used in the <complete atlas program structure> shall appear in an <extend statement>. A <bus parametre name> is used where none of the standardized keywords associated with BUS-PARAMETRE in the <define exchange statement> is applicable to a parametre of the databus that needs to be defined to produce a complete test program. The mod-type and <dimension extension> establish the number type and dimensions associated with the associated characteristic. MNEMONIC-ONLY indicates that the parametre has no associated value.

6.17.4.6 Rules for EXTEND PROTOCOL-PARAMETRES statement

Every <protocol parametre name> used in the <complete atlas program structure> shall appear in an <extend statement>. A <protocol parametre name> is used where none of the standardized keywords associated with PROTOCOL-PARAMETRE in the <define exchange statement> is applicable to a parametre of the databus protocol that needs to be defined to produce a complete test program. The mod-type and <dimension extension> establish the number type and dimensions associated with the associated characteristic. MNEMONIC-ONLY indicates that the parametre has no associated value.

6.17.4.7 Rules for EXTEND BUS-MODE statement

Every <bus mode name> used in the <complete atlas program structure> shall appear in an <extend statement>. A <bus mode name> is used where none of the standardized keywords associated with BUS-MODE in the <define exchange statement> is applicable to an operational mode of the databus protocol that needs to be used to produce a complete test program.

6.17.4.8 Rules for EXTEND TEST-EQUIP-ROLE statement

Every <test equip role name> used in the <complete atlas program structure> shall appear in an <extend statement>. A <test equip role name> is used where none of the standardized keywords associated with TEST-EQUIP-ROLE in the <do exchange statement> is applicable to a role the test equipment has to play during databus testing to produce a complete test program.

6.17.4.9 Rules applicable to the EXTEND EXCHANGE-PIN statement

1. Every <pin descriptor name> used in the <complete atlas program structure> shall appear in an <extend statement>. A <pin descriptor name> is used where the standard <pin descriptor> described in 14.13 (<conn set>) is not applicable to the databus.

2. <pin descriptor name>s introduced by this option can only be referenced in an <establish protocol statement> and its associated <enable exchange configuration statement>s, <connect exchange configuration statement>s, and <disconnect exchange configuration statement>s.

6.17.5 Guidance for the use of extensibility

6.17.5.1 Introduction

The capabilities provided by the extensibility facilities in C/ATLAS permit users to establish local definitions of additional capabilities beyond those defined by this standard. It is highly desirable that these should be used in ways that are consistent within the underlying principles of standardization. The following guidance will support this function without inordinately inhibiting the expeditious use of C/ATLAS.

The whole user community will benefit if uses of extensibility are reported back to the C/ATLAS Committee. This will permit the Committee to determine which uses of extensibility address test situations that are of sufficiently general applicability to justify the addition of new material to C/ATLAS. In practice, the service that the C/ATLAS Committee provides to the user community is totally dependent upon the feedback of problems and solutions employed.

It is intended that the extensibility facilities should have the capability to add any required extension. If it is found that this objective has not been achieved and an extension can only be defined by adapting C/ATLAS, then the problem should be reported to the C/ATLAS Committee to enable it to consider ways of improving C/ATLAS.

6.17.5.2 Objective

The objective of the extensibility facilities is to allow specialists in a particular test discipline to develop C/ATLAS to meet their immediate needs without the delays necessarily involved in the formal approval procedures needed for new C/ATLAS language proposals.

While the extensibility facilities help to avoid inordinate growth in the standard that would make it difficult to use, there is clearly a danger of weakening the C/ATLAS standard if extensibility is used in an excessive and ill-considered way.

6.17.5.3 Guidance

In order to prevent potential weakening of the C/ATLAS standard, the following guidance for the use of extensibility is essential.

1. It is one criterion of the language requirements document that there shall not be alternative capabilities in C/ATLAS. It is not a proper use of the extensibility constructs to use them to define an alternative method of writing material that can be written in C/ATLAS without recourse to extensibility.

2. If a required construct does not exist in C/ATLAS, the extensibility facilities should be used to generate a new construct to provide the required capability.

3. Only if extensibility facilities are found to lack the capability to define a construct with the required capability should other facilities, such as NON-ATLAS, be employed.

6.17.6 Examples

The textural references in the following statements are imaginary and are supplied only as 'fill' characters in the appropriate fields. The new noun RATTLE is nearly synonymous with the existing noun VIBRATION.

```
000100  EXTEND, ATLAS NOUN 'RATTLE',
            FOR VERBS APPLY/MEASURE/VERIFY,
            SPEC 'AS DEFINED IN IEEE/ANSI-100 PARA.  XX'$
    01  EXTEND, ATLAS MODIFIERS FOR RATTLE,
            'RATTLE-AMPLITUDE' MOD-TYPE VALUE DECIMAL
            MOD-DIM MM,
            USAGE STIMULUS-RESPONSE-MEASUREMENT,
            SPEC 'AS DEFINED IN PARA. 17.XX OF XXX
            PROGRAMMING MANUAL'$
    03  EXTEND, ATLAS MODIFIERS FOR RATTLE,
            FREQ, USAGE STIMULUS, SPEC 'XXX' $
    05  EXTEND, ATLAS CONNECTIONS FOR RATTLE,
            CNX 'MOUNT-1' 'MOUNT-2' 'STEADY',
            SPEC 'AS DEFINED IN TDL (PROG1)' $
            .   .   .   .   .   .   .   .   .   .   .   .   .

123456  APPLY, RATTLE, RATTLE-AMPLITUDE 5 MM,
            FREQ 3 HZ,
            CNX MOUNT-1 WIDGET-TOP
            MOUNT-2 WIDGET-BOTTOM
            STEADY WIDGET-FLEXURE $
```

6.18 ESTABLISH PROTOCOL statement

6.18.1 Function

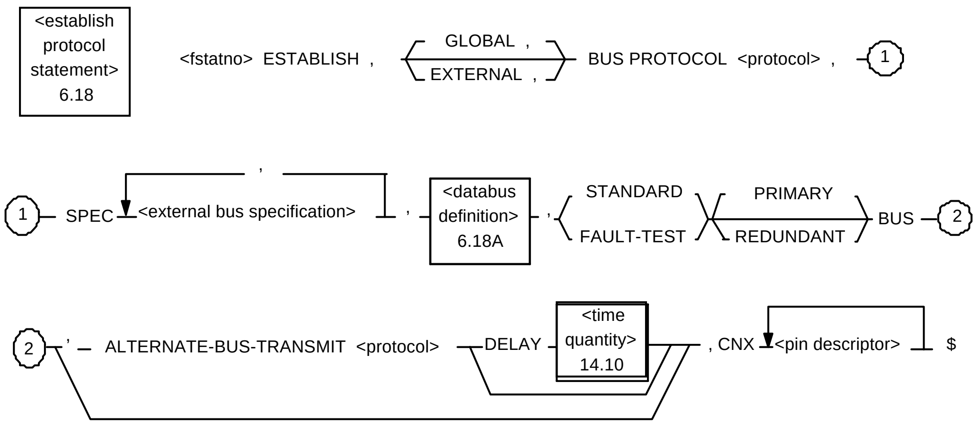
The <establish protocol statement> specifies a bus protocol, associated connections, and a format for all the fields that can be subsequently encountered in the C/ATLAS test program structure. The protocol may be invoked later by use of the <define exchange statement>, <define exchange configuration statement>, <enable exchange configuration statement>, <disable exchange configuration statement>, <connect exchange configuration statement>, <disconnect exchange configuration statement>, <update protocol statement>, and <fetch protocol statement>.

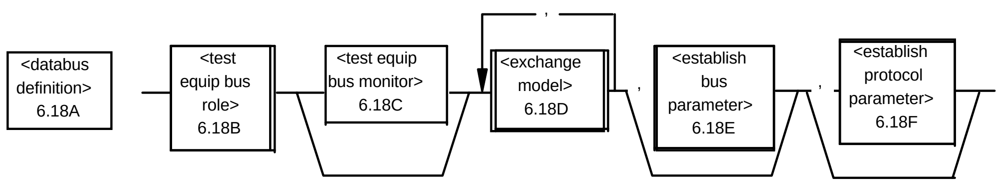
Whenever an <exchange> in a <do exchange statement>, <update exchange statement>, or <fetch exchange statement> references the <protocol> label defined in the <establish protocol statement>, the execution of that <exchange> implies that the specified <protocol> is already enabled by an <enable exchange configuration statement>.

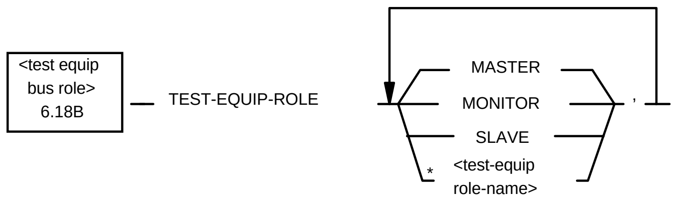
6.18.2 Formal syntax

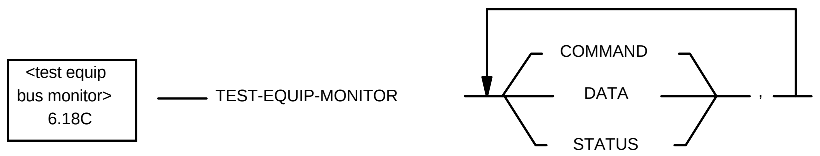
Reference establish-protocol-statement

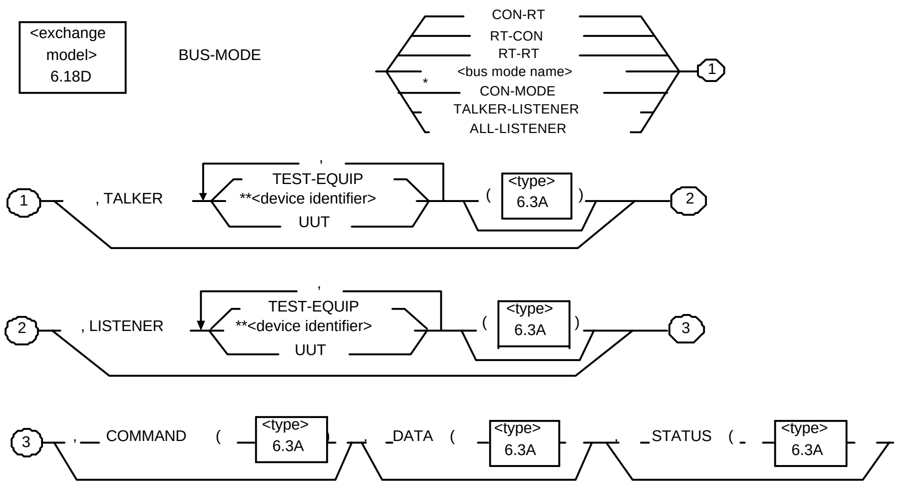
6.18.3 Syntax diagram

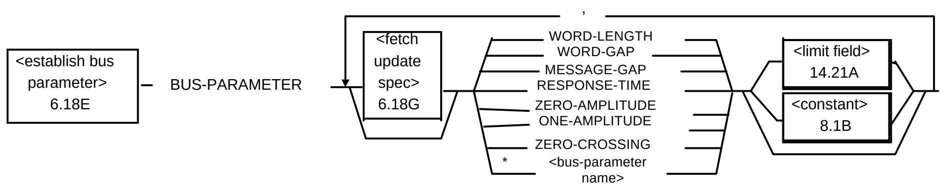

<establish protocol statement> 6.18
<fstatno> ESTABLISH ,
GLOBAL ,
EXTERNAL ,
BUS PROTOCOL <protocol> ,
1
SPEC
<external bus specification>
,
<databus definition> 6.18A
STANDARD
FAULT-TEST
PRIMARY
REDUNDANT
BUS
2
ALTERNATE-BUS-TRANSMIT <protocol>
DELAY
<time quantity> 14.10
, CNX
<pin descriptor>
$

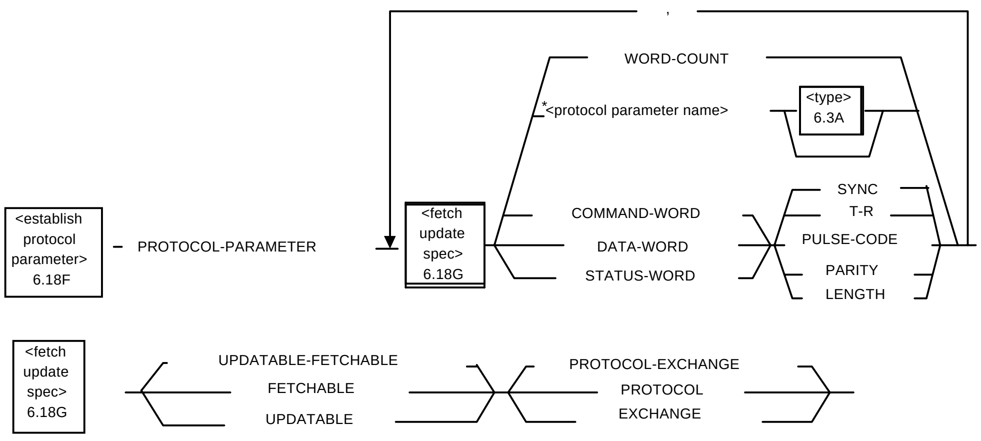

<databus definition> 6.18A
<test equip bus role> 6.18B
<test equip bus monitor> 6.18C
,
<exchange model> 6.18D
,
<establish bus parameter> 6.18E
,
<establish protocol parameter> 6.18F


<test equip bus role> 6.18B
TEST-EQUIP-ROLE
MASTER
MONITOR
SLAVE
* <test-equip role-name>
,


<test equip bus monitor> 6.18C
TEST-EQUIP-MONITOR
COMMAND
DATA
STATUS
,


<exchange model> 6.18D
BUS-MODE
CON-RT
RT-CON
RT-RT
* <bus mode name>
CON-MODE
TALKER-LISTENER
ALL-LISTENER
1
1
, TALKER
,
TEST-EQUIP
**<device identifier>
UUT
(<type> 6.3A)
2
2
, LISTENER
,
TEST-EQUIP
**<device identifier>
UUT
(<type> 6.3A)
3
3
, COMMAND (<type> 6.3A
, DATA (<type> 6.3A
, STATUS (<type> 6.3A


<establish bus parameter> 6.18E
BUS-PARAMETER
,
<fetch update spec> 6.18G
WORD-LENGTH
WORD-GAP
MESSAGE-GAP
RESPONSE-TIME
ZERO-AMPLITUDE
ONE-AMPLITUDE
ZERO-CROSSING
* <bus-parameter name>
<limit field> 14.21A
<constant> 8.1B



NOTES

* The name shall be one that has been introduced by an <extend statement> of the associated type.

** The name shall be one that has been introduced by an enumeration declaration.

6.18.4 Rules

1. A databus is a line or group of lines used to transfer <exchange>s. A databus may include both data and control lines.

2. Use of the GLOBAL attribute makes the <protocol> global. The GLOBAL attribute may only appear in the preamble of an <atlas program structure>. The statement ESTABLISH, EXTERNAL may only appear in the preamble of an <atlas module structure>. An ESTABLISH, GLOBAL statement shall define each label defined by an ESTABLISH, EXTERNAL statement. The global <establish protocol statement> shall encompass the requirements of the ESTABLISH, EXTERNAL statement.

3. <external bus specification> is a text string that fully identifies a specification in which the databus operations in the context are fully or partially defined. Where one exists, such a referenced specification shall be a standard produced by an organization that issues standards for the discipline. If no such comprehensive specification exists, the set of referenced specifications may include one or more produced locally. Any referenced external specification is a part of the test documentation and shall be available to all users of the C/ATLAS test program.

The <external bus specification> text string may include any characters except the dollar sign and apostrophe. Apostrophes shall always enclose the text string.

The first entry in the <external bus specification> set shall reference the military/industry standard identification or nomenclature for the bus. The full set shall fully define the protocol of the normal operation of the databus, all normal signal levels and the connections to the UUT. Statements that reference the databus directly or indirectly shall only contain vocabulary that is fully defined by one of the referenced documents.

4. When the keyword STANDARD is used for a bus protocol, the options specified shall be

within the specified parametre limits for that bus and BUS-PARAMETRE shall not be used within <bus parametre>. If no options are specified, the bus characteristic is according to the standard/mean rates and values specified by the <external bus specification>.

Similarly, when the word STANDARD is used, INSERT-INVALID shall not be used within <protocol parametre>.

5. The keyword FAULT-TEST is used in an <establish protocol statement> to cause the test equipment to simulate faults of any parametre as its default operation. Optional fields have to be specified using the <bus parametre> option to identify the faults that are required to be present in normal operation.

6. The REDUNDANT bus mode may only be used when there are two separate <establish protocol statement>s using different protocol labels and establishing interrelated buses independently connected to the UUT.

The second of the <establish protocol statement>s shall include the keyword PRIMARY and shall contain the ALTERNATE-BUS-TRANSMIT field that references the previous <establish protocol statement> containing REDUNDANT.

The referenced <establish protocol statement> shall include the keyword REDUNDANT and no ALTERNATE-BUS-TRANSMIT field.

The two interrelated buses shall be invoked simultaneously in the <define exchange configuration statement>s.

7. The optional DELAY field determines a fixed time interval between the PRIMARY and REDUNDANT transmissions.

8. The subsequent keyword names, if used, shall be defined in an earlier <extend statement>.

a) <test equip role name> identifies the test equipment's role that is associated with a particular protocol.

b) The <bus mode name> identifies an extended bus mode that is associated with a particular protocol.

c) The <bus parametre name> identifies the physical parametre that is associated with the particular bus parametre, and identifies the appropriate dimensions for the <constant> entries in the associated <bus parametre range>.

d) The <protocol parametre name> identifies the discrete parametre that is associated with the particular bus parametre, and identifies the appropriate data type.

6.18.4.1 Rules applicable to <databus definition>

1. <databus definition> establishes the only keywords that are permissible entries for the associated fields of subsequent statements required for the databus tests. The entries may be standardized keywords associated with the field, or they may be <enumeration element>s from a DEFINE statement that defines the set of <device identifier>s, or they may be new keywords introduced by an <extend statement>. Every keyword included in a field in the <databus definition> shall be fully defined in the context of the particular databus by a specification reference in the SPEC field.

2. <test equip bus role>, together with <test equip bus monitor>, <exchange model>s, <bus parametre>s, and <protocol parametre>s establish the vocabulary associated with the particular <protocol> the number and the type of parametres for each field.

3. The <test equip bus role> specification defines the set of permitted test equipment role(s) during the <do exchange statement>s. When the MONITOR role is specified, the <test equip bus monitor> field shall be used to define the set of items that may be monitored on an exchange during a <do exchange statement>.

6.18.4.2 Rules applicable to the <exchange model> fields

1. Each occurrence of <exchange model> defines one consistent set of parametres on the databus. When the <protocol> is referenced in a <define exchange statement> all the associated parametres shall match one of the <exchange model>s. The set of <exchange model>s shall define all the vocabulary associated with the particular installation. It declares the number and the type of parametre for each field used in an exchange that references the <protocol>.

2. The various fields define the permitted entries in the corresponding fields of subsequent <define exchange statement>s, <do exchange statement>s, <update exchange statement>s, and <fetch exchange statement>s.

3. All the subsequent <define exchange statement>s shall match one of the <exchange model>s established in the referenced <protocol>. The rules for a <define exchange statement> to match an <exchange model> are as follows:

a) The set of <define exchange statement> fields shall match the <exchange model> set.

b) A variable or constant value type used in a <define exchange statement> field shall match the corresponding <exchange model> field <type>.

c) For the TALKER and LISTENER fields, if one or more <device identifier name> and/or TEST-EQUIP and/or UUT are used in the <define exchange statement>, this device list shall be a subset of the <exchange model> list. If a <type> is established in the <exchange model>, each device in the <define exchange statement> shall have a variable or constant value matching this <type>.

4. In the TALKER and LISTENER fields, TEST-EQUIP indicates that the test equipment is required to be an operational device on the databus during some of the testing of the unit under test. UUT indicates that the unit under test is required to be an operational device on the databus and is not referenced by a <device identifier name>.

If indices are required in the TALKER and LISTENER fields, the optional <type> in parentheses indicates the type to fully specify the TALKER or LISTENER.

The valid types are restricted to INTEGER, DECIMAL, LONG-DECIMAL, STRING-OF-BIT, STRING-OF-CHAR, and ENUMERATION. In addition to the above simple types, the type can be specified as a RECORD. This RECORD can contain any combination of the above simple types except ENUMERATION, but shall not contain any RECORD or ARRAY types.

This <type> is used where a UUT has several channels or other sub-addresses. If this subfield is omitted, the calling statement shall not attempt to use any constant or variable value for the device.

5. The <device identifier>s used in a single ESTABLISH statement shall be enumeration elements from a single declared ENUMERATION type.

6. The <type> definitions specified in the COMMAND, DATA, and STATUS fields provide a template for the specification of constants and variables to represent databus words in subsequent <define exchange statement>s. The valid types are restricted to INTEGER, DECIMAL, STRING-OF-BIT, STRING-OF-CHAR, and ENUMERATION. In addition to the above simple types, the type can be specified as a RECORD. This RECORD can contain any combination of the above simple types except ENUMERATION, but shall not contain any RECORD or ARRAY types. A RECORD permits both field structured information and multiple words to be transmitted in a single <exchange>.

6.18.4.3 General rules applicable to <bus parametre>s and <protocol parametre>s

1. The presence of FETCHABLE indicates that the following bus or protocol parametre may be read and stored for subsequent analysis. It implies that the bus or protocol parametre may be used in the <fetch exchange configuration statement>s, and the TEST-EQUIP-MONITOR field of <do exchange statement>s.

2. The presence of UPDATABLE indicates that the following bus or protocol parametre may be updated during testing without inhibiting continuing tests. It implies that the bus or protocol parametre may be used in the <define exchange statement>s, <enable exchange configuration statement>s, and <update exchange configuration statement>s.

3. The use of FETCHABLE-UPDATABLE means that the following bus or protocol parametre can be either read or updated.

4. The PROTOCOL option is used when the following bus or protocol parametre may only be updated or fetched globally for all exchanges using the protocol. It implies that the bus or protocol parametre may be used only in the <update protocol statement>, <fetch protocol statement>s, and <enable exchange configuration statement>s (if UPDATABLE).

5. The EXCHANGE option is used when the following bus or protocol parametre may be updated or fetched for a specific exchange. It implies that the bus or protocol parametre may be used only in the <fetch exchange statement>,<update exchange statement>, and <define exchange statement>s (if UPDATABLE).

6. If a bus/protocol parametre is to be updated/fetched both globally at the protocol level and specifically for an exchange, PROTOCOL-EXCHANGE shall be used.

6.18.4.4 Rules applicable to <bus parametre>

1. The <establish bus parametre> defines the set of analog parametres that can be fetched or updated on the databus.

2. If a value is required to fully specify the bus parametre, it shall be followed by a <limit field> field that defines the extremes of the value, unless the bus parametre type is STRING-OF-CHAR or STRING-OF-BIT, in which case the <limit field> shall be bypassed.

3. The <limit field> defines the extremes of the values for the associated parametre that occur in the <atlas program structure> or <atlas module structure>. In the case of an <atlas program structure>, the <limit field> shall encompass the complete <limit field> for every <atlas module structure> associated with the <complete atlas test program structure>.

4. The optional <dim>s in <dimension number> of the <limit field> field shall be compatible with the bus parametre standard or extended keyword definition. All <dim>s in the <limit field> shall be the same.

5. For definitions of the predefined bus parametres see 14.39.4 rule 3.

6.18.4.5 Rules applicable to <protocol parametre>

1. The <establish protocol parametre> defines the set of discrete parametres that can be accessed in the subsequent databus statements.

2. The COMMAND-WORD, DATA-WORD, and STATUS-WORD keywords indicate the ability to modify or read a characteristic of one of the relevant words in a transmission package in a manner defined by a subsequent keyword. If the keyword is preceded by UPDATABLE it may be set in an invalid or standard state by using the INSERT-INVALID word. If the keyword is preceded by FETCHABLE, the default count of words may be fetched.

3. The WORD-COUNT protocol parametre, if preceded by UPDATABLE, allows an invalid WORD-COUNT value to be set in the subsequent statements. If preceded by FETCHABLE, it allows an erroneous number of words to be read.

4. The <protocol parametre name>, if used, shall have been defined in an earlier <extend statement>. The optional <type> field defines the type of data to be updated or fetched.

5. For definitions of the predefined protocol parametres see 14.40.4 rule 3.

6.18.5 Notes and examples

1. Tests employing databuses require the use of the compulsory statements ESTABLISH, DEFINE, ENABLE, DISABLE, DO, and the optional statements CONNECT, DISCONNECT, UPDATE, FETCH. It is essential to understand the relationship between these statements to appreciate fully the role of any individual statement. This note describes the theoretical role of each:

<establish statement>. This preamble statement labels a databus and references the specifications that define the databus. It defines vocabulary and parametres used with this protocol.

<define exchange configuration>. This preamble statement associates one or several databuses that have to work together.

<define exchange statement>. This preamble statement defines and labels a single operation on the databus. It defines the bus mode, the devices that are operational on the bus, and the talker, listener, command, data, or status information associated with the exchange.

<enable exchange configuration statement>. This procedural statement initializes parametres that are common to all the exchanges. It connects a bus configuration to a particular UUT pin set.

<disable exchange configuration statement>. This procedural statement renders inactive the set of protocols and disconnects the bus configuration.

<do exchange statement>. This procedural statement defines a test situation on the databus. It defines which sequences of exchanges are executed on the databus, and for each exchange, the role of the ATE.

<connect exchange configuration statement>. This procedural statement connects one or more protocols of a bus configuration to a particular UUT pin set.

<disconnect exchange configuration statement>. This procedural statement disconnects one or more protocols of a bus configuration from a particular UUT pin set.

<update exchange statement>. This procedural statement changes data, status, and command information of one of the defined exchanges and/or any bus or protocol parametres initialized by the proceeding <do exchange statement> without interrupting the databus operation.

<update protocol statement>. This procedural statement changes any bus or protocol parametres initiated by the preceding <do exchange statement> without interrupting the databus operation.

<fetch exchange statement>. This procedural statement reads data, status, and/or any bus or protocol parametres of one of the defined exchanges initiated by the preceding <do exchange statement> without interrupting the databus operation.

<fetch protocol statement>. This procedural statement reads any bus or protocol parametres of one of the established protocols initiated by the preceding <do exchange statement> without interrupting the databus operation.

6.19 DEFINE EXCHANGE statement

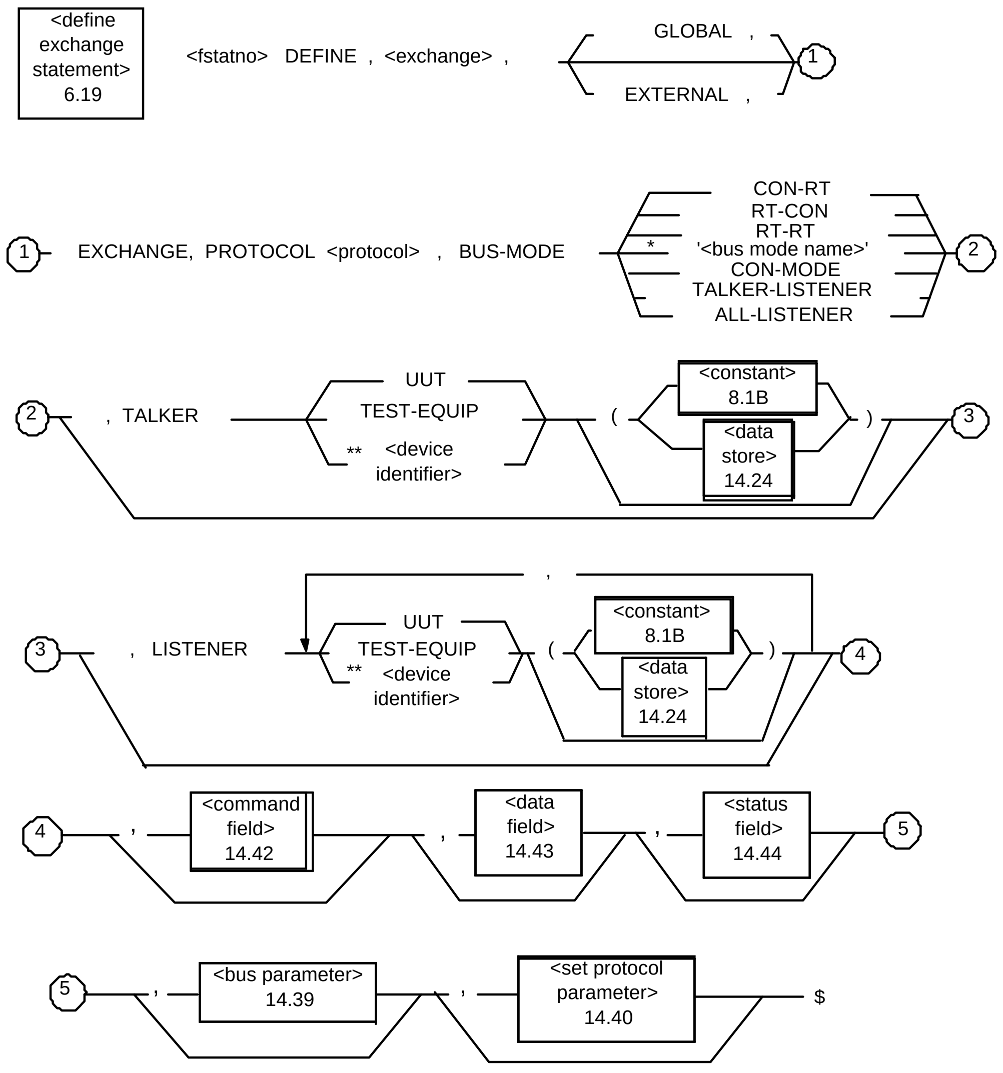
6.19.1 Function

The <define exchange statement> establishes an <exchange> label corresponding to a particular exchange (or transaction) on a databus.

6.19.2 Formal syntax

Reference define-exchange-statement

6.19.3 Syntax diagram



NOTES

* The name shall be one that has been introduced by an <extend statement> of the associated type.

** The name shall be one that has been introduced by an enumeration declaration.

6.19.4 Rules

1. The <define exchange statement> establishes and labels one <exchange> on a databus.

2. An <exchange> involves one block of information traveling on the databus. An <exchange> can involve control information, data, status information, or a combination of two or all three.

3. The global <define exchange statement> uses the GLOBAL attribute. This may appear only in the preamble of an <atlas program structure>. The statement DEFINE, <exchange>, EXTERNAL may only appear in the preamble of an <atlas module structure>. Each label defined by an EXTERNAL <exchange> definition statement shall have a corresponding definition in a GLOBAL <exchange> definition statement that encompasses the requirements of the EXTERNAL <exchange> definition statement.

4. PROTOCOL field. The PROTOCOL field references the previously established <protocol> that controls the databus. The specifications referenced by the <establish protocol statement> shall include all the information required to select the databus with the correct connections. They shall also include the information necessary to control operations, to transfer data, to handle synchronization, and to handle timing. The <define exchange statement> supplies the required mode of bus operation, the default bus devices involved, and any default data, command, or status information. The <do exchange statement> supplies any variations to the default information.

5. The <exchange definition> shall be consistent with an <exchange model> within the <establish protocol statement> of the referenced <protocol>.

6. BUS-MODE field. The BUS-MODE field identifies the mode of bus operation for the exchange. It is used to indicate which of the possible operating modes of the bus is to be used. It may be one of the default entries listed below or a <bus mode name>. Whichever is employed shall be fully defined as it applies to that bus in the set of <external bus specification>s referenced in the <establish protocol statement> and if it is a <bus mode name>, it shall be introduced by a databus <extend statement>.

The standard bus modes are as follows:

CON-RT. Bus controller to remote terminal transfers. The controller is talking and one device is listening. The TALKER field is not used.

RT-CON. Remote terminal to bus controller transfers. One device is talking and the controller is listening. The LISTENER field is not used.

RT-RT. Remote terminal to remote terminal transfers. One device is talking and one device is listening.

CON-MODE. Transfer(s) from bus controller to remote terminal(s). The controller is talking to all devices. The TALKER field is not used.

TALKER-LISTENER. The talker and listener(s) are explicitly designated.

ALL-LISTENER. One talker is explicitly designated and all other devices are listening. The LISTENER field is not used.

7. TALKER field. The TALKER field identifies the device that will talk (send data) when the <exchange> transfers data (as opposed to control signals). For this purpose, command and status information is a special form of data. The TALKER field shall contain one of the <device identifier>s listed in the corresponding TALKER field of the <establish protocol

statement>. The keyword TEST-EQUIP indicates that the TALKER is the test equipment. The keyword UUT indicates that the TALKER is the unit under test.

If the transaction transfers data, command, or status information, the TALKER field is normally required. The only exception is when a particular bus mode defines the TALKER by default.

The optional <constant> or <data store> allows addressing and sub-addressing within a device. The <constant> or <data store> <type> shall be consistent with the corresponding <exchange model>.

This <data store> shall reference a previously declared variable. The value of the <data store> will be evaluated at execution of any statement referencing the defined exchange.

If multi-level addressing or sub-addressing is required, a variable of type RECORD shall be used.

8. LISTENER field. The LISTENER field establishes the device(s) that will listen (receive data). They will receive data (as opposed to control signals) transferred during the <exchange>. The LISTENER field shall contain one of the <device identifier>s listed in the corresponding <exchange model> LISTENER field of the <establish protocol statement>. The keyword TEST- EQUIP indicates that the LISTENER is the test equipment. The keyword UUT indicates that the LISTENER is the unit under test.

The number and selection of listeners shall be consistent with the bus mode selected for the <exchange>.

If the transaction transfers data, command, or status information, the LISTENER field is normally required. The only exception is when a particular bus mode defines the LISTENERs by default.

The optional <constant> or <data store> allows addressing and sub-addressing within a device. The <constant> or <data store> <type> shall be consistent with the corresponding <exchange model>.

This <data store> shall reference a previously declared variable. The value of the <data store> will be evaluated at execution of any statement referencing the defined exchange.

If multi-level addressing or sub-addressing is required, a variable of type RECORD shall be used.

9. COMMAND field. The COMMAND field occurs if the selected bus mode requires a command to be transmitted. The <type> of this field shall be consistent with the corresponding <exchange model>.

10. DATA field. The DATA field occurs if the selected bus mode requires data to be transmitted. The <type> of this field shall be consistent with the corresponding <exchange model>.

11. STATUS field. The STATUS field occurs if the selected bus mode requires status to be transmitted. The <type> of this field shall be consistent with the corresponding <exchange model>.

12. Appropriate bus modes allow control-only <exchange>s. This case might require the DATA field although the TALKER and LISTENER fields are unused.

13. Each bus mode requires a predetermined set of fields. The BUS-MODE determines how these fields are converted into command, status, and data transmission information on the data bus. All fields that are required by the particular bus mode shall be included in the <define exchange statement>. Fields that a bus mode cannot use may occur. If they do, the data included in the field is not transmitted on the databus.

14. COMMAND, STATUS, and DATA information shall be included in a <define exchange statement> for transmission by the TEST-EQUIP or by another device. If the test equipment simulates this device, the simulated device shall be identified in the TEST-EQUIP-SIMULATE field of a <do exchange statement>. Any information transmitted by another device which it is desired to collect and store for future analysis is identified in the TEST-EQUIP-MONITOR field of a <do exchange statement>.

15. The <bus parametre> and <set protocol parametre> fields may be used to specify a bus and/or protocol parametre value to be applied when the <exchange> is invoked in a <do exchange statement>.

6.19.5 Notes and examples

TALKER and LISTENER are terms used with the exchange of information. They are unrelated to the concepts of master and slave associated with the control of many databuses.

Subclause 6.18.5 describes the fundamental philosophy of databus testing.

Subclause 13.2.5 includes examples of databus tests.

6.20 DEFINE DIGITAL TIMING statement

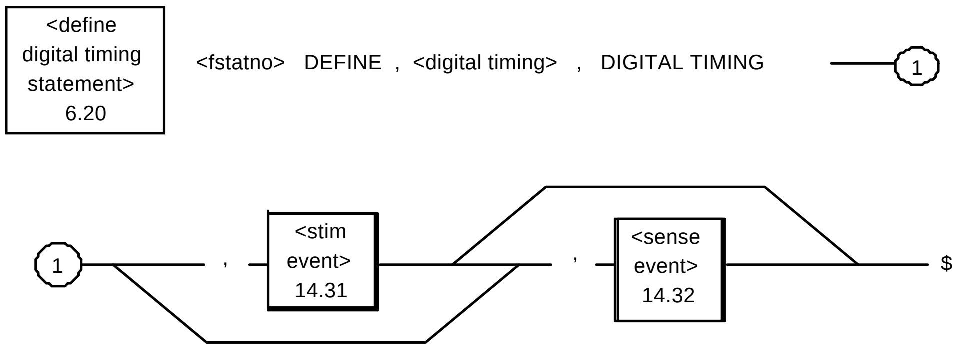
6.20.1 Function

The <define digital timing statement> assigns a <label> to a <stim event> and/or <sense event> list to be used in one or more <do timed digital statement>s.

6.20.2 Formal syntax

Reference define-digital-timing-statement

6.20.3 Syntax diagram


<define digital timing statement> 6.20
<fstatno> DEFINE , <digital timing> , DIGITAL TIMING
1
1
,
<stim event> 14.31
,
<sense event> 14.32
$

6.20.4 Rules

1. A valid C/ATLAS statement shall result when the information associated with a <digital timing> label is substituted within a <do timed digital statement>.

2. Only complete <stim event> and/or <sense event> fields may be included in a <define digital timing statement>.

3. A <digital timing> label may be used to replace a <stim event>, <sense event> or combination of <stim event> followed by <sense event> in a <do timed digital statement>.

6.20.5 Notes and examples

```
          .
          .
          .
100000 DEFINE, 'CYCLE 1', DIGITAL TIMING,
          STIM-EVENT 'S1', 'S2', 'S3', 'S4',
          SENSE-EVENT 'P1', 'P2', 'P3' $
          .
          .
          .
200000 DO, TIMED DIGITAL, 'CYCLE 1', ITERATE 3 TIMES,
          PROCEED, MAX-TIME 10 MSEC $
          .
          .
          .
```

6.21 COMPLEX SIGNAL definition

6.21.1 DEFINE COMPLEX SIGNAL structure

6.21.1.1 Function

The <define complex signal structure> defines one complex signal that is the resultant signal created by operation(s) upon a set of input signals. It provides the means for specifying the characteristics that completely define a COMPLEX SIGNAL as consisting of one or more signals combined or operated on by some function to form a single signal. This structure provides for defining the component signals, as well as the relationships and complex functions that comprise a COMPLEX SIGNAL. The resultant signal defined in this structure may be referenced by any statement as a source or sensor signal or as a component signal of another <define complex signal structure> by using the noun COMPLEX SIGNAL <label>.

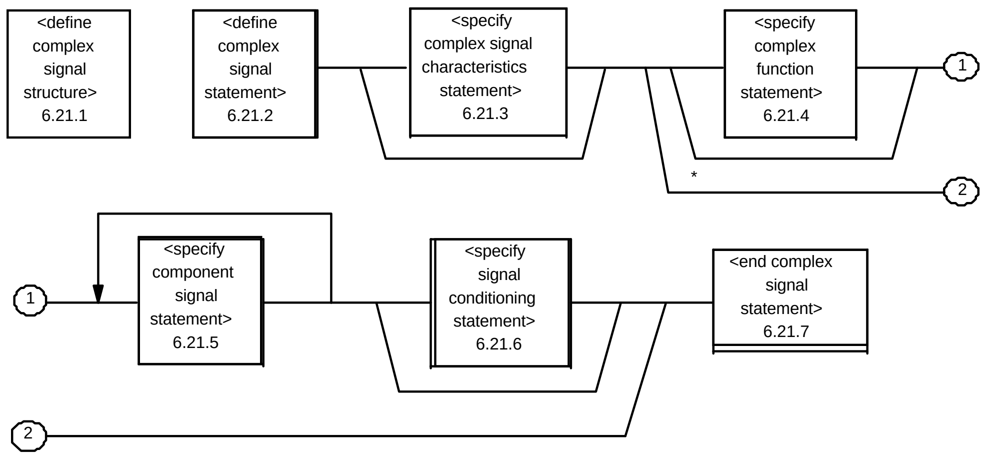
The <define complex signal structure> identifies the nature of simultaneous resources required for the source or sensing of the signal.

This structure contains three statement types: one to specify the type of complex function being performed to define the complex signal, one to specify the component signal(s) required to define the COMPLEX SIGNAL, and one to specify any necessary modifying or conditioning required in the generation of the resultant complex signal.

6.21.1.2 Formal syntax

Reference define-complex-signal-structure

6.21.1.3 Syntax diagram



<define complex signal structure> 6.21.1
<define complex signal statement> 6.21.2
<specify complex signal characteristics statement> 6.21.3
<specify complex function statement> 6.21.4
1
*
2
<specify component signal statement> 6.21.5
<specify signal conditioning statement> 6.21.6
<end complex signal statement> 6.21.7

NOTE

* This bypass is valid for measurement statements as defined in 6.21.1.4, rule 1.

6.21.1.4 Rules

1. The bypass subject to the note in 6.21.1.3 is used to define a label for a signal that is referenced as a component signal of a complex signal in a measure statement. The bypass is not allowed in stimulus statements, or when the bypass around 6.21.3 is taken.

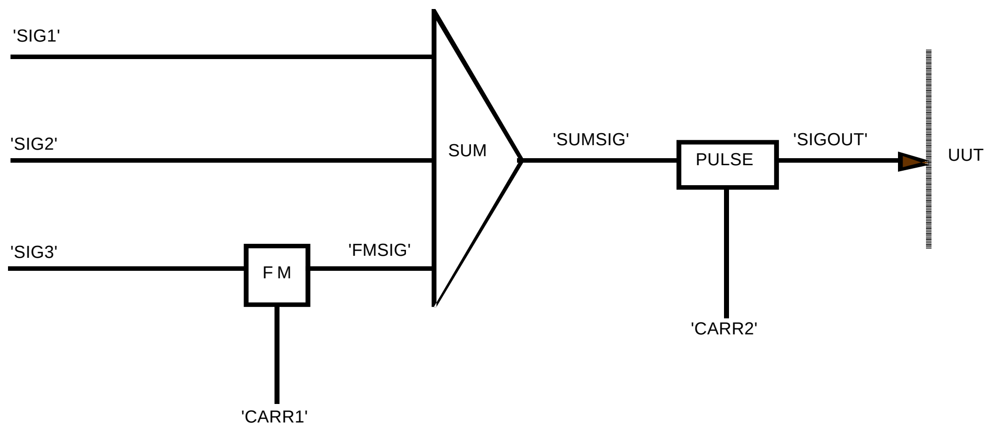
2. A <define complex signal structure> is not active until referenced by an appropriate ENABLE, COMPLEX SIGNAL statement.

3. If synchronization or timing fields are included within statements in a <define complex signal structure>, their timing applies only to that signal component.

4. When making measurements of parametres of a component of a COMPLEX SIGNAL, the sensor statement shall contain the COMPONENT <complex signal> subfield. This shall reference a previous <define complex signal structure>.

6.21.1.5 Notes and examples

1. The following defines a complex source signal 'SIGOUT', which is a pulse modulation of the sum of two IF signals plus an FM signal. The model is shown below.

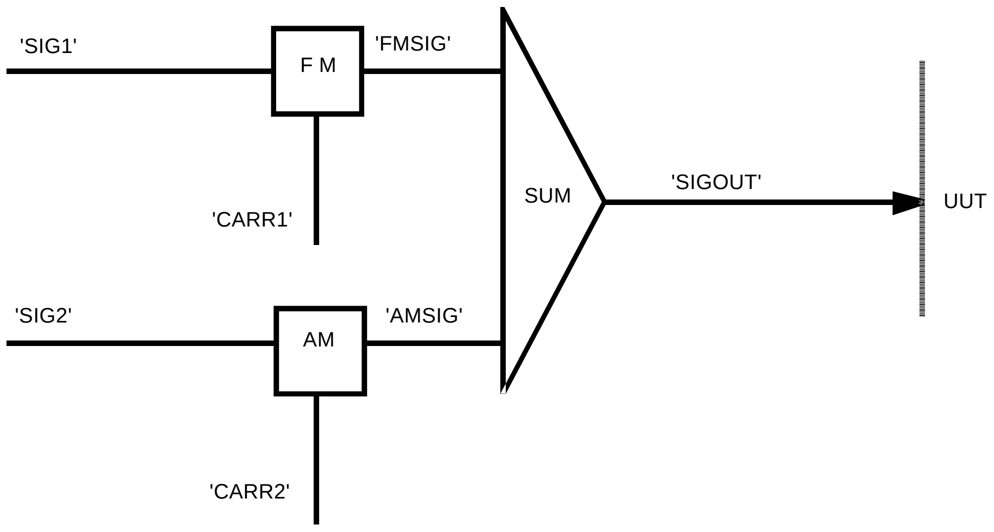


The example ATLAS DEFINE structures and references in a test program for this model are as follows:

```
      DEFINE, 'FMSIG', COMPLEX SIGNAL $
      SPECIFY, COMPLEX FUNCTION, FM, FM-SENSITIVITY 200 HZ/VOLT $
      SPECIFY, AS MOD-SIGNAL, WAVEFORM, STIM 'SIG3' (1 THRU 4000),
VOLTAGE-INST RANGE -1V TO 3V PERIOD 0.3 MSEC $
      SPECIFY, AS CARRIER, AC SIGNAL, FREQ 40 MHZ, VOLTAGE 5 V $
      END, 'FMSIG' $
      DEFINE, 'SUMSIG', COMPLEX SIGNAL $
      SPECIFY, COMPLEX FUNCTION, SUM $
      SPECIFY, AS COMPONENT, AC SIGNAL, FREQ 'SIG1' RANGE 1MHZ TO 3MHZ,
        VOLTAGE 'V1' RANGE 2V TO 5V $
      SPECIFY, AS COMPONENT, AC SIGNAL, FREQ 'SIG2' RANGE 1MHZ TO 3MHZ,
        VOLTAGE 'V2' RANGE 2V TO 5V $
      SPECIFY, AS COMPONENT, COMPLEX SIGNAL 'FMSIG' $
      END, 'SUMSIG' $
      DEFINE, 'SIGOUT', COMPLEX SIGNAL $
      SPECIFY, COMPLEX FUNCTION, PULSE, MOD-INDEX 1.2 $
            SPECIFY, AS CARRIER, PULSED DC, PERIOD 55 NSEC, VOLTAGE 2.4 V,
        PULSE-WIDTH 20 NSEC $
      SPECIFY, AS MOD-SIGNAL, COMPLEX SIGNAL 'SUMSIG' $
```

```
END, 'SIGOUT' $
  :
  :
ENABLE, COMPLEX SIGNAL 'SIGOUT' $
APPLY, COMPLEX SIGNAL 'SIGOUT', POWER -10 DBM,
  CNX HI J1-1 LO J1-2 $
  :
  :
REMOVE, COMPLEX SIGNAL 'SIGOUT', POWER -10 DBM,
  CNX HI J1-1 LO J1-2 $
  :
  :
DISABLE, COMPLEX SIGNAL 'SIGOUT' $
```

2. The following defines a complex source signal 'SIGOUT', which is the sum of the FM modulated signal 'FMSIG' and the AM modulated signal 'AMSIG'. In addition, the carrier for the FM modulated signal is swept and the AM modulating signal is established at the time the complex signal is applied. The model is shown below.

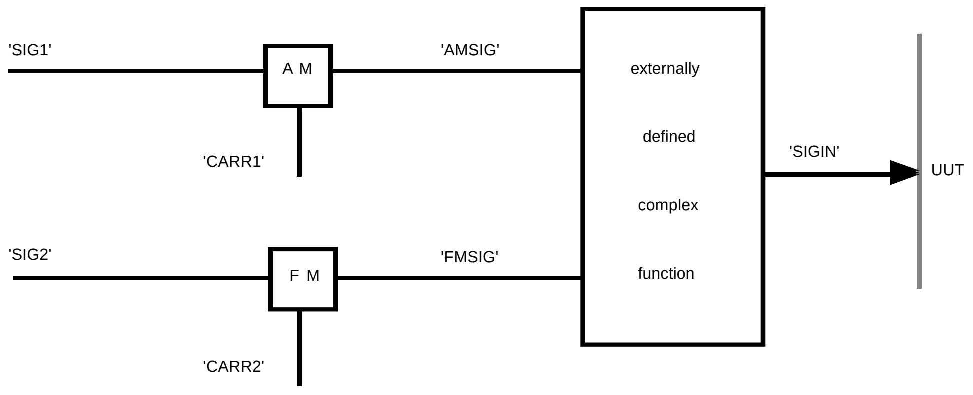


The example ATLAS DEFINE structures and references in a test program for this model are as follows:

```
      DEFINE, 'FMSIG', COMPLEX SIGNAL $
      SPECIFY, COMPLEX FUNCTION, FM, FREQ-DEV 1 KHZ $
      SPECIFY, AS CARRIER, AC SIGNAL, POWER -5 DBM,
         SWEEP-CONFIGURATION SMOOTH,
         FREQ RANGE 150 MHZ TO 200 MHZ, FORWARD-REVERSE,
         SWEEP-TYPE LIN, SWEEP-TIME 1 SEC $
      SPECIFY, AS MOD-SIGNAL, TRIANGULAR WAVE SIGNAL, VOLTAGE-PP 5
V,
          FREQ 100 HZ $
      END, 'FMSIG' $
      DEFINE, 'CARR2', COMPLEX SIGNAL $
      SPECIFY, COMPLEX SIGNAL, SQUARE WAVE,
         FREQ RANGE 0.9GHZ TO 1.5GHZ, VOLTAGE-PP 10 V $
      END, 'CARR2' $
      DEFINE, 'AMSIG', COMPLEX SIGNAL $
```

```
SPECIFY, COMPLEX FUNCTION, AM, MOD-INDEX 0.75,
   MOD-FUNCTION LOG $
SPECIFY, AS CARRIER, COMPLEX SIGNAL 'CARR2' $
SPECIFY, AS MOD-SIGNAL, AC SIGNAL, VOLTAGE-PP 5 V,
   FREQ 100 HZ $
END, 'AMSIG' $
DEFINE, 'SIGOUT', COMPLEX SIGNAL $
SPECIFY, COMPLEX FUNCTION SUM $
SPECIFY, AS COMPONENT, COMPLEX SIGNAL 'FMSIG' $
SPECIFY, AS COMPONENT, COMPLEX SIGNAL 'AMSIG' $
END 'SIGOUT' $
   :
   :
ENABLE, COMPLEX SIGNAL 'SIGOUT' $
 APPLY, COMPLEX SIGNAL 'SIGOUT', POWER +2 DBM, COMPONENT
  'CARRIER2' FREQ 1 GHZ, CNX HI J4-1 LO J4-2 $
   :
   :
REMOVE, COMPLEX SIGNAL 'SIGOUT', POWER +2 DBM,
   CNX HI J4-1 LO J4-2$
   :
   :
DISABLE, COMPLEX SIGNAL 'SIGOUT' $
```

3. The following defines a complex input signal 'SIGIN', which is the result of an operation described by external semantics on two signals of which only one is of concern, 'AMSIG'. The model is shown below.

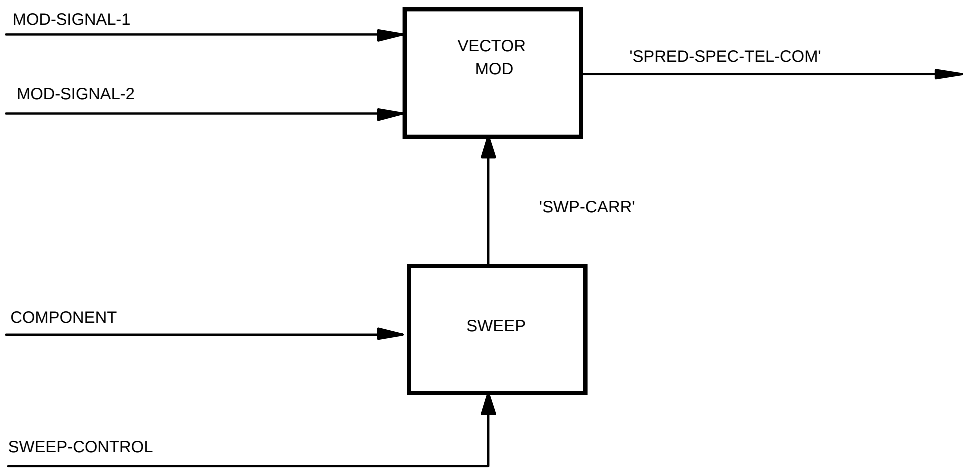


The example ATLAS DEFINE structures and references in a test program for this model to measure the frequency of both 'SIG1' and 'CARR1' and the power of 'AMSIG' are shown below. All three are defined as well as 'SIGIN' so that they may be referenced in the measure statement.

```
DEFINE, 'SIG1', COMPLEX SIGNAL $
SPECIFY, COMPLEX SIGNAL, AC SIGNAL, FREQ RANGE 100 HZ TO 105 HZ,
   VOLTAGE-PP 4 V $
END, 'SIG1' $
DEFINE, 'CARR1', COMPLEX SIGNAL $
SPECIFY, COMPLEX SIGNAL, AC SIGNAL, FREQ RANGE 1GHZ TO 1.001GHZ, POWER
   -5 DBM $
```

```
END, 'CARR1' $
DEFINE, 'AMSIG', COMPLEX SIGNAL $
SPECIFY, COMPLEX SIGNAL 'AMSIG', POWER RANGE 0 DBM TO 1 DBM        $
SPECIFY, COMPLEX FUNCTION, AM, MOD-INDEX 0.75 $
SPECIFY, AS CARRIER, COMPLEX SIGNAL 'CARR1' $
SPECIFY, AS MOD-SIGNAL, COMPLEX SIGNAL 'SIG1' $
END, 'AMSIG' $
DEFINE, 'SIGIN', COMPLEX SIGNAL $
SPECIFY, COMPLEX FUNCTION, SPEC '<external complex function
   specification>'......$
SPECIFY, AS COMPONENT, COMPLEX SIGNAL 'AMSIG' $
SPECIFY, AS COMPONENT, COMPLEX SIGNAL 'FMSIG' $
END, 'SIGIN' $
   :
ENABLE, COMPLEX SIGNAL 'SIGIN' $
   :
   :
MEASURE, (COMPONENT 'SIG1' FREQ INTO 'MEAS'), COMPLEX SIGNAL
   'SIGIN', COMPONENT 'SIG1'
   FREQ MAX 200 HZ,
   RF-BANDWIDTH 'XX' HZ, VIDEO-BANDWIDTH 'YY' HZ,
   POWER 2 DBM, POS-SAMPLING, AVG-SWEEPS 4,
   CNX HI J1-4 LO J1-3 $
   :
   :
MEASURE, (COMPONENT 'CARR1' FREQ INTO 'MEAS'), COMPLEX SIGNAL
   'SIGIN', COMPONENT 'CARR1'
   FREQ MAX 1.1 GHZ,
   RF-BANDWIDTH 'AA' HZ, VIDEO-BANDWIDTH 'BB' HZ,
   POWER 2 DBM, POS-SAMPLING, AVG-SWEEPS 4,
   CNX HI J1-4 LO J1-3 $
   :
   :
MEASURE, (COMPONENT 'AMSIG' POWER INTO 'MEAS'), COMPLEX SIGNAL
   'SIGIN', COMPONENT 'AMSIG' POWER RANGE 0 DBM TO 1 DBM,
   RF-BANDWIDTH 'CC' HZ, VIDEO-BANDWIDTH 'DD' HZ,
   POWER 2 DBM, POS-SAMPLING, AVG-SWEEPS 4,
   CNX HI J1-4 LO J1-3 $
```

4. The following defines a complex telecommunications signal 'SPRED-SPEC-TEL-COM'.



```
    DEFINE 'SWP-CARR', COMPLEX SIGNAL $
    SPECIFY, COMPLEX FUNCTION, SWEEP,
       FREQ PROPORTIONAL TO VOLTAGE-PP,
       FREQ RANGE 100 KHZ TO 1 GHZ,
       VOLTAGE-PP RANGE O V TO 1 V $
    SPECIFY, AS COMPONENT, AC SIGNAL,
       VOLTAGE-PP 1O V, FREQ MAX 1 GHZ,
       CNX HI J1-4 LO J5-2 $
    SPECIFY, AS SWEEP-CONTROL, RAMP SIGNAL,
       VOLTAGE-PP 1 V, FREQ 100 HZ,
       CNX HI J5-9 LO J5-2 $
    END, 'SWP-CARR' $
    DEFINE, 'SPRED-SPEC-TEL-COM', COMPLEX SIGNAL $
    SPECIFY, COMPLEX FUNCTION, VECTOR, VECTOR-TYPE 64QAM $
    SPECIFY, AS CARRIER, 'SWP-CARR' $
    SPECIFY, AS MOD-SIGNAL-I, AC SIGNAL, FREQ 400HZ,
       VOLTAGE-PP 1V $
    SPECIFY, AS MOD-SIGNAL-Q, AC SIGNAL, FREQ 400HZ,
       VOLTAGE-PP 1V $
     END, 'SPRED-SPEC-TEL-COM' $
        :
        :
    ENABLE, COMPLEX SIGNAL 'SPRED-SPEC-TEL-COM' $
    APPLY, COMPLEX SIGNAL 'SPRED-SPEC-TEL-COM', POWER +2 DBM,
       CNX HI J4-1 LO J4-2 $
```

6.21.2 DEFINE COMPLEX SIGNAL statement

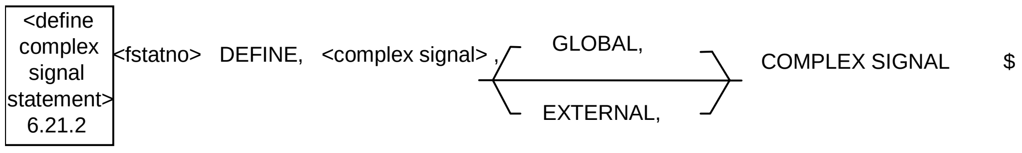
6.21.2.1 Function

This statement labels and introduces a <define complex signal structure>.

6.21.2.2 Formal syntax

Reference define-complex-signal-statement

6.21.2.3 Syntax diagram



6.21.2.4 Rules

The statement DEFINE, <complex signal>, GLOBAL may only appear in the preamble of an <atlas program structure>. The DEFINE, <complex signal>, EXTERNAL may only appear in the preamble of an <atlas module structure>. Each <complex signal> label defined by a DEFINE, <complex signal>, EXTERNAL statement shall also be defined by a DEFINE, <complex signal>, GLOBAL statement in which the requirements of the DEFINE, <complex signal>, GLOBAL statement encompass those of the DEFINE, <complex signal>, EXTERNAL statement.

6.21.3 SPECIFY COMPLEX SIGNAL CHARACTERISTICS statement

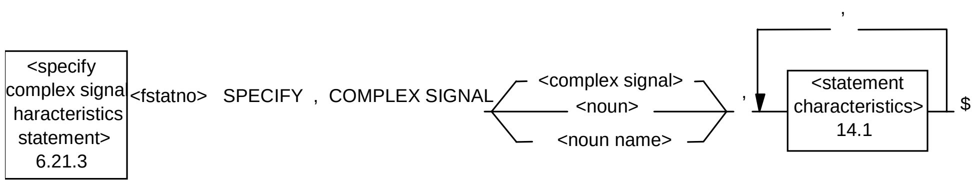
6.21.3.1 Function

This statement specifies the characteristics of the total COMPLEX SIGNAL.

6.21.3.2 Formal syntax

Reference specify-complex-signal-characteristics-statement

6.21.3.3 Syntax diagram


<specify complex signa haracteristics statement> 6.21.3
<fstatno>
SPECIFY
,
COMPLEX SIGNAL
<complex signal>
<noun>
<noun name>
,
,
<statement characteristics> 14.1
$

6.21.3.4 Rules

1. This statement shall be used when a component of a complex signal is referenced by a sensor statement.

2. This statement shall be used to specify the range of modifiers that are referenced when using the keyword COMPONENT in a signal oriented statement.

6.21.4 SPECIFY COMPLEX FUNCTION statement

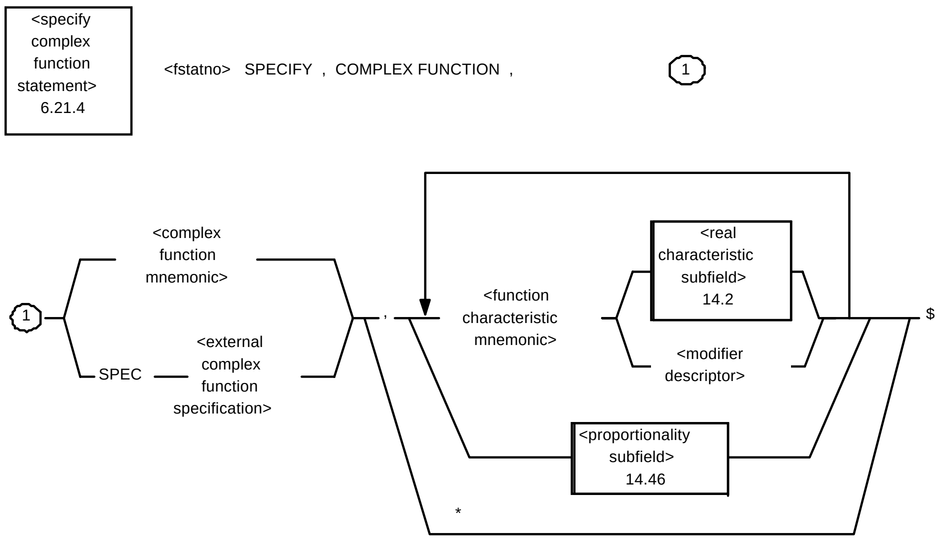
6.21.4.1 Function

This statement specifies a complex function that operates on a set of input signals to create the COMPLEX SIGNAL.

6.21.4.2 Formal syntax

Reference specify-complex-function-statement

6.21.4.3 Syntax diagram


<specify complex function statement> 6.21.4
<fstatno> SPECIFY , COMPLEX FUNCTION ,
1
<complex function mnemonic>
SPEC
<external complex function specification>
,
<function characteristic mnemonic>
<real characteristic subfield> 14.2
<modifier descriptor>
<proportionality subfield> 14.46
*
$

NOTE

* Bypass required for SUM.

6.21.4.4 Rules

1. Only one complex function can be specified in a <define complex signal structure>.

2. The <external complex function specification> option is used when the type and specification of the complex function of a COMPLEX SIGNAL are defined in an existing specification. This option shall be used if components of the COMPLEX SIGNAL refer to external specifications in the <specify component signal statement> (see 6.21.5).

3. The <complex function mnemonic> defines the type of function applied to the component signals to generate the COMPLEX SIGNAL. The types of functions and their defining characteristics for this statement are listed in the following two tables:

TABLE 6-1.

<complex function mnemonic> descriptions

Description	<complex function mnemonic>
Amplitude modulation	AM
Frequency modulation	FM
Lower single side band modulation	LSSB
Mixing	MIX
Phase lock loop	PHASE-LOCK
Phase modulation	PM
Pulse amplitude modulation	PAM
Pulse modulation	PULSE
Summing	SUM
Suppressed carrier modulation	SUPP-CAR
Sweep	SWEEP
Upper single side band modulation	USSB
Vector modulation	VECTOR

The applicable <function characteristic mnemonic>s are then as shown in table 6.2:

Table 6-2. <function characteristics mnemonics>

<function mnemonic>	<Function characteristic mnemonic>	Type usage (as defined in clause 16)		Quantity
AM	MOD-INDEX	R	SR-	Ratio
	MOD-FUNCTION	MD	SR-	(none)
	AM-SENSITIVITY	R	SR-	PC/voltage
FM	FREQ-DEV	R	SR-	Frequency
	FM-SENSITIVITY	R	SR-	Frequency/voltage
	MOD-INDEX	R	SR-	Ratio
LSSB	MOD-INDEX	R	SR-	Ratio
	MOD-FUNCTION	MD	SR-	(none)
	AM-SENSITIVITY	R	SR-	PC/voltage
	SUPPRESS-LEVEL	R	SR-	Ratio
MIX	SUMMATION	MD	SR-	(none)
	DIFFERENCE	MD	SR-	(none)
PAM	MOD-INDEX	R	SR-	Ratio
	MOD-FUNCTION	MD	SR-	(none)
	AM-SENSITIVITY	R	SR-	PC/voltage
PHASE-LOCK	PHASE-ANGLE	R	SR-	Angle, plane
	PHASE-JIT	R	SR-	Angle, plane, or ratio
PM	PHASE-SENSITIVITY	R	SR-	Angle/voltage
	MOD-INDEX	R	SR-	Ratio
	PHASE-DEV	R	SR-	Angle
PULSE	PULSE-SENSITIVITY	R	SR-	Voltage
	MOD-INDEX	R	SR-	Ratio
SUM	(none)			
SUPP-CAR	MOD-INDEX	R	SR-	Ratio
	MOD-FUNCTION	MD	SR-	(none)
	AM-SENSITIVITY	R	SR-	PC/voltage
	SUPRESS-LEVEL	R	SR-	Ratio
SWEEP	<Proportionality subfield> 14.46			
USSB	MOD-INDEX	R	SR-	Ratio
	MOD-FUNCTION	MD	SR-	(none)
	AM-SENSITIVITY	R	SR-	PC/voltage
	SUPRESS-LEVEL	R	SR-	Ratio
VECTOR	VECTOR-TYPE	MD	SR-	BPSK, QPSK, 8PSK, OQPSK, OQAM, 16QAM, 64QAM, 256QAM, CPFSK, BFSK, OR MFSK

4. The <function mnemonic>s and the <function characteristic mnemonic>s are defined in 17.5.

5. When MOD-FUNCTION is used, the modifier descriptor can be LIN or LOG.

6.21.4.5 Notes and examples

Figure 6-2 shows the basic signal models referenced by the different function types of this statement:

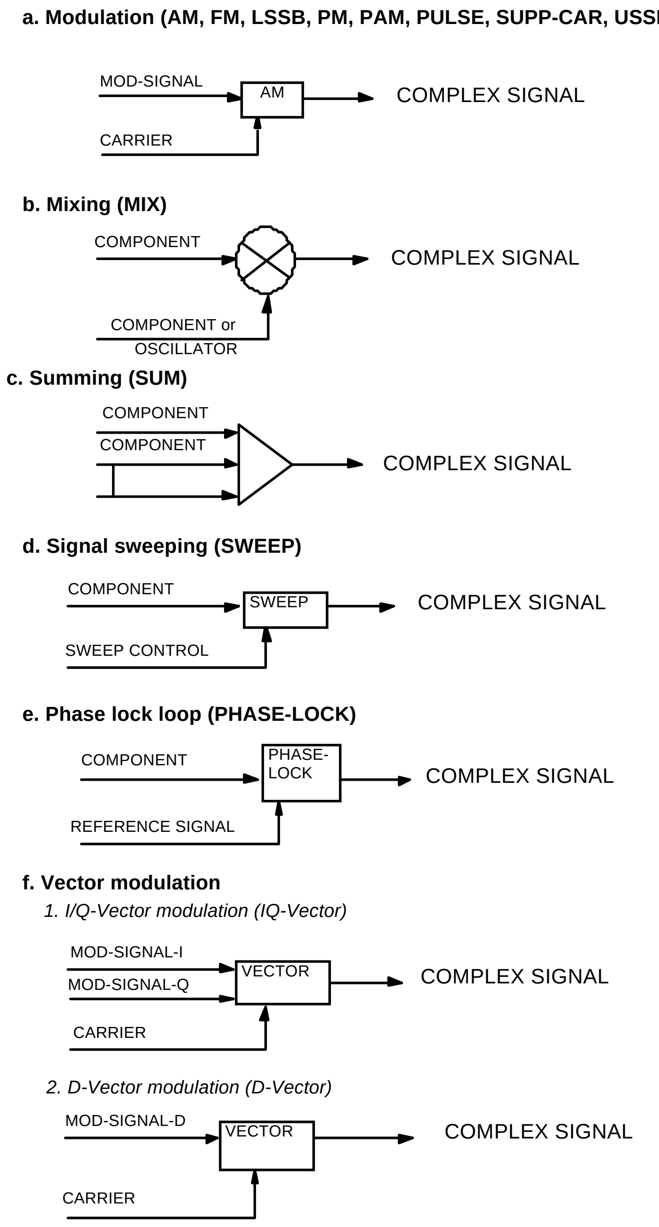


Figure 6-2. Basic signal models

6.21.5 SPECIFY COMPONENT SIGNAL statement

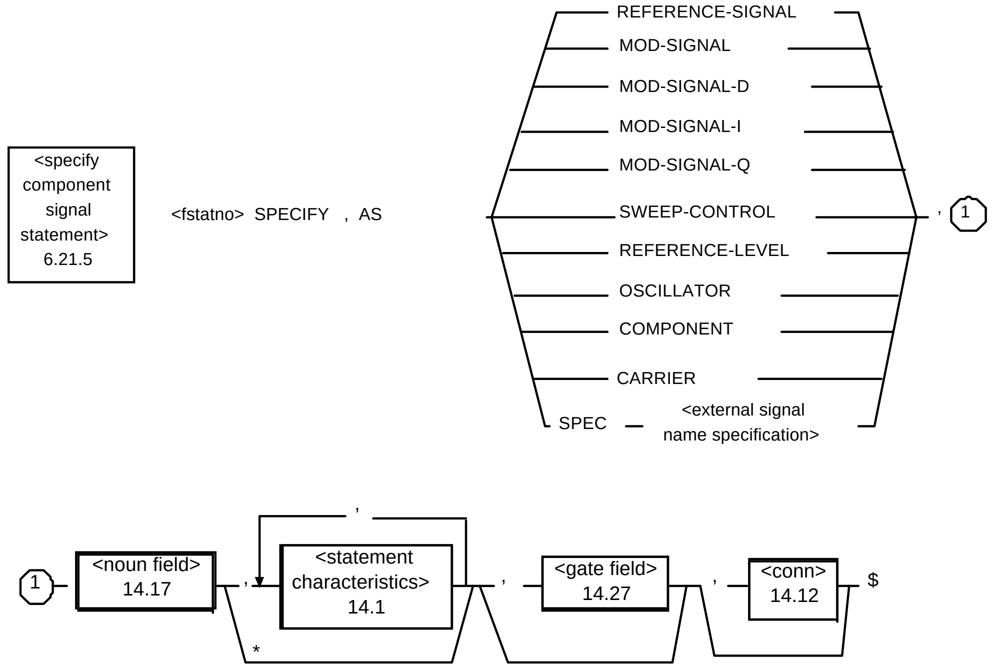
6.21.5.1 Function

This statement specifies an input signal to a COMPLEX FUNCTION and its role in the creation of a COMPLEX SIGNAL.

6.21.5.2 Formal syntax

Reference specify-component-signal-statement

6.21.5.3 Syntax diagram



NOTE

* Bypass used when the noun is a COMPLEX SIGNAL.

6.21.5.4 Rules

1. The <noun> and <statement characteristics> may be of any digital (LOGIC DATA) or analog signal type listed in clause 16 appropriate to the COMPLEX SIGNAL's waveform (carrier, modulating signal, oscillator, or other component signals). The <noun> field may reference a separately defined COMPLEX SIGNAL as a component of this COMPLEX SIGNAL by using the <noun field> entry COMPLEX SIGNAL <complex signal>.

2. The <external signal name specification> branch permits the identification of other standard signal types as specified in an external specification. This external specification shall be identified in the SPECIFY, COMPLEX FUNCTION statement.

	SWEEP-CONTROL	REFERENCE LEVEL	CARRIER	MOD-SIGNAL	OSCIL-LATOR	COM-PONENT	REFERENCE SIGNAL
AM		P	X	X			
FM		P	X	X			
LSSB		P	X	X			
MIX		P			P	X	
PAM		P	X	X			
PHASE-LOCK						X	X
PM		P	X	X			
PULSE		P	X	X			
SUM		P				X	
SUPP -CAR		P	X	X			
SWEEP	X	P				X	
USSB		P	X	X			
VECTOR		P	X	X			

P-Permitted

X-Required

If the complex function is a new one introduced with the SPEC subfield, the set of specification types shall be only those listed in the function definition.

6.21.6 SPECIFY SIGNAL CONDITIONING statement

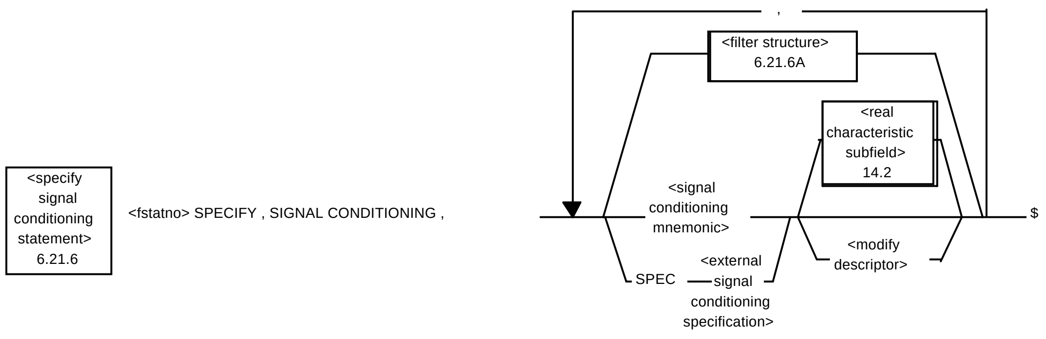
6.21.6.1 Function

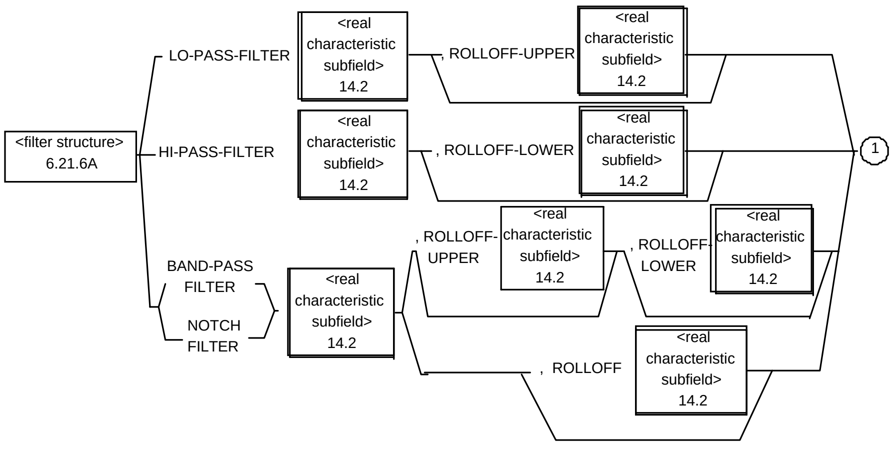
This statement establishes the characteristics of any signal conditioning or modifying to be applied to the resultant COMPLEX SIGNAL generated in this structure.

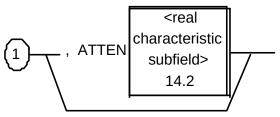
6.21.6.2 Formal syntax

Reference specify-signal-conditioning-statement

6.21.6.3 Syntax diagram


<specify signal conditioning statement> 6.21.6
<fstatno> SPECIFY , SIGNAL CONDITIONING ,
,
<filter structure> 6.21.6A
<real characteristic subfield> 14.2
<signal conditioning mnemonic>
SPEC
<external signal conditioning specification>
<modify descriptor>
$





6.21.6.4 Rules

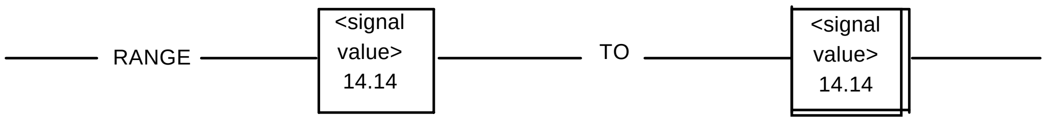
1. The <signal conditioning mnemonic> describes the type of conditioning to be applied to the COMPLEX SIGNAL. The types of conditioning and their legal characteristics for this subfield are listed in the following table:

signal conditioning characteristic	<signal conditioning mnemonic>	Type	Usage	Quantity
Attenuation	ATTEN	R	SR-	Ratio
De-Emphasis	DE-EMPHASIS	R	SR-	Time
Frequency divider	DIVIDE	I	SR-	Ratio
Gain	GAIN	R	SR-	Ratio
Isolation	ISOLATION	R	SR-	Ratio
Phase shift	PHASE-SHIFT	R	SR-	Angle
Pre-Emphasis	PRE-EMPHASIS	R	SR-	Time

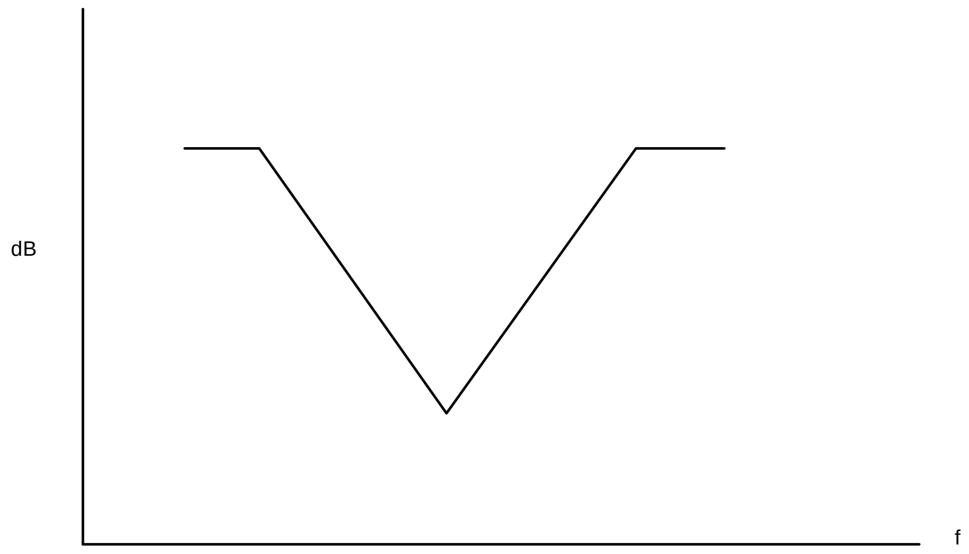
The <modifier descriptor> as well as the <real characteristic subfield> may be used with the <external signal conditioning specification>.

2. All of the ROLLOFF parametres are in dB/octave or W/Hz.

3. The <real characteristic subfield> for BAND-PASS-FILTER is limited to the following:



4. A notch filter with a signal frequency specified appears as shown below in the frequency domain:



The center frequency of the notch is specified by NOTCH-FILTER in Hz. ROLLOFF in dB/octave describes the rate of change of the sides. They are assumed to be the same above and below the NOTCH-FILTER frequency. If they are not, then ROLLOFF-UPPER and ROLLOFF-LOWER may be used. The depth of attenuation shall be specified by ATTEN in dBm. A NOTCH-FILTER with a frequency RANGE describes a band elimination filter.

5. The <external signal conditioning specification> branch permits the identification of other signal conditioning parametres as specified in an external specification.

6. The keywords are defined in 17.6.

6.21.7 END COMPLEX SIGNAL statement

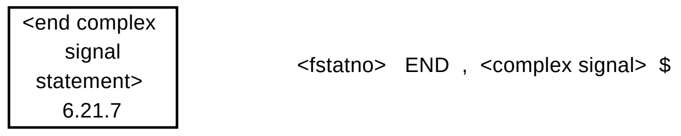
6.21.7.1 Function

This statement terminates a <define complex signal structure>.

6.21.7.2 Formal syntax

Reference end-complex-signal-statement

6.21.7.3 Syntax diagram



6.22 DEFINE EXCHANGE-CONFIGURATION statement

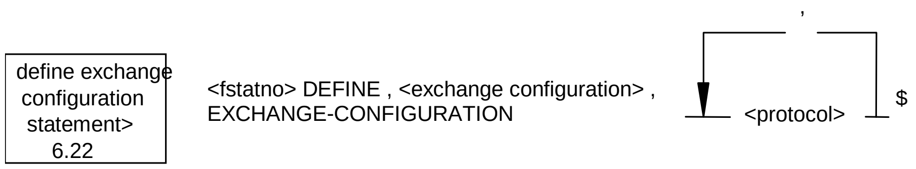
6.22.1 Function

The <define exchange configuration statement> defines a set of databus protocols that are employed simultaneously in a <do exchange statement>. There is a <define exchange configuration statement> for each combination of protocols.

6.22.2 Formal syntax

Reference define-exchange-configuration-statement

6.22.3 Syntax diagram



define exchange configuration statement> 6.22
<fstatno> DEFINE , <exchange configuration> , EXCHANGE-CONFIGURATION
,
<protocol>
$

6.22.4 Rules

1. Every <do exchange statement>, <enable exchange configuration statement>, <connect exchange configuration statement>, <disconnect exchange configuration statement>, <update exchange configuration statement>, <fetch exchange configuration statement>, and <disable exchange configuration statement> in a <complete atlas program structure> shall reference an <exchange configuration>.

2. A second <define exchange configuration statement> is necessary if two independent <do exchange statement>s have to be active simultaneously. Simultaneous <do exchange statement>s shall not reference the same <exchange configuration statement>.

3. If there are two or more <define exchange configuration statement>s some <protocol>s may appear in more than one. If they do, it is an error for a <protocol> to be active in two <exchange configuration>s concurrently. If a <protocol> appears in more than one <define exchange configuration statement> it shall not be active in more than one <exchange configuration> simultaneously.

4. A <protocol> becomes active when it is referenced in an <enable exchange configuration statement> and remains active until the associated <exchange configuration> is referenced in a <disable exchange configuration statement>.

5. The GLOBAL attribute may only appear in the <program preamble structure> and the <exchange configuration> shall then be unique to the entire <complete atlas program structure>.

6. The EXTERNAL attribute may only appear in a <module preamble structure>. There shall be a corresponding GLOBAL <exchange configuration> in the <program preamble structure> and the list of protocols shall be identical in the two definitions.

7. Each <protocol> shall be unique and refer to an <establish protocol statement> within the same <atlas program structure> or <atlas module structure>. The use of the GLOBAL and EXTERNAL attribute shall be the same for the <define exchange configuration statement> and each of its <establish protocol statement>s.

7.0 Procedural structure

7.1 <main procedural structure>

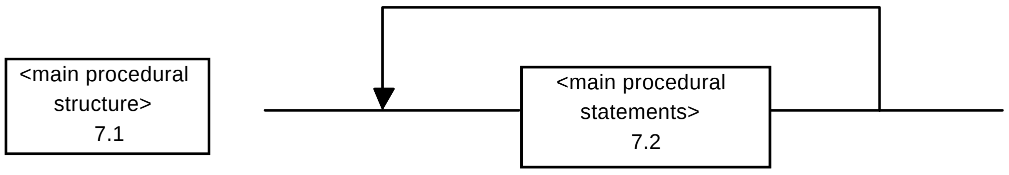
7.1.1 Function

The <main procedural structure> is a series of one or more <main procedural statements>.

7.1.2 Formal syntax

Reference main procedural structure

7.1.3 Syntax diagram



7.2 <main procedural statements>

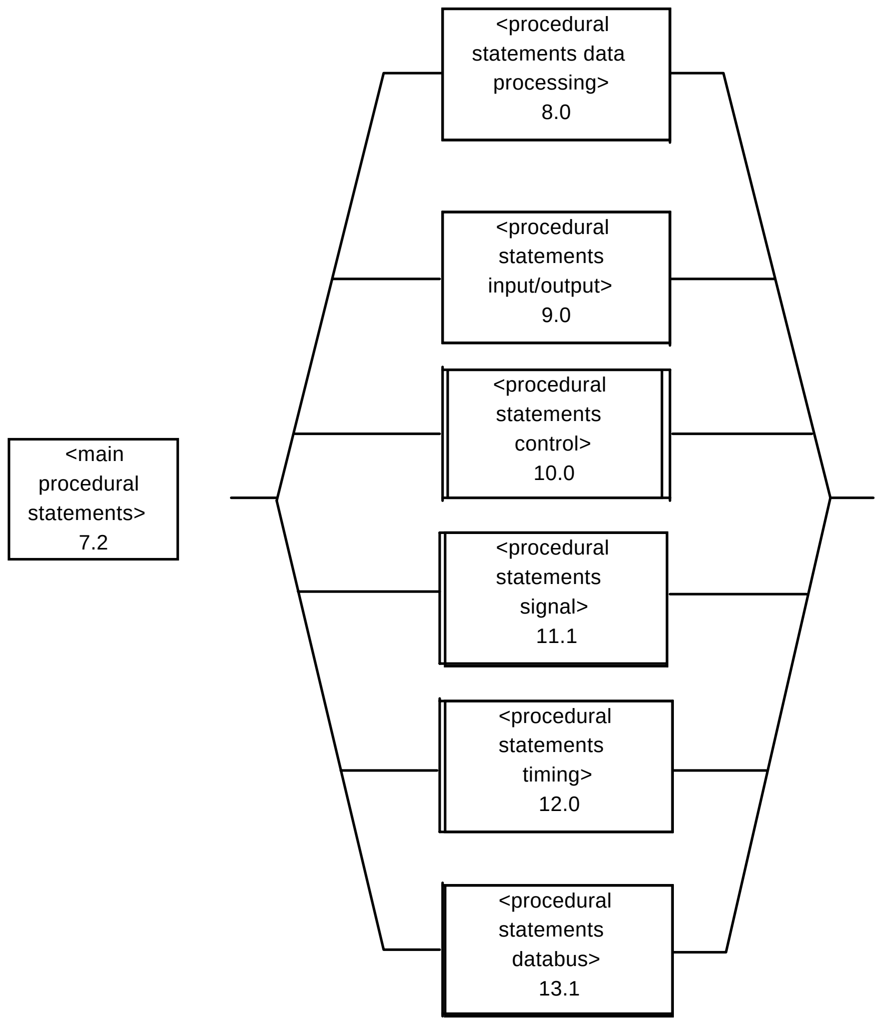
7.2.1 Function

The <main procedural statements> are a series of statements or structures, each of which describes a portion of the required test that shall be completed prior to the next statement or structure. Each statement or structure implies an instruction to proceed to the following statement or structure immediately after completion of the present one unless directed otherwise in a branching statement. The referenced paragraph numbers in the syntax diagram contain the detailed definition of each of the classes of C/ATLAS procedural statements.

7.2.2 Formal syntax

Reference main procedural statements

7.2.3 Syntax diagram


<main procedural statements> 7.2
<procedural statements data processing> 8.0
<procedural statements input/output> 9.0
<procedural statements control> 10.0
<procedural statements signal> 11.1
<procedural statements timing> 12.0
<procedural statements databus> 13.1

8.0 Procedural statements, data processing

a) <procedural statements data processing>

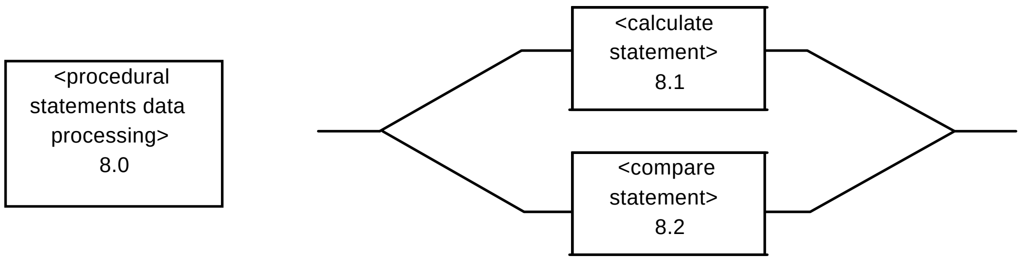
b) Function

Data processing statements provide the capability to perform calculations, compare test values with specified limits, and retain test results expressed in GO, NOGO, HI, and LO flag status for subsequent use in the test procedure.

c) Formal syntax

Reference procedural statements data processing

d) Syntax diagram



8.1 CALCULATE statement

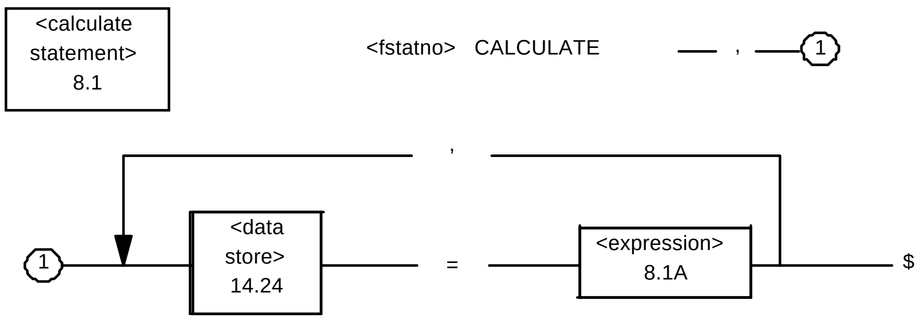
8.1.1 Function

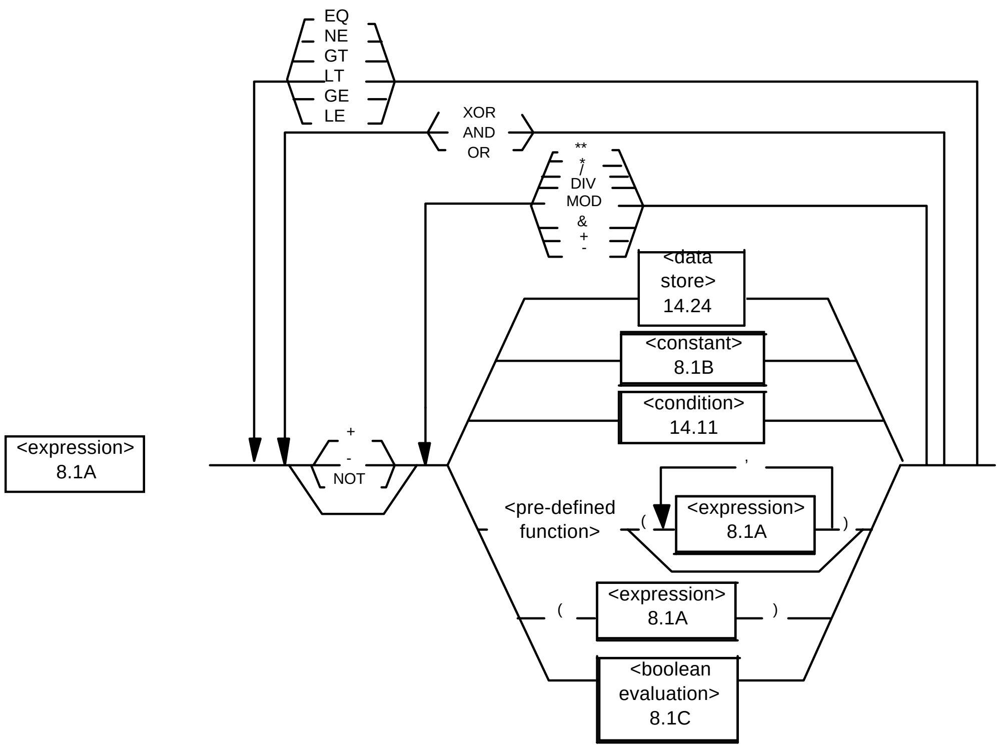
To evaluate an <expression> on the right side of an "=" and assign the value of that <expression> to the variable on the left. One or more such evaluations and assignments may be made within one CALCULATE statement.

8.1.2 Formal syntax

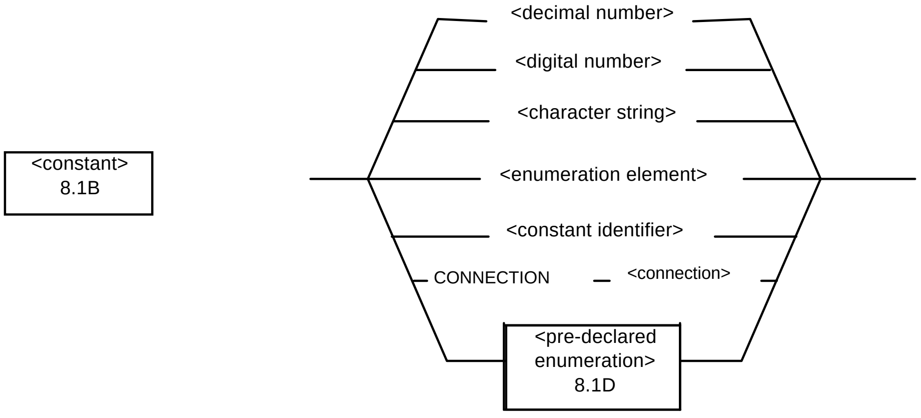
Reference calculate statement

8.1.3 Syntax diagram

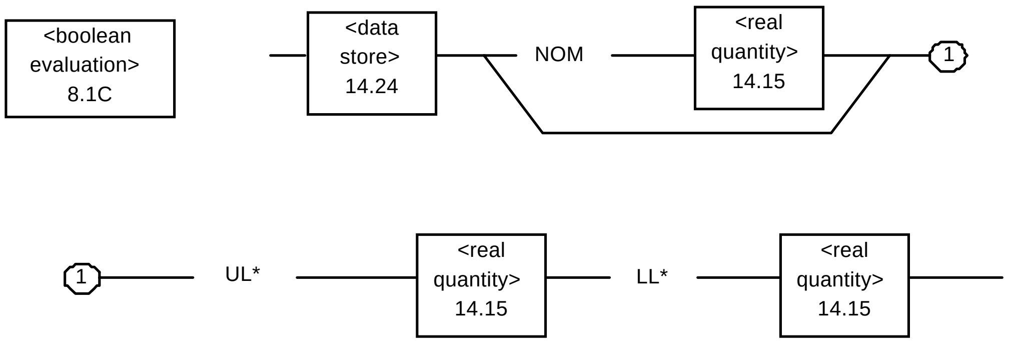




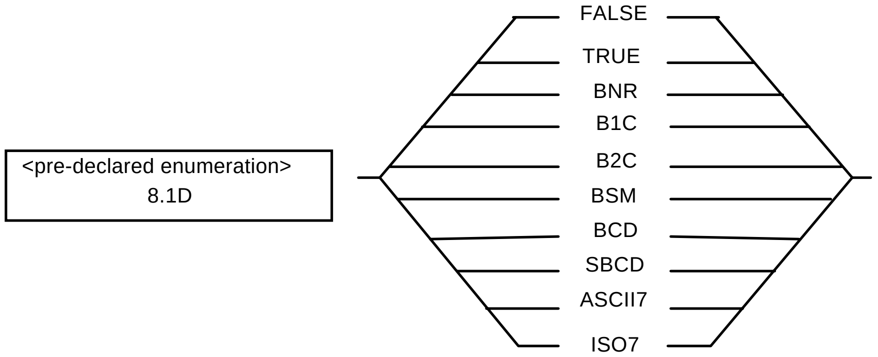
<expression>
8.1A
EQ
NE
GT
LT
GE
LE
XOR
AND
OR
**
*
/
DIV
MOD
&
+
-
+
-
NOT
<data store>
14.24
<constant>
8.1B
<condition>
14.11
<pre-defined function>
(
,
<expression>
8.1A
)
(
<expression>
8.1A
)
<boolean evaluation>
8.1C



<constant>
8.1B
<decimal number>
<digital number>
<character string>
<enumeration element>
<constant identifier>
CONNECTION
<connection>
<pre-declared enumeration>
8.1D



NOTE
* UL and LL may be interchanged



8.1.4 Rules for the CALCULATE statement

8.1.4.1 General

1. The <expression> can be of any <type> except FILE.

2. The <data store> to which assignment is made shall be assignment compatible with the <expression>.

8.1.4.2 General rules for <expression>

1. An <expression> may contain literal constants, labeled constants, variables, <pre-defined function>s, and operators.

2. The operators that are permitted within an <expression> are tabulated in table 8-1. This table also specifies, for each operator, the precedence level, the number and types of operands, and the type of the resultant <expression>.

3. Each <predefined function> operates upon a set of parametres of predetermined types and yields a result of predetermined type. Table 8-2 tabulates the set of <predefined function>s.

4. For nested parenthesized expressions, the innermost expression is evaluated first.

5. At any one depth of parentheses the precedence of evaluation is given by the level of the operators tabulated in 8.1.4.3. Operators are evaluated in the order of their increasing level numbers (i.e., operators at precedence level 1 are evaluated before operators at precedence level 2, etc.).

6. For operators of equal precedence level, evaluation is from left to right.

7. Signed DECIMAL and signed LONG_DECIMAL <constant>s are not permitted.

8. A <condition> is treated as a special BOOLEAN operand that is established and maintained by the system.

8.1.4.3 Rules associated with permitted operations

1. Table 8-1 lists the permitted operations. Each operation is located in a precedence level. Each operation has an operator with permitted types of operands and associated result types. Operations between compatible parts of dissimilar structures are allowed.

2. LONG_DECIMAL is allowed everywhere that DECIMAL is allowed. An operand of <type> LONG_DECIMAL always results in LONG_DECIMAL.

The subsequent rules apply only where referenced from table 8-1.

3. If an operation on a STRING OF BIT is applied to strings of unequal current dynamic length, the shorter is treated as though it had extra B'0's at the left hand (MSB) end. The dynamic length of the resultant is equal to that of the longer operand.

4. The resultant of the "&" operator is the concatenation of the operands in the order written. The dynamic length of the resultant is the sum of the dynamic lengths of the operands.

5. The operands shall be of identical types, except as follows:

a) INTEGER, DECIMAL, and LONG_DECIMAL can be compared with each other, and if these types are mixed, INTEGER operands will be converted to DECIMAL and DECIMAL to LONG_DECIMAL before the comparison is made.

b) STRING OF CHAR can be compared with CHAR. In this case the CHAR is treated as a unit length STRING OF CHAR.

c) STRING OF BIT can be compared with BIT. In this case the BIT is treated as a unit length STRING OF BIT.

If the operands are RECORDs, they shall be of identical structure and have identical components.

If the operands are ARRAYs, the number of dimensions, the number of components within each dimension, and the type of the components shall be identical.

If the operands are <enumeration element>s, they shall be of the same ENUMERATION type.

6. Comparison of strings proceeds as follows:

a) If the strings are of unequal current dynamic length, the shorter STRING OF BIT is treated as though it were padded on the left with B'0's and the shorter STRING OF CHAR is treated as though it were padded on the right with ASCII NUL characters.

b) Comparison proceeds from left to right until either corresponding unequal values are found or until the longer dynamic length is reached. If an unequal position value is found, the relationship between the STRINGs is determined by the relationship of the unequal BITs or CHARs. If STRINGs OF CHAR are being compared and no inequality is found, the relationship between the strings is determined by the relationship between their current dynamic lengths. For STRING OF BIT, dynamic length is not considered in the comparison.

c) The relationship of characters is determined by the ordinal values of the characters in the ISO or ASCII character set to which they belong.

7. The NOT operator does not alter the dynamic length of a STRING OF BIT.

8. The exponentiation operation is defined over the following ranges:

For INTEGER y: -_ < x < _, -_ < y < _ (x not = 0)

For DECIMAL,

and LONG_DECIMAL y: 0 < x < _, -_ < y < _

For INTEGER, DECIMAL,

and LONG_DECIMAL y: x = 0, 0 _ y < _

9. ENUMERATION includes the <pre-declared type>s, 6.3C.

Table 8-1. Permitted operations

Prec level	Operation	Operand type structure or base types	Action	Result type	Resultant
1	NOTx	BOOLEAN	logical NOT	BOOLEAN	TRUE if x is FALSE, FALSE if x is TRUE
		BIT	logical NOT	BIT	complement of x
		STRING OF BIT	logical NOT	STRING OF BIT	Bit-by-bit complement of x. See rule 7
1	-x	INTEGER	unary-	INTEGER	negation of x
		DECIMAL	unary-	DECIMAL	
		LONG_ DECIMAL	unary-	LONG_ DECIMAL	
1	+x	INTEGER	unary+	INTEGER	x
		DECIMAL	unary+	DECIMAL	
		LONG_ DECIMAL	unary+	LONG_ DECIMAL	
2	x XOR y	BOOLEAN	exclusive OR	BOOLEAN	((x AND NOT y) OR (y AND NOT x))
		BIT	exclusive OR	BIT	
		STRING OF BIT	exclusive OR	STRING OF BIT	
2	x**y	INTEGER	exponentiate	INTEGER	x to y power See rule 8
		INTEGER and DECIMAL	exponentiate	DECIMAL	
		INTEGER or DECIMAL or LONG_ DECIMAL and LONG_ DECIMAL	exponentiate	LONG_ DECIMAL	

Table 8-1. Permitted operations (continued)

Prec level	Operation	Operand type structure or base types	Action	Result type	Resultant
3	x*y	INTEGER	multiply	INTEGER	product of x and y
		DECIMAL	multiply	DECIMAL	
		INTEGER and DECIMAL	multiply	DECIMAL	
		INTEGER and LONG_ DECIMAL or DECIMAL and LONG_ DECIMAL	multiply	LONG_ DECIMAL	
3	x/y	INTEGER	division	DECIMAL	quotient of x and y
		DECIMAL	division	DECIMAL	
		INTEGER and DECIMAL	division	DECIMAL	
		INTEGER or DECIMAL or LONG_ DECIMAL and LONG_ DECIMAL	division	LONG_ DECIMAL	
3	x DIV y	INTEGER	division	INTEGER	INT(x/y)
3	x MOD y	INTEGER	modulo	INTEGER	Remainder of (x/y)
3	x AND y	BOOLEAN	logical AND	BOOLEAN	TRUE if both x and y are TRUE else FALSE
		BIT	logical AND	BIT	B'1' if both x and y are B'1' else B'0'
		STRING OF BIT	logical AND	STRING OF BIT	Each bit as for BIT. See rule 3
4	x & y	STRING OF CHAR or CHAR	concatenate	STRING OF CHAR	See rule 4
		STRING OF BIT or BIT	concatenate	STRING OF BIT	

Table 8-1. Permitted operations (continued)

Prec level	Operation	Operand type structure or base types	Action	Result type	Resultant
4	x + y	INTEGER	addition	INTEGER	Sum of x and y
		DECIMAL	addition	DECIMAL	
		INTEGER and DECIMAL	addition	DECIMAL	
		INTEGER or DECIMAL or LONG_ DECIMAL and LONG_ DECIMAL	addition	LONG_DECIMAL	
4	x - y	INTEGER	subtraction	INTEGER	difference of x and y
		DECIMAL	subtraction	DECIMAL	
		INTEGER and DECIMAL	subtraction	DECIMAL	
		INTEGER or DECIMAL or LONG_ DECIMAL and LONG_ DECIMAL	subtraction	LONG_DECIMAL	
4	x OR y	BOOLEAN	logical OR	BOOLEAN	TRUE if x or y are TRUE else FALSE
		BIT	logical OR	BIT	B'1' if x or y are B'1' else B'0'
		STRING OF BIT	logical OR	STRING OF BIT	Each bit as for BIT. See rule 3
5	x EQ y	All except FILE See rule 5 and 6	equality	BOOLEAN	TRUE for identical x and y else FALSE
5	x NE y	All except FILE See rule 5 and 6	inequality	BOOLEAN	FALSE for identical x and y else TRUE
5	x GT y	INTEGER, CHAR, DECIMAL, LONG_ DECIMAL, STRING OF BIT, STRING OF CHAR, BIT, ENUMERATION See rule 5, 6 and 9	greater than	BOOLEAN	TRUE for x greater than y else FALSE

Table 8-1. Permitted operations (continued)

Prec level	Operation	Operand type structure or base types	Action	Result type	Resultant
5	x LT y	as for GT	less than	BOOLEAN	TRUE for x less than y else FALSE
5	x GE y	as for GT	GT or EQ	BOOLEAN	TRUE for (x GT y) or (x EQ y) else FALSE
5	x LE y	as for GT	LT or EQ	BOOLEAN	TRUE for (x LT y) or (x EQ y) else FALSE

8.1.4.4 Rules associated with <predefined function>s

1. Table 8-2 lists the <predefined function>s. The number and permitted types of parametre for each function are listed with the associated result <type>.

The subsequent rules apply only where referenced from table 8-2.

2. For STRING OF CHAR, each CHAR is numbered starting from the left with number 1. For STRING OF BIT, each BIT is numbered starting from the right with number 1, which is the least significant bit.

a) The value of LOCN(x,y) is the first position within y for which the strings match for the LEN(x).

b) For COPY, if z > (LEN(x) - y + 1), then only (LEN(x) - y + 1) CHARs or BITs are copied. If (LEN(x) - y + 1) < 1, then no copying results.

c) For DELETE, 1 <= p <= LEN(s) and if w > (LEN(s) - p + 1), then the function returns a copy of s with only (LEN(s) - p + 1) CHARs or BITs deleted.

d) For INSERT, 1 <= p <= LEN(t). The string s is inserted starting at position p + 1.

3. For the function DIG, w shall be greater than or equal to the value k as shown below.

BNR: k is the smallest number such that 2 ** k > ABS(y).

B1C, B2C and BSM: k is the smallest number such that

2 ** (k-1) > ABS (y).

BCD: k is four times the number of digits in INT(y) and w and f shall be a multiple of four.

SBCD: k is one plus four times the number of digits in INT(y), w shall be one plus a multiple of four, and f shall be a multiple of four.

If w exceeds the above value, then extra bits are inserted as follows:

BNR and BCD: Leading B'0's are inserted.

B1C and B2C: The first sign bit is replicated as required.

BSM and SBCD: B'0's are inserted after the sign bit.

If f is too small to represent the precision of the fractional part of y, then y is truncated.

The resultant STRING OF BIT is of length w + f.

Examples of the DIG function are:

<function>	y = 6	y = 3	y = 1.5	y = -1.5	y=-3	
DIG (BNR,y,3,3)	110000	011000	001100	ILLEGAL		ILLEGAL
DIG (B2C,y,3,3)	ILLEGAL		011000	001100	110100	101000
DIG (B1C,y,3,3)	ILLEGAL		011000	001100	110011	100111
DIG (BSM,y,3,3)	ILLEGAL		011000	001100	101100	111000
DIG (BCD,y,4,4)	01100000		00110000		00010101	
ILLEGAL	ILLEGAL					
DIG (SBCD,y,5,4)	001100000		000110000		000010101	
100010101	100110000					

4. For the DEC function, w shall be less than or equal to the dynamic length of y. Bits in excess of w are considered to be fractional.

5. The result of SUCC(x), where ORD(x) is the last of a range of values, is undefined.

6. The result of PRED(x), where ORD(x) is the first of a range of values, is undefined.

7. For the DATE function, the returned STRING OF CHAR will always contain 15 characters in the format YYMMDDhhmmsssss where:

YY represents two digits (00-99) indicating the year,

MM represents two digits (01-12) indicating the month in the year,

DD represents two digits (01-31) indicating the day in the month,

hh represents two digits (00-23) representing the hour in the day,

mm represents two digits (00-59) indicating the minute in the hour,

sssss represents five digits (00000-59999) indicating the millisecond in the minute.

8. The functions ROTATE, SHIFT, and A-SHIFT operate over the dynamic length of a STRING OF BIT.

Table 8-2. <predefined function>s

Function	Parametre <type>	Action	Result<type>	Resultant
EOF(x)	FILE	test for end of file	BOOLEAN	TRUE if at end of file else FALSE
ODD(x)	INTEGER or BIT	odd	BOOLEAN	For INTEGER: TRUE if x is odd else FALSE For BIT: TRUE if BIT is a 1 else FALSE
ORD(x)	ENUMERATION, CONNECTION, or CHAR	ordinal	INTEGER	Ordinal position where 0 is the first in permitted range of values of x.
LEN(x)	STRING OF BIT or STRING OF CHAR	length	INTEGER	Current dynamic length of string x, 0 if empty
LOCN (x,y)	STRING OF BIT or STRING OF CHAR	location	INTEGER	Position of the first occurrence of string x within string y, 0 if not found. See rule 2
COUNT (x,y)	x is a BIT or CHAR and y is a compatible string	count occurrences of x in y	INTEGER	The number of occurrences of BIT or CHAR in the string y.
SIZE(f)	f is a FILE	size	INTEGER	The number of bytes in an UNTYPED file, the number of CHAR in a TEXT file or the number of data items in a typed file. Zero for an empty file.
SUCC(x)	ENUMERATION, CONNECTION or CHAR	successor	same as x	Next sequential value of x. See rule 5
PRED(x)	ENUMERATION, CONNECTION or CHAR	predecessor	same as x	Previous sequential value of x. See rule 6
DATE	none	date-time	STRING OF CHAR	[YYMMDDhhmmssss] See rule 7
CHAR (x,y)	x is CHAR_ CLASS y is STRING OF BIT	conversion to character	CHAR	Character represented in CHAR_CLASS by STRING OF BIT
BITS (x,y)	x is CHAR_ CLASS y is CHAR	conversion of character to bits	STRING OF BIT	STRING OF BIT representing character y in CHAR_CLASS x
COPY (x,y,z)	y and z are INTEGER and x is STRING OF BIT or STRING OF CHAR	copy substring	same as x	Copy of STRING x from location y for z elements. See rule 2
ROTATE (x,y)	x is a STRING OF BIT and y is an INTEGER	rotate	STRING OF BIT	x rotated y places. Rotation is to the right (toward the LSB) for positive values of y and left for negative values of y. See rule 8
SHIFT (x,y)	as for ROTATE	shift	STRING OF BIT	x shifted y places. Shifting is to the right (toward the LSB) for positive values of y and left for negative values of y. Bits shifted off the end are lost. Vacated positions set to B'0'. See rule 8
A-SHIFT (x,y)	as for ROTATE	arithmetic shift	STRING OF BIT	As SHIFT except that an assumed sign bit (at MSB) is not shifted and for a right shift it is replicated in the vacated bit positions. See rule 8

Table 8-2. <predefined function>s (continued)

Function	Parametre <type>	Action	Result<type>	Resultant
DIG (x,y,w,f)	x is DIG_CLASS y is INTEGER, LONG_ DECIMAL, or DECIMAL, w and f are INTEGER	string conversion	STRING OF BIT	A STRING OF BIT representation of y in BNR, B1C, etc. form allowing w bits for the whole part and f bits for the fractional part. For signed number types, w includes the sign bit. See rule 3
DEC (x,y,w)	x is DIG_CLASS y is a STRING OF BIT, w is INTEGER	float	LONG_DECIMAL	A DECIMAL representation of y in BNR, B1C, etc. form, where w bits is the whole part including a sign bit for signed number types. See rule 4
INT(x)	DECIMAL	integer part	INTEGER	x truncated. The fractional part is ignored.
ROUND (x)	DECIMAL	rounding	INTEGER	x rounded. INT (x+0.5) for x GT 0 else INT(x-0.5)
TRIM(x)	LONG_ DECIMAL	conversion to decimal	DECIMAL	x is reduced from LONG_ DECIMAL. The least significant digits are lost.
ABS(x)	DECIMAL, LONG_ DECIMAL or INTEGER	absolute	same as x	x if x is GE 0 else -x
SIN(x)*	DECIMAL, LONG_ DECIMAL or INTEGER. x in degrees	trig function	DECIMAL or LONG_DECIMAL	sine of x
COS(x)*	as for SIN	trig function	DECIMAL or LONG_DECIMAL	cosine of x
TAN(x)*	as for SIN	trig function	DECIMAL or LONG_DECIMAL	tangent of x
LN(x)*	DECIMAL, LONG_ DECIMAL or INTEGER x GT 0	natural log	DECIMAL or LONG_DECIMAL	log base e
LOG(x)*	as for LN	common log	DECIMAL or LONG_DECIMAL	log base 10
ALOG(x) *	DECIMAL, LONG_DECIMAL or INTEGER	antilog	DECIMAL or LONG_DECIMAL	number whose log base 10 equals x
ATAN(x) *	DECIMAL, LONG_ DECIMAL or INTEGER	angle of TAN	DECIMAL or LONG_DECIMAL	angle (in degrees) whose TAN is x
EXP(x)*	DECIMAL, LONG_ DECIMAL or INTEGER	e to power x	DECIMAL or LONG_DECIMAL	e power x
SQRT(x) *	DECIMAL ,LONG_ DECIMAL or INTEGER x GE 0	square root	DECIMAL or LONG_DECIMAL	square root of x
DELETE (s,p,w)	s is a STRING OF BIT or STRING OF CHAR, p and w are INTEGER	delete a substring	same as s	A copy of s with w elements deleted starting at position p. See rule 2

Table 8-2. <predefined function>s (continued)

Function	Parametre <type>	Action	Result<type>	Resultant
INSERT (s,t,p)	s and t are both of type STRING OF BIT or both of type STRING OF CHAR, p is INTEGER	Insert a string into a copy of another string	same as s and t	A copy of t with s inserted at position p+1. See rule 2
PE(x)	STRING OF BIT or BIT	Parity Even	BIT	B'1' if x has even parity else B'0'
PO(x)	STRING OF BIT or BIT	Parity Odd	BIT	B'1' if x has odd parity else B'0'

NOTE:

* If the parametre <type> is LONG_DECIMAL, the result <type> is LONG_DECIMAL. If the parametre <type> is INTEGER or DECIMAL, the result <type> is DECIMAL.

8.1.4.5 Rules for <constant>

1. For literal <constant>s, the type of the <constant> is determined by inspection of its textual form. The keyword CONNECTION is used to resolve ambiguities that <connection> would otherwise introduce.

2. Each constant has a specific type and may be used in expressions only as permitted by the type rules for variables. The constant type is determined by the form in which it is written as follows:

Type	Constant form
INTEGER	<unsigned integer number>
DECIMAL	decimal fixed-point number (15.2, item b) or decimal floating-point number (15.2, item c)
LONG_DECIMAL	<long decimal number> (see 15.2.5)
BIT	<digital number> specifying one bit
STRING OF BIT	<digital number> specifying zero or more bits
CHAR	<character string> specifying one character
STRING OF CHAR	<character string> specifying zero or more characters
ENUMERATION	<enumeration element>
CONNECTION	<connection>, COMMON, EARTH, or ATMOS preceded by the word CONNECTION

8.1.4.6 Rules for <boolean evaluation>

1. A <boolean evaluation> produces a BOOLEAN result.

2. The <condition>s of 14.11 are not affected by processing a <boolean evaluation>.

3. The types of the <data store> and the quantities following NOM, UL, and LL in a <boolean evaluation> shall be identical, except that INTEGER, DECIMAL, and LONG_DECIMAL may be compared with each other. If types are mixed, INTEGER operands will be converted to DECIMAL and DECIMAL to LONG_DECIMAL before the comparison is made.

4. The dimensions following NOM, UL, and LL in a <boolean evaluation> shall be identical.

8.2 COMPARE statement

8.2.1 Function

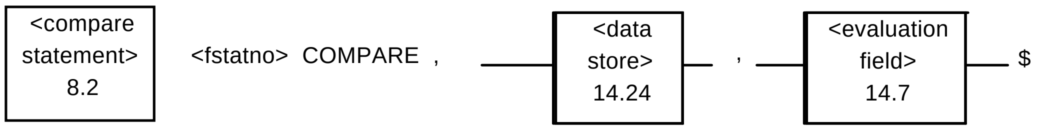
The function is to establish the state of HI, LO, GO, and NOGO conditions, in accordance with table 14-1, based on the relationship between a variable identifier and the specified limit(s).

8.2.2 Formal syntax

Reference compare statement

8.2.3 Syntax diagram

8.2.4 Rules



1. Following the verb in the <compare statement>, a <data store> is written to indicate the object of the evaluation. An evaluation field completes the statement.

2. The <compare statement> shall be confined to assist in setting the HI, LO, GO and NOGO flags needed to display or log failures. The <compare statement> shall not be used in place of the IF THEN ELSE structure which is considered as the only C/ATLAS branching facility that supports structured programming practice.

8.2.5 Notes and examples

```
 611600 COMPARE, 'OUT 23', NOM 23V UL 23.75V LL 22.25V $
 711711 COMPARE, 'INDEX', UL 'GAIN' LL 'APPLES' $
```

NOTE: The use of NOM (nominal) in the evaluation field is not required to accurately specify limits. However, this capability is an option for better understanding by the human reader.

9.0 Procedural statements, input/output

a) Input/output statements

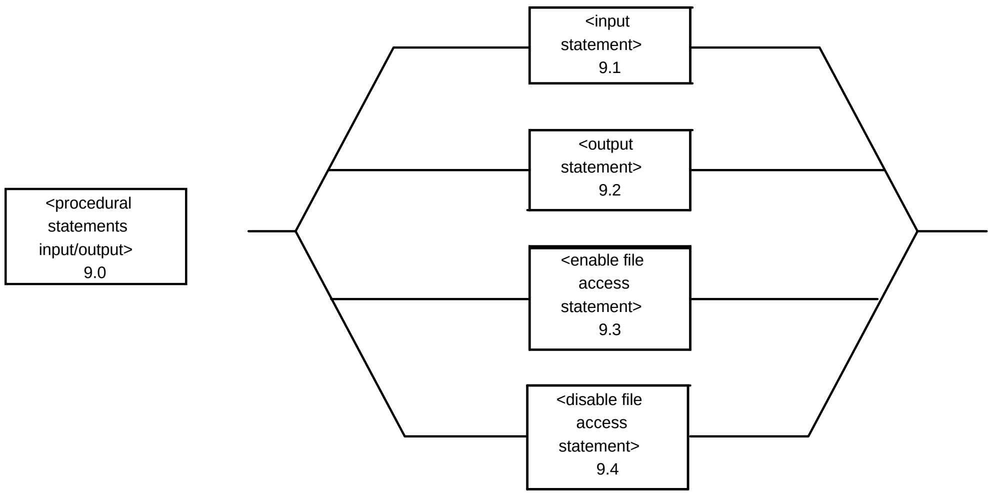
b) Function

Input and output procedural statements transfer data to and from a C/ATLAS program.

c) Formal syntax

Reference procedural statements input output

d) Syntax diagram


<procedural statements input/output> 9.0
<input statement> 9.1
<output statement> 9.2
<enable file access statement> 9.3
<disable file access statement> 9.4

9.1 INPUT statement

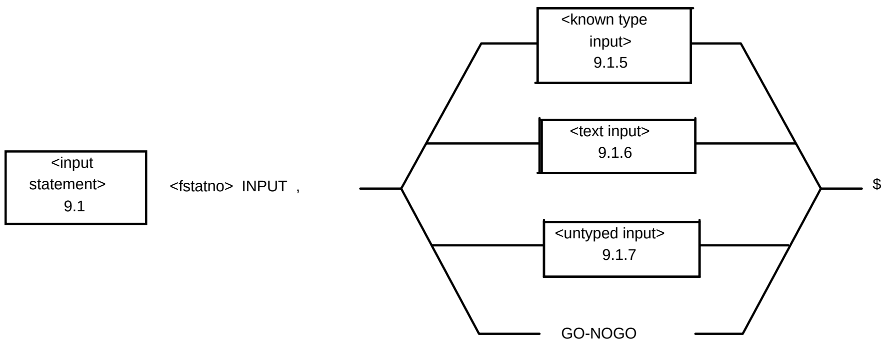
9.1.1 Function

The function is to transfer external non-signal data into a C/ATLAS program. C/ATLAS INPUT statements operate on virtual I/O resources.

9.1.2 Formal syntax

Reference input statement

9.1.3 Syntax diagram



9.1.4 Rules

1. GO/NOGO signifies that an operator response will establish new complementary states for the GO and NOGO condition identifiers.

9.1.5 <known type input>

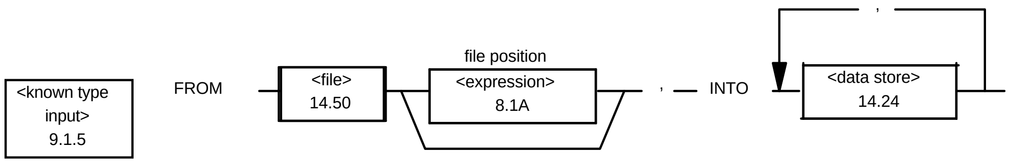
9.1.5.1 Function

This form of input transfers data items from a file to <data store>s of compatible type.

9.1.5.2 Formal syntax

Reference known type input

9.1.5.3 Syntax diagram



9.1.5.4 Rules

1. The <data store>s following INTO shall be assignment compatible with the components of the <file>.

2. The <file> shall have been ENABLEd previously.

3. If used, the file position <expression> preceding INTO shall evaluate to a non-negative INTEGER that is less than SIZE (<file>).

4. Reading of data items into variables is sequential in a forward direction from the current position. The current position is set as follows:

a) If an ENABLE is executed on the <file>, the current position is set to zero corresponding to the first item in the <file>.

b) If an INPUT statement containing a file position <expression> is executed, the current position is set to the value of the <expression> and input begins at that position. After execution of the INPUT statement, the current position will have advanced by the number of data elements needed to fill the specified INTO <data store>s.

c) If an INPUT statement not containing a file position <expression> is executed, input begins at the current position. After execution of the INPUT statement, the current position will have advanced by the number of data elements needed to fill the specified INTO <data store>s.

5. For sequential reads, the pre-defined function EOF(<file>) will return TRUE if invoked after the last data item of the file has been read.

9.1.6 <text input>

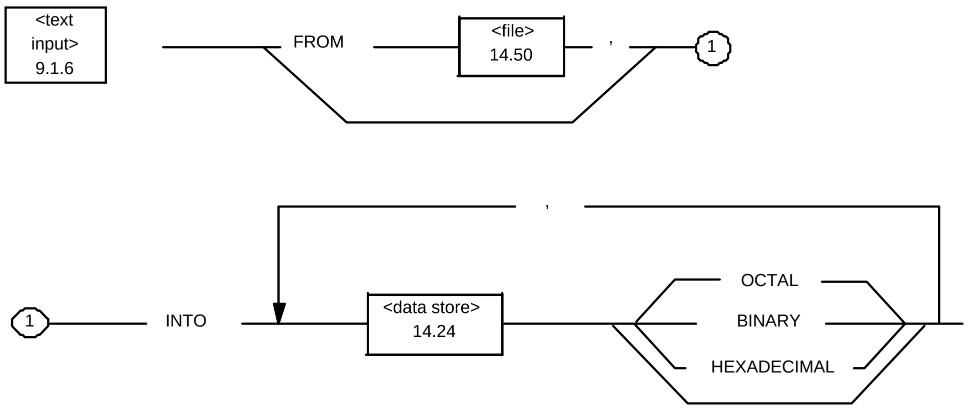
9.1.6.1 Function

This form of input takes characters from FILE OF TEXT and converts the input to match the destination VARIABLE type. A simple default form allows easy text input from the system dependent standard input device.

9.1.6.2 Formal syntax

Reference text input

9.1.6.3 Syntax diagram



9.1.6.4 Rules

1. If INTO directly follows INPUT, then input is to be from the system dependent standard input device.

2. The <file> label immediately following FROM shall have been declared as FILE OF TEXT and the associated file shall be ENABLEd for INPUT. The input is sequential forwards.

3. Table 9-1 specifies permitted <type>s of INTO <data store>s. Each permitted <type> has a set of legal characters. For each type, INPUT characters are ignored until the first legal character is seen. Type conversion of the input proceeds until an illegal character as defined by table 9-1 is found. This allows meaningful numeric data to be separated by blank, new line, comma, etc.

4. HEXADECIMAL, OCTAL, and BINARY can only be used when the INTO variable is a STRING OF BIT.

5. Data types for which automatic input conversion is specified and their legal characters are specified in table 9-1.

6. INPUT to a STRING OF BIT, unqualified by HEXADECIMAL, OCTAL, or BINARY involves a logical conversion from an unsigned INTEGER input.

7. When inputting to a STRING, the dynamic length is set as follows:

a) STRING OF CHAR : Number of characters input.

b) STRING OF BIT with HEXADECIMAL : Four times the number of characters input.

c) STRING OF BIT with OCTAL : Three times the number of characters input.

d) STRING OF BIT with BINARY : Number of characters input.

e) STRING OF BIT : Minimum required to represent the INTEGER input.

Table 9-1.

Permitted data types for automatic conversion on input

Type of variable	Allowable characters
DECIMAL	'+', '-', '0' thru '9', '.', 'E'
INTEGER	'+', '-', '0' thru '9'
CHAR *	Any except new line
BIT *	'0', '1'
STRING OF CHAR	Any except new line
STRING OF BIT	'0' thru '9'
STRING OF BIT with HEXADECIMAL	'0' thru '9', 'A' thru 'F'
STRING OF BIT with OCTAL	'0' thru '7'
STRING OF BIT with BINARY	'0' thru '1'
CONNECTION	See 14.12.4

NOTE

* INPUT to these types only inputs one CHAR or BIT.

9.1.7 <untyped input>

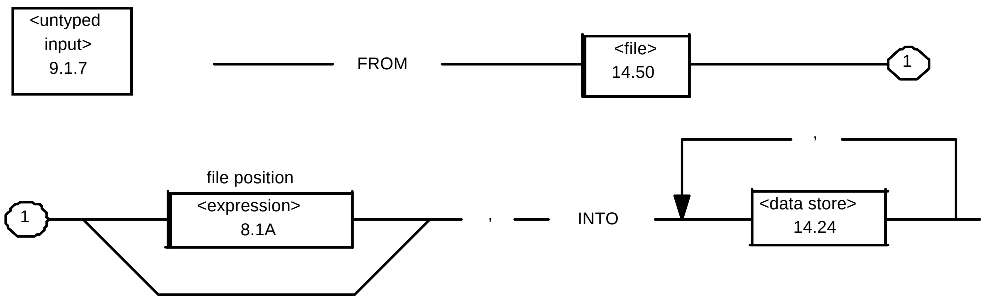
9.1.7.1 Function

This form of INPUT allows data items from a FILE OF UNTYPED to be assigned to variables. Its primary use is for inter-system data transfer of mixed data types.

9.1.7.2 Formal syntax

Reference untyped input

9.1.7.3 Syntax diagram



9.1.7.4 Rules

1. The <file> immediately following FROM shall be that of a FILE OF UNTYPED and the associated <file> shall be ENABLEd for INPUT.

2. The <data store>s following INTO that designate where the input is stored can be of any <type>.

3. If used, the file position <expression> preceding INTO shall evaluate to a non-negative INTEGER.

4. One or more 8-bit bytes are read in for each <data store> contained within an INPUT statement.

a) If the implementation creates a variable stored as a multiple of eight bits, the number of bytes input is equal to that multiple.

b) If the implementation creates a variable that is not stored as an exact multiple of eight bits, the number of bytes input will be those that fill the variable and any remaining bits in the last byte will be lost. The lost bits are determined by particular implementations.

5. Reading of the bytes is sequential in a forward direction from the position of the file pointer. The file pointer is positioned as follows:

a) If an INPUT statement contains an <expression>, the pointer is set to the value of the <expression>.

b) Otherwise, if the most recently executed operation associated with the <file> was ENABLE, the file position is set to zero.

c) In all other circumstances, the file pointer remains as set by the execution of the previous statement associated with the <file>.

6. For each execution, the file pointer is advanced by the number of bytes input.

7. For sequential reads, the pre-defined function EOF (<file>) will return TRUE if invoked after the last byte of the file has been read.

8. Before a file of untyped data can be transferred from a particular data processing system

that does not use a C/ATLAS data type, information shall be provided that defines the typing and formatting of data as related to the processing system. This information is used to generate a conversion routine to translate the input data into a defined data type in a C/ATLAS statement. Conversion routines are dependent upon the data representation format of particular implementations.

9.2 OUTPUT statement

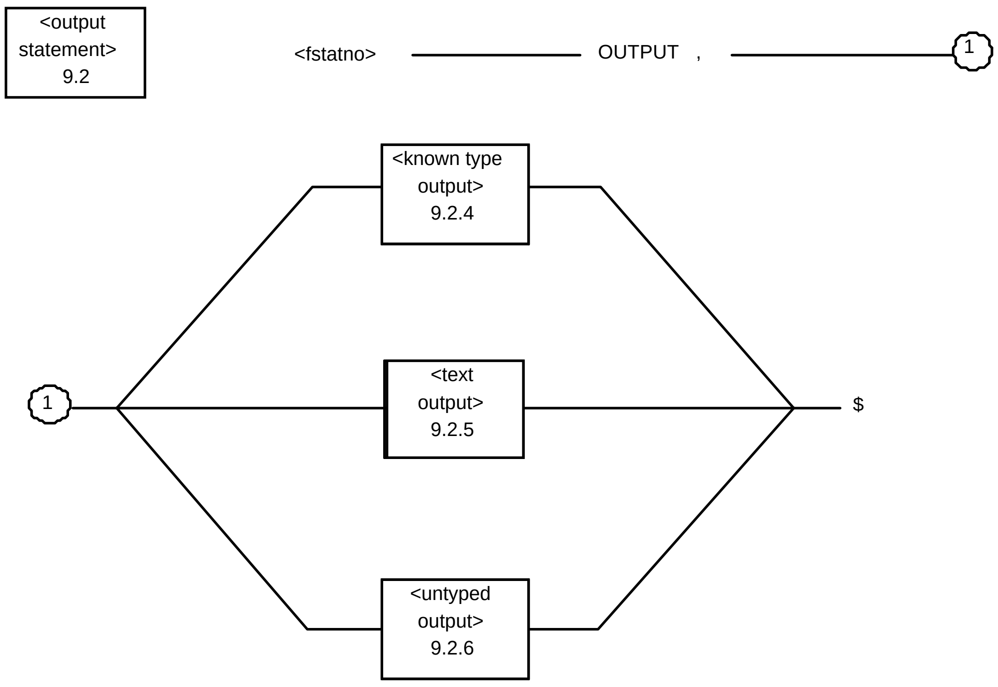
9.2.1 Function

This function is to transfer non-signal data from a C/ATLAS test specification to an external resource. C/ATLAS OUTPUT statements operate on virtual I/O resources.

9.2.2 Formal syntax

Reference output statement

9.2.3 Syntax diagram



9.2.4 <known type output>

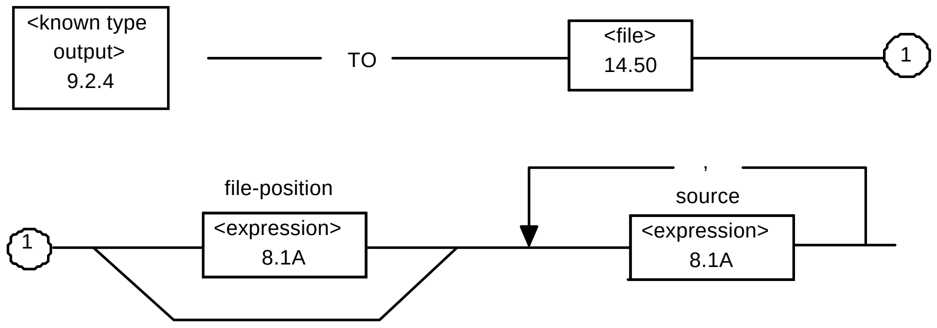
9.2.4.1 Function

This form of OUTPUT transfers the value of an <expression> to a file of a declared assignment compatible <type>.

9.2.4.2 Formal syntax

Reference known type output

9.2.4.3 Syntax diagram



9.2.4.4 Rules

1. OUTPUT statements are of this form if the <file> following TO is neither a FILE OF TEXT nor a FILE OF UNTYPED.

2. The data item resulting from the evaluation of the source <expression> shall be assignment compatible with the components of the <file> whose label follows TO. A binary image of this data item will be transferred to the file at the position of the file pointer.

3. The <file> shall have been ENABLEd previously.

4. If used, the file position <expression> shall evaluate to a non-negative integer.

5. Writing of the items of data into the file is sequential in a forwards direction from the current position. The current position is set as follows:

a) If an OUTPUT statement contains a file position <expression>, the file pointer is set to the value of that <expression>.

b) If the most recently executed operation associated with the <file> was ENABLE, the file position is set to zero.

c) In all other circumstances, the current position remains as set by the execution of the previous statement associated with the <file>.

6. For sequential writes, the pre-defined function EOF (<file>) will return TRUE if invoked after data has been transferred to the last data item of the file.

7. After execution of the OUTPUT statement, the current file position will have advanced by the number of data items transferred to the file.

9.2.5 <text output>

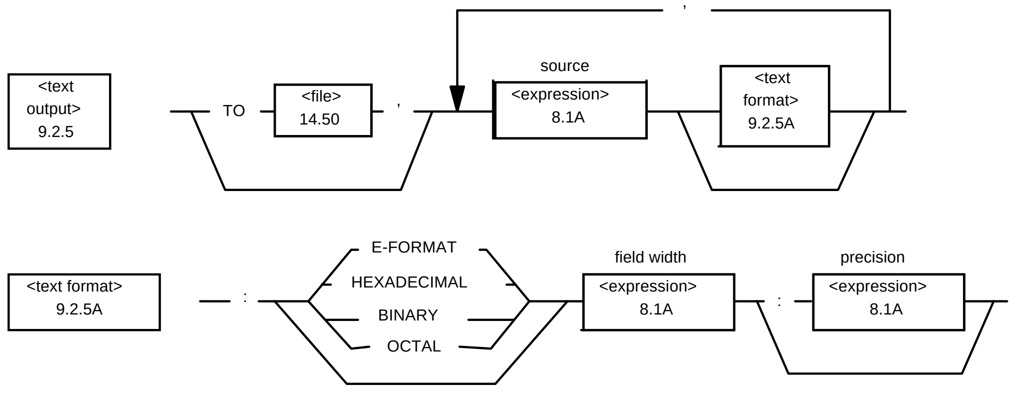
9.2.5.1 Function

This form of output writes characters to a FILE OF TEXT, converting the value of an <expression> to a text string. A simple default form allows easy text output to the system dependent standard output device.

9.2.5.2 Formal syntax

Reference text output

9.2.5.3 Syntax diagram



9.2.5.4 Rules

1. If TO <file> is omitted, output is to the system dependent standard output device.

2. The <file> immediately following TO shall have been declared as FILE OF TEXT and be enabled for OUTPUT. Output is forward sequential.

3. Table 9-2 specifies the permitted <type>s of source <expression>. Each permitted type has a set of legal characters. Type conversion of the value of the <expression> into its character representation occurs automatically on output.

4. HEXADECIMAL, OCTAL, and BINARY can be used only when the source <expression> is a STRING OF BIT.

5. E_FORMAT can be used only when the source <expression> is of type DECIMAL or LONG_ DECIMAL. E_FORMAT outputs a decimal floating-point number as defined in 15.2, item C.

6. Data types for which automatic output conversion is specified and their legal characters are specified in table 9-2.

7. The <expression>s in the <text format> subdiagram, 9.2.5A, must evaluate to positive integers. The field-width <expression> indicates the total number of characters to be written. Leading blank spaces will be added as required to match this format. An output longer than this total will not be truncated, but will be written using the smallest number of characters that will represent the data items. The precision <expression> is only used with a source <expression> of type DECIMAL or LONG_DECIMAL. It indicates the number of characters to be written to represent the fractional part of the number. Trailing 0's will be added as necessary. Truncation at the specified number of digits will occur.

8. If DECIMAL or LONG_DECIMAL numbers have no format, the output is implementation dependent.

9. OUTPUT from a STRING OF BIT, unqualified by HEXADECIMAL, OCTAL, or BINARY, involves a logical conversion to an unsigned INTEGER output with leading zeros suppressed.

Table 9-2.

Permitted data types for automatic conversion on output

Type of variable	Allowable characters
DECIMAL	'+', '-', '0' thru '9', '.', 'E'
INTEGER	'+', '-', '0' thru '9'
CHAR *	Any
BIT *	'0', '1'
STRING OF CHAR	Any
STRING OF BIT	'0' thru '9' (see 9.2.5.4 rule 9)
STRING OF BIT with HEXADECIMAL	'0' thru '9', 'A' thru 'F' (character per 4 bits)
STRING OF BIT with OCTAL	'0' thru '7' (character per 3 bits)
STRING OF BIT with BINARY	'0', '1' (character per 1 bit)
CONNECTION	See 14.12.4

NOTE
* OUTPUT from these types only outputs one character.

9.2.6 <untyped output>

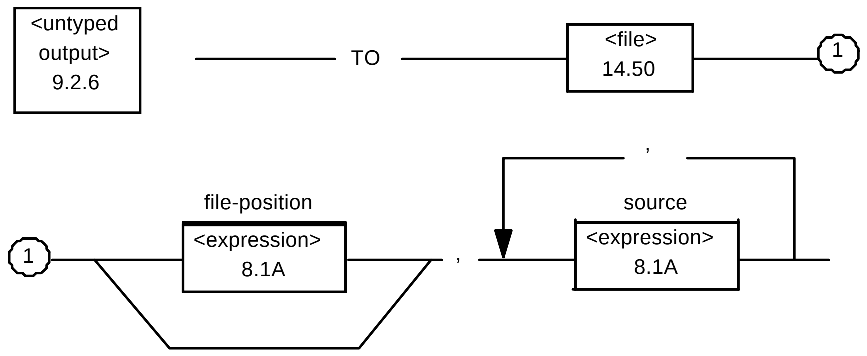
9.2.6.1 Function

This form of OUTPUT allows data to be transferred to a FILE OF UNTYPED. Its primary use is for inter-system data transfer of mixed data types. For graphic applications, this form of OUTPUT is used to display, print, and draw pictures.

9.2.6.2 Formal syntax

Reference untyped output

9.2.6.3 Syntax diagram



9.2.6.4 Rules

1. The type of the <file> immediately following TO shall be FILE OF UNTYPED.

2. The source <expression> designates the data that is to be output. It can be of any type.

3. The file position <expression> shall evaluate to a non-negative INTEGER.

4. Each source <expression> is written without data conversion for each iteration of the OUTPUT statement.

5. Writing of the bytes is sequential in a forwards direction from the position of the file pointer. The file pointer is positioned as follows:

a) If an OUTPUT statement contains a file position <expression>, the pointer is set to the value of the <expression>.

b) If the most recently executed operation associated with the <file> was ENABLE, the file position is set to zero.

c) In all other circumstances the file pointer remains as set by the execution of the previous operation associated with the <file>.

6. For sequential writes, the pre-defined function EOF(<file>) will return TRUE if invoked after the last byte of the file has been written.

7. After execution of the OUTPUT statement, the current position will have advanced by the number of bytes transferred to the file.

8. For graphic applications, the variables in the source <expression> are used to address and modify the pictures.

9.3 ENABLE FILE ACCESS statement

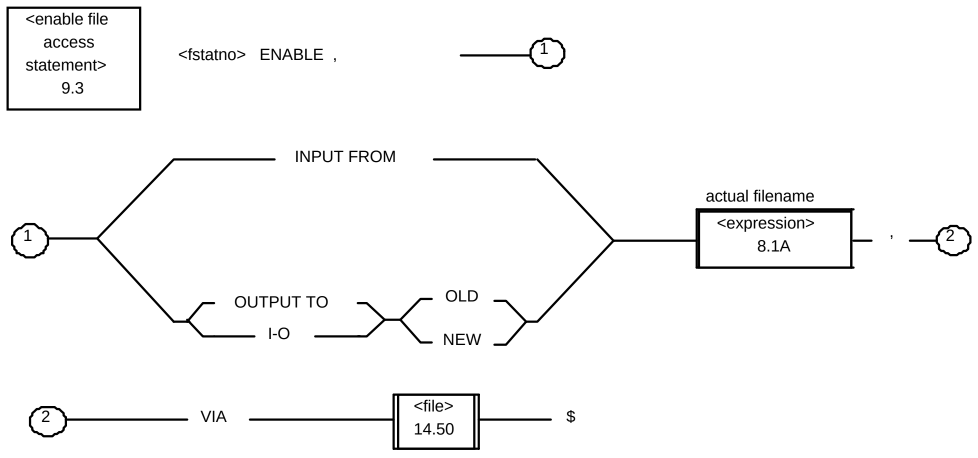
9.3.1 Function

This function is to associate a declared <file> variable with an actual file to allow subsequent access to the file.

9.3.2 Formal syntax

Reference enable file access statement

9.3.3 Syntax diagram



9.3.4 Rules

1. The actual filename <expression> shall evaluate to a STRING OF CHAR. This string identifies, to the operating system, the actual file. It may include identification of volume and/or physical device as well as the file name. For graphic applications, the actual filename <expression> allows the selection of the device(s) and the needed graphic functions for displaying, printing, and drawing pictures with an OUTPUT verb. The format of this string is implementation dependent.

2. The <file> shall be currently in a disabled state.

3. Only one file variable can be associated with a <file>.

4. Subsequent references to the file are by means of the <file> variable.

5. The file pointer is positioned to provide access to the first file record or byte. Immediately following this statement, calls to the pre-defined function EOF (<file>) will return FALSE for non-empty files.

6. Specifying NEW creates a file designated by the actual filename <expression>. The file is created with no data items in it and the current position is set to zero. Specifying OLD allows access to an existing file and sets the current position to zero.

7. For graphic applications, the VIA <file> is used for displaying, printing, and drawing pictures with an OUTPUT verb, and should have been previously declared as an untyped file. This file contains all the needed data for the graphic functions. Graphic data shall be expressed in a format compatible with the graphic processor. This requirement should be met by using a graphic standard.

9.4 DISABLE FILE ACCESS statement

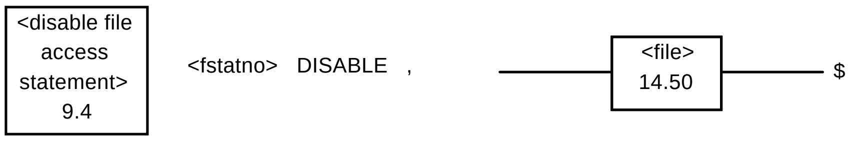
9.4.1 Function

To dissociate a declared <file> variable from an actual file and thereby release the <file> variable for future use.

9.4.2 Formal syntax

Reference disable file access statement

9.4.3 Syntax diagram



9.4.4 Rules

1. The statement has no effect unless the <file> is currently enabled.

2. Following execution of this statement a call to EOF (<file>) will return an undefined result.

10.0 Procedural statements, control

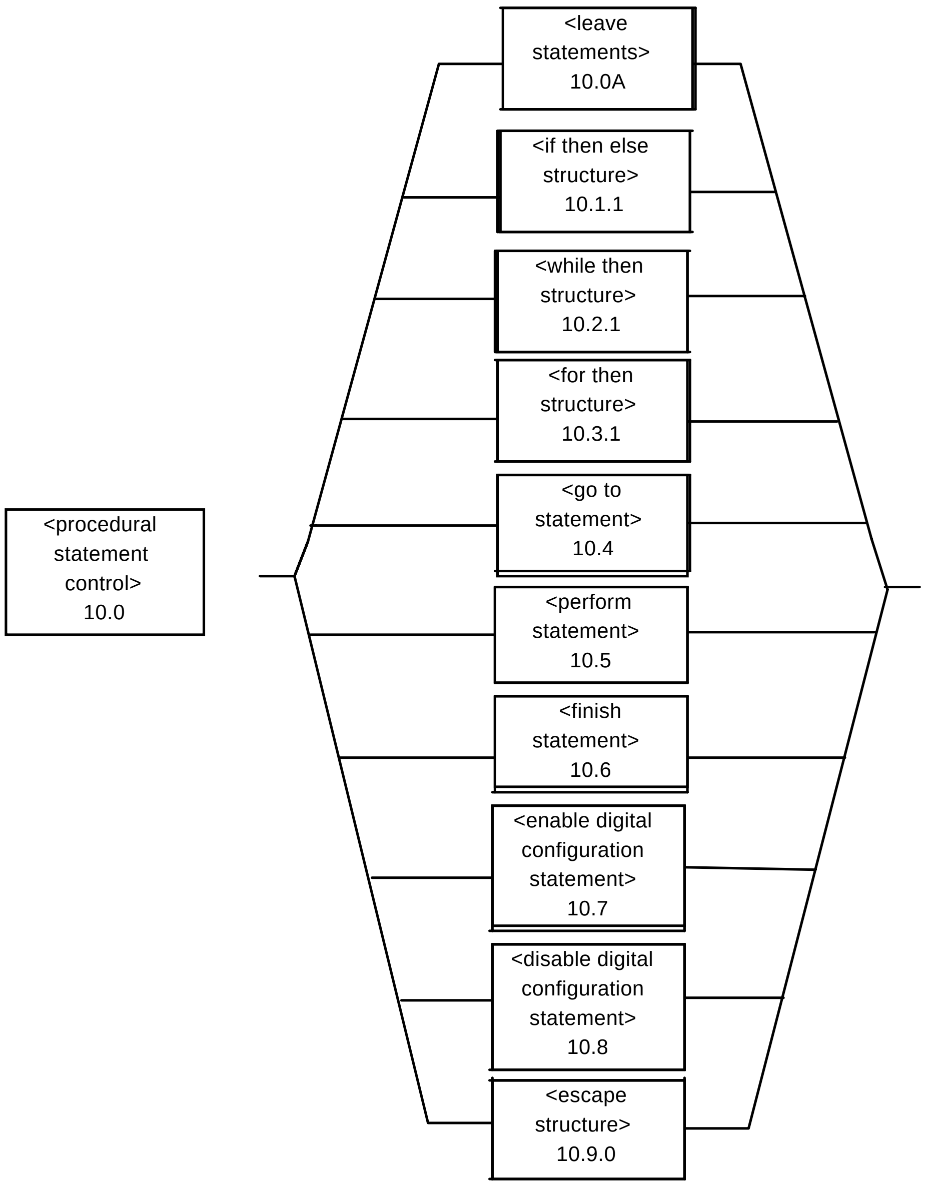
a) Function

The procedural control statements determine program execution sequences.

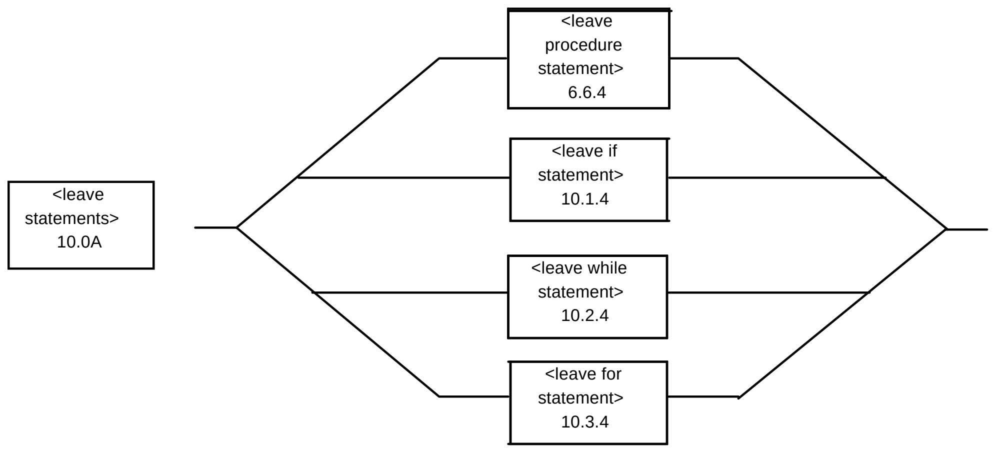
b) Formal syntax

Reference procedural statements control

c) Syntax diagram



<procedural statement control> 10.0
<leave statements> 10.0A
<if then else structure> 10.1.1
<while then structure> 10.2.1
<for then structure> 10.3.1
<go to statement> 10.4
<perform statement> 10.5
<finish statement> 10.6
<enable digital configuration statement> 10.7
<disable digital configuration statement> 10.8
<escape structure> 10.9.0



d) Rules

Each <leave statement> shall be used inside the relevant <procedure body> or <procedural segment ...>, (e.g., a <leave while statement> can only be used inside the <procedural segment ...> of a <while then structure>).

10.1 IF THEN ELSE capability

10.1.1 IF THEN ELSE structure

10.1.1.1 Function

The IF THEN structure permits the conditional execution of one or more statements based upon the logical value (TRUE or FALSE) of an expression. If the optional ELSE sub-structure is included, then the execution of one of two sets of statements, or single statements, is dependent upon the logical value of the expression. The functional flow of these structures is shown in figures 10-1 and 10-2.

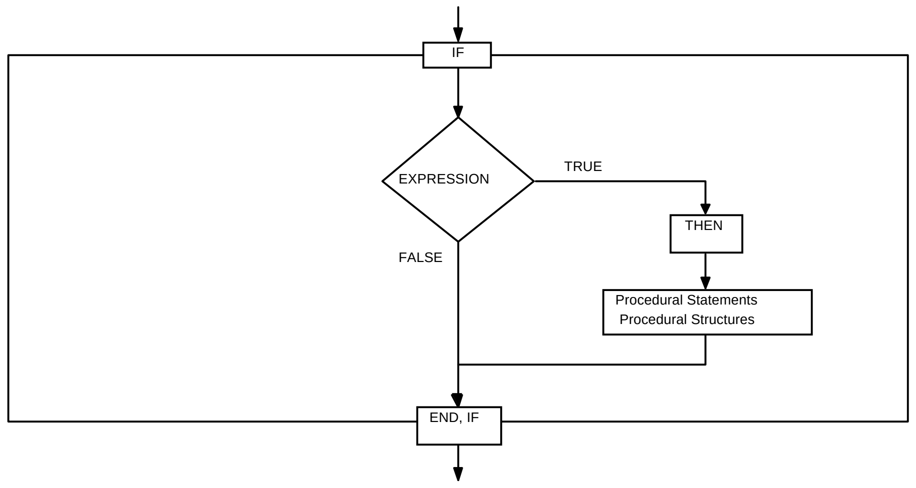


Figure 10-1. Functional flow diagram of the IF THEN structure

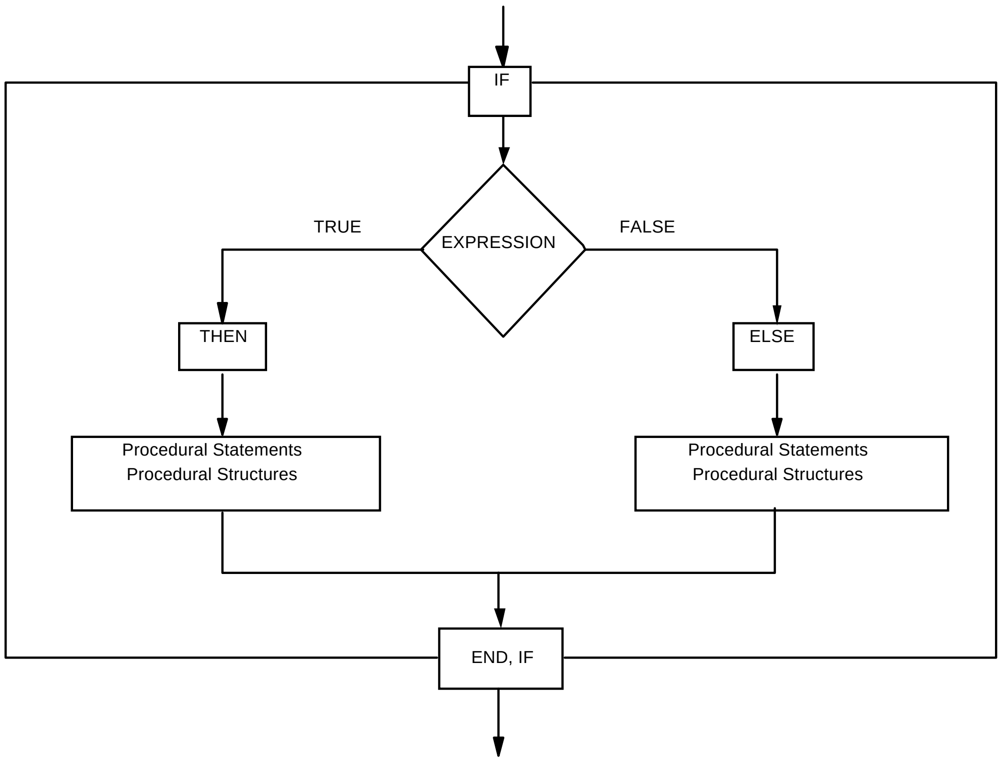
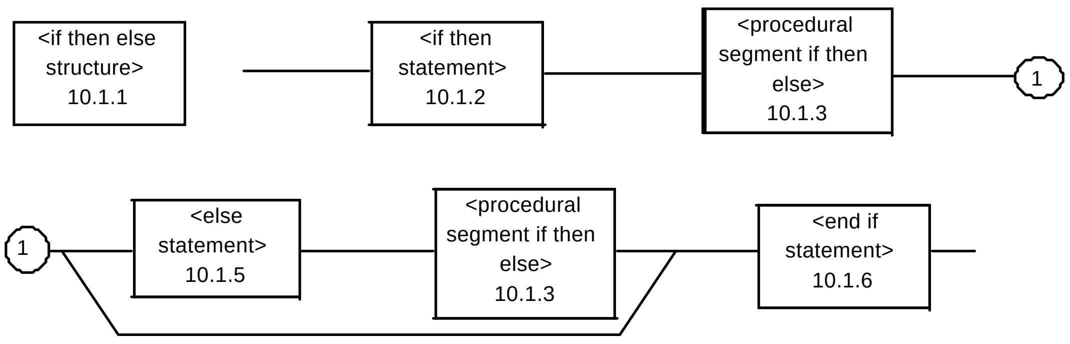


Figure 10-2. Functional flow diagram of the IF THEN ELSE structure

10.1.1.2 Formal syntax

Reference if then else structure

10.1.1.3 Syntax diagram


<if then else structure> 10.1.1
<if then statement> 10.1.2
<procedural segment if then else> 10.1.3
1
1
<else statement> 10.1.5
<procedural segment if then else> 10.1.3
<end if statement> 10.1.6

10.1.1.4 Rules

10.1.1.4.1 Rules applicable to the IF THEN structure

The <procedural segment> following an IF statement will be executed only if the stated logical condition is fulfilled, otherwise execution will continue directly at the statement following the END, IF statement.

10.1.1.4.2 Rules applicable to the IF THEN ELSE structure

1. The <procedural segment> following an IF statement will be executed only if the stated logical condition is fulfilled. If the stated condition is not fulfilled, the <procedural segment> following the ELSE statement will be executed.

2. Irrespective of which <procedural segment> was executed, execution will continue at the statement following the END, IF statement.

10.1.1.5 Notes and examples

See 10.1.3 <procedural segment if then else>

10.1.2 IF THEN statement

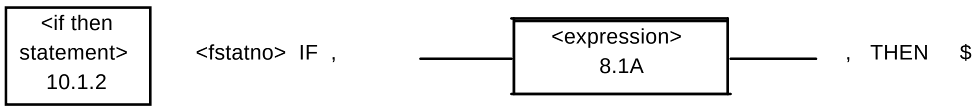
10.1.2.1 Function

The IF THEN statement serves to introduce the IF THEN and IF THEN ELSE structures by specifying the logical conditions which will cause the appropriate <procedural segment> to be executed.

10.1.2.2 Formal syntax

Reference if then statement

10.1.2.3 Syntax diagram



10.1.2.4 Rules

<expression> shall evaluate to a BOOLEAN result. The <procedural segment> immediately following the IF, THEN statement will be executed if the result evaluates to TRUE.

10.1.2.5 Notes and examples

See 10.1.3.

10.1.3 <procedural segment if then else>

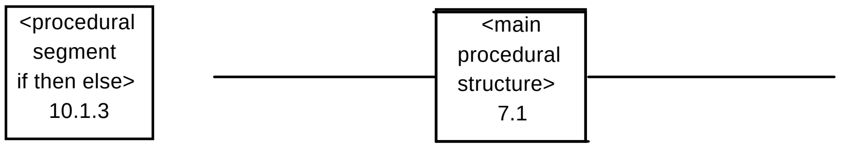
10.1.3.1 Function

A <procedural segment if then else> includes those statements to be performed if the stated logical condition is fulfilled or not fulfilled. A <procedural segment if then else> consists of one or more complete C/ATLAS procedural statements or structures.

10.1.3.2 Formal syntax

Reference procedural segment if then else

10.1.3.3 Syntax diagram



10.1.3.4 Rules

1. Statement numbers within a <procedural segment if then else> may only be referenced from within the bounds of that segment.

2. Entry points shall not appear within a <procedural segment if then else>.

10.1.3.5 Notes and examples

```
000100  IF, 'TEST', THEN $
    01     OUTPUT, C'UUT IS GOOD' $
    02  ELSE $
    03     OUTPUT, C'UUT IS BAD' $
    04  END, IF $
000205  IF, ('TESTOUT' + 'BIAS') ** 2 GT 12.5, THEN $
    07     SETUP, DC SIGNAL, VOLTAGE 5.0 V,
                  CNX HI J1-1 LO J1-2 $
    09  END, IF $

001000  IF, ( ('TEST1' OR 'TEST2') AND NOT 'TEST3'), THEN $
    01     IF, 'TESTOUT' EQ DEC(BNR,B'101110',6), THEN $
    02            OUTPUT, C'UUT IS GOOD' $
    03     ELSE $
    04            OUTPUT, C'UUT IS BAD' $
    05     END, IF $
    06  ELSE $
    08     APPLY, AC SIGNAL, ... $
    09     WAIT FOR, 10 SEC $
    10     MEASURE, (VOLTAGE_P INTO 'X'), ... $
    11     IF, 'X' UL 25.0 LL 15.0, THEN $
    12            CALCULATE, 'OK' = TRUE $
    13     END, IF $
    15  END, IF $
```

10.1.4 LEAVE, IF statement

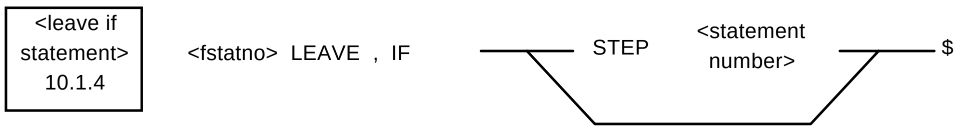
10.1.4.1 Function

Causes execution of a <procedural segment> of an IF THEN ELSE structure to terminate prior to normal sequential execution of the last statement of the <procedural segment>. Control is transferred to the first executable statement following the IF THEN ELSE structure.

10.1.4.2 Formal syntax

Reference leave if statement

10.1.4.3 Syntax diagram



10.1.4.4 Rules

1. When a statement number reference is used, it shall be the statement number of an IF statement which introduces an IF THEN ELSE structure containing the LEAVE, IF statement.

2. When a statement number reference is omitted, the LEAVE, IF statement is taken to refer to the innermost IF THEN ELSE structure containing the LEAVE, IF statement.

3. Any structures that contain a LEAVE, IF statement, and are contained within the IF THEN ELSE structure to which it applies, are also exited upon execution of the LEAVE, IF statement.

10.1.4.5 Notes and examples

In the following example, execution of statement number 423003 will transfer control to statement number 423011.

```
423000   IF, 'X', THEN $
    01    APPLY, DC SIGNAL ... $
    02    IF, 'Y', THEN $
    03                 LEAVE, IF STEP 423000 $
    04    END, IF $
    05    MEASURE, ... $
    10   END, IF $
    11   OUTPUT, ... $
```

10.1.5 ELSE statement

10.1.5.1 Function

The ELSE statement serves to introduce the optional alternate <procedural segment> of the IF THEN ELSE structure and, as such, is referred to as a connective verb.

10.1.5.2 Formal syntax

Reference else statement

10.1.5.3 Syntax diagram

```
<else
statement>          <fstatno>  ELSE    $
10.1.5
```

10.1.5.4 Rules

The ELSE statement, as a component of the IF THEN ELSE structure, can only be used in that context and has no other usage within the C/ATLAS language.

10.1.5.5 Notes and examples

See 10.1.3.

10.1.6 END, IF statement

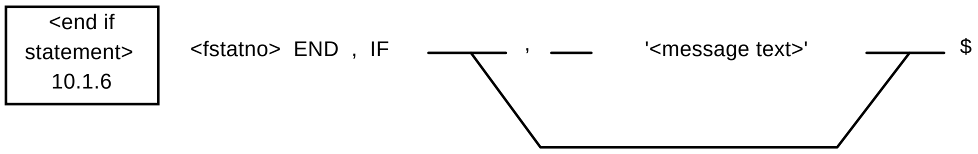
10.1.6.1 Function

The END, IF statement serves to delimit an IF THEN or IF THEN ELSE structure. Optional commentary may be included following the IF field to identify the IF structure which it delimits.

10.1.6.2 Formal syntax

Reference end if statement

10.1.6.3 Syntax diagram



10.1.6.4 Rules

The END, IF statement, as a component of the IF THEN ELSE structure, can only be used in that context and has no other usage within the C/ATLAS language.

10.1.6.5 Notes and examples

See 10.1.3.

10.2 WHILE THEN capability

10.2.1 WHILE THEN structure

10.2.1.1 Function

The WHILE THEN structure identifies the logical condition under which a <procedural segment while then> of a C/ATLAS test specification is to be iteratively performed and establishes the boundaries of that <procedural segment while then>. The functional flow of this structure is shown in figure 10-3.

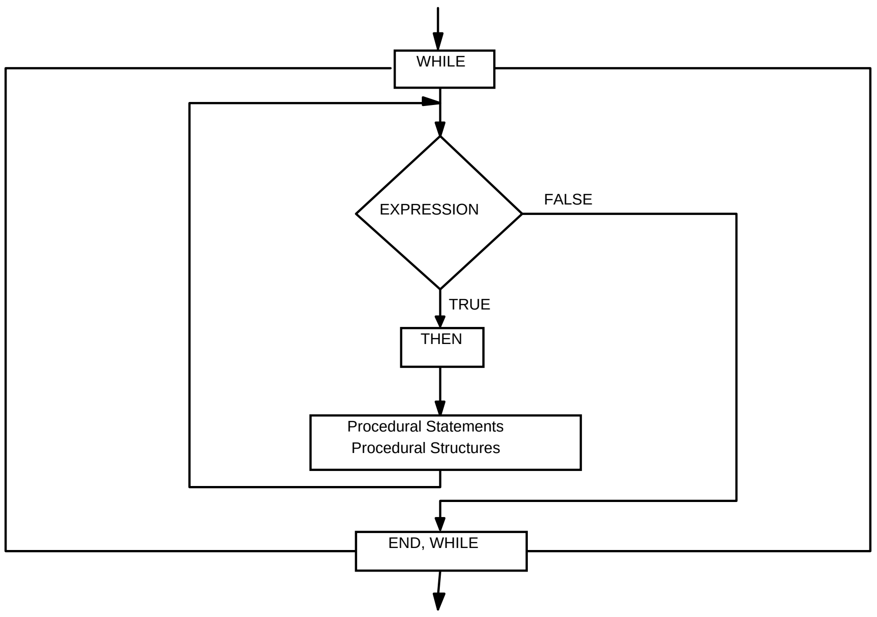
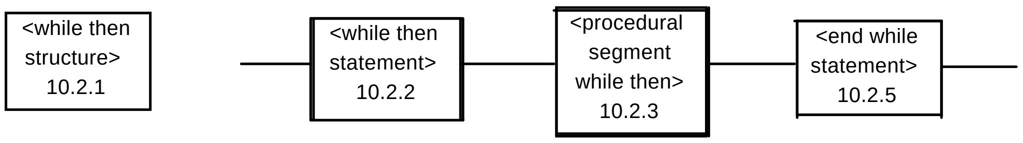


Figure 10-3. Functional flow diagram of the WHILE THEN structure

10.2.1.2 Formal syntax

Reference while then structure

10.2.1.3 Syntax diagram



10.2.1.4 Rules

1. The <procedural segment while then> will be iteratively executed while the stated condition is fulfilled.

2. If the stated condition is not fulfilled, execution continues with the next statement following the END, WHILE statement.

10.2.1.5 Notes and examples

See 10.2.3.

10.2.2 WHILE THEN statement

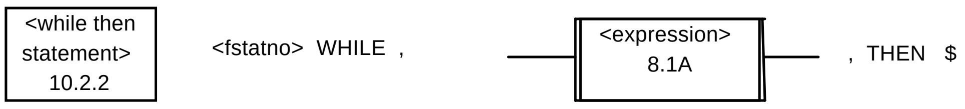
10.2.2.1 Function

The WHILE THEN statement serves to introduce the WHILE THEN structure by specifying the logical conditions that will cause the <procedural segment while then> to be executed at each iteration.

10.2.2.2 Formal syntax

Reference while then statement

10.2.2.3 Syntax diagram



10.2.2.4 Rules

<expression> shall evaluate to a BOOLEAN result. The <procedural segment while then> of the WHILE THEN structure will be executed if the <expression> evaluates to TRUE.

10.2.2.5 Notes and examples

See 10.2.3.

10.2.3 <procedural segment while then>

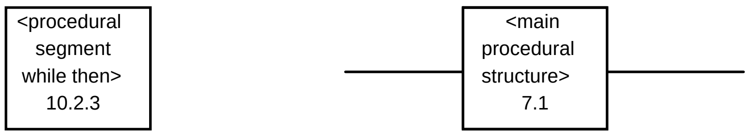
10.2.3.1 Function

The <procedural segment while then> describes the required logic to be performed during each iteration of the statements bounded by the WHILE THEN and END, WHILE statements.

10.2.3.2 Formal syntax

Reference procedural segment while then

10.2.3.3 Syntax diagram



10.2.3.4 Rules

See 10.1.3.

10.2.3.5 Notes and examples

```
READ, (VOLTAGE INTO 'DCW') ... $
WHILE, 'DCW' LT 15.3, THEN $
   READ, (VOLTAGE INTO 'DCW') ... $
END, WHILE $
```

10.2.4 LEAVE, WHILE statement

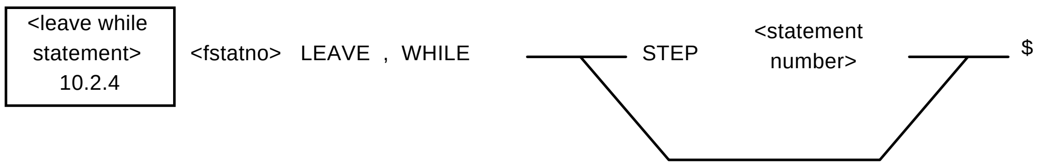
10.2.4.1 Function

The LEAVE, WHILE statement causes termination of a WHILE THEN structure.

10.2.4.2 Formal syntax

Reference leave while statement

10.2.4.3 Syntax diagram



10.2.4.4 Rules

1. When a statement number reference is used, it shall be the statement number of a WHILE THEN statement that introduces a WHILE THEN structure containing the LEAVE, WHILE statement.

2. When a statement number reference is omitted, the LEAVE, WHILE statement is taken to refer to the innermost WHILE THEN structure containing the LEAVE, WHILE statement.

3. Any structure that contains a LEAVE, WHILE statement and is contained within the WHILE THEN structure to which it applies, is also exited upon execution of the LEAVE, WHILE statement.

10.2.4.5 Notes and examples

```
WHILE, 'X' LT 10.0, THEN $
  CALCULATE, 'BIAS' = 'BIAS' -0.5 $
IF, 'BIAS' LT 0.0, THEN $
  LEAVE, WHILE $
END, IF $
MEASURE, (FREQ INTO 'X') ... $
END, WHILE $
OUTPUT, ... $
```

In the preceding example, the WHILE loop will be terminated when variable 'BIAS' becomes negative, regardless of the value of 'Xí.

10.2.5 END, WHILE statement

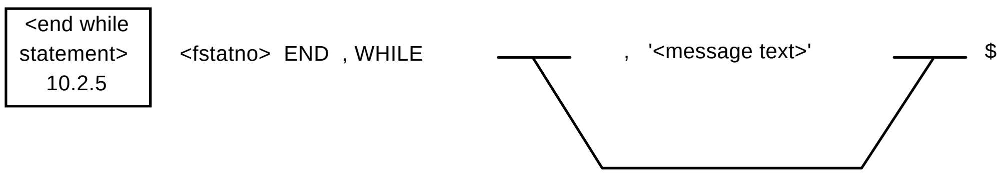
10.2.5.1 Function

The END, WHILE statement serves to delimit the WHILE THEN structure. Optional commenting may be included following the WHILE field to identify the WHILE structure which it delimits.

10.2.5.2 Formal syntax

Reference end while statement

10.2.5.3 Syntax diagram



10.2.5.4 Rules

The END, WHILE statement, as a component of the WHILE THEN structure, can be used only in that context and has no other usage within the C/ATLAS language.

10.2.5.5 Notes and examples

See 10.2.3.

10.3 FOR THEN capability

10.3.1 FOR THEN structure

10.3.1.1 Function

The FOR THEN structure permits the repetitive execution of a <procedural segment for then>, establishes the bounds of that <procedural segment for then>, and identifies a control variable that will be assigned a value prior to each iteration of that <procedural segment for then>. The functional flow of this structure is shown in figure 10-4.

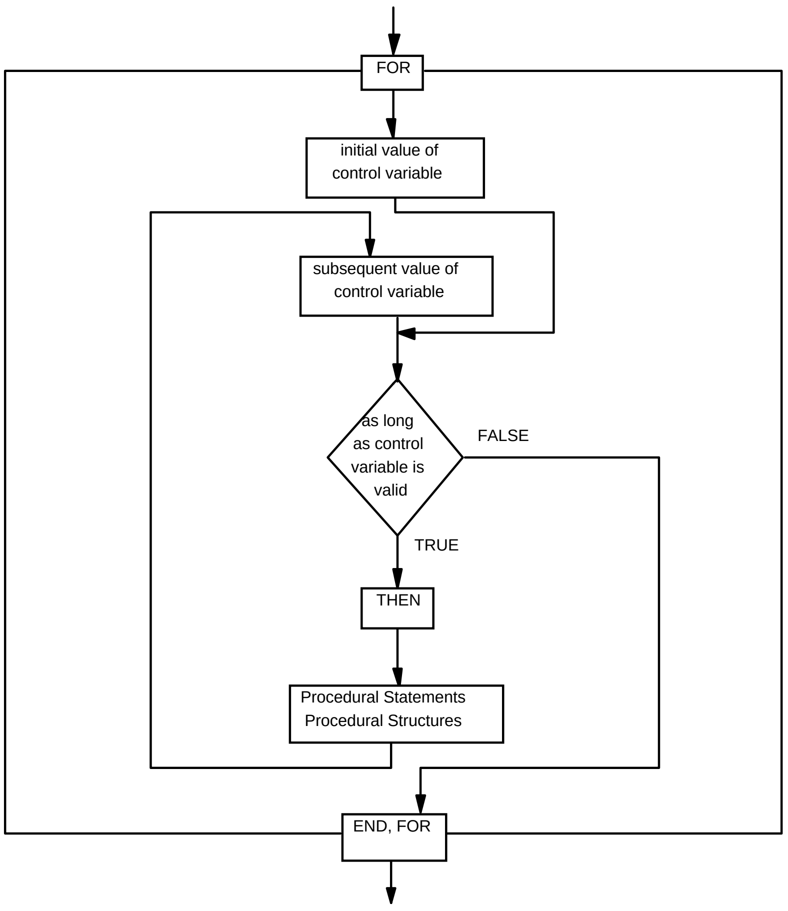
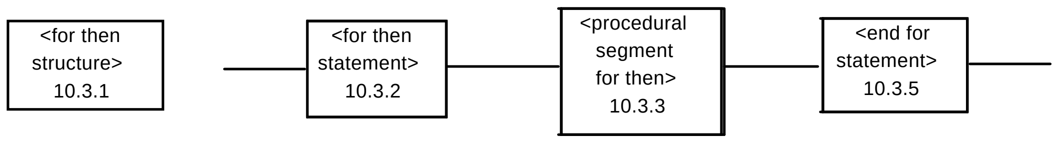

FOR
initial value of control variable
subsequent value of control variable
as long as control variable is valid
FALSE
TRUE
THEN
Procedural Statements
Procedural Structures
END, FOR

Figure 10-4. Functional flow diagram for the FOR THEN structure

10.3.1.2 Formal syntax

Reference for then structure

10.3.1.3 Syntax diagram



<for then structure> 10.3.1
<for then statement> 10.3.2
<procedural segment for then> 10.3.3
<end for statement> 10.3.5

10.3.1.4 Rules

1. The number of iterations of the <procedural segment for then> is specified by the form of the FOR THEN statement.

2. Upon completion of the final iterative execution of the <procedural segment for then>, execution continues with the next statement following the END, FOR statement.

10.3.1.5 Notes and examples

See 10.3.3.

10.3.2 FOR THEN statement

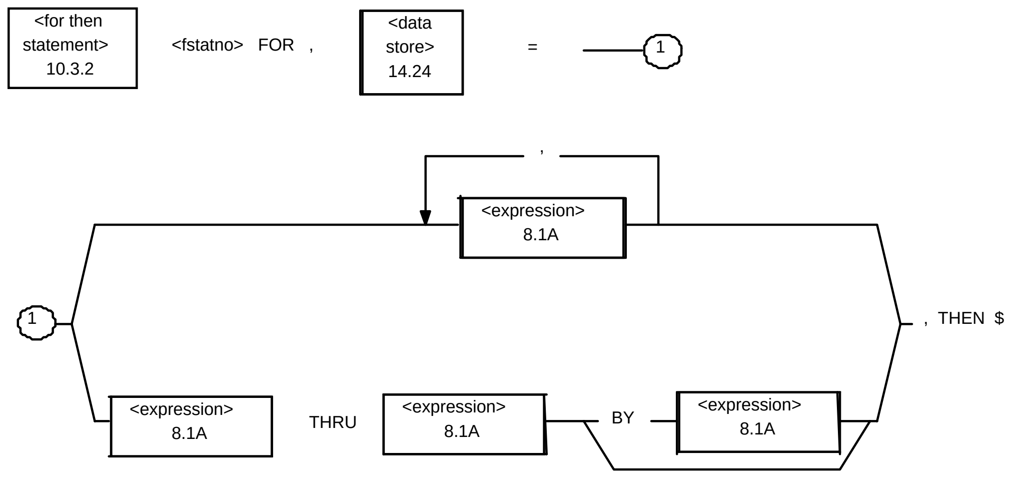
10.3.2.1 Function

The FOR THEN statement serves to introduce the FOR THEN structure, identifies the control variable, and specifies the values that the control variable will take for each iteration of the <procedural segment for then>. The FOR THEN statement has two forms: one form specifies a list of independent values for the control variable; the other form identifies a computed sequence of values for the control variable.

10.3.2.2 Formal syntax

Reference for then statement

10.3.2.3 Syntax diagram



10.3.2.4 Rules

10.3.2.4.1 Rules applicable to both forms of the FOR THEN statement

1. The control variable shall be a single element of base type INTEGER, DECIMAL, LONG_ DECIMAL, ENUMERATION, CHAR, or CONNECTION.

2. <expression> shall evaluate to a base type INTEGER, DECIMAL, LONG_DECIMAL, ENUMERATION, CHAR, or CONNECTION. The results of evaluation shall be assignment compatible with the <type> of the control variable.

3. Upon completion of the final iterative execution of the procedural segment, the value of the control variable is undefined.

10.3.2.4.2 Rules applicable to the value list form of the FOR THEN statement

The list of values for the control variable has the form e_1, e_2, e_3, ... e_n, where e_1 thru e_n represent any valid arithmetic expression that is consistent with the control variable type. On the first iteration, the control variable is assigned the value of e_1. On successive iterations, the control variable is assigned the values of e_2, e_3, ...e_n. Thus, the number of iterations will equal the number of items in the value list. Each expression is evaluated once upon initial entry into the FOR THEN structure.

10.3.2.4.3 Rules applicable to the value sequence form of the FOR THEN statement for INTEGER, DECIMAL, and LONG_DECIMAL control variables

The range of values for the control variable has the form e_1 THRU e_2 or e_1 THRU e_2 BY e_3. On the first iteration, the control variable is assigned the value of e_1. On successive iterations, e_3 (or 1.0 by default, if "BY e_3" is not specified) is added to the control variable. If the value of e_3 is negative, the value of e_1 is decremented. The iterative sequence is continued while e_1 <= control variable <= e_2 if $e_1 < e_2$ or while e_2<= control variable <= e_1 if

$e_2 < e_1$. When the value of the control variable goes outside of these boundaries, control is transferred to the statement following the <end for statement>. The values of e_1, e_2, and e_3 are each evaluated once upon entry into the <for then structure>.

10.3.2.4.4 Rules applicable to the value sequence form of the FOR THEN statement for ENUMERATION, CONNECTION, and CHAR control variables

The range of values for the control variable has the form e_1 THRU e_2, where ORD(e_2) shall be greater than or equal to ORD(e_1). On the first iteration, the control variable is assigned the value of e_1. On successive iterations, the control variable takes the value of its successor. The iterative sequence is continued while the control variable is in the range e_1 THRU e_2. When the value of the control variable is not in the range e_1 THRU e_2, control is transferred to the statement following the END, FOR statement. The values of e_1 and e_2 are each evaluated once upon entry into the <for then structure>.

10.3.2.5 Notes and examples

See 10.3.3.

10.3.3 <procedural segment for then>

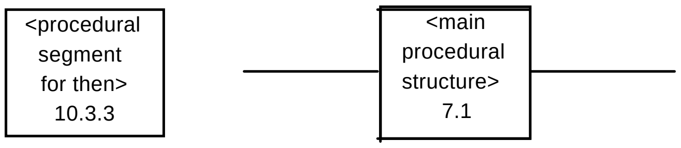
10.3.3.1 Function

The <procedural segment for then> describes the required logic to be performed during each iteration of the statements bounded by the FOR THEN and END, FOR statements.

10.3.3.2 Formal syntax

Reference procedural segment for then

10.3.3.3 Syntax diagram



10.3.3.4 Rules

See 10.1.3.

10.3.3.5 Notes and examples

```
000100  FOR,  'I' = 1 THRU 20, THEN $
    10     MEASURE, (VOLTAGE INTO 'RESULT' ('I')),
               DC SIGNAL,
               VOLTAGE RANGE -20.0V TO +20.0V,
                         CNX HI J5 LO EARTH $
    20     WAIT FOR, 250MSEC $
    99  END, FOR $
```

```
000200  FOR, 'STEP' = 'X', 'X' + 2, 'X', 'X' - 2, 'X', THEN $
    10      SETUP, DC SIGNAL,
                VOLTAGE 'STEP' RANGE -20.0V TO 20.0V,
                          CNX HI J14-7 LO COMMON $
    20      WAIT FOR, 500 MSEC $
    99  END, FOR $
```

10.3.4 LEAVE, FOR statement

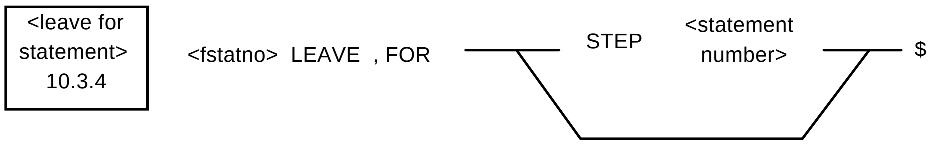
10.3.4.1 Function

The LEAVE, FOR statement causes termination of a FOR THEN structure.

10.3.4.2 Formal syntax

Reference leave for statement

10.3.4.3 Syntax diagram



10.3.4.4 Rules

1. When a statement number reference is used, it shall be the statement number of a FOR THEN statement that introduces a FOR THEN structure containing the LEAVE, FOR statement.

2. When a statement number reference is omitted, the LEAVE, FOR statement is taken to refer to the innermost FOR THEN structure containing the LEAVE, FOR statement.

3. Any structure that contains a LEAVE, FOR statement, and is contained within the FOR THEN structure to which it applies, is also exited upon execution of the LEAVE, FOR statement.

10.3.5 END, FOR statement

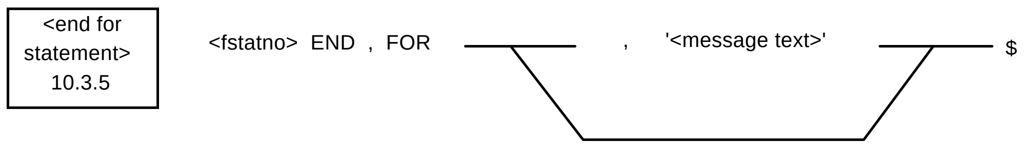
10.3.5.1 Function

The END, FOR statement serves to delimit the FOR THEN structure. Optional commentary may be included following the FOR field to identify the FOR THEN structure which it delimits.

10.3.5.2 Formal syntax

Reference end for statement

10.3.5.3 Syntax diagram



10.3.5.4 Rules

The END, FOR statement, as a component of the FOR THEN structure, can only be used in that context and has no other usage within the C/ATLAS language.

10.3.5.5 Notes and examples

See 10.3.3.

10.4 GO TO statement

10.4.1 Function

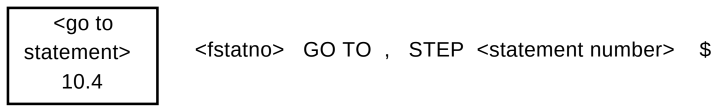
The GO TO statement is used in a test specification to direct the tester to some statement other than the one following it. The destination statement shall be preceded by a "B" Flag (see 15.3). A GO TO statement causes the sequence to be diverted as indicated every time the GO TO statement is executed.

The GO TO statement in conjunction with the IF structure may be used for conditional branching.

10.4.2 Formal syntax

Reference go to statement

10.4.3 Syntax diagram



10.4.4 Rules

1. The target statement number referenced by a GO TO statement in a <define procedure structure> shall be within the structure containing the GO TO statement.

2. Only the first statement of an <if then else structure>, a <while then structure>, or a <for then structure> may be the destination of a GO TO statement that is located outside of the structure.

3. A GO TO statement within a <while then structure> or <for then structure> that references the END statement transfers control to the head of the structure where the condition for further iteration is evaluated.

10.5 PERFORM statement

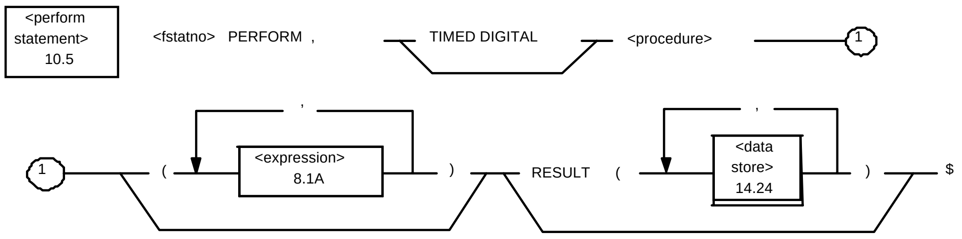
10.5.1 Function

This function is to invoke a single or series of C/ATLAS statement(s) or test information in a NON-ATLAS module previously defined as a procedure.

10.5.2 Formal syntax

Reference perform statement

10.5.3 Syntax diagram



10.5.4 Rules

1. The respective values for the <parametre> labels of the DEFINE <procedure> statement are written next. Only the constants or labels are written in the PERFORM statement. Labels, in general, are defined in clause 15. These numbers or labels shall be written in the PERFORM statement in exactly the same sequence as the <parametre> labels are listed in the DEFINE <procedure> statement.

2. The type of each variable in the actual parametre list shall be assignment compatible with the type of the corresponding variable in the formal parametre list as defined in 6.3.4.5 for assignment compatibility between <expression> and <data store>.

3. If a multi-element <structured type> such as a RECORD, STRING, or ARRAY is used as an actual parametre (e.g., array) it shall have at least as many elements as the declared array from the formal parametres. If a formal parametre has n elements and the corresponding actual parametre has m elements, where m is greater than n, then the first n elements of the actual array are employed.

4. An actual parametre that corresponds to a formal parametre established in the RESULT field of a <define procedure statement> shall be a declared variable. (i.e., constants, literals, and <expression>s may not be used as RESULT parametres of a <procedure>).

5. The TIMED DIGITAL path may only be used for procedures defined as TIMED DIGITAL.

10.5.5 Notes and examples

See 6.6.1.

10.6 FINISH statement

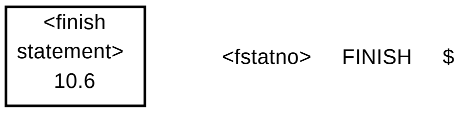
10.6.1 Function

This function is to terminate testing and to return the test equipment to a quiescent state.

10.6.2 Formal syntax

Reference finish statement

10.6.3 Syntax diagram



10.6.4 Rules

1. FINISH is always written alone with no noun or modifier.

2. No statements are executed after the execution of a FINISH statement.

10.7 ENABLE DIGITAL CONFIGURATION statement

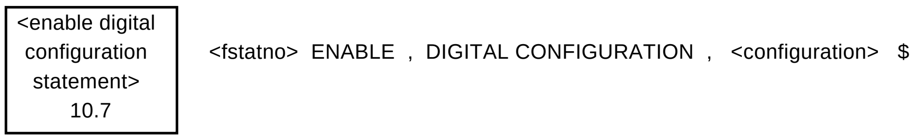
10.7.1 Function

To ENABLE a DEFINEd <configuration> so that the <configuration> can be referenced by procedural STIMULATE, SENSE or PROVE statements (or their compounds).

10.7.2 Formal syntax

Reference enable digital configuration statement

10.7.3 Syntax diagram



10.7.4 Rules

1. Only one <configuration> can be ENABLEd at a time. When an <enable digital configuration statement> is encountered, a previously ENABLEd <configuration> cannot then be referenced for digital testing.

2. The specific actions of ENABLE are

a) To first disable a previously enabled <configuration>. If an already enabled <configuration> is re-enabled, there will be no disconnect and connect actions.

b) To then set up drivers/sensors of the <configuration> to the required conditions.

c) To lastly connect the <configuration> to the UUT.

3. The first time that a <configuration> is ENABLEd, the initial state of each connection will be LOGIC_HIZ state if defined, else LOGIC_QUIES if defined, else LOGIC_ZERO.

4. The subsequent ENABLE of a DISABLEd <configuration> shall be as if it was ENABLEd for the first time in accordance with rule 3.

5. Upon execution of an ENABLE statement, the value of each <real quantity> in the set of <define digital source statement>s and <define digital sensor statement>s in the identified <define digital configuration structure> is computed and used as the corresponding BIT_RATEs and BIT_PERIODs until a subsequent ENABLE or DISABLE of a <configuration>.

10.8 DISABLE DIGITAL CONFIGURATION statement

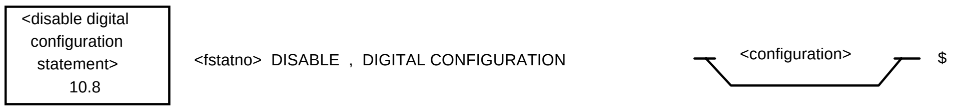
10.8.1 Function

To disable a previously enabled <configuration>.

10.8.2 Formal syntax

Reference disable digital configuration statement

10.8.3 Syntax diagram


<disable digital configuration statement> 10.8
<fstatno> DISABLE , DIGITAL CONFIGURATION
<configuration>
$

10.8.4 Rules

1. Only the previously enabled configuration can be disabled. When disabled, the <configuration> cannot then be referenced for digital testing.

2. The specific action of the DISABLE is to disconnect the <configuration> from the UUT.

3. If an already disabled <configuration> is re-disabled, no action is taken.

4. A REMOVE, ALL statement will disable any enabled <configuration>.

5. The optional <configuration> shall be identical to the <configuration> in the previous <enable digital configuration statement>.

10.9 Escape structure

10.9.0 <escape structure>

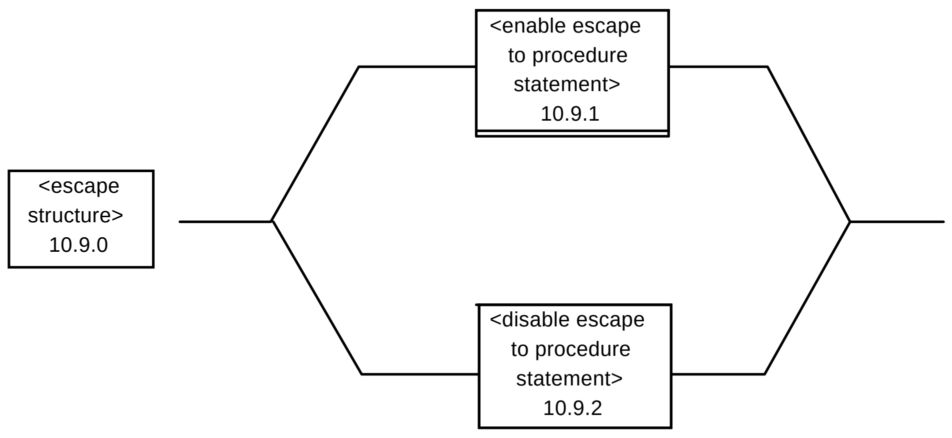
10.9.0.1 Function

The escape mechanism allows a limited form of parallel monitoring of a test situation. If a stated test condition occurs, the escape mechanism causes the program execution sequence to be interrupted and a procedure to be executed before program execution is resumed.

10.9.0.2 Formal syntax

Reference escape structure

10.9.0.3 Syntax diagram


<escape structure> 10.9.0
<enable escape to procedure statement> 10.9.1
<disable escape to procedure statement> 10.9.2

10.9.1 ENABLE ESCAPE TO PROCEDURE statement

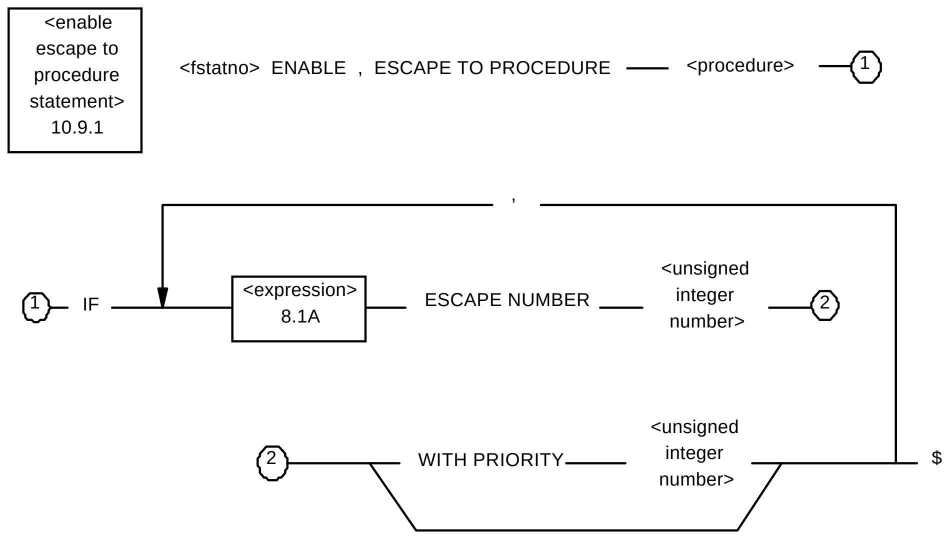
10.9.1.1 Function

This function is to enable an escape mechanism that will occur if a specified <expression> evaluates to TRUE. The escape mechanism is active until disabled by a <disable escape to procedure statement>.

10.9.1.2 Formal syntax

Reference enable escape to procedure statement

10.9.1.3 Syntax diagram


<enable escape to procedure statement> 10.9.1
<fstatno> ENABLE , ESCAPE TO PROCEDURE
<procedure>
1
,
1
IF
<expression> 8.1A
ESCAPE NUMBER
<unsigned integer number>
2
2
WITH PRIORITY
<unsigned integer number>
$

10.9.1.4 Rules

1. The definition of an escape procedure structure follows rules 1, 2, and 3 of <define procedure structure> (reference 6.6.1).

2. An escape procedure is invoked after finishing the execution of the current executed ATLAS statement. However, an escape procedure with priority zero shall be treated as an emergency exit from the program execution and shall be executed immediately. After execution of a priority zero escape procedure, program execution is not resumed, but finished.

3. An <expression> can contain only <event indicator>s.

4. The procedure is executed when <expression> evaluates TRUE.

5. The lowest <unsigned integer number> for PRIORITY aligns to the highest priority of execution. If no priority is given, it shall be treated as the lowest priority.

6. An escape procedure shall not be invoked during the execution of another escape procedure with equal or higher priority, but will be executed in priority order after completion of the current procedure.

7. The procedure referenced in the <enable escape to procedure statement> shall be defined with a single parametre of type INTEGER, which at time of invocation will contain the escape number of the invoking escape condition.

8. All <event>s indirectly referred to from an <enable escape to procedure statement> via an <event indicator> shall be explicitly enabled in an <enable event statement>.

10.9.1.5 Notes and examples

1. During resource allocation, all resources used in an escape procedure shall be treated as allocated when the escape mechanism is enabled, and will remain so until the escape mechanism is disabled.

2. Regardless of note 1, resources used in an escape procedure, except in an escape procedure with priority zero, may be allocated in a program for signal activities that begin and terminate in the same ATLAS statement because they will not interfere with signal activities in the escape procedure.

3. Reset of an <event indicator> shall be done explicitly in a <calculate statement> (reference 6.14.4, rule 2).

```
   :
   :
C DEFINITION OF THE ESCAPE PROCEDURE $

   DEFINE, 'LOCSUB' PROCEDURE ('ENR' IS INTEGER) $
   :
   :
   END, 'LOCSUB' $
   :
   :
   IDENTIFY, EVENT INDICATOR 'EVEN 1' AS SET BY 'EVENT 1' $
   IDENTIFY, EVENT INDICATOR 'EVEN 2' AS SET BY 'EVENT 2' $
   IDENTIFY, EVENT INDICATOR 'EVEN 3' AS SET BY 'EVENT 3' $
   :
   :

C PROGRAM START $

 100000 ENABLE, ESCAPE TO PROCEDURE 'LOCSUB'
        IF ('EVEN 1' AND 'EVEN 2') ESCAPE NUMBER 2 WITH PRIORITY 2,
        'EVEN 3' ESCAPE NUMBER 4 WITH PRIORITY 18 $
        :
        :

 109900 DISABLE, ESCAPE TO PROCEDURE ESCAPE NUMBER 2, 4 $

        :
        :
```

10.9.2 DISABLE ESCAPE TO PROCEDURE statement

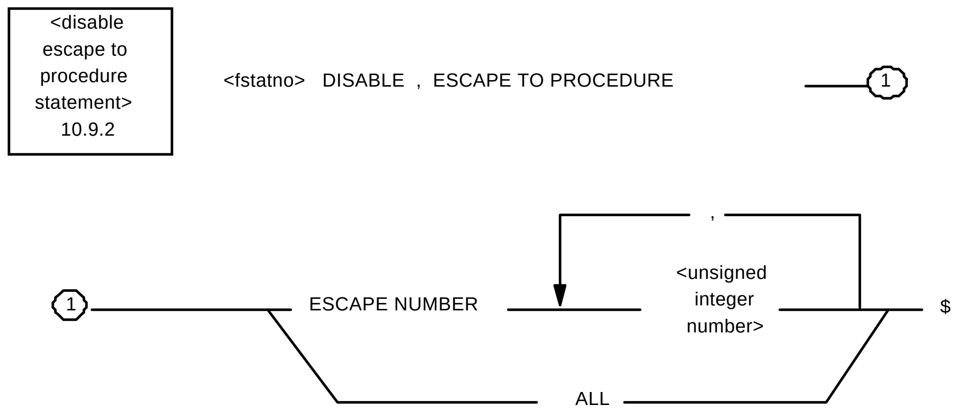
10.9.2.1 Function

This function is to disable an escape mechanism previously activated by an <enable escape to procedure statement>.

10.9.2.2 Formal syntax

Reference disable escape to procedure statement

10.9.2.3 Syntax diagram



<disable escape to procedure statement> 10.9.2
<fstatno> DISABLE , ESCAPE TO PROCEDURE
1
1
ESCAPE NUMBER
,
<unsigned integer number>
$
ALL

11.0 Signal oriented statements

11.1 <procedural statements signal>

11.1.1 Function

Signal oriented procedural statements consist of single-action, multiple-action, and digital statements that describe source, sensor, load, and digital functions of signals relative to the UUT. The multiple-action statements are functionally equivalent to sequences of single-action signal procedural statements and other non-signal procedural statements. Where two or more statements specify operations on the same signal, the set of characteristics describing that signal shall be compatible.

11.1.1.1 Critical actions

Each multiple-action signal oriented verb has a critical action for the purpose of referencing timing and synchronization of test sequences.

11.1.1.2 Timing control

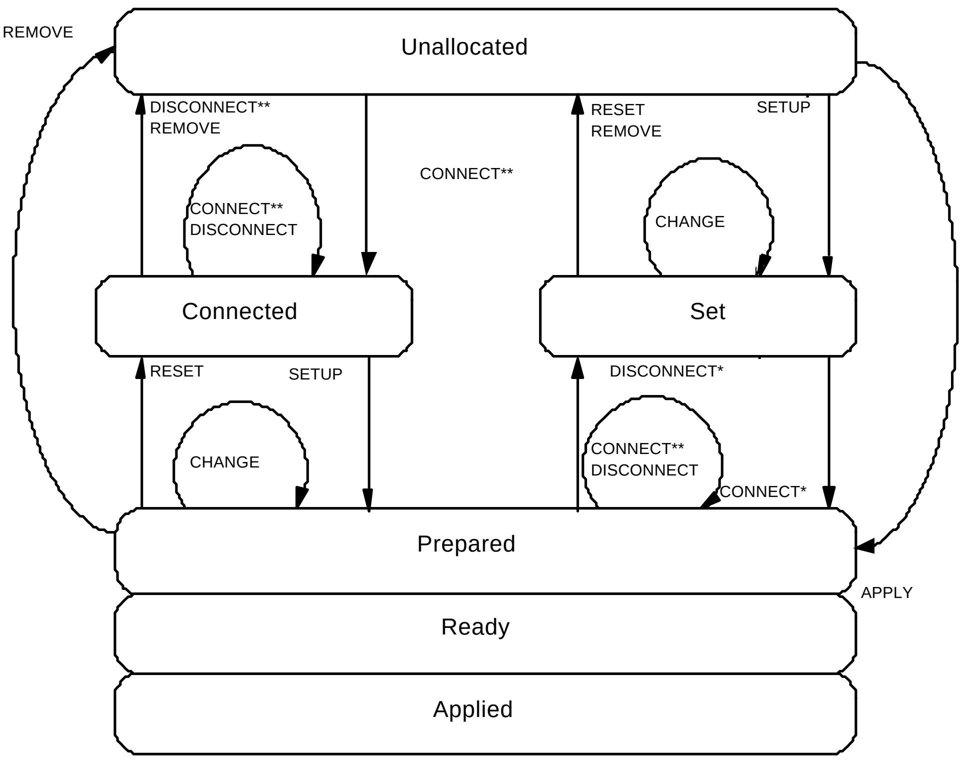
Statement timing is realized by controlling the time of execution of <single action statements>, or for a <multiple action statements>, the time of execution of the critical action.

11.1.1.3 UUT failures

For analog testing, C/ATLAS provides for the identification of failures by the use of the flags GO/NOGO/HI/LO following a VERIFY or a COMPARE statement.

11.1.1.4 State diagrams

Figures 11-1, 11-2, 11-3, and 11-4 are the general cases for the optimum resource state sequences of signal oriented statements that can be implemented by a single virtual resource.



WARNINGS:

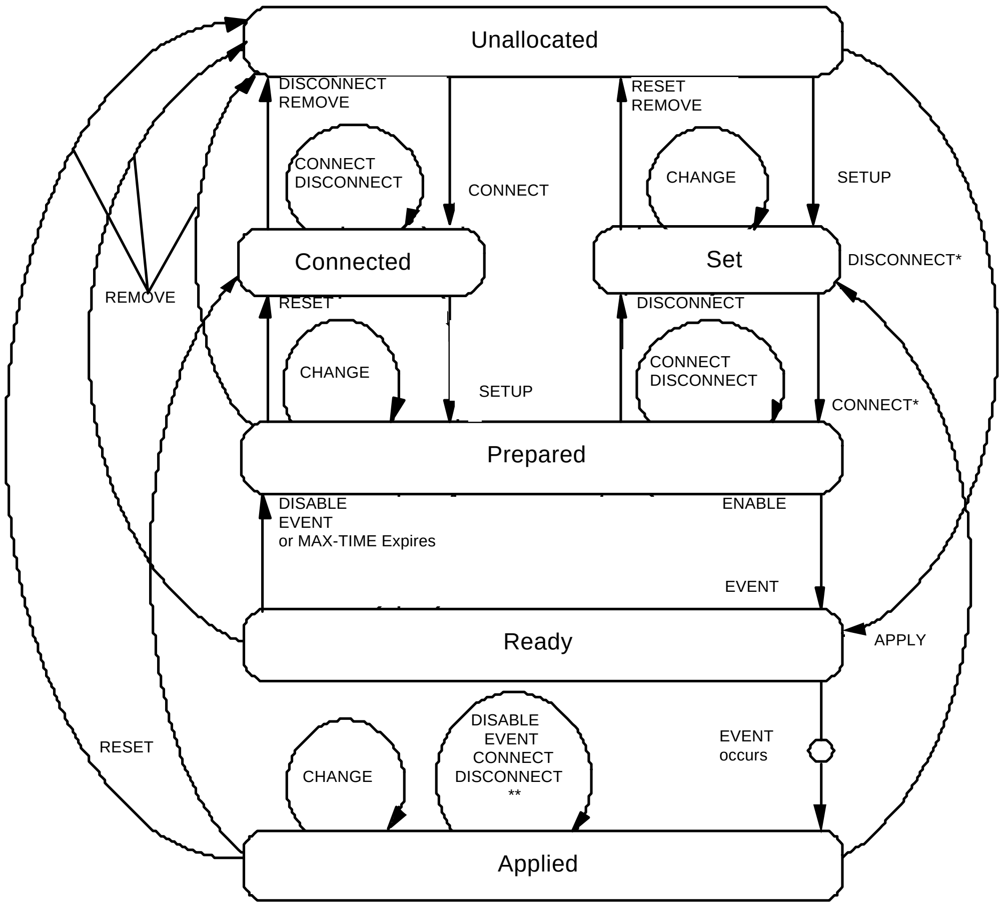
*	Hot Source Switching
**	Hot Load Switching
***	Hot Switching

NOTES

1) The preferred single action path for sources goes through the Connected state, while the preferred single action path for loads goes through the Set state.

2) Multiple connections of a source or a load resource to separate UUT interface points are permitted. Thus, DISCONNECT actions will keep the resource in the Prepared/Ready/Applied state unless all UUT interface connections of the resource are disconnected in which case the state changes to the Set state. Likewise, the resource will stay in the Connected state, where multiple connections could have been made until the last connection is opened at which time the state changes to the Unallocated state.

Figure 11-1. State diagram for a virtual analog source or load resource without an event monitor



WARNINGS:	
*	Hot Source Switching
**	Hot Switiching

NOTES

1) The preferred single action path for sources goes through the Connected state, while the preferred single action path for loads goes through the Set state.

2) Multiple connections of a source or a load resource to separate UUT interface points are permitted. Thus, DISCONNECT actions will keep the resource in the Prepared state unless all UUT interface connections of the resource are disconnected in which case the state changes to the Set state. Likewise, the resource will stay in the Connected state, where multiple connections could be made, until the last connection is opened at which time the state changes to the Unallocated state.

Figure 11-2. State diagram for a virtual analog source or load resource with an event monitor

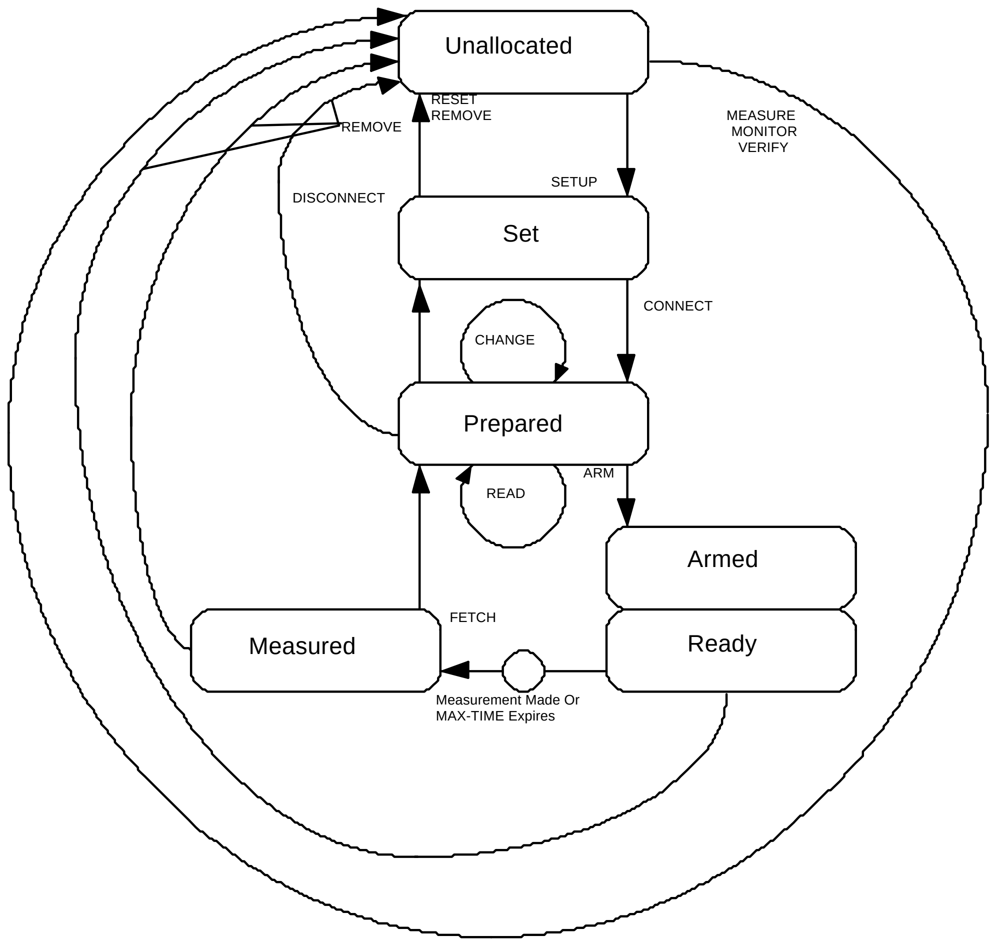


Figure 11-3. State diagram for a virtual analog sensor resource without an event monitor

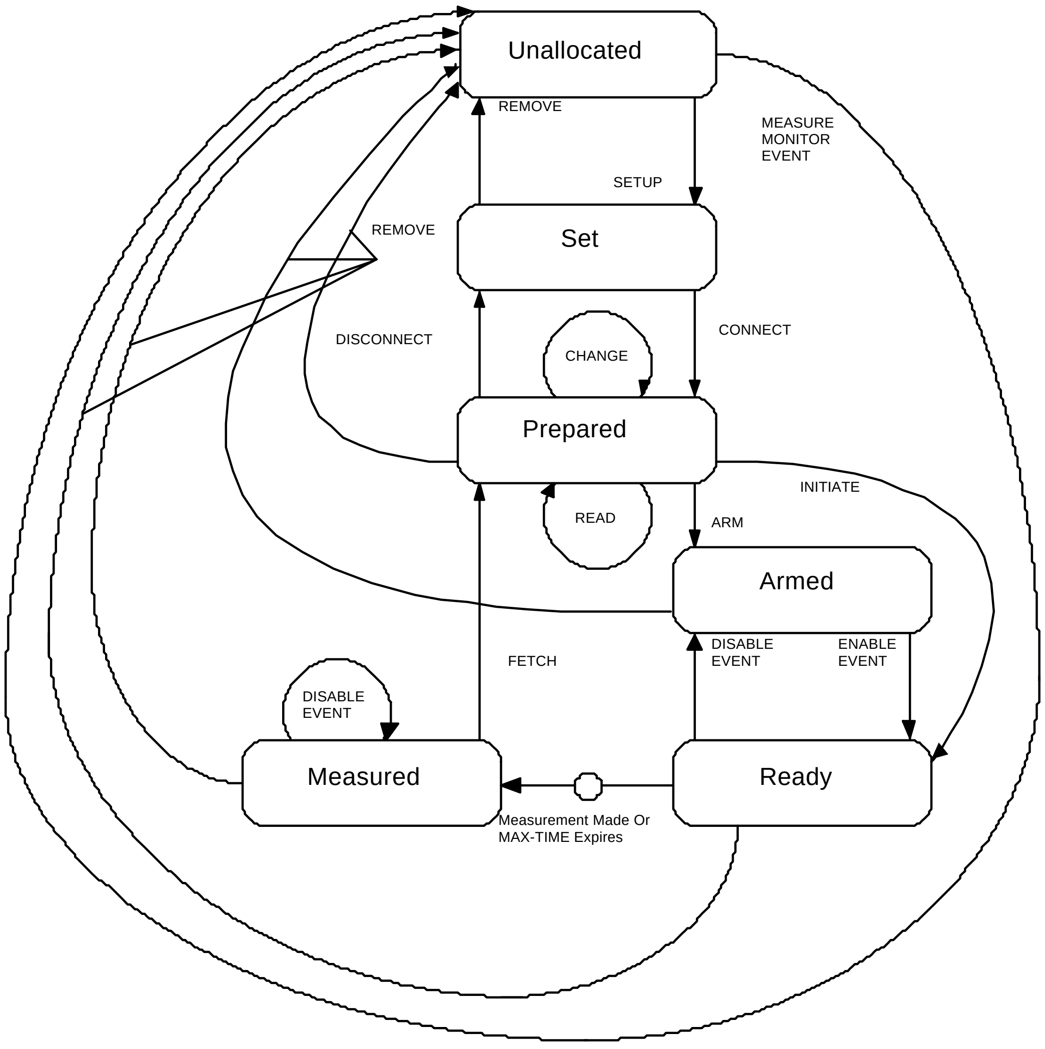


Figure 11-4. State diagram for a virtual analog sensor resource with an event monitor

The state diagrams are pictorial representations of the typical behavior of a virtual resource. The diagrams illustrate the transitions between the various resource states resulting from the imposition of external actions caused by single-action and multiple-action analog verbs. Only state transitions within the given states of one and the same state diagram are defined. Transitions between states belonging to different diagrams are not permitted. The single action verbs may or may not result in state transitions depending on the nature of the action. For example, the CHANGE verb causes a virtual resource to remain in the same state, the CONNECT or DISCONNECT verbs only cause state transitions when the first connection is included in the connection set to be CONNECTed and the last connection is included in the connection set to be DISCONNECTed. There is the greatest possible degree of freedom of single-action verb transitions, especially in the diagrams covering SOURCE and LOAD virtual resources. "Hot Switching" (opening or closing of contacts causing or interrupting current flow) is a well known problem to be avoided, where applicable. Warnings on certain CONNECT or DISCONNECT transitions are included and should be passed on to the language processor implementor or to writers of the user's manual.

The multiple-action verb transitions usually cover more than one resource state and allow smaller degrees of freedom for the implementer and user. However, these multiple-action verb transition paths are intended for the majority of analog test situations. Where a transition path of any multiple- action verb is not suited for a particular test situation, single-action verb sequences may be used.

All single-action and multiple-action paths shown in all state diagrams are fully consistent with the definition of the corresponding test action verb and with the model for interactions between the UUT and the virtual test station given in figure 11-5.

Figures 11-1 through 11-4 illustrate the states that can occur in virtual resources, with or without the requirement to monitor events, of SOURCE, LOAD, and SENSOR type.

The diagrams assume that once a SOURCE, LOAD, or SENSOR resource has been prepared for generating a signal or making a measurement, the test function is completed before the resource is available for the next function. Changes can be made to a resource without necessarily reverting back to a prior state. Such allowable changes are shown in the state diagrams as loops back to the current state. The connections between the test resource and the UUT are maintained during all state diagram transitions, except where the C/ATLAS verb expressly modifies the connections (e.g., REMOVE, CONNECT, DISCONNECT).

Each state diagram represents the combinational sequences of signal oriented statements that cause a particular single virtual resource to assume discrete states during the execution of one test sequence using the particular resource. It shall be recognized that other test functions involving both signal and non-signal oriented statements can be executed in addition to the test sequence that is the subject of a particular state diagram. These other test functions are also controlled by sequences of C/ATLAS statements which, if necessary, can be interpolated into the sequence of statements controlling a test resource related to a particular state diagram. State diagrams represent the states of their affected virtual resources at any given time according to their particular controlling sequence of C/ATLAS statements that impose the states.

The conventions in these state diagrams are as follows:

1. Boxed enclosures contain the states that can be assumed by a resource.

2. Directional linking lines denote those changes of state imposed by C/ATLAS statements.

3. Each directional linking line is annotated with an external action verb that joins the state which is exited because of the action that is imposed to the final state which is entered after the action has occurred.

4. Progress through a state diagram is from the initial unallocated state through functional states with or without recursion to intermediate states and back to the unallocated state.

5. A directional linking line between two states, which bypasses other intermediate states on other lines between the same two states, means that all the bypassed states have in effect been passed through during the transition.

6. A directional line in the form of a loop from one state back to the same state without passing through any other intermediate states means that the external action which annotates the loop causes all the appropriate states bypassed by the loop to be, in effect, passed through during the transition.

7. Directional linking lines flowing from a less functional to a more functional state illustrate the transitions that occur for either the application of a SOURCE or LOAD or the making of a measurement towards completion.

8. Directional linking lines flowing from a more functional to less functional state illustrate the transitions that occur after either the application of a SOURCE or LOAD or the making of a measurement has been completed.

9. When the boxed enclosures of two states are contiguous (in state diagrams with a restricted set of virtual resources) so that a directional linking line is non-existent between the two states, no external action is required for the transition to occur between the two states because they are equivalent for that restricted set of virtual resources.

10. For Source or Load Nouns the DISCONNECT verb may cause more than one transition to be made from a given state. In those instances, the current state will be effectively exited only when the last of a set of connections is broken.

The states that may be attained in the state diagrams are defined in detail by the definitions of the single-action verbs later in this chapter. The following brief descriptions are intended to assist in the comprehension of the state diagrams, but are not to be interpreted as superseding the definitions included for each verb below.

STATE	DESCRIPTION
UNALLOCATED	The instrument is not committed to any particular UUT function and is in a quiescent state available for use.
SET	The resource has been set so that it will operate at the required signal level.
CONNECTED	The resource has been connected to particular UUT pins for subsequent interaction.
PREPARED	The resource is ready to interact with the UUT. However, if the resource requires any form of synchronization to interact with the UUT, it is prepared for these transitions.
ARMED	This state is only used with sensors. It indicates that the sensor is ready to take a measurement, but its associated event detectors, if any are required to take the measurement, are disabled so the measurement will await them.
READY	The instrument is waiting for any events that are required to cause a source to change from quiescent to active or a sensor to make a measurement.
APPLIED	This state is only used with sources and loads. It indicates that the signal has achieved its required active state.
MEASURED	This state is only used with sensor statements. It indicates that the instrument has taken its measurement that is available to be collected by a FETCH action.

A few nouns have attributes that render some single-action verbs inappropriate, reduce the meaning of multiple-action verbs, and impose limits on state diagrams.

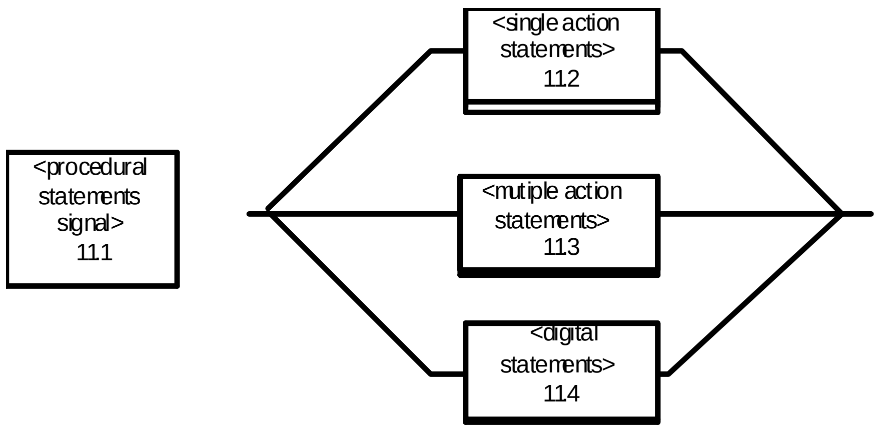
The first category consists of all unsettable nouns such as COMMON, SHORT, EARTH. The single- action verbs SETUP, RESET, and CHANGE are inappropriate. The multiple-action verbs APPLY and REMOVE are used without the SETUP or RESET components, respectively. In the virtual source/load state diagrams in figures 11-1 and 11-2, the Set state vanishes into the Unallocated state and the Connected state vanishes into the Prepared state.

The second category consists of unconnectable nouns such as AMBIENT CONDITIONS. The single-action verbs CONNECT and DISCONNECT are inappropriate. The multiple-action verbs APPLY and REMOVE are used without the CONNECT and DISCONNECT components, respectively. In the virtual source/load state diagrams, the Connected state vanishes into the Unallocated state and the Set state vanishes into the Prepared state.

11.1.2 Formal syntax

Reference procedural-statements-signal

11.1.3 Syntax diagram


<procedural statements signal> 111
<single action statements> 112
<mutiple action statements> 113
<digital statements> 114

11.1.4 Rules

11.1.4.1 Rules for signal routing using multiple action verbs

1. Analog source and load type resources that are attached to the UUT in a procedural statement remain attached to the UUT until an appropriate REMOVE or DISCONNECT statement is executed.

2. Analog sensor type resources that are attached to the UUT in a procedural statement remain attached to the UUT only for the duration of the measurement.

3. Digital source and sensor resources associated with STIMULATE, SENSE, and PROVE statements are attached to the UUT by an ENABLE, DIGITAL CONFIGURATION, and remain attached to the UUT until a REMOVE, ALL, a DISABLE, DIGITAL CONFIGURATION, or an ENABLE, DIGITAL CONFIGURATION of another <configuration> utilizing the same pins is executed.

11.1.4.2 Rules for ANALOG critical actions

1. For Source or Load virtual resources without EVENT MONITOR (figure 11-1), the critical action is any transition that causes the virtual resource to enter the Prepared state or leave the Applied state.

2. For Source or Load virtual resources with EVENT MONITOR (figure 11-2), the critical action is any transition that causes the virtual resource to enter or leave the Applied state.

3. For Sensor virtual resources, the critical action is the transition between the Ready and Measured states.

11.1.4.3 Rules for timing control

1. Unless a digital source or sensor signal statement includes a <when field> or is included in a DO structure, testing proceeds at the natural speed of operation of the test system.

2. A WAIT FOR statement can be used to cause test sequences to be executed at a slower speed than the natural speed of operation of the test system.

3. The <gate field> in an analog signal statement permits the specification of a time interval during which the signal is to be applied or sensed. If a <gate field> is used, it specifies both the instant when the signal starts (the critical action) and the instant when the signal ends.

4. If a <when field> is used in a digital source or sensor statement, it specifies the instant when the critical action is to occur.

5. The <do simultaneous structure> contains a sequence of signal oriented statements whose critical actions are to be executed simultaneously within the limitations of determining time differences for the sequence.

6. The <do timed digital structure> contains a sequence of digital signal oriented statements whose critical actions are to occur sequentially at a rate specified by the timing information given in the <do timed digital structure>.

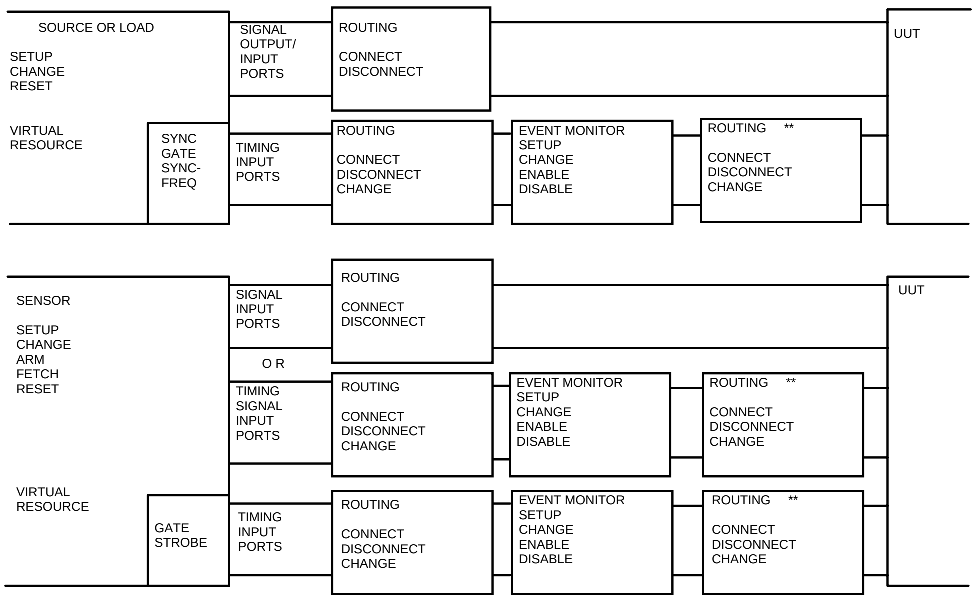
7. When more than one <measured characteristic> is identified in analog signal oriented statements, the initiation of the measurements shall be at the same time.

11.2 Single-action statements

11.2.1 General description

11.2.1.1 Function

Single-action statements are considered basic to the language since each one describes a testing action that cannot be further subdivided with respect to the UUT.



NOTES

* INPUT for LOAD Resources OUTPUT for SOURCE Resources.

** Routing is not applicable to the generation of time based <event>s.

Figure 11-5. Single-action statements illustration

Figure 11-5 illustrates the single action functions (commands) associated with the C/ATLAS single action verbs and their effects on the C/ATLAS virtual resources and associated signal and event paths. All commands, associated parametres, and the handling of measured values are specified in the C/ATLAS single-action statements.

In Figure 11-5, a virtual resource for SOURCE or LOAD has two sets of ports. Signal output ports are those ports through which the signals to the UUT are transmitted. Timing input ports are those ports through which timing information associated with subfields used to generate initial coincidence synchronization, frequency locking, or gated signals is passed from the UUT.

The SENSOR virtual resource has three sets of ports. Signal input ports are those ports through which the signals from the UUT are transmitted in all cases but time measurement. Timing Signal input ports are used instead of Signal input ports for timing measurement to communicate information on the events to be timed into the SENSOR. Timing input ports are those ports associated with the GATE and STROBE fields and are used to control the timing of the measurement.

The system in figure 11-5 is simplified and may be altered in several ways as follows:

1. The TIMING inputs to the SOURCE, SENSOR and LOAD, which control when the signal is generated or measured, may not be present.

2. Each EVENT MONITOR may comprise an EVENT MONITOR with two or more

EVENT MONITORs feeding it through a routing system. In this case the single EVENT MONITOR is closer to the SOURCE, SENSOR, or LOAD on the diagram and the multiple EVENT MONITORs are closer to the UUT. In this case the set of verbs acting on all the EVENT MONITORs is the same. The set of verbs acting on the routing is the same as for other routing in the same path.

3. Each of the EVENT MONITORs described in 2, above, can be similarly constructed.

4. Each of the EVENT MONITORs with only ROUTING between it and the UUT includes a SENSOR. It may have this ROUTING replaced by all of the items that can be between a SENSOR and a UUT. This is because it is conceptually a SENSOR but with the restriction that it is controlled only by those verbs acting on that part of the path in which it occurs, and which are also applicable to the corresponding element in a full sensor.

The single action functions are defined in 11.2.2 through 11.2.12. However, for the purpose of understanding figure 11-5, the following paragraphs give general descriptions:

SETUP

This command provides control and data to the specified resource type (SOURCE, SENSOR, or LOAD), and any associated EVENT MONITOR resource types. SETUP prepares a SOURCE, SENSOR, or LOAD virtual resource and any associated EVENT MONITOR virtual resources for a subsequent test action.

The quantity of data is a function of the resource type and signal. For example, for the DC SIGNAL noun, VOLTAGE may be the only control parametre for a SOURCE; for the AC SIGNAL noun, VOLTAGE and FREQ may be required; for the PULSED DC noun, VOLTAGE, PERIOD, PULSE-WIDTH, etc. may be required.

CONNECT/DISCONNECT

These commands control the operation of some signal routing function that defines a signal path from the C/ATLAS resource to the specified port(s) on the UUT or between C/ATLAS virtual resources. For electrical signals this function may be a relay switching unit, for pneumatic signals it may be a manifold with automatic control of the ports on the UUT side. Note that this active signal path requires that the appropriate connections between the test equipment interface and the UUT shall exist.

ARM

This command is used only with SENSOR type resources and causes the SENSOR to measure the signal value or set of signal values that exists at its input. If any events are referenced by the statement in a GATE, STROBE, FROM/TO, or OF field, the measurement occurs after the events have been enabled by an ENABLE EVENT statement and the events have occurred.

FETCH

This command is used only with SENSOR type resources in association with INITIATE or ARM. It causes a value which has been measured and stored in storage that is local to the SENSOR to be transferred to storage indicated by the statement.

ENABLE EVENT

This command is used to reset any storage in any EVENT MONITOR resources referenced directly by the statement and to cause the EVENT MONITORs to generate an event whenever the event definition is fulfilled. An ENABLE EVENT command that references an event which is already enabled, acts as a RESET of any storage in that EVENT MONITOR.

DISABLE EVENT

This command is used to inhibit an EVENT MONITOR resource from generating further events until it is enabled by an ENABLE EVENT command.

CHANGE

This command is used to alter the SOURCE or LOAD value or the SENSOR range of a virtual resource and some associated EVENT MONITOR resources.

RESET

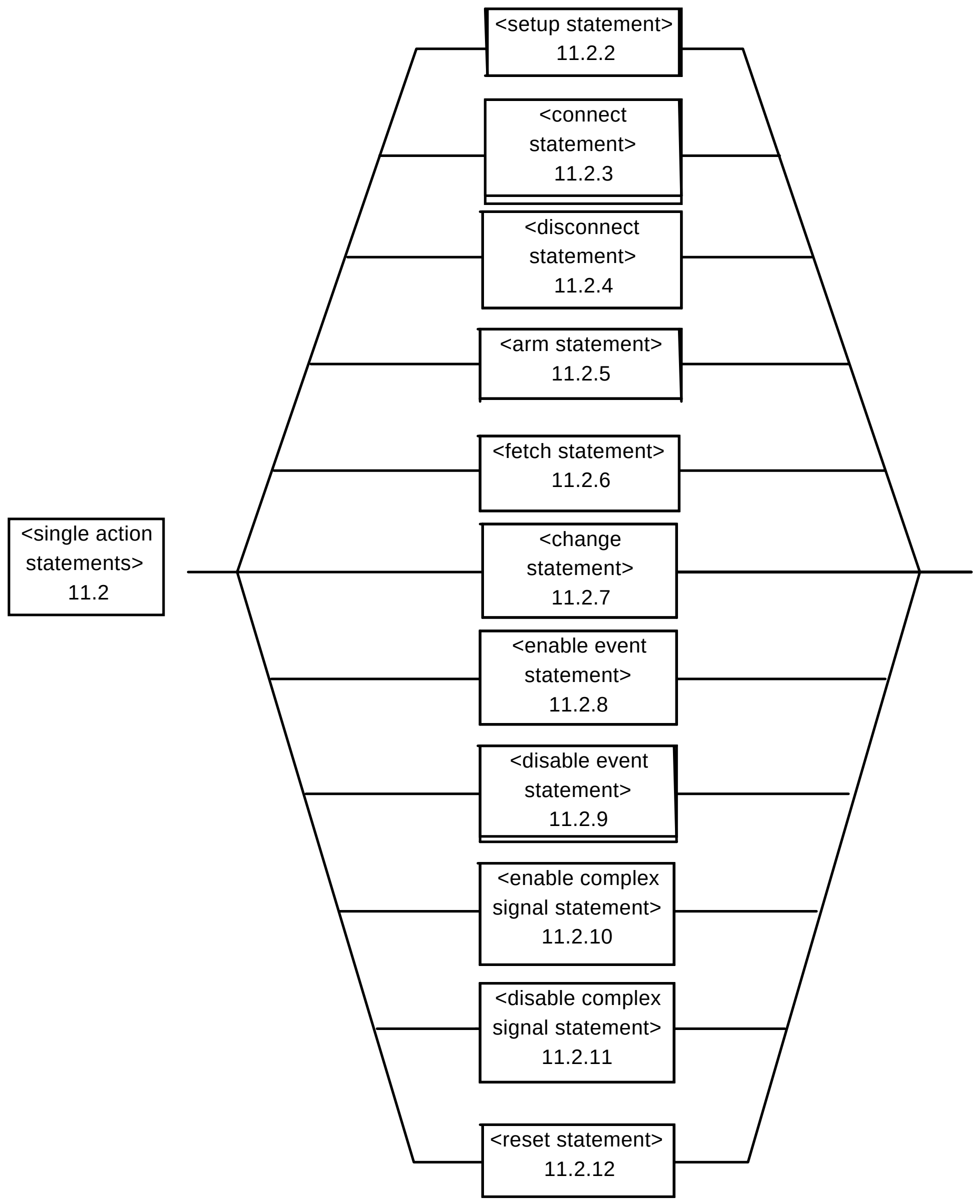
This command causes the virtual resource to be returned to its quiescent state.

The above commands exist for all C/ATLAS resources used in a C/ATLAS procedure. The C/ATLAS procedure may be implemented on a specific test system by assigning specific resources to the virtual resources in the procedure. Specific resources may not have the signal routing functions and will give a null or modified response to the corresponding commands.

11.2.1.2 Formal syntax

Reference single-action-statements.

11.2.1.3 Syntax diagram


<single action statements> 11.2
<setup statement> 11.2.2
<connect statement> 11.2.3
<disconnect statement> 11.2.4
<arm statement> 11.2.5
<fetch statement> 11.2.6
<change statement> 11.2.7
<enable event statement> 11.2.8
<disable event statement> 11.2.9
<enable complex signal statement> 11.2.10
<disable complex signal statement> 11.2.11
<reset statement> 11.2.12

11.2.1.4 Rules

1. A signal oriented statement may refer to one or more <event>s. In this case the signal described in the statement employs one or more virtual resources for monitoring events as well as the virtual resource for the SOURCE, SENSOR, or LOAD.

2. The execution of a single action verb statement affects all the virtual test resources that monitor an <event> as defined by an <identify event based event statement> and called by the single action verb statement.

3. When a number of principal <event>s are defined by more than one <identify event based event statement>, each of which has an OCCURRENCES OF subfield containing a secondary <event> identified previously by a unique name, all <event>s shall be made ready by a single <enable event statement>.

11.2.1.5 Notes and examples

Normally, it will not be necessary to write a UUT test procedure in the level of detail afforded by single-action verbs. The verbs were incorporated in the language primarily as a means of properly defining the multiple-action verbs. These multiple-action verbs will be used for most test statements because they are easier to write than repeating a single-action series each time a test is written. However, it is recognized that there may be special cases where the multiple-action verb is not appropriate. The single action sequence can then be written out.

11.2.2 SETUP statement

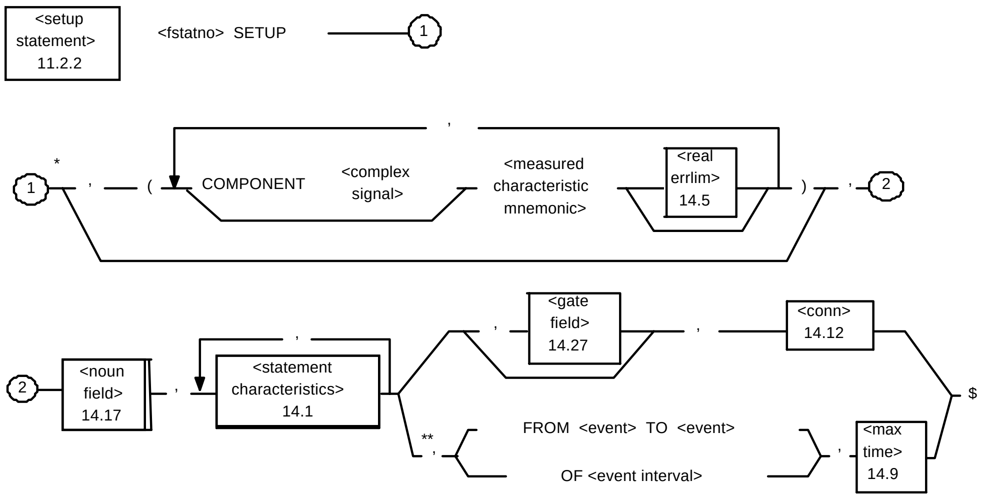
11.2.2.1 Function

The SETUP statement function is to allocate a virtual resource and any associated EVENT MONITORs and to control them so as to generate or receive a defined signal.

11.2.2.2 Formal syntax

Reference setup-statement

11.2.2.3 Syntax diagram



NOTES

*   This path is for measurements only.

**   This path is allowed only for use with the TIME INTERVAL noun.

11.2.2.4 Rules

1. Resources not controlled:

A <setup statement> does not control resources for signal routing.

2. Effect on source and load resources:

A signal provided by a resource is that described by the noun and statement characteristic fields.

The effect on synchronization is as follows:

a) Initial coincidence type of synchronization:

   For initial coincidence type of synchronization the signal will change from a quiescent condition to the stated non-quiescent condition when the synchronization condition is both enabled and fulfilled for the first time after the execution of the SETUP statement.

b) Gate field:

   The signal will be at the stated non-quiescent condition in the time slot defined by the <gate field>. The signal will be at a stated or default quiescent condition at all other times.

c) Frequency type of synchronization:

   For frequency type of synchronization the signal will be in phase locked synchronization with a frequency defined by a repetitive event.

3. The SETUP statement prepares the test resource to support all the specified signal requirements. For all specifications utilizing the <real control characteristic> of the <real characteristic subfield>, the test resource is set to deliver a signal with a specified parametre value. For all specifications utilizing the <real capability characteristic> form of the <real characteristic subfield>, the test resource is ranged to ensure that test operations can proceed in the specified range. For all specifications utilizing the <real limit characteristic> form of the <real characteristic subfield>, the test resource is ranged to provide the signal limits specified.

4. If a noun has no parametres that can be used to control, range, or limit test resources as defined in clause 16 (e.g., COMMON), the setup statement cannot be used.

5. The keyword COMPONENT is only permitted for the <noun field> COMPLEX SIGNAL. The <complex signal> shall be the one defined as being a COMPONENT of the <complex signal> in the <noun field>.

6. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the definition for the referenced component in the <complex signal>.

11.2.3 CONNECT statement

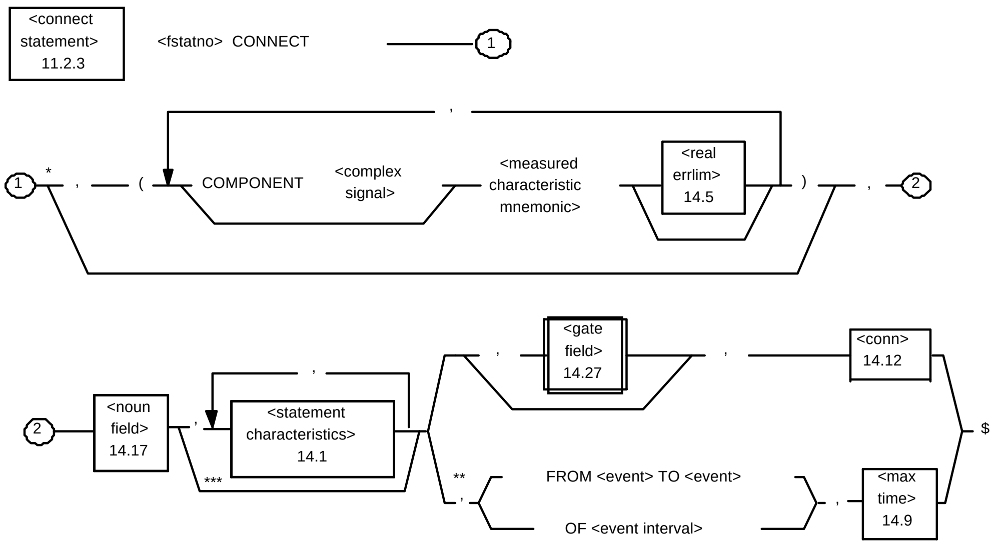
11.2.3.1 Function

The CONNECT statement fastens or joins together the interface points (pin connectors) of the UUT and the interface points of the SOURCE, SENSOR, or LOAD device and any associated EVENT MONITORs and to initiates or gates the signal between the virtual resource and the UUT.

11.2.3.2 Formal syntax

Reference connect-statement

11.2.3.3 Syntax diagram



NOTES

* This path is for measurements only.

** This path is allowed only for use with the TIME INTERVAL noun.

*** This path is allowed for use with COMMON, EARTH, and SHORT only.

11.2.3.4 Rules

1. Resources not controlled:

A <connect statement> has no effect on EVENT MONITOR, SOURCE, SENSOR, or LOAD virtual resources.

2. Effect on UUT to SOURCE, SENSOR, LOAD, or EVENT MONITOR connections:

The signal routing required to carry the signal between the virtual resource and the UUT is completed.

3. Effect on SIGNAL to EVENT MONITOR, and EVENT MONITOR to EVENT MONITOR connections:

If an EVENT MONITOR is employed, all connections between the EVENT MONITOR and the signal virtual resources are established.

4. A <connect statement> cannot be used if a noun has no parametres to connect test resources as defined in clause 16 (e.g., AMBIENT CONDITIONS).

5. The keyword COMPONENT is only permitted for the <noun field> COMPLEX SIGNAL. The <complex signal> shall be the one defined as being a COMPONENT of the <complex signal> in the <noun field>.

6. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the definition for the referenced component in the <complex signal>.

11.2.4 DISCONNECT statement

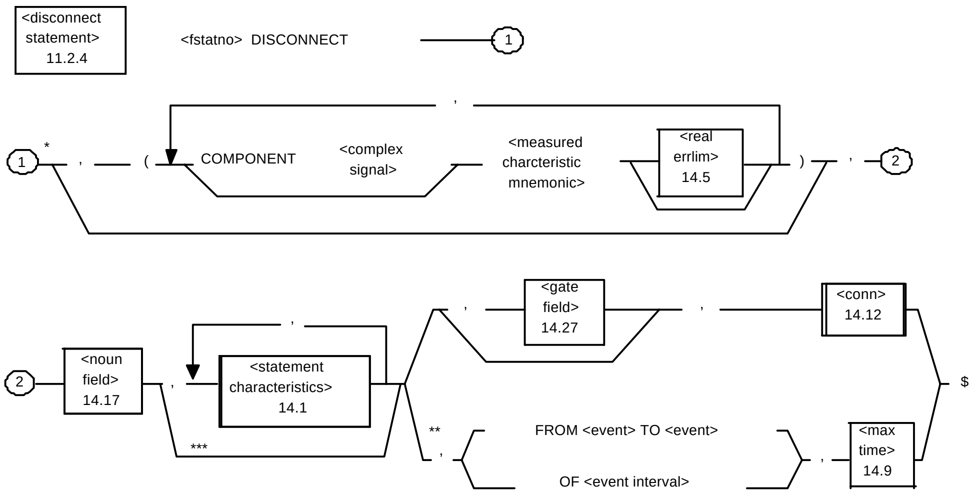
11.2.4.1 Function

The DISCONNECT function is to disconnect the interface points (pin connectors) of the UUT and the interface points of the SOURCE, SENSOR, or LOAD device and any associated EVENT MONITORs (i.e., opposite of CONNECT).

11.2.4.2 Formal syntax

Reference disconnect-statement

11.2.4.3 Syntax diagram



NOTES
* This path is for measurements only.
** This path is allowed only for use with the TIME INTERVAL noun.
*** This path is allowed for use with COMMON, EARTH, and SHORT only.

11.2.4.4 Rules

1. Resources not controlled:

A <disconnect statement> does not control SOURCE, SENSOR, LOAD, or EVENT MONITOR virtual resources.

2. Effect on resources for signal routing:

When a DISCONNECT statement is executed, only the connection paths specified in the CNX field are rendered ineffectual. When the last remaining connections are all covered in the CNX field, then all the connections between associated signal and EVENT MONITOR resources and between these resources and the UUT are rendered ineffectual by setting all controlled switching within a test system associated with the specified signal to a condition that inhibits the transmission of the signal.

3. A <disconnect statement> cannot be used if a noun has no parametres to connect test resources as defined in clause 16 (e.g., AMBIENT CONDITIONS).

4. The keyword COMPONENT is only permitted for the <noun field> COMPLEX SIGNAL. The <complex signal> shall be the one defined as being a COMPONENT of the <complex signal> in the <noun field>.

5. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the definition for the referenced component in the <complex signal>.

11.2.5 ARM statement

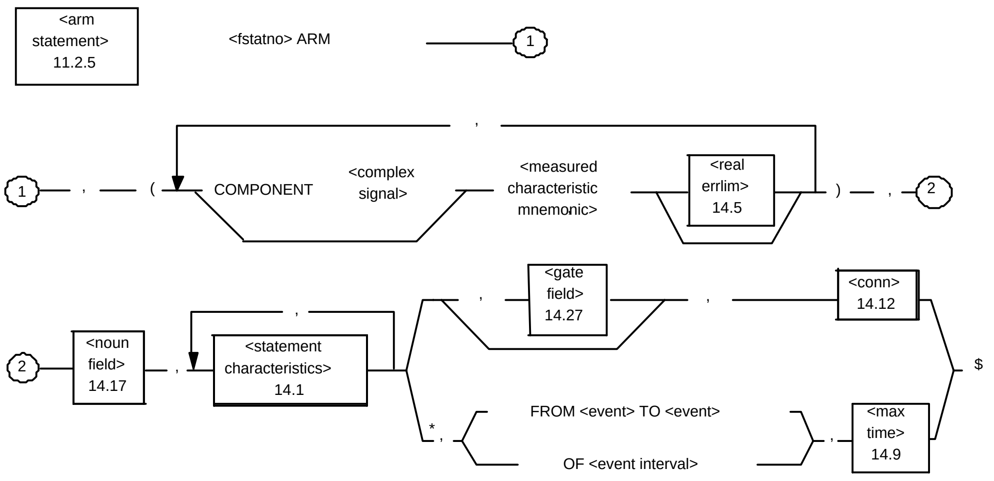
11.2.5.1 Function

The ARM statement function is to induce a measurement cycle by a SENSOR during which the results of the measurement will be determined before transfer by the associated <fetch statement>. The timing and duration of a measurement is described in the statement characteristics field.

11.2.5.2 Formal syntax

Reference arm-statement

11.2.5.3 Syntax diagram



NOTE
* This path is allowed only for use with the TIME INTERVAL noun

11.2.5.4 Rules

1. Resources not controlled:

An <arm statement> does not control SOURCE, LOAD, CONNECTION, or EVENT MONITOR resources. A SOURCE or LOAD virtual resource cannot be used in an ARM statement.

2. Effect on SENSOR resources:

An <arm statement> indicates that a SENSOR resource is required to execute a measurement. A measurement will be taken as soon as the associated timing considerations, which may be dependent on subsequent statements, are satisfied.

A succeeding statement is executed as soon as a SENSOR is ready to take a measurement. This may be prior to the measurement commencing (in fact it may be a necessary preliminary to the measurement commencing). The measurement is then required to be executed asynchronously with the succeeding statements. The SENSOR resource is required to store the measurement in local internal storage until it is extracted by a FETCH statement.

3. The keyword COMPONENT is only permitted for the <noun field> COMPLEX SIGNAL. The <complex signal> shall be the one defined as being a COMPONENT of the <complex signal> in the <noun field>.

4. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the COMPLEX SIGNAL definition for the referenced component in the <complex signal>.

11.2.5.5 Notes and examples

There are a small number of measurements such as MEASURE, (SAMPLE), WAVEFORM where a single measurement returns many numerical values. In these cases, despite the limitation of the definition of ARM, the many numerical values are obtained and are held for subsequent transfer by a <fetch statement> to a RESP <array range> for storage.

11.2.6 FETCH statement

11.2.6.1 Function

The FETCH statement function is to wait for the completion of a measurement cycle induced by an ARM or INITIATE statement and to transfer the measured value(s) as follows:

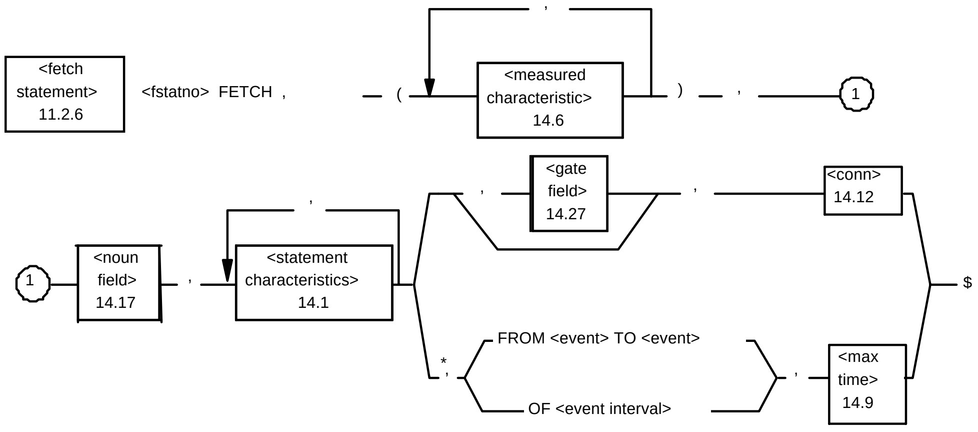
a) For a single measurement value: Into any <data store> in the INTO branch.

b) For multiple measured values: Into the locations within a RESP <array range>.

11.2.6.2 Formal syntax

Reference fetch-statement

11.2.6.3 Syntax diagram



NOTE
* This path is allowed only for use with the TIME INTERVAL noun.

11.2.6.4 Rules

1. If the measurement enabled by the corresponding ARM or INITIATE statement is incomplete, the FETCH statement waits for it to be completed prior to storing the measured value(s).

2. In the measurement of a TIME INTERVAL, where <event>s or an <event interval> is referenced and a MAX-TIME field is specified, if the measurement does not complete

within the time specified by the MAX-TIME field, the measurement will be terminated and the MAX-TIME flag set.

3. FETCH has no direct effect on any resource, other than a SENSOR resource from which it extracts data without affecting the SENSOR status.

4. A <fetch statement> cannot be executed unless an associated explicit <arm statement> or <initiate statement> has been executed. In the case of an <arm statement>, a <fetch statement> cannot be executed unless an associated <enable event statement> has been executed.

11.2.6.5 Notes and examples

In those cases where a single ARM or INITIATE statement returns many numerical values, data is transferred to the designated RESP <array range>.

11.2.7 CHANGE statement

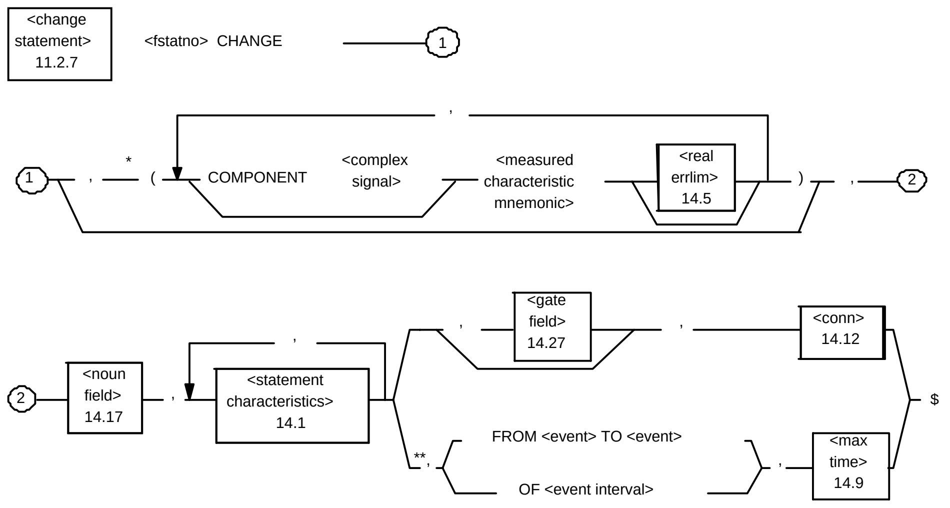
11.2.7.1 Function

This function is to alter some of the characteristics of a test requirement that are active in the testing of a UUT.

11.2.7.2 Formal syntax

Reference change-statement

11.2.7.3 Syntax diagram



NOTES
* This path is for measurements only.
** This path is allowed only for use with the TIME INTERVAL noun.

11.2.7.4 Rules

1. A <change statement> can only be used to alter some of the characteristics of C/ATLAS statements that are active in the testing of a UUT.

2. A <change statement> cannot be used to alter in a statement being executed the content of one of the following:

<noun field>,

<gate field>,

<sync subfield>, (used for synchronization reference)

<conn> field.

3. A <change statement> can be used to alter in the statement being executed the content of a:

<real errlim> in the <measured characteristic mnemonic> branch,

<statement characteristics> field except <sync subfield> when used for synchronization reference,

FROM <event> TO <event> field,

OF <event interval> field,

<max time> field.

4. A <change statement> maintains the already established connections between the UUT and the SENSOR, SOURCE, or LOAD test resources.

5. A <change statement> can modify connections to EVENT MONITORs.

6. A <change statement> cannot be used to alter the characteristics of a SENSOR between the executions of ARM and FETCH actions.

7. A <change statement> neither enables nor disables any EVENT MONITOR.

8. The keyword COMPONENT is only permitted for the <noun field> COMPLEX SIGNAL. The <complex signal> shall be the one defined as being a COMPONENT of the <complex signal> in the <noun field>.

9. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the COMPLEX SIGNAL definition for the referenced component in the <complex signal>.

11.2.8 ENABLE EVENT statement

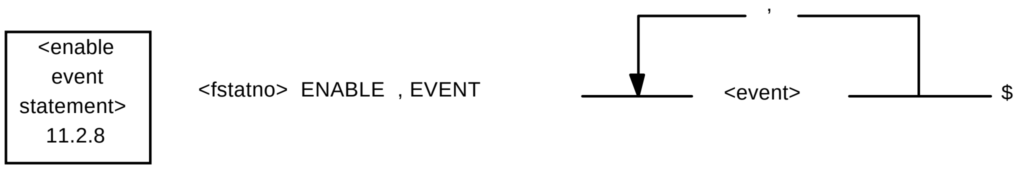
11.2.8.1 Function

This statement makes ready an EVENT MONITOR for the detection and indication that its identified <event> has occurred.

11.2.8.2 Formal syntax

Reference enable-event-statement

11.2.8.3 Syntax diagram



11.2.8.4 Rules

1. An <enable event statement> does not control SOURCE, SENSOR, LOAD, or signal routing resources.

2. If an EVENT definition includes an occurrence subfield, then the occurrence count is reset to zero.

3. When an IDENTIFY <event> definition references other <event>s, all <event>s shall either be in the enabled state or shall be made ready explicitly by the same ENABLE statement.

4. If the event referenced is already enabled, that event is reset by the <enable event statement>.

5. If variables were used for any of the quantities or connections in the associated <identify statement>, values shall be assigned to these variables prior to the execution of the associated <enable event statement>. When this ENABLE statement is executed, the current value of these variables is used to complete the <event> specification. Succeeding changes of these values do not affect the enabled <event>.

11.2.9 DISABLE EVENT statement

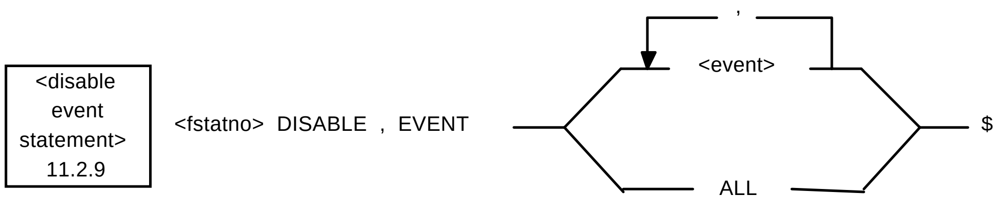
11.2.9.1 Function

This statement function is to prevent an EVENT MONITOR from detecting and indicating that its identified <event> has occurred.

11.2.9.2 Formal syntax

Reference disable-event-statement

11.2.9.3 Syntax diagram



11.2.9.4 Rules

1. A <disable event statement> does not control SOURCE, SENSOR, LOAD, or signal routing resources.

2. When an IDENTIFY <event> definition is referenced by other <event>s, all referencing <event>s shall either be in the disabled state or shall be disabled explicitly by the same DISABLE statement.

3. A DISABLE, EVENT ALL statement disables all currently ENABLEd <event>s.

11.2.10 ENABLE COMPLEX SIGNAL statement

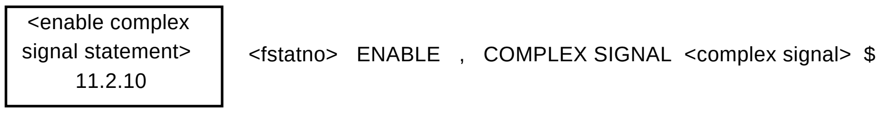
11.2.10.1 Function

This statement enables a defined COMPLEX SIGNAL so that it may be referenced and operated upon by a procedural signal oriented statement.

11.2.10.2 Formal syntax

Reference enable-complex-signal-statement

11.2.10.3 Syntax diagram


<enable complex signal statement> 11.2.10
<fstatno> ENABLE , COMPLEX SIGNAL <complex signal> $

11.2.10.4 Rules

1. The enabling of a COMPLEX SIGNAL will enable all component signals referenced within the identified <complex signal structure>, which are themselves separately defined COMPLEX SIGNALs.

2. The specific actions of an enable are:

a) First, connect any interface/UUT signal connections specified in the SPECIFY statements as component signals.

b) Next, set up and connect any virtual resources identified in the SPECIFY statements for component signals.

c) Next, set up and connect any EVENT MONITORS required by any <event>s identified in the SPECIFY statement.

3. The COMPLEX SIGNAL itself is not connected to any UUT interface by the ENABLE actions, this is only accomplished within the procedural statement referencing this signal <label>.

11.2.11 DISABLE COMPLEX SIGNAL statement

11.2.11.1 Function

This statement disables a defined COMPLEX SIGNAL so that it may not be referenced and operated upon by a procedural signal oriented statement.

11.2.11.2 Formal syntax

Reference disable-complex-signal-statement

11.2.11.3 Syntax diagram

```
<disable
complex signal
statement>          <fstatno>  DISABLE  ,  COMPLEX SIGNAL <complex signal>  $
11.2.11
```

11.2.12 RESET statement

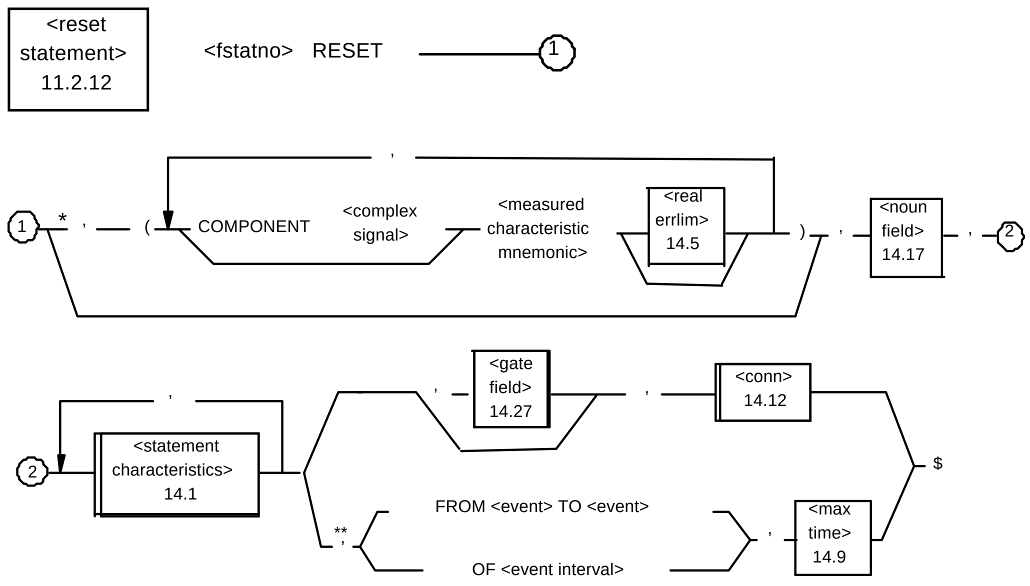
11.2.12.1 Function

The RESET statement causes a signal to return to its unallocated quiescent state and to make its virtual resources available for re-use in a SETUP or multiple action statement.

11.2.12.2 Formal syntax

Reference reset-statement

11.2.12.3 Syntax diagram



NOTES
* This path is for measurements only.
** This path is allowed only for use with the TIME INTERVAL noun.

11.2.12.4 Rules

1. If a noun has no parametres that can be used to control, range, or limit test resources as defined in clause 16 (e.g., COMMON), the <reset statement> cannot be used.

2. SOURCE, SENSOR, and LOAD virtual resources are set to a defined or default quiescent input level.

3. A <reset statement> does not control EVENT MONITOR or connection of virtual resources, but it does release them for use by a subsequent SETUP statement.

4. The keyword COMPONENT is only permitted for the <noun field> COMPLEX SIGNAL. The <complex signal> shall be the one defined as being a COMPONENT of the <complex signal> in the <noun field>.

5. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the COMPLEX SIGNAL identified for the referenced component in the <complex signal>.

11.3 Multiple-action statements

11.3.1 General description

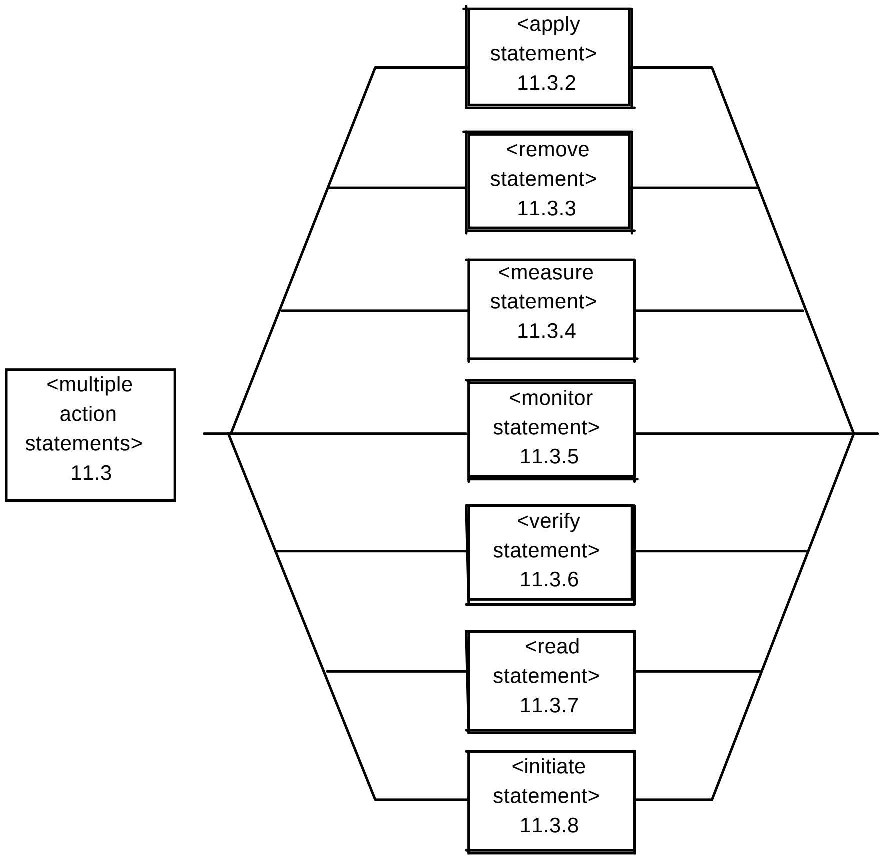
11.3.1.1 Function

Multiple-action statements are functionally equivalent to sequences of single-action signal statements and other non-signal procedural statements.

11.3.1.2 Formal syntax

Reference multiple-action-statements

11.3.1.3 Syntax diagram


<multiple action statements> 11.3
<apply statement> 11.3.2
<remove statement> 11.3.3
<measure statement> 11.3.4
<monitor statement> 11.3.5
<verify statement> 11.3.6
<read statement> 11.3.7
<initiate statement> 11.3.8

11.3.2 APPLY statement

11.3.2.1 Function

The APPLY statement accomplishes the following actions for a SOURCE or LOAD resource type supporting the different analog stimulus or load noun categories:

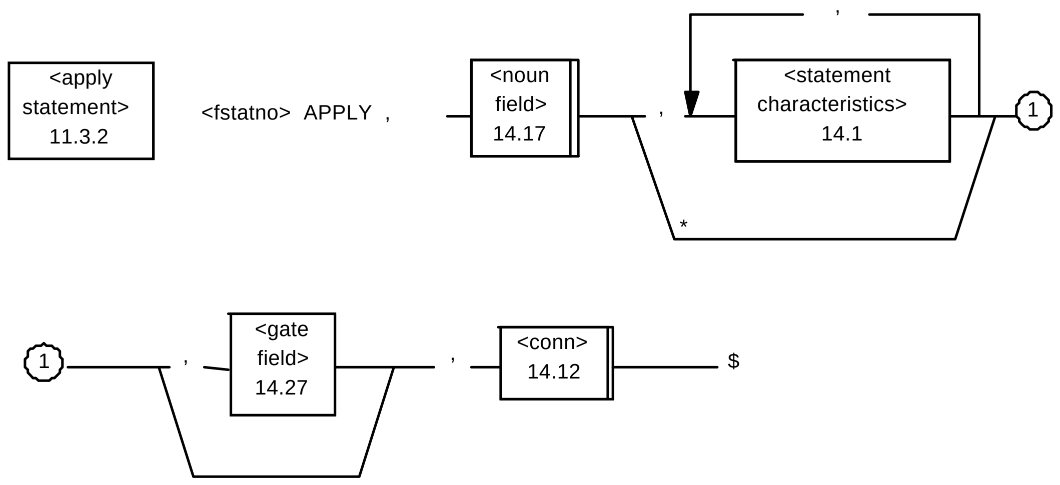
Source noun	Load noun
CONNECT	SETUP
SETUP	CONNECT
* ENABLE, EVENT	ENABLE, EVENT

NOTE
* For resources with EVENT MONITOR capabilities

11.3.2.2 Formal syntax

Reference apply-statement

11.3.2.3 Syntax diagram



NOTE

* This path is only used for nouns without capability to control, range, or limit resource parametres as defined in clause 16 (e.g., COMMON).

11.3.2.4 Rules

For resources with EVENT MONITOR capabilities the APPLY statement completes all single actions necessary to alter the affected resource from its Unallocated to its Ready state awaiting the triggering event. The Applied state is reached after the enabled EVENT has occurred.

11.3.3 REMOVE statement

11.3.3.1 Function

The REMOVE statement accomplishes the following actions for SOURCE, SENSOR, or LOAD resource types supporting the different analog stimulus, sensor, or load noun categories:

Source noun	Load noun	Sensor Noun
* DISABLE, EVENT	* DISABLE, EVENT	* DISABLE, EVENT
RESET	DISCONNECT	DISCONNECT
DISCONNECT	RESET	RESET

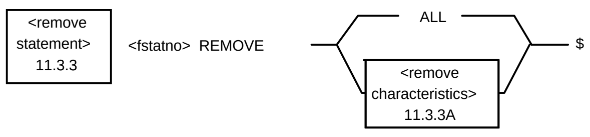
NOTE
* For resources with EVENT MONITOR capability.

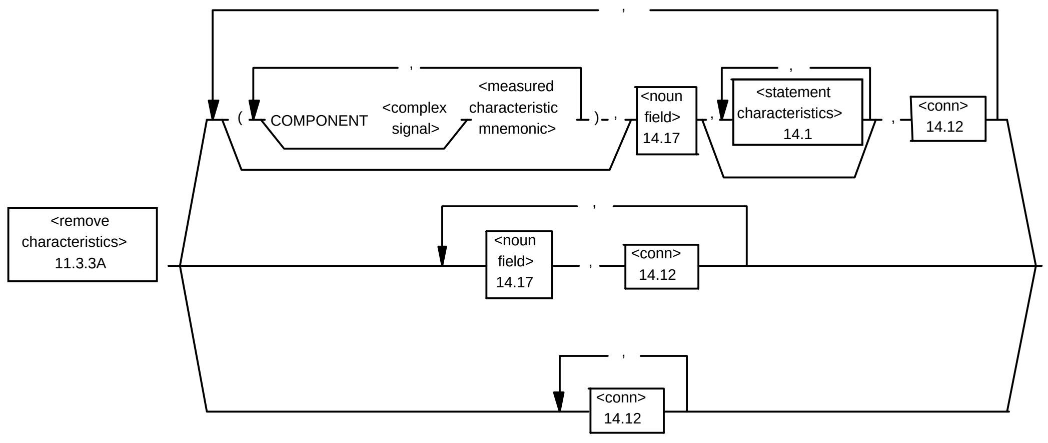
The REMOVE statement may perform fewer actions than those listed when the affected resource is in a transitional state between fully active and unallocated.

11.3.3.2 Formal syntax

Reference remove-statement

11.3.3.3 Syntax diagram





NOTE

* This branch is used only for COMMON, SHORT, and EARTH nouns.

11.3.3.4 Rules

1. Previously made signal paths may be referenced in a REMOVE statement by citing the UUT connection points (e.g., J1-9, J2-0). If more than one signal path has been made to the same UUT connection point, the appropriate stimulus-measurement noun may be written to identify one particular path. (e.g., DC VOLTAGE and J1-9 J1-10.)

2. A REMOVE, ALL statement may be used without referencing any signal paths in order to remove every existing signal to the UUT.

3. If a measurement has been initiated by an ARM or INITIATE statement, execution of a <remove statement> affecting the associated virtual resource causes the measurement to be terminated with undefined data in any storage referenced in an INTO subfield or a RESP field.

4. The keyword COMPONENT is only permitted for a COMPLEX SIGNAL noun. The <complex signal> shall be the one defined as being a COMPONENT of the <complex signal> in the <noun field>.

5. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the COMPLEX SIGNAL definition for the referenced component in the <complex signal>.

11.3.3.5 Notes and examples

```
654321 REMOVE, CNX HI J1-2 LO J1-3,
       CNX X J1-A Y J1-BB Z J1-C $
    23 REMOVE, DC SIGNAL, CNX HI J1-5 LO J1-6,
       PULSED DC, CNX HI J8-1 LO J8-0 $
```

11.3.4 MEASURE statement

11.3.4.1 Function

The MEASURE statement accomplishes the following actions for a SENSOR resource type in the sequence listed:

SETUP

CONNECT

ARM

* ENABLE, EVENT

FETCH

* DISABLE, EVENT

DISCONNECT

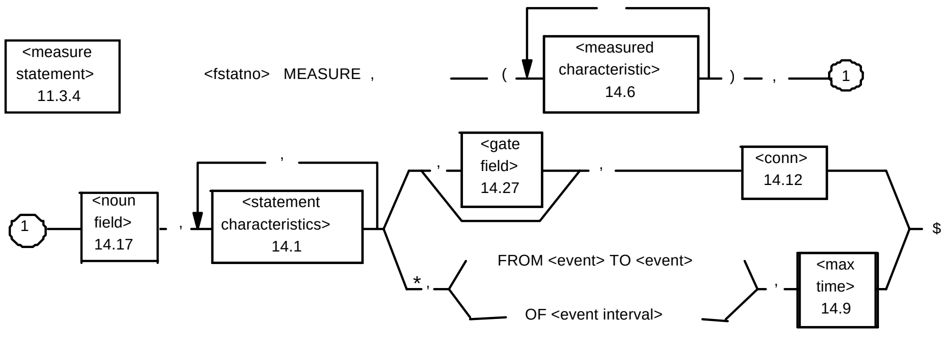
RESET

NOTE
* For resources with EVENT MONITOR capabilities.

11.3.4.2 Formal syntax

Reference measure-statement

11.3.4.3 Syntax diagram



NOTE

* This path is allowed only for use with the TIME INTERVAL noun.

11.3.5 MONITOR statement

11.3.5.1 Function

The MONITOR function accomplishes the following sensor functions in the sequence/loop shown until a manual intervention event directs the program to proceed to the next step:

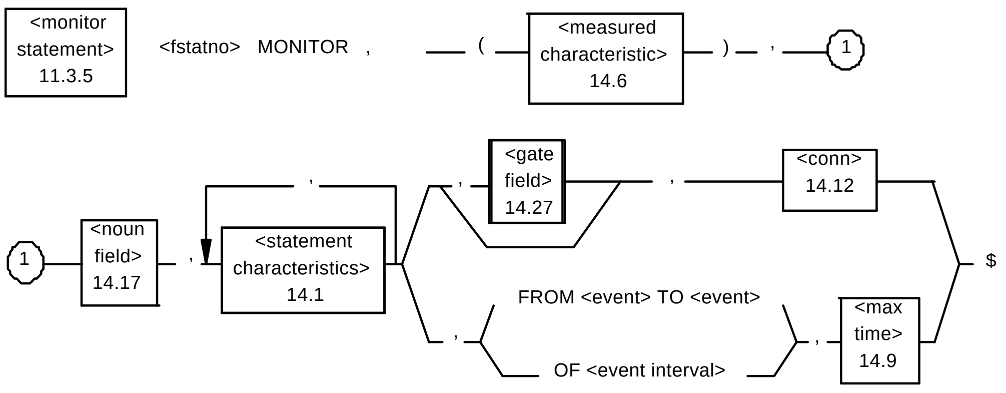
```
      SETUP
      CONNECT
*     ENABLE, EVENT
      ENABLE, manual intervention event
      WHILE, no manual intervention event indicated,
              THEN
              ARM
              FETCH
              OUTPUT, reading
      END, WHILE
      DISABLE, manual intervention event
*     DISABLE, EVENT
      DISCONNECT
      RESET
```

NOTE
* For resources with EVENT MONITOR capabilities.

11.3.5.2 Formal syntax

Reference monitor-statement

11.3.5.3 Syntax diagram



NOTE

* This path is allowed only for use with the TIME INTERVAL noun.

11.3.6 VERIFY statement

11.3.6.1 Function

The VERIFY statement accomplishes the following sensor functions in the sequence listed:

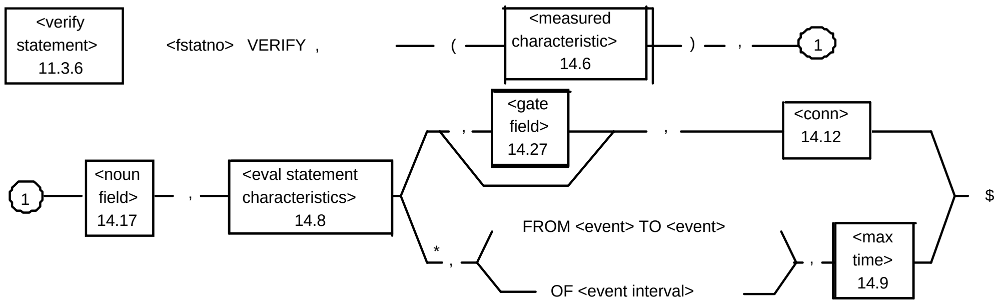
```
          SETUP
          CONNECT
          ARM
     *    ENABLE, EVENT
          FETCH
          COMPARE
     *    DISABLE, EVENT
          DISCONNECT
          RESET
```

NOTE
* For resources with EVENT MONITOR capability

11.3.6.2 Formal syntax

Reference verify-statement

11.3.6.3 Syntax diagram



NOTE
* This path is allowed only for use with the TIME INTERVAL noun.

11.3.7 READ statement

11.3.7.1 Function

The READ statement establishes the present value of a sensor function that exists at the instant of measuring and retains that value in the INTO location or RESP locations specified.

READ accomplishes the following functions in the sequence listed:

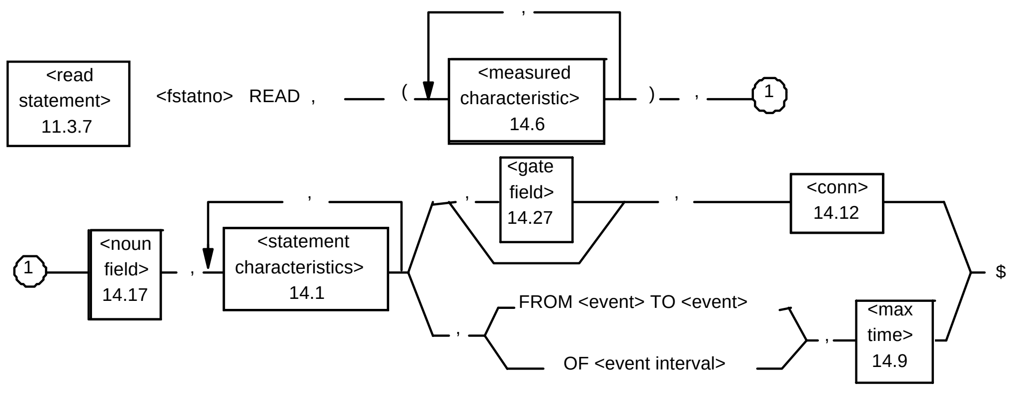
```
            ARM
      *     ENABLE, EVENT
            FETCH
```

NOTE
* For resources with EVENT MONITOR capability

11.3.7.2 Formal syntax

Reference read-statement

11.3.7.3 Syntax diagram



NOTE
* This path is allowed only for use with the TIME INTERVAL noun.

11.3.8 INITIATE statement

11.3.8.1 Function

An <initiate statement> is a multiple action statement that executes the following single-action statements in the order listed.

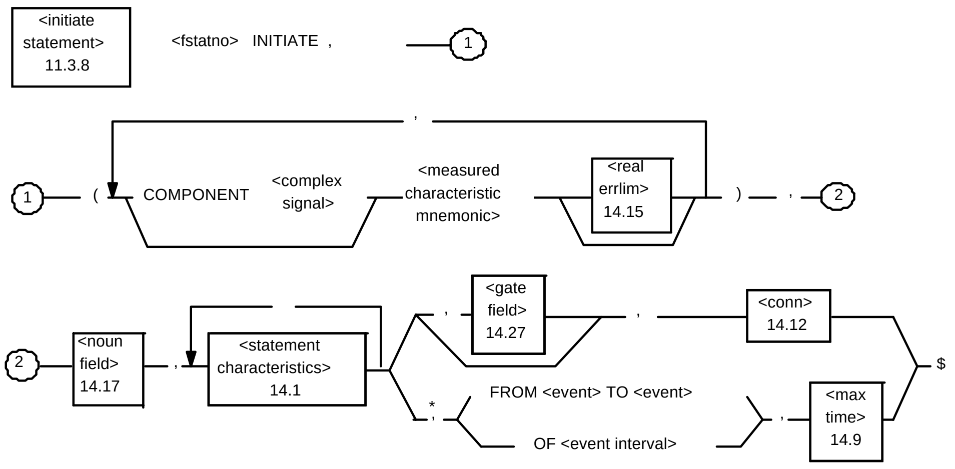
```
          ARM
     *    ENABLE, EVENT
```

NOTE
* For resources with EVENT MONITOR capability

11.3.8.2 Formal syntax

Reference initiate-statement

11.3.8.3 Syntax diagram



NOTE
*   This path is allowed only for use with the TIME INTERVAL noun.

11.3.8.4 Rules

1. After execution of an <initiate statement>, the only statements permitted to operate on the measurement resource thus initiated are the <fetch statement> and the <remove statement>.

2. The keyword COMPONENT is only permitted for a COMPLEX SIGNAL noun. The <complex signal> shall be the one defined as being a COMPONENT of the <complex signal> in the <noun field>.

3. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the COMPLEX SIGNAL definition for the referenced component in the <complex signal>.

11.4 Digital statements

11.4.1 General description

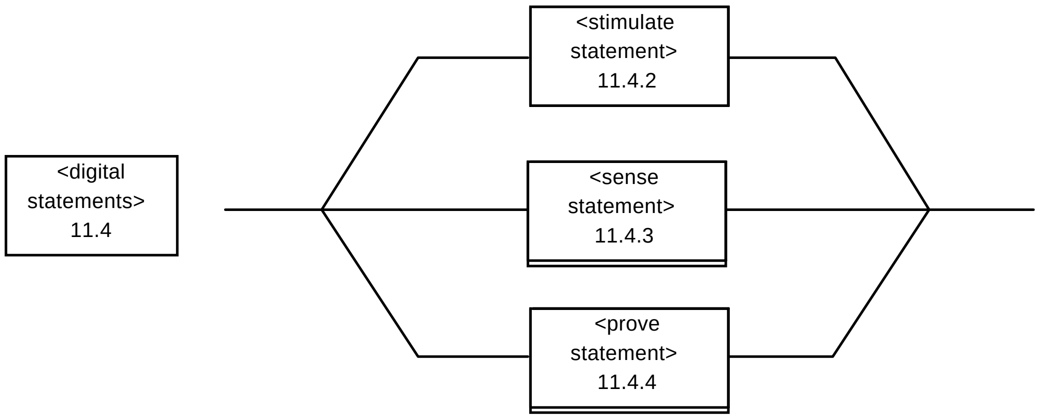
11.4.1.1 Function

The functions of the digital verbs are to provide the testing actions necessary to transmit and receive digital logic signals to and from the UUT. The relationship between the digital data transferred via these statements and the source logic signal drivers and sensor logic signal receivers is defined in the <define digital configuration structure>.

11.4.1.2 Formal syntax

Reference digital-statements

11.4.1.3 Syntax diagram



11.4.2 STIMULATE statement

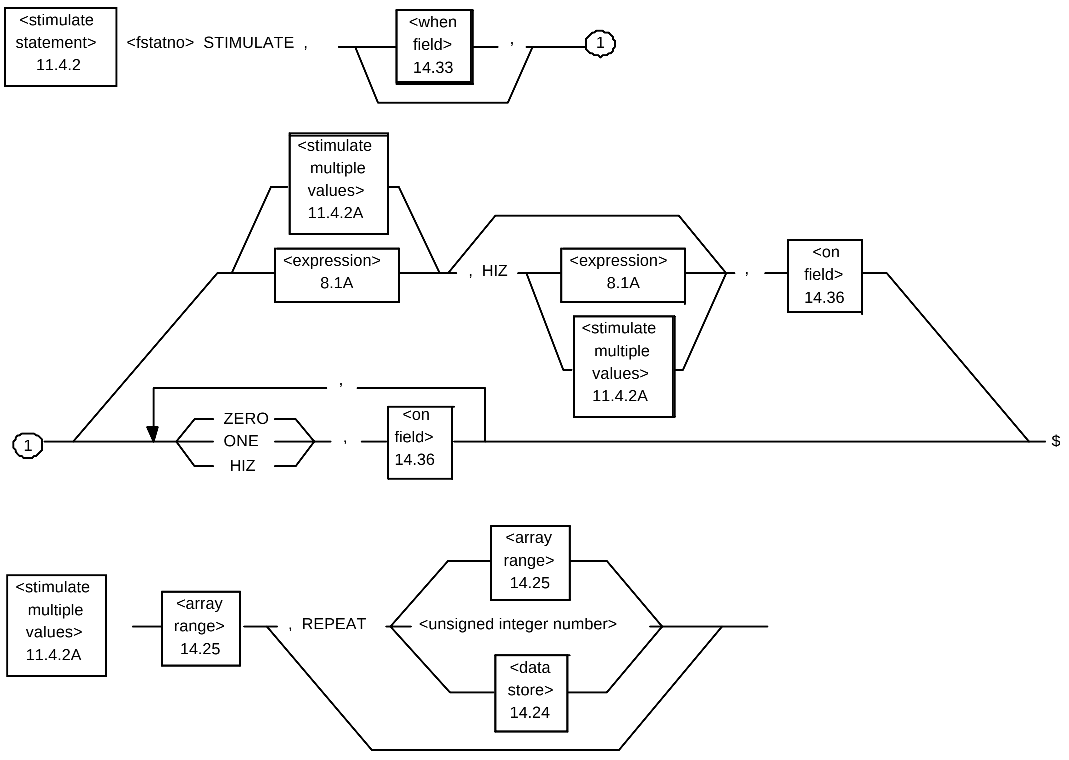
11.4.2.1 Function

The function and critical action of the <stimulate statement> is to update the logic-signal drivers of an ENABLEd <configuration>. Logic states can be manipulated using either word or bit level constructs.

11.4.2.2 Formal syntax

Reference stimulate-statement

11.4.2.3 Syntax diagram



11.4.2.4 Rules

1. ZERO, ONE, and HIZ correspond to LEVEL-LOGIC-ZERO, LEVEL-LOGIC-ONE, and LEVEL-LOGIC-HIZ respectively of the ENABLEd <configuration>. When the drivers of the ENABLEd <configuration> have been set to the high impedance condition, they will remain in this condition until re-driven by a subsequent STIMULATE, ONE/ZERO/digital expression statement.

2. Parallel logic signals can be operated on by word-level and bit-level constructions of STIMULATE. Serial logic signals can only be operated on by word level constructions. Multiple words can only be operated on using the <array range> subfield within a STIMULATE statement included in a DO TIMED DIGITAL structure. The transfer rate will be in accordance with the STIM-RATE or STIM-EVENT specified in the DO TIMED DIGITAL structure.

3. Although a <stimulate statement> may contain any number of ZERO, ONE, and HIZ fields, the updating of logic-signal drivers will still occur as a single (critical) action. The updating will represent the cumulative states specified as determined by the order of writing.

4. For parallel signals, the <expression> or each element of <array range> shall evaluate to a STRING OF BIT whose dynamic length is equal to the number of signals as defined by the <on field>. If the <on field> has no CNX, the dynamic length shall equal the number of

connections defined in the <define digital source statement>. If HIZ is present, a logic ONE in the HIZ <expression> will result in an HIZ state on the corresponding pin, and a logic ZERO in the HIZ <expression> will result in the digital value in the preceding <expression>, appearing on the corresponding pin.

5. For serial signals, the <expression> or each element of <array range> shall evaluate to a STRING OF BIT whose dynamic length is equal to the WORD-LENGTH as defined by the <digital source> being STIMULATEd.

6. The time of the critical action of STIMULATE can be specified in the <when field>. If the <when field> is omitted the timing of the critical action is uncontrolled unless it is specified by DO/END DO structures.

7. The critical action of a <stimulate statement> for a serial signal is the start of the first bit period.

8. The REPEAT field, when present, specifies the number of times that an array of STIMULATE data words is to be repeated before executing the next statement.

a) A single REPEAT value, either an integer constant or a <data store> of type integer, specifies the number of times to transfer the entire STIMULATE array. The value of the integer constant or <data store> shall be a non-zero positive integer.

b) A REPEAT array specifies the number of times to transfer each individual word of the STIMULATE array.

1) Each element of the REPEAT array specifies the number of times that the corresponding indexed STIMULATE word is to be transferred before the next word in the STIMULATE array.

2) The number of elements of the REPEAT array used shall equal the number of elements used from the STIMULATE array.

3) Each element of the REPEAT array shall be a non-zero positive integer.

9. In one <stimulate statement> it is only allowed to stimulate parallel data or one logical serial channel.

10. If the <stimulate statement> or its referenced <define digital source statement> contains references to <event>s that are not already enabled, the <event>s are enabled at the commencement of execution of the <stimulate statement>.

11.4.3 SENSE statement

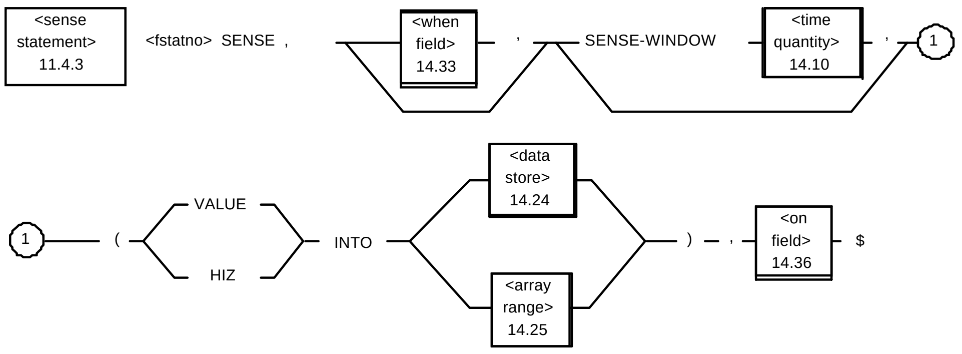
11.4.3.1 Function

The <sense statement> has two principal functions. The first and critical function is to cause the sensors of an ENABLEd <configuration> to sense and digitize UUT responses. The second action fills data stores with the digitized data. Logic states (VALUE) and high-impedance states (HIZ) can be SENSEd.

11.4.3.2 Formal syntax

Reference sense-statement

11.4.3.3 Syntax diagram



11.4.3.4 Rules

1. The INTO <data store> and element of INTO <array range> shall be of type STRING OF BIT. For parallel signals, the dynamic length will be set equal to the number of signals defined in the <on field>. For serial signals, the dynamic length will be set equal to the WORD-LENGTH in the associated <define digital sensor statement>. In both cases the declared length shall be equal to or greater than the dynamic length.

2. The time of the critical action of SENSE can be specified in the <when field>. If the <when field> is omitted the timing of the critical action is uncontrolled unless it is specified by DO/END DO structures.

3. When the ENABLEd <configuration> includes SERIAL signals, the critical action of SENSE is the time at which the first bit is expected.

4. Multiple sense actions can only be operated on by using the <array range> subfield within a SENSE statement included in a DO TIMED DIGITAL structure. The sense rate will be in accordance with the SENSE-RATE, SENSE-EVENT, or SENSE-DELAY specified in the DO TIMED DIGITAL structure.

5. For SENSE, HIZ statements, the high impedance state is represented in the <data store> or <array range> by binary "1" and all other states are represented by binary "0".

6. If the referenced <digital sensor> has an ILLEGAL-STATE-INDICATOR and the SENSEd physical characteristics are not within any of the defined logic limits, then the associated <boolean variable> will be set TRUE.

7. In one <sense statement> it is only allowed to sense parallel data or one logical serial channel.

8. If the <sense statement> or its referenced <define digital sensor statement> contain references to <event>s that are not already enabled, the <event>s are enabled at the commencement of execution of the <sense statement>. Any <event>s that are enabled as a result of this rule are disabled at the completion of the execution of the <sense statement>.

9. If the <on field> has no CNX, the dynamic length equals the number of connections defined in the <define digital sensor statement>.

10. The SENSE-WINDOW path is used to allow parallel data to be sensed over a period of time.

For parallel data, where the logic states are defined in the <define digital configuration structure> as LEVEL-LOGIC-ONE, LEVEL-LOGIC-ZERO, LEVEL-LOGIC-HIZ, or LEVEL-LOGIC- QUIES, the logic state is only achieved if it is constant throughout the entire period. If it is not constant, then the ILLEGAL-STATE-INDICATOR variable is set to TRUE. (Reference 6.16.4.4, rule 19.)

For parallel data, where the logic levels are defined in the <define digital configuration structure> as TRANS-LOGIC-ONE or TRANS-LOGIC-ZERO, the logic state is achieved if only one transition occurs throughout the period. If no transitions occur, or more than one transition occurs, then the ILLEGAL-STATE-INDICATOR variable is set to TRUE.

11.4.4 PROVE statement

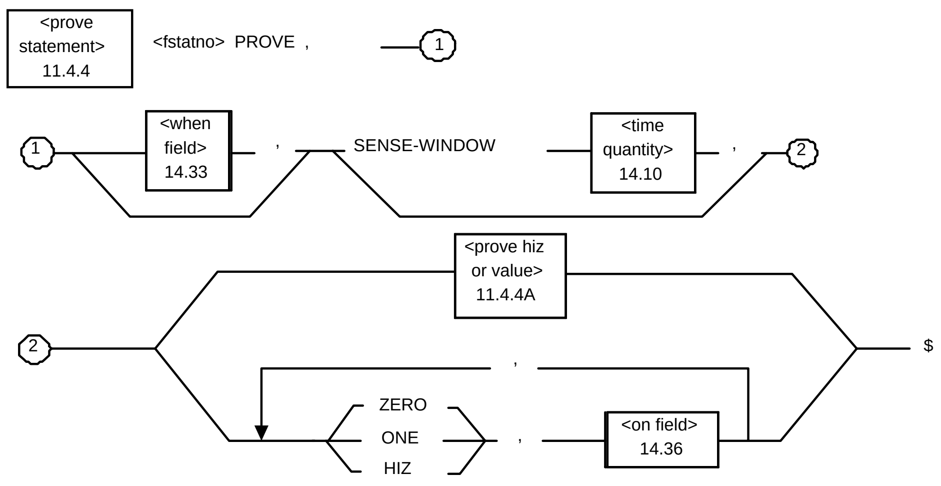
11.4.4.1 Function

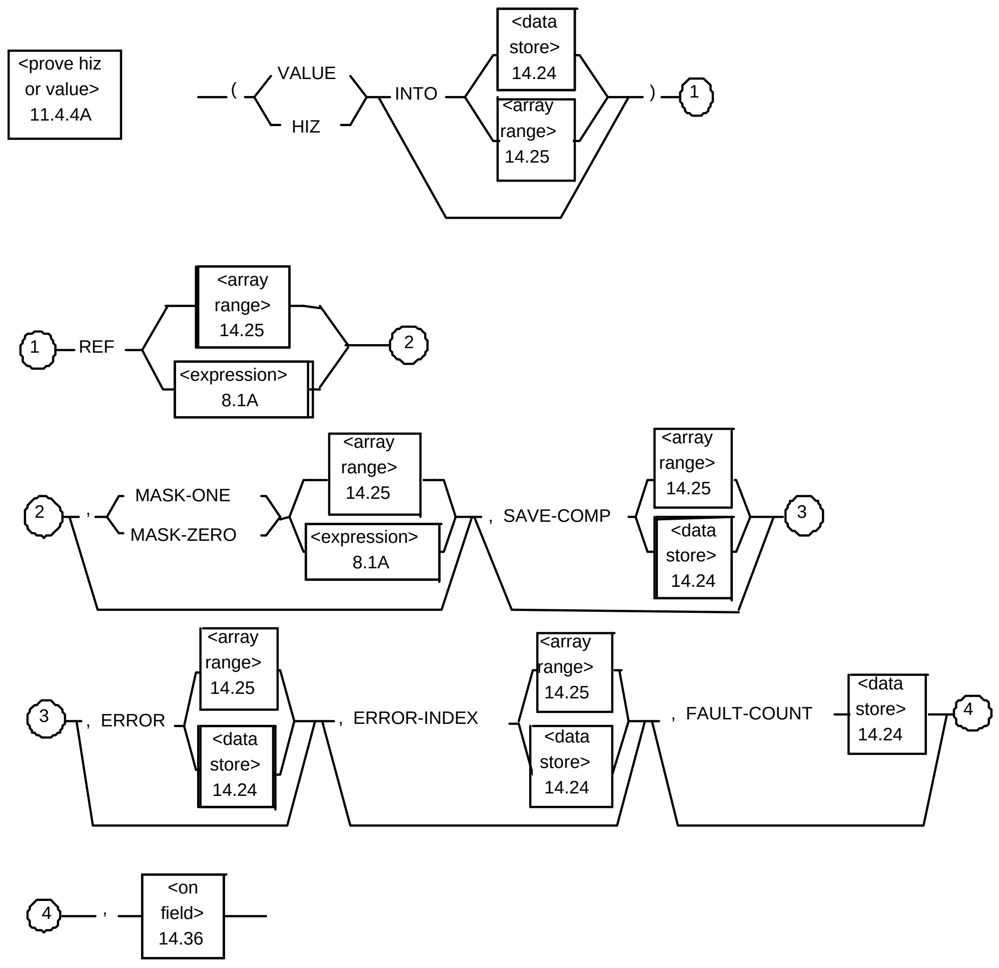
PROVE is a combination of the SENSE verb and an evaluation function to prove UUT responses are as expected. The evaluation function carries out comparisons and digital calculations that would be impractical to express routinely using explicit COMPARE and CALCULATE statements. Expected UUT responses can be expressed using either word or bit level constructs.

11.4.4.2 Formal syntax

Reference prove-statement

11.4.4.3 Syntax diagram





11.4.4.4 Rules

1. All the rules of SENSE apply (see 11.4.3.4). With PROVE, the INTO facilities (of SENSE) become optional.

2. For parallel signals, the <array range> or <expression> used with REF, MASK-ONE, or MASK- ZERO shall evaluate to a STRING OF BIT whose dynamic length is equal to the number of signals as defined by the <on field>. Also for parallel signals, the number of bits in the SAVE-COMP and ERROR <data store> or <array range> shall be DECLAREd to be at least as many bits as the number of signals SENSEd as defined by the <on field>.

3. For serial signals, the <array range> or <expression> used with REF, MASK-ONE, or MASK- ZERO shall evaluate to a STRING OF BIT whose dynamic length is equal to the WORD-LENGTH as defined by the <digital sensor> performing the sensing. Also, the number of bits in the SAVE- COMP and ERROR <data store> or <array range> shall be DECLAREd to be at least as many bits as the WORD-LENGTH as defined by the <digital sensor> performing the sensing.

4. ZERO and ONE correspond to LEVEL-LOGIC-ZERO and LEVEL-LOGIC-ONE respectively of the ENABLEd <configuration>. Specific connections identified with these two states represent the expected UUT response. Although a PROVE statement may contain any number of ZERO and ONE fields, the sensing of UUT responses will still occur as a single (critical) action. This sensing will represent the cumulative states specified as determined by their order of writing.

5. ZERO and ONE constructions can only reference parallel signals of the ENABLEd <configuration>.

6. The <expression> following REF represents the expected UUT responses. When used, the HIZ statement for expected high impedance states are represented by binary "1" and all other states by binary "0".

7. Multiple sense actions can be operated on by using the <array range> subfield within a PROVE statement included in a DO TIMED DIGITAL structure. The sense rate will be in accordance with the SENSE-RATE, SENSE-EVENT, or SENSE-DELAY specified in the DO TIMED DIGITAL structure.

8. The UUT response is XORed with the evaluation of the REF expression. If neither MASK-ONE nor MASK-ZERO fields are present the NOGO flag is set if at least one XOR result is non-zero. If MASK-ONE is present, the evaluation of the MASK-ONE expression is ANDed with the XOR result and the NOGO flag is set if at least one result is non-zero. If MASK-ZERO is present, the evaluation of the MASK-ZERO expression is complemented and treated as MASK-ONE. When bit by bit evaluations are used, each of the referenced sense pins is compared against the given logic condition (ZERO, ONE or HIZ). The NOGO flag is set if there is a mismatch.

9. The result of the evaluation according to rule 8 is stored in SAVE-COMP <data store> or <array range> element, if present.

10. When a set of response words is being processed and the statement contains an ERROR and/or ERROR-INDEX field and a comparison failure occurs, then

a) The result of the comparison described in rules 4, 6, and 7 is saved in the ERROR variable of type ARRAY.

b) The current value of the test step index is saved in the ERROR-INDEX variable of type ARRAY.

The fault error index counts the number of comparison failures that have occurred. The ERROR and ERROR-INDEX variables are always filled sequentially, starting with the first specified element. The final value of the fault error index is saved in the FAULT-COUNT.

11. The SENSE-WINDOW path is used to allow parallel data to be sensed over a period of time.

For parallel data, where the logic states are defined in the <define digital configuration structure> as LEVEL-LOGIC-ONE, LEVEL-LOGIC-ZERO, LEVEL-LOGIC-HIZ, or LEVEL-LOGIC- QUIES, the logic state is only achieved if it is constant throughout the entire period. If it is not constant, then the ILLEGAL-STATE-INDICATOR variable is set to TRUE. (Reference 6.16.4.4, rule 19.)

For parallel data, where the logic levels are defined in the <define digital configuration structure> as TRANS-LOGIC-ONE or TRANS-LOGIC-ZERO, the logic state is achieved if only one transition occurs throughout the period. If no transitions occur, or more than one transition occurs, then the ILLEGAL-STATE-INDICATOR variable is set to TRUE.

In both cases where the ILLEGAL-STATE-INDICATOR variable is TRUE, the NOGO flag shall also be set and the data stored in accordance with rule 10, above.

12. The FAULT-INDICATOR boolean variable provides an indication of any test failures within the <do timed digital structure>. The results of individual test steps can be determined by interrogating the information captured in the ERROR, ERROR-INDEX, FAULT-COUNT, and SAVE-COMP stores.

13. If the SENSE-WINDOW option is selected, for "window" mode testing, the individual test passes only if the sensor-signal equals the reference value for the whole duration of the SENSE- WINDOW. The test fails if the state does not equal the reference value or if there is a change in state while the window is open.

14. The fault error index is reset to 0 by the <do timed digital statement> and is incremented each time a test step containing a failure is encountered. The final value of the fault error index is saved in the FAULT-COUNT variable, which is of type INTEGER. If the fault error index is greater than 0, it is used as the index for the ERROR and ERROR-INDEX arrays.

15. Each time a failed test step is encountered, the current value of the test step index that is used to count the number of test steps within the <do timed digital structure> is saved in the ERROR-INDEX variable, which is of type ARRAY OF INTEGER. The result of the comparison is saved in the ERROR variable, which is of type ARRAY OF STRING OF BIT. The width of the ERROR string shall be the same as the width of the REF pattern.

16. For <do timed digital structure>s specifying a fixed STIM-RATE with a fixed SENSE-DELAY, the test step index is reset to 0 by the <do timed digital statement> and is incremented at the end of each SENSE-DELAY period.

17. For <do timed digital structure>s specifying a DIGITAL TIMER, the test step index is reset to 0 by the <do timed digital statement> and is incremented each time the DIGITAL TIMER is reset.

11.4.4.5 Notes and examples

The forms (HIZ) and (VALUE) are to be used when the user wants the result of the <prove statement> only in the form of GO or NOGO, but does not want the data to be returned explicitly.

12.0 Procedural statements, timing

12.1 Timing statements (general)

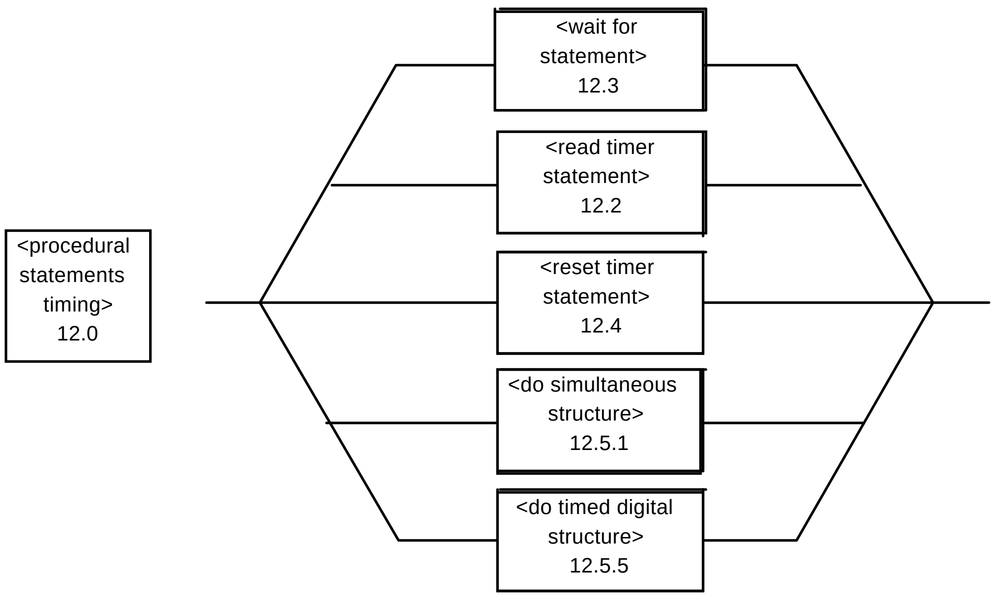
12.1.1 Function

Timing procedural statements consist of statements that describe the timing of source, sensor, or load functions, or which allow the delay of the execution of C/ATLAS statements.

12.1.2 Formal syntax

Reference procedural statements timing

12.1.3 Syntax diagram


<procedural statements timing> 12.0
<wait for statement> 12.3
<read timer statement> 12.2
<reset timer statement> 12.4
<do simultaneous structure> 12.5.1
<do timed digital structure> 12.5.5

12.2 READ TIMER statement

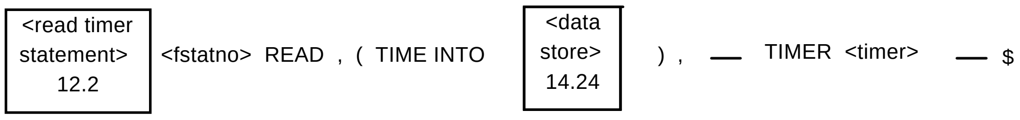
12.2.1 Function

This statement function is to read the current time value of a <timer> and retain that value in the INTO <data store> location. This allows measurements to be made of the elapsed time between C/ATLAS test statements or actions.

12.2.2 Formal syntax

Reference read timer statement

12.2.3 Syntax diagram



12.2.4 Rules

1. The current time value of the specified <timer> is established and transferred into the <data store> specified by the INTO field.

2. The <timer> specified is not affected by this statement; time counting continues without interruption.

3. The time value units and resolution are established by the IDENTIFY, TIMER statement.

12.3 WAIT FOR statement

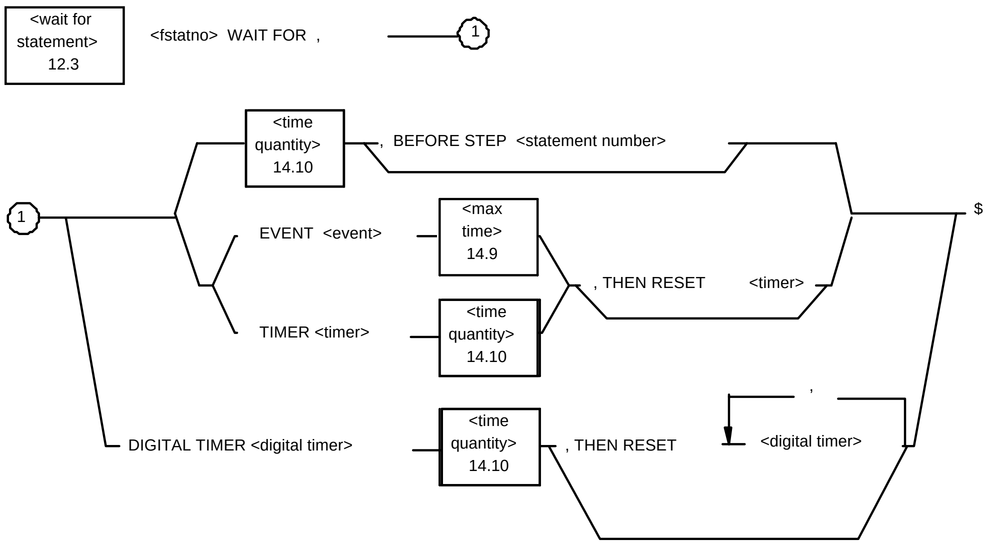
12.3.1 Function

This statement function is to postpone execution of the specified statement in the test until the specified time or event has occurred.

12.3.2 Formal syntax

Reference wait for statement

12.3.3 Syntax diagram



12.3.4 Rules

1. If the desired time interval is written after WAIT FOR followed by a BEFORE entry, the specified interval includes the time required to accomplish intervening statements.

2. If the BEFORE entry is omitted from the WAIT FOR statement, the next step in the sequence is postponed until the condition is satisfied.

3. Wait time is measured from the commencement of the execution of the WAIT FOR statement to the initiation of the STEP mentioned in the statement or the next executable statement when the BEFORE STEP is bypassed.

4. When the WAIT FOR <event> option is used, the program execution will pause at that statement until the labelled <event> occurs or the MAX_TIME is exceeded. Events occurring prior to the execution of the WAIT FOR statement will have no effect on the statement execution and the execution of the statement will be delayed until another <event> occurs or MAX_TIME condition is established.

5. The optional THEN RESET <timer> or THEN RESET <digital timer> branch specifies a reset of the <timer> or <digital timer>s simultaneous with the end of the WAIT state (i.e., either the occurrence of the <event>, the MAX_TIME condition becoming true, or the occurrence of a specified time on a <timer> or <digital timer>).

6. The <wait for statement> containing the DIGITAL TIMER branch can only be used inside a <do timed digital structure> that includes a DIGITAL TIMER option in the <do timed digital structure>.

12.3.5 Notes and examples

```
543211 APPLY, AC SIGNAL, VOLTAGE 115.0V,
           CNX HI J9-8 LO J9-9 $
    12 WAIT FOR, 15 SEC,  BEFORE STEP 543216 $
    13 MEASURE,  DC ...
    14 ...
    15 VERIFY, ...
    16 APPLY, PULSED AC, ...
```

12.4 RESET TIMER statement

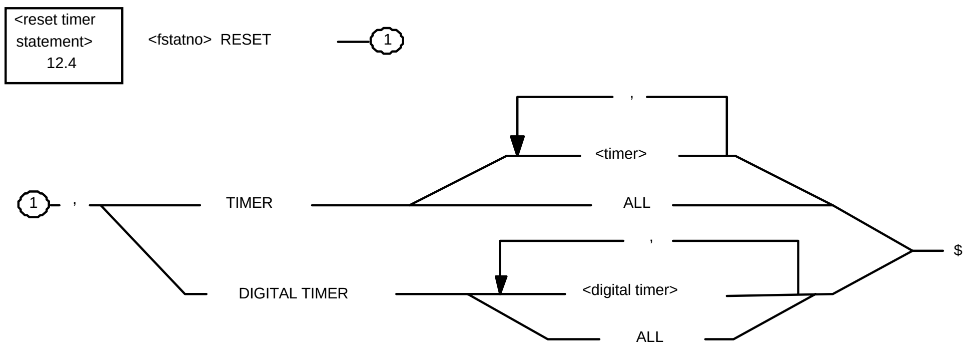
12.4.1 Function

This statement function is to reset a <timer> or <digital timer> to zero.

12.4.2 Formal syntax

Reference reset timer statement

12.4.3 Syntax diagram



12.4.4 Rules

1. A <timer> starts counting time from zero as soon as it is reset. It has no "idle state."

2. A <digital timer> is set to zero as soon as it is reset. It has no "idle state."

3. All <timer>s or <digital timer>s specified in a single statement shall be reset simultaneously.

4. The <reset timer statement> containing the DIGITAL TIMER branch can only be used inside a <do timed digital structure> that includes a DIGITAL TIMER option in the <do timed digital statement>.

12.5 DO/END DO capability

The DO/END DO capability provides a method to delimit a series of C/ATLAS statements for which the timing is specified by the opening DO statement.

12.5.1 <do simultaneous structure>

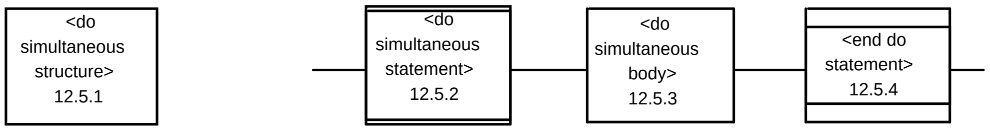
12.5.1.1 Function

This function is to delimit a set of statements of which the critical actions shall be executed simultaneously.

12.5.1.2 Formal syntax

Reference do simultaneous structure

12.5.1.3 Syntax diagram


<do simultaneous structure> 12.5.1
<do simultaneous statement> 12.5.2
<do simultaneous body> 12.5.3
<end do statement> 12.5.4

12.5.1.4 Rules

The critical action of the <do simultaneous structure> is the execution of the first critical action of the embedded statements.

12.5.2 DO SIMULTANEOUS statement

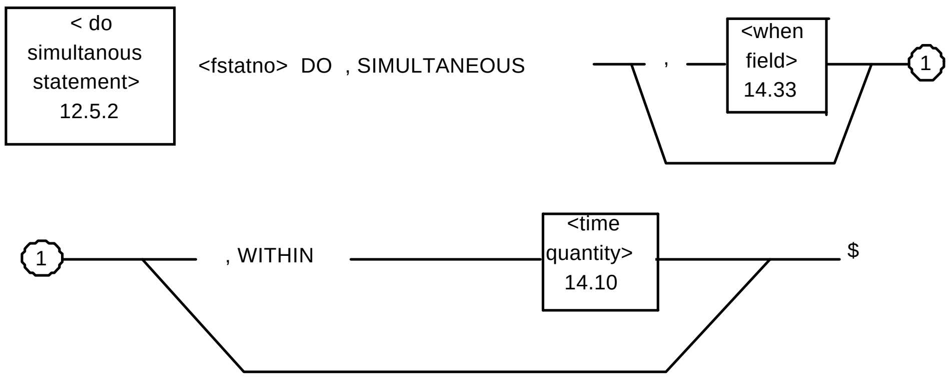
12.5.2.1 Function

To introduce a <do simultaneous structure>.

12.5.2.2 Formal syntax

Reference do simultaneous statement

12.5.2.3 Syntax diagram


< do simultanous statement> 12.5.2
<fstatno> DO , SIMULTANEOUS
,
<when field> 14.33
1
1
, WITHIN
<time quantity> 14.10
$

12.5.2.4 Rules

1. The optional <when field> specifies the point in time when the <do simultaneous structure> is executed.

2. The optional <within> subfield defines the maximum amount of time for execution of the structure.

3. If the <do simultaneous statement> or any statement in the <do simultaneous body> contains references to <event>s which are not already enabled, the <event>s are enabled at the commencement of execution of the <do simultaneous structure>. Any <event> that is enabled as a result of this rule is disabled on completion of the execution of the <do simultaneous structure>.

12.5.3 <do simultaneous body>

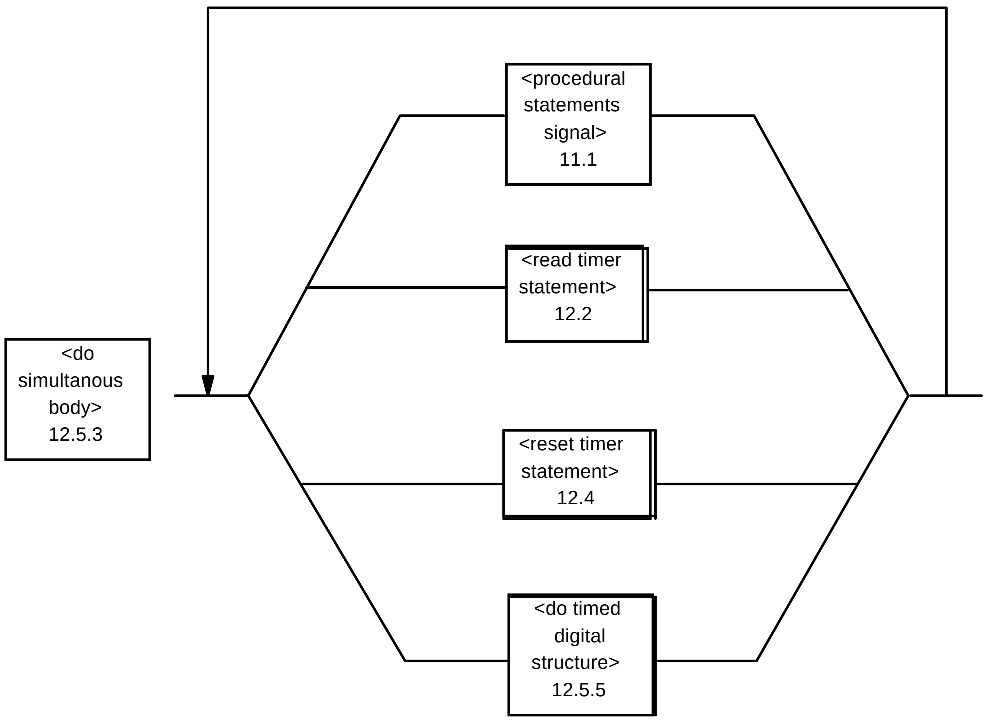
12.5.3.1 Function

The <do simultaneous body> consists of the C/ATLAS procedural statements having critical actions that shall be executed simultaneously.

12.5.3.2 Formal syntax

Reference do simultaneous body

12.5.3.3 Syntax diagram


<do simultanous body> 12.5.3
<procedural statements signal> 11.1
<read timer statement> 12.2
<reset timer statement> 12.4
<do timed digital structure> 12.5.5

12.5.3.4 Rules

1. A <do simultaneous body> shall include only statements for which a critical action is defined.

2. Statements within a <do simultaneous body> shall not include timing subfields ruling their critical actions.

3. Statements within a <do simultaneous body> shall not be the target of a <go to statement>.

12.5.4 END DO statement

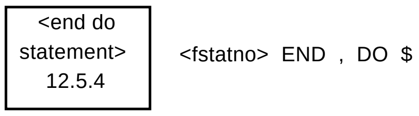
12.5.4.1 Function

This function is to close a <do simultaneous structure> or a <do timed digital structure>.

12.5.4.2 Formal syntax

Reference end do statement

12.5.4.3 Syntax diagram



12.5.5 <do timed digital structure>

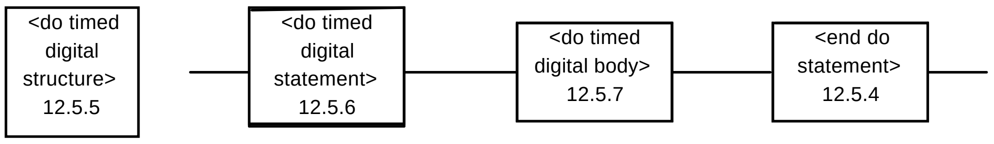
12.5.5.1 Function

This function is to delimit a digital test sequence of which the timing is specified in the <do timed digital statement>.

12.5.5.2 Formal syntax

Reference do timed digital structure

12.5.5.3 Syntax diagram


<do timed digital structure> 12.5.5
<do timed digital statement> 12.5.6
<do timed digital body> 12.5.7
<end do statement> 12.5.4

12.5.5.4 Rules

The critical action of the <do timed digital structure> is the execution of the first critical action within the <do timed digital body>.

12.5.6 DO TIMED DIGITAL statement

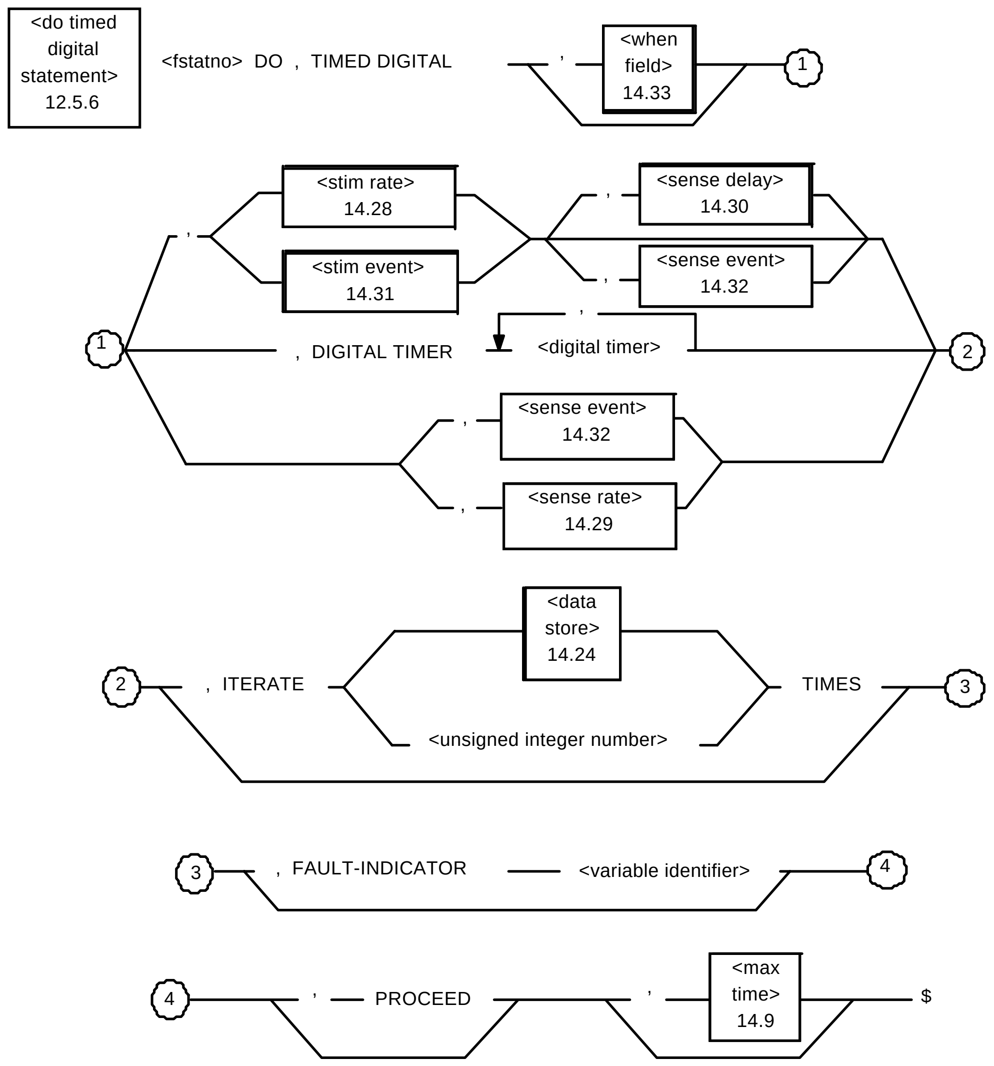
12.5.6.1 Function

This statement function is to introduce a <do timed digital structure>.

12.5.6.2 Formal syntax

Reference do timed digital statement

12.5.6.3 Syntax diagram



12.5.6.4 Rules

1. The optional <when field> defines the following:

a) Critical time for resetting the <digital timer>s if the DIGITAL TIMER field is selected or

b) Time for starting to look for stim/sense events if a <stim event> or <sense event> is selected or

c) Time of execution of the first critical action within the <do timed digital structure> if a <stim rate> or <sense rate> is selected

2. The subfields after the DO, TIMED DIGITAL, or DO, TIMED DIGITAL, <when field> of the syntax diagram specify the timing of the critical actions of the STIMULATE,

SENSE, or PROVE statements contained in the following <do timed digital body>. These subfields act as follows:

a) <stim rate> or <stim event> used singly. Successively encountered STIMULATE statements will be executed at the rate specified in <stim rate>, or alternatively, at every occurrence of the referenced <event>s if <stim event> is used. In this case SENSE or PROVE statements shall not be included in the following <do timed digital body>.

b) <sense rate> or <sense event> used singly. Successively encountered SENSE or PROVE statements will be executed at the rate specified in <sense rate>, or alternatively, at every occurrence of the referenced <event>s if <sense event> is used. In this case STIMULATE statements shall not be included in the following <do timed digital body>.

c) <sense delay> used with either <stim rate> or <stim event>. The <sense delay> specified shall be less than the expected stimulus interval. If a SENSE or PROVE statement is encountered following a STIMULATE statement, then the critical action of the SENSE or PROVE will be delayed from the critical action of the STIMULATE by the time specified in <sense delay>. If there are no SENSE or PROVE statements between any successively encountered STIMULATE statements, the timing of these successive STIMULATEs will be as specified by the associated DO, TIMED DIGITAL statement. If successive SENSE or PROVE statements are encountered with no intervening STIMULATE statements, their critical actions will still be timed as if there had been intervening STIMULATEs.

d) <sense event> used with either <stim rate> or <stim event>. If a SENSE or PROVE statement is encountered following a STIMULATE statement, then the critical action of the SENSE or PROVE will take place at the next occurring <sense event> following the STIMULATE statement. If there are no SENSE or PROVE statements between successively encountered STIMULATE statements the timing of these successive STIMULATEs will be as specified in the associated DO, TIMED DIGITAL statement. If the time between one stimulus and the next <sense event> is longer than the stimulus interval, then the next occurring stimulus will be on the next beat of <stim rate> or the next <stim event> following the <sense event>. If successive SENSE or PROVE statements are encountered with no intervening STIMULATE statements their critical actions will be timed by successively occurring <sense event>s.

e) If a list of <event>s is specified in the DO, TIMED DIGITAL, the number of events in the list shall equal the number of critical actions of the corresponding "statements" within the <do timed digital body>.

f) The DIGITAL TIMER option can only be used when all STIMULATE, SENSE, and PROVE statements within the <do timed digital structure> specify a <digital timer> in their <when field>s to dynamically control the timing of the execution of these statements. The <digital timer> in these statements shall be identical to one of the <do timed digital statement> <digital timer>s.

g) On the start of execution of this statement, every referenced <digital timer> is RESET simultaneously.

h) FAULT_INDICATOR specifies a boolean variable that is set TRUE if a fail has occurred during the execution of a <do timed digital structure>. Otherwise, the boolean variable is set to FALSE. During the execution of a <do timed digital structure>, the boolean variable is undefined.

3. The ITERATE field, when present, specifies the number of times the entire <do timed digital body> is to be executed.

4. The PROCEED field, when present, specifies that the test sequence will continue, in order to execute other statements following the <do timed digital structure> while the digital test sequence is executing.

5. The <max time> field, when used, specifies the maximum time within which the complete <do timed digital structure> is to be executed. If the digital test sequence is not completed within the time limit, a MAX_TIME condition is set and the test resumes execution following the <do timed digital structure>.

6. If the <do timed digital statement> or any statement in the <do timed digital body> or their referenced <define digital source statement>s or <define digital sensor statement>s contain references to <event>s that are not already enabled, the <event>s are enabled at the commencement of execution of the <do timed digital structure>. Any <event> that is enabled as a result of this rule is disabled on completion of the execution of the <do timed digital structure>.

7. A <stim event>, <sense event>, or combination of <stim event> followed by <sense event> in a <do timed digital statement> may be replaced with a <digital timing> label introduced by a <define digital timing statement>.

12.5.7 <do timed digital body>

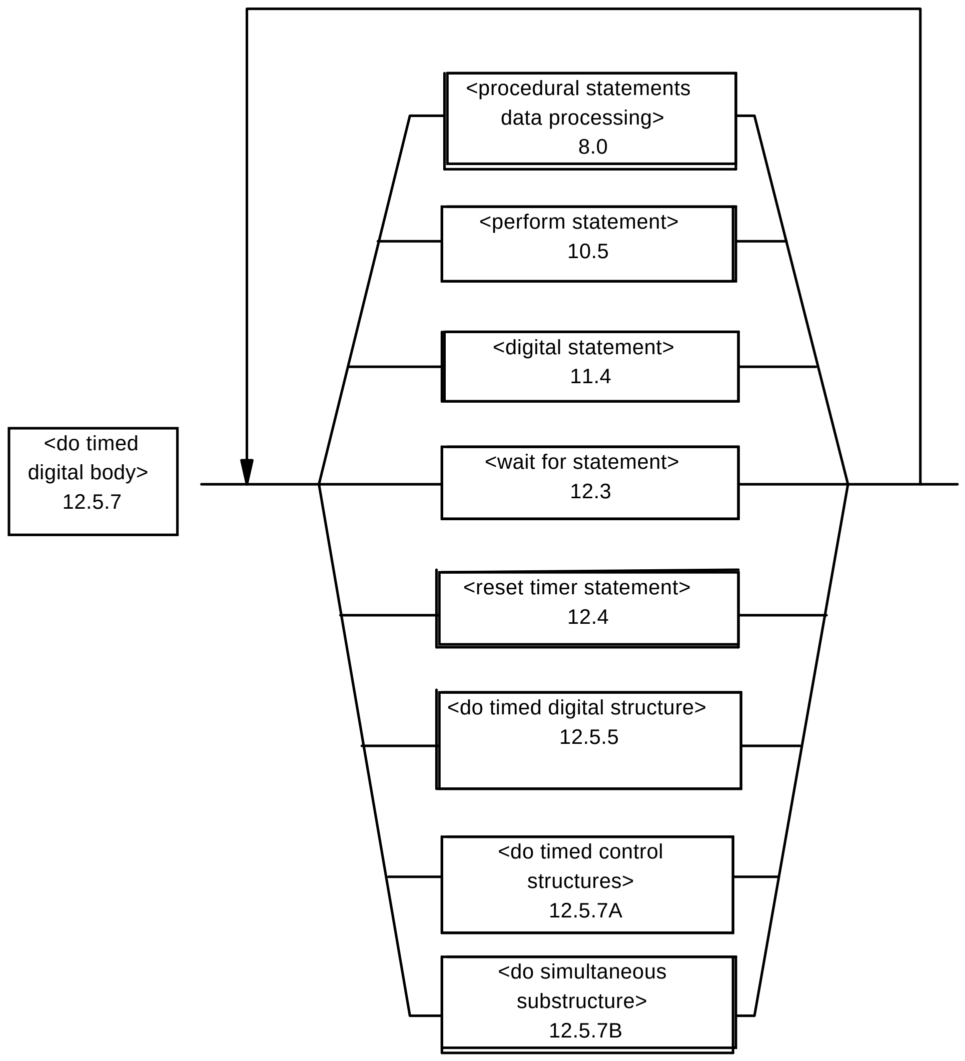
12.5.7.1 Function

The <do timed digital body> consists of C/ATLAS procedural statements that constitute the timed digital test sequence.

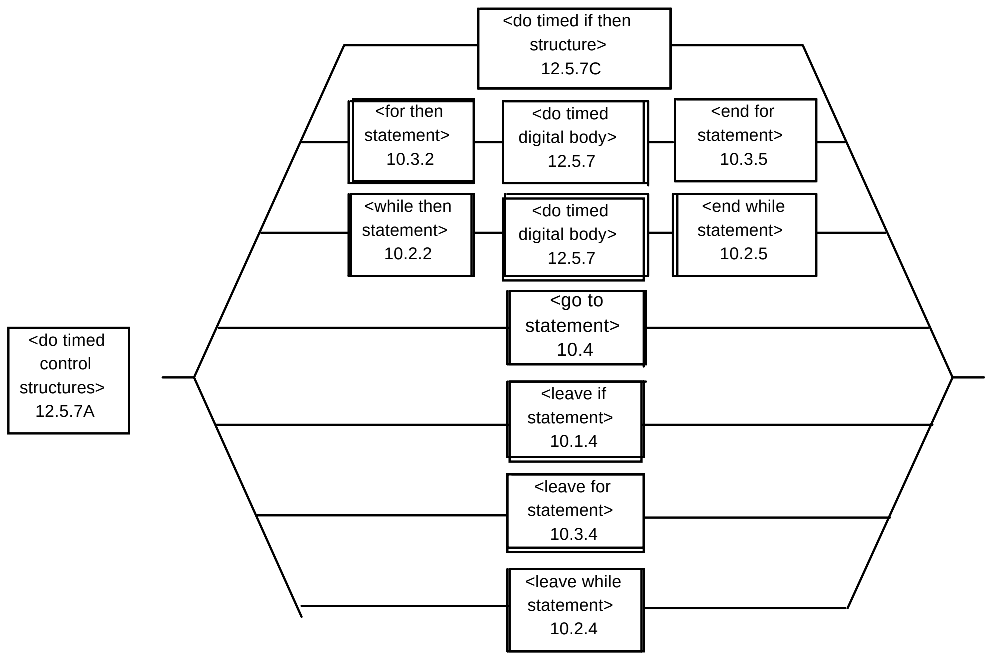
12.5.7.2 Formal syntax

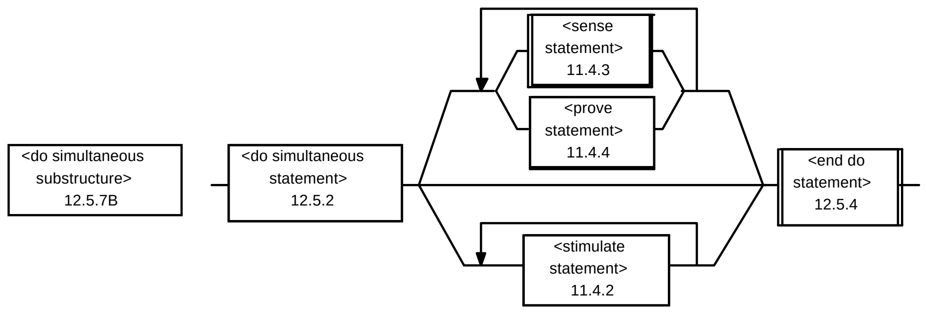
Reference do timed digital body

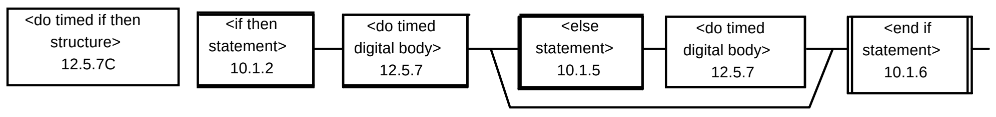
12.5.7.3 Syntax diagram



<do timed digital body> 12.5.7
<procedural statements data processing> 8.0
<perform statement> 10.5
<digital statement> 11.4
<wait for statement> 12.3
<reset timer statement> 12.4
<do timed digital structure> 12.5.5
<do timed control structures> 12.5.7A
<do simultaneous substructure> 12.5.7B







12.5.7.4 Rules

1. <do timed control structure>s are a limited subset of the program control structures of the C/ATLAS language for use within the context of a <do timed digital body>.

2. Statements within a <do timed digital structure> shall not include timing subfields that conflict with the timing specified in the <do timed digital statement> of that structure.

3. Statements within a <do timed digital body> shall not be the target of a <go to statement> external to the body.

4. Whenever a <do simultaneous substructure> is used within a <do timed digital body>, it shall only contain STIMULATE and SENSE/PROVE statements.

5. For timing the start of an embedded <do timed digital structure>, the structure is treated as a STIMULATE statement. If a <do simultaneous structure> used within a <do timed digital body> contains only SENSE or PROVE statements, it shall be treated as a SENSE statement. Otherwise, it shall be treated as a STIMULATE statement.

6. A <do timed digital structure> embedded in a <do timed digital structure> using <stim event> or <stim rate>, shall not include a <when field>.

7. Nested <do timed digital structure>s shall not contain a PROCEED field.

8. Where a <perform statement> is used inside a <do timed digital body>, it shall use the TIMED DIGITAL attribute.

9. In a <do timed digital body>, the <reset timer statement> or the <wait for statement> can only be used if the <do timed digital statement> contains the <digital timer> field. Additionally, the <digital timer>(s) in the <reset timer statement> or the <wait for statement> shall be identical to one of the <do timed digital statement> <digital timer>s.

12.5.7.5 Notes and examples

1. There are many requirements for a sequence of different combinations of STIMULATE, SENSE, and PROVE statements within a <do timed digital structure>. These can be accomplished by using FOR loops. Arrays may be used within the FOR loops with either the loop index or other <data store> employed as the array indices.

2. Different timing cycles for digital test can be described in separate <do timed digital structure>s that may be defined in DIGITAL TIMER procedures.

<do timed digital structure>s can be performed in another <do timed digital structure> using a <stim event> as the basis for digital synchronization.

The duration of a <do timed digital structure> embedded in another <do timed digital structure> can be defined using the <wait for statement> that shall contain the TIMED DIGITAL attribute.

13.0 Procedural statements, databus

Most C/ATLAS signal-oriented statements describe operations that are logically simple in nature. They usually do not describe simultaneous multiple critical actions, combined stimulus and measurement actions, or more than one event that controls test timing. The macro capability is provided to specify test actions that are more complex, typically requiring synchronized stimulus and/or measurement sequences or cases where the test tactics would be hidden by the details of other test statements and structures. Such macro capabilities are provided in two forms: macro structures and macro statements. The macro structures can be found in 12.5 and its subclauses; the macro statements are described in this clause. These two forms of macro capability address the three types of test capability currently needing such an implementation. They are as follows:

1. Simultaneous actions macro capability

Most signal-oriented statements require that during execution, all actions including the critical actions take place before beginning execution of the next statement. This naturally precludes simultaneous or concurrent test actions involving multiple statements. Many test sequences require simultaneous application of stimululi, simultaneous measurements, or measurements during stimulus application. For these types of test sequences, the <do simultaneous structure> is provided.

2. Digital test macro capability

The capability to synchronize the application of a sequence of digital stimulus patterns, the reading back of a sequence of UUT response values, and the evaluation of these responses is so complex as to fall within the province of C/ATLAS macro capabilities. These test requirements are addressed by the <do timed digital structure>.

3. Bus testing macro capability

A common requirement is to test a UUT or group of UUTs that communicate via one or multiple digital data buses. Although the bus communications could be handled by standard digital testing techniques, the enormous amount of data involved makes it difficult, if not impossible, to discern the logical operations being performed. Frequently, only part of the information involved is of interest to the test being performed. On occasion, the real time test actions are dependent on results of some current action. In order to provide a more readable and understandable description of such situations, the data bus testing macro statement described in this clause is provided.

Data bus processing, as discussed in this clause, assumes that the data bus itself is functioning correctly and that the UUT communicates via a bus or buses. Several statements support the testing of UUTs that utilize buses. The first is the <establish protocol statement>, which is described in 6.18. The second is the <define exchange statement>, which is described in 6.19. This permits describing a single complete transaction or exchange on the bus. Both the <establish protocol statement> and the group of <define exchange statement>s can be used in various test programs for different UUTs that are normally tied together on the same bus because they only identify the functioning of the bus itself and the types of transactions that can appear on the bus. The DO EXCHANGE described in 13.2 permits the description of the actual traffic on the bus in terms of exchanges and specifies the type of test equipment participation in each exchange.

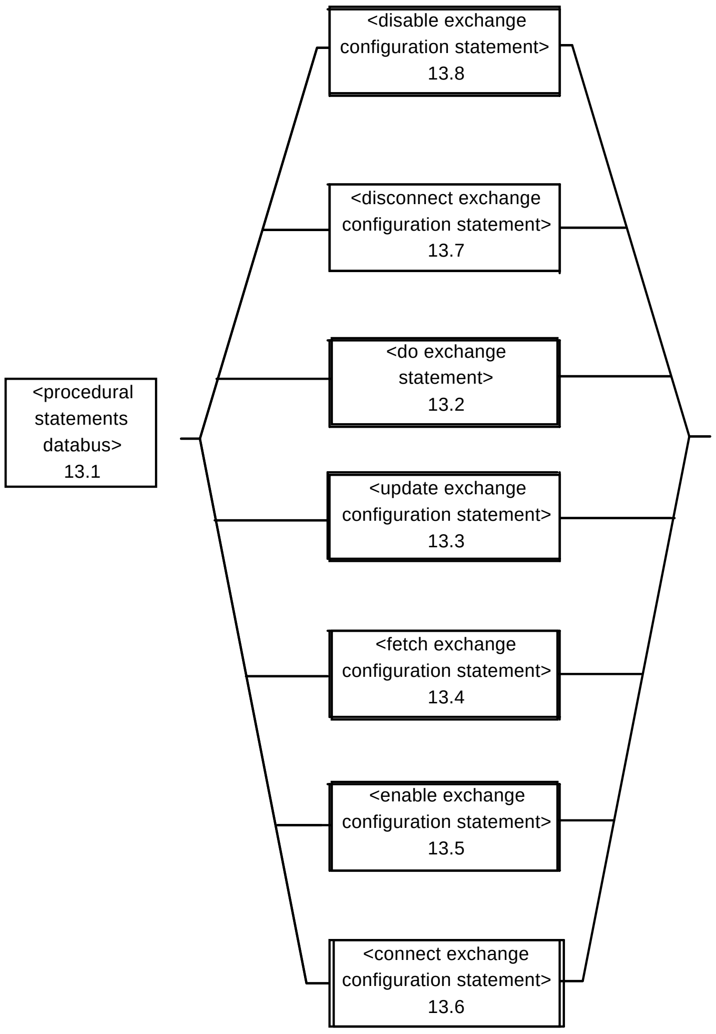
13.1 <procedural statements databus>

13.1.1 Function

13.1.2 Formal syntax

Reference procedural-statements-databus

13.1.3 Syntax diagram


<procedural statements databus> 13.1
<disable exchange configuration statement> 13.8
<disconnect exchange configuration statement> 13.7
<do exchange statement> 13.2
<update exchange configuration statement> 13.3
<fetch exchange configuration statement> 13.4
<enable exchange configuration statement> 13.5
<connect exchange configuration statement> 13.6

13.2 DO EXCHANGE statement

13.2.1 Function

The DO EXCHANGE statement describes a series of transactions (exchanges) on a databus in a single C/ATLAS statement. A separate <define exchange statement> defines each transaction. An <establish protocol statement> references the protocol of the databus. The <do exchange statement> describes sequences of exchanges that define the overall bus traffic of interest to the test being performed. It also specifies the test equipment participation in each exchange.

13.2.1.1 Bus participation modes

Databus systems classify different functions performed by the devices on the bus. A device can be a listener or a talker and communications usually consist of one talker and one or more listeners. On many databuses a device can be either a master or a slave. In these, the master generates the control information on the bus and a slave responds to the control information. Normally, only one device at a time is master. There is no fixed relationship between master and talker or slave and listener. Master/slave concerns control functions. Talker/listener concerns data functions. To perform a particular test the test-equipment can be performing the tasks of master, slave, listener, talker, or a combination of the above. In addition, the test equipment can monitor and/or control data information on the bus without actively participating in the transaction. The role played by the test equipment can vary from exchange to exchange.

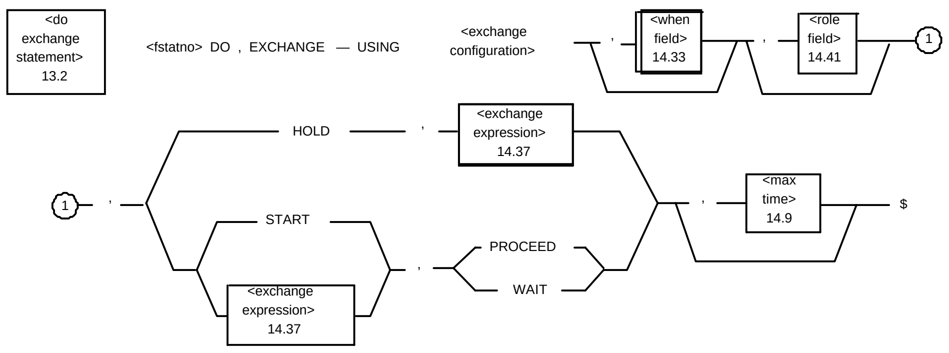
13.2.1.2 Bus timing considerations

The timing of various exchanges on a bus can be quite complex. Various timing events occurring either on the bus or external to the bus can synchronize the databus. The <do exchange statement>, in effect, establishes the concurrent execution of various bus related operations. Since conflicts can arise, each operation can be prioritized. The execution of a <do exchange statement> continues until the termination of every operation or of the MAX-TIME field, whichever occurs first.

13.2.2 Formal syntax

Reference do-exchange-statement

13.2.3 Syntax diagram


<do exchange statement> 13.2
<fstatno> DO , EXCHANGE — USING
<exchange configuration>
,
<when field> 14.33
,
<role field> 14.41
1
1
,
HOLD
,
<exchange expression> 14.37
START
<exchange expression> 14.37
,
PROCEED
WAIT
,
<max time> 14.9
$

13.2.4 Rules

1. The USING subfield references an <exchange configuration> that shall have been previously enabled by an <enable exchange configuration statement>. All the <exchange>s invoked in the <exchange frame> shall reference one of the <protocol>s listed in the <define exchange configuration statement>, and this <protocol> shall be made active by the previous <enable exchange configuration statement>.

2. The optional <when field> at the beginning of the <do exchange statement> defines a condition that shall be fulfilled before the start of the execution of the first transaction included in the <do exchange statement>. It precedes the starting condition associated with each <exchange frame>.

3. The PROCEED field requires that the execution of the <do exchange statement> continues in parallel with the execution of other statements in the C/ATLAS program. The WAIT field requires that the execution of the next C/ATLAS statement starts after completion of the <do exchange statement>.

4. The <do exchange statement> shall only reference <exchange>s that are defined in the preamble. An <exchange frame> within an <exchange expression> references <exchange>s by their defined labels. They execute as defined when the <do exchange statement> is executed and continue to do so unless a subsequent <update exchange statement> modifies them.

5. Only one <do exchange statement> can be active for a single databus at one time. A number of databuses can be active simultaneously by executing two or more <do exchange statement>s with the PROCEED option.

6. The <do exchange statement> is terminated as soon as one of the following conditions occurs:

a) All expressed <exchange expression>s are complete

b) The MAX-TIME condition established by either the <exchange expression> or the <do exchange statement> has expired

c) A <finish statement> occurs

7. If the <role field> is used, the parameters represent the functions or roles the test equipment can assume: MASTER, SLAVE, MONITOR, or others by use of the <test equip role name> field. Whichever is employed shall be listed in the <test equip bus role> section of the referenced <establish protocol statement>. When used at this level, the test equipment role is specified for the whole <do exchange statement>. If the <role field> is bypassed in the <do exchange statement>, the role of the test equipment is to be specified for each <exchange> within the <exchange frame> definition. Where <role field> is specified both within the <do exchange statement> and within an <exchange frame> for a given <exchange>, the <role field> for the <exchange frame> overrides that specified for the entire <do exchange statement> only for that <exchange>.

8. An <exchange> shall only be referenced once in an <exchange frame>.

9. The HOLD option prepares the device, but does not start the exchanges. The START option initiates the actions previously prepared by the statement containing the HOLD field.

13.2.5 Notes and examples

The fundamentals of databus testing are described in 6.18.5.

13.2.5.1 Example 1. Mono-bus capability

For the following example we shall consider a bus system which, in operational mode, includes the following:

— An inertial platform (INS)

— An autopilot

— An air-data-computer (ADC)

— A main computer responsible for bus traffic

Information exchange between components is needed to run, and therefore to test, each component. To test one of them, when out of the system, the test equipment must simulate the operational environment.

The frequency in parentheses indicates the performance frequency for each exchange. The following portions of a C/ATLAS test program show use of the capabilities of the language.

In the C/ATLAS test program preamble that follows, the <establish protocol statement> identifies the protocol for the databus and indicates the format of the fields used. The <define exchange configuration statement> identifies the exchange configuration and gives the name of the protocol used. Each <define exchange statement> in the preamble references the <establish protocol statement>.

The DEFINE, EXCHANGE accomplishes the following:

a) Refers to the <establish protocol statement> for the exchange.

b) States the bus mode for the exchange.

c) States the talker, listener(s), and data as required by the bus mode.

d) Identifies the exchange by a label.

```
000100 DECLARE, TYPE, 'DEVICE-IDENTIFIERS' IS ENUMERATION
        ('INS-GAMMA-FUNCTION',
            'INS-GEO-FUNCTION',
            'INS-ATT-FUNC',
            'ADC-B-FUNCTION',
            'MASTER-COMP',
            'ADC-M-FUNC',
            'ADC-S-FUNC',
            'ADC-TAS-FUNCTION',
            'MASTER-COMP-ACC-FUNC',
            'MASTER-COMP-GEO-FUNC',
            'AUTO-PILOT',
            'MASTER-COMP-B-FUNC',
            'INS',
            'AUTOPILOT-H-FUNC',
            'MASTER-COMP-MACH-FUNC',
            'MASTER-COMP-SPO-FUNC',
            'MASTER-COMP-TAS-FUNC' ) $
000110 DECLARE, VARIABLE, 'ATTITUDES' IS RECORD OF
```

```
          ['PHI1','PHI2',
          'THETA1',THETA2',
          'PSI1','PSI2' IS INTEGER ] $

000120 DECLARE, VARIABLE, 'SPEED' IS RECORD OF
          ['X1','X2','X3',
          'Y1','Y2','Y3',
          'Z1','Z2','Z3' IS INTEGER ] $

000130 DECLARE, VARIABLE, 'POSITION' IS ARRAY (1 THRU 5)
          OF RECORD OF
          ['L1','L2','L3',
          'G1','G2','G3',
          'Z1','Z2','Z3' IS INTEGER ] $

000140 DECLARE, VARIABLE, 'ZB','HEADING','MACH',
          'SZ','TAS','II'
          IS INTEGER $

002000 ESTABLISH, BUS PROTOCOL 'BUS-1',
          SPEC 'AIR-STANDARD-A1',
          TEST-EQUIP-ROLE MASTER, MONITOR, SLAVE,
          TEST-EQUIP-MONITOR COMMAND, DATA, STATUS,
          BUS-MODE ALL-LISTENER,
          TALKER 'MASTER-COMP' (INTEGER),
          LISTENER 'AUTOPILOT-H-FUNC',
          DATA (INTEGER),
          BUS-MODE TALKER-LISTENER,
          TALKER 'ADC-B-FUNCTION',
          'ADC-M-FUNC',
          'ADC-S-FUNC',
          'ADC-TAS-FUNCTION',
          LISTENER 'MASTER-COMP-B-FUNC',
          'MASTER-COMP-MACH-FUNC',
          'MASTER-COMP-TAS-FUNC',
          'MASTER-COMP-SPO-FUNC',
          DATA (INTEGER),
          BUS-MODE TALKER-LISTENER,
          TALKER 'INS-ATT-FUNCTION',
          LISTENER 'MASTER-COMP' (INTEGER),
          DATA (RECORD OF
               ['FIRST','SECOND','THIRD',
               'FOURTH','FIFTH','SIXTH', IS INTEGER]),
          BUS-MODE TALKER-LISTENER,
          TALKER 'INS-GAMMA-FUNCTION',
          'INS-GEO-FUNCTION',
          LISTENER 'MASTER-COMP-ACC-FUNC',
          'MASTER-COMP-GEO-FUNC',
          DATA (RECORD OF
               ['A','B','C',
               'D','E','F','G','H','J' IS INTEGER]),
          STANDARD PRIMARY BUS ,
                CNX TRUE COMP $
```

```
002500 DEFINE, 'CONF-1', EXCHANGE-CONFIGURATION 'BUS-1'  $

003000 DEFINE, 'SYN-10', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE ALL-LISTENER,
           DATA 21  $

    10 DEFINE, 'SYN-40', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE ALL-LISTENER,
           DATA 22 $

    20 DEFINE, 'SPEED', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE TALKER-LISTENER,
           TALKER 'INS-GAMMA-FUNCTION',
           LISTENER 'MASTER-COMP-ACC-FUNC',
           DATA 'SPEED' $

    30 DEFINE, 'GEO-POSITION', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE TALKER-LISTENER,
           TALKER 'INS-GEO-FUNCTION',
           LISTENER 'MASTER-COMP-GEO-FUNC',
           DATA 'POSITION' (1 THRU 5) $

    40 DEFINE, 'ATTITUDE', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE TALKER-LISTENER,
           TALKER 'INS-ATT-FUNC',
           LISTENER 'MASTER-COMP' (22),
           DATA 'ATTITUDES'   $

    50 DEFINE, 'BARO-ALT', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE TALKER-LISTENER,
           TALKER 'ADC-B-FUNCTION',
           LISTENER 'MASTER-COMP-B-FUNC',
           DATA 'ZB' $

    60 DEFINE, 'HEADING', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE TALKER-LISTENER,
           TALKER 'MASTER-COMP' (14),
           LISTENER 'AUTO-PILOT-H-FUNC',
           DATA 'HEADING'  $

    70 DEFINE, 'MACH', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE TALKER-LISTENER,
           TALKER 'ADC-M-FUNC',
           LISTENER 'MASTER-COMP-MACH-FUNC',
           DATA 'MACH'  $

    80 DEFINE, 'VERTICAL-SPEED', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE TALKER-LISTENER,
           TALKER 'ADC-S-FUNC',
           LISTENER 'MASTER-COMP-SPO-FUNC',
           DATA 'SZ'  $
    90 DEFINE, 'TRUE-AIR-SPEED', EXCHANGE, PROTOCOL 'BUS-1',
           BUS-MODE TALKER-LISTENER,
           TALKER 'ADC-TAS-FUNCTION',
           LISTENER 'MASTER-COMP-TAS-FUNC',
           DATA 'TAS'  $
```

Both the bus modes and the test-equip modes in this example have descriptive names, as do all the talkers and listeners and most of the subaddresses. The greater clarity of using descriptive names is clear from the relative unfriendliness of the numerical subaddresses.

Figure 13-2 and table 13-1 illustrate the clarity of using descriptive names.

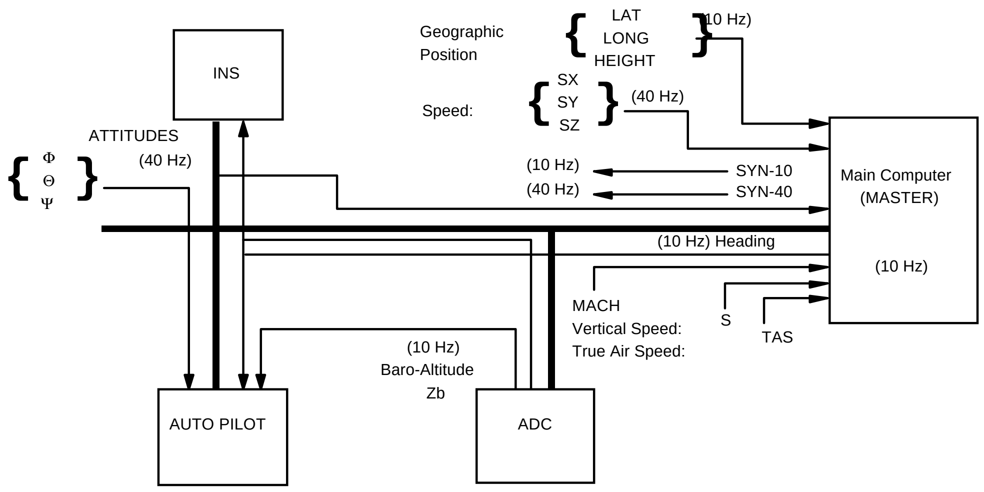


Figure 13-1. Example mono-bus system

Table 13-1. Sample bus traffic

EXCHANGE	TIME	TEST-EQUIP	BUS MODE	TALKER		LISTENER		DATA
	MSEC	ROLE	ADDRESS	SUB-ADDRESS		ADDRESS	SUB-ADDRESS	
:								
SYN-40'	2 27 52 77	MASTER	ALL-LISTENER					22
BARO-ALT'	4	MONITOR	TALKER-LISTENER	'ADC'	'B-FUNC'	'AUTO-PILOT' 'INS'	'B-FUNC' 18	'2B'
SPEED'	8 33 58 83	MONITOR 'SX', 'SY' 'SZ' SLAVE	TALKER-LISTENER	'INS'	'GAMMA-FUNCTION'	'MASTER-COMP'	'ACC-FUNC'	'SX', 'SY', 'SZ'

GEO-POSITION	14	MONITOR	TALKER-LISTENER	'INS'	'GEO-FUNCTION'	'MASTER-COMP'	'GEO-FUNC'	LAT(1)
								LONG(1)
								HEIGHT(1)

ATTITUDE'	23 48 73 98	MONITOR 'PHI' 'PSI' 'THETA' SLAVE	TALKER-LISTENER	'INS'	'ATT-FUNC'	'MASTER-COMP' 'AUTO PILOT'	22 25	'PHI' 'THETA' 'PSI'
HEADING'	40	SIMULATE 'AUTOPILOT' SLAVE	TALKER-LISTENER	'MASTER-COMP'	14	'AUTO-PILOT'	'H-FUNC'	'HEADING'
MACH'	43	SIMULATE 'ADC' SLAVE	TALKER-LISTENER	'ADC'	'M-FUNC'	'MASTER-COMP'	'MACH-FUNC'	'MACH'
VERTICAL SPEED'	63	SIMULATE 'ADC' SLAVE	TALKER-LISTENER	'ADC'	'S-FUNC'	'MASTER-COMP'	'SPD-FUNC'	'SZ'
TRUE-AIR	91	MONITOR 'TAS' SLAVE	TALKER-LISTENER	'ADC'	'TAS-FUNC'	'MASTER-COMP'	'TAS-FUNC'	'TAS'

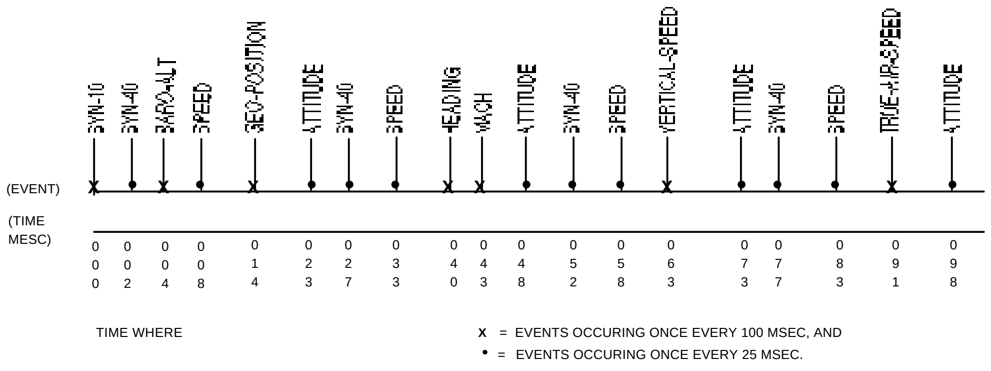


Figure 13-2. Sample bus traffic

The following C/ATLAS procedural statement would implement the traffic:

```
004000 ENABLE, EXCHANGE-CONFIGURATION 'CONF-1',
        PROTOCOL 'BUS-1', CNX TRUE BUS-SIG COMP BUS-RET  $
005000 DO, EXCHANGE USING 'CONF-1', TEST-EQUIP-ROLE SLAVE,
        (STARTING WHEN 'CLK1' 0 MSEC,
        WITH PRIORITY 1,  EVERY 100 MSEC, UNTIL 500 MSEC,
        EXCHANGE  'SYN-10', TEST-EQUIP-ROLE MASTER,
        DELAY 4 MSEC,
          EXCHANGE  'BARO-ALT',
          TEST-EQUIP-ROLE MONITOR,
          TEST-EQUIP-MONITOR DATA 'ZB',
        DELAY 10 MSEC,
          EXCHANGE  'GEO-POSITION',
          TEST-EQUIP-ROLE MONITOR,
          TEST-EQUIP-MONITOR DATA 'POSITION' (1 THRU 5),
        DELAY 26 MSEC,
          EXCHANGE 'HEADING',
          TEST-EQUIP-SIMULATE LISTENER 'AUTO-PILOT-H-FUNC',
          TEST-EQUIP-MONITOR DATA 'HEADING',
        DELAY 3 MSEC,
          EXCHANGE  'MACH',
          TEST-EQUIP-SIMULATE TALKER 'ADC-M-FUNC',
        DELAY 20 MSEC,
          EXCHANGE 'VERTICAL-SPEED',
          TEST-EQUIP-SIMULATE TALKER 'ADC-S-FUNC',
        DELAY 28 MSEC,
          EXCHANGE  'TRUE-AIR-SPEED',
          TEST-EQUIP-ROLE MONITOR,
          TEST-EQUIP-MONITOR DATA 'TAS',
          AND STARTING WHEN 'CLK1' 2 MSEC, WITH PRIORITY 2,
          EVERY 25 MSEC, UNTIL 500 MSEC, EXCHANGE  'SYN-40',
          TEST-EQUIP-ROLE 'MASTER',
        DELAY 6 MSEC,
          EXCHANGE  'SPEED',
          TEST-EQUIP-ROLE MONITOR,
          TEST-EQUIP-MONITOR DATA 'SPEED',
        DELAY 15 MSEC,
        EXCHANGE  'ATTITUDE',
        TEST-EQUIP-ROLE MONITOR,
        TEST-EQUIP-MONITOR DATA 'ATTITUDES'
        'PHI1' 'THETA1'
    ),
    PROCEED,
    MAX-TIME 505 MSEC $
```

13.3 UPDATE EXCHANGE-CONFIGURATION statement

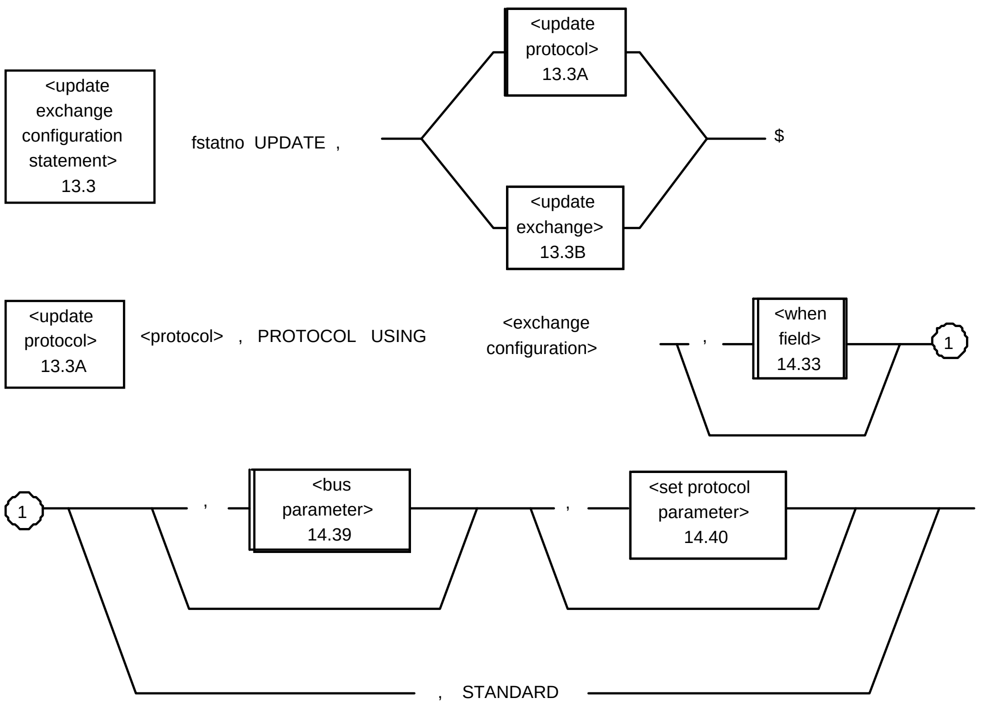
13.3.1 Function

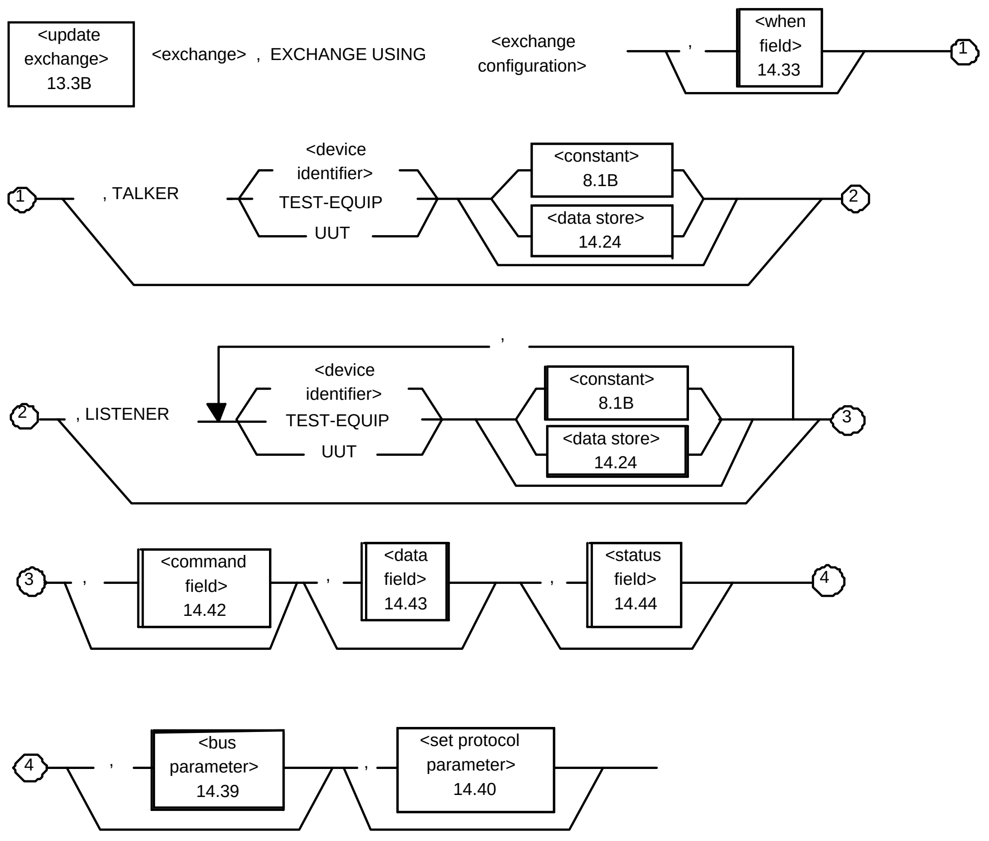
The <update exchange configuration statement> enables the modification of <bus parameter> and <protocol parameter> values without interrupting the databus operation. It can operate at the <protocol> level or at the level of an individual <exchange>. The statement can also be used to update the COMMAND, DATA, and STATUS words that are sent by the test equipment at the individual <exchange> level.

13.3.2 Formal syntax

Reference update-exchange-configuration-statement

13.3.3 Syntax diagram





13.3.4 Rules

13.3.4.1 Rules applicable to an <update exchange configuration statement> that references an <exchange> label

1. The invoked <exchange configuration> shall be the same as that employed in the associated <do exchange statement>.

2. The <exchange> label shall be one that was defined previously in a <define exchange statement>. It shall be currently active following a reference in a <do exchange statement> with a PROCEED field.

3. Only those keywords (COMMAND, DATA, STATUS, TALKER, LISTENER, BUS-PARAMETER, or PROTOCOL-PARAMETER) that occurred in the <define exchange statement> may appear in the <update exchange statement>. BUS-PARAMETER and PROTOCOL-PARAMETER may only occur if the corresponding BUS-PARAMETER or PROTOCOL-PARAMETER field entry in the associated <establish protocol statement> are established with both the UPDATABLE and EXCHANGE attributes. In this case, only those fields that were so established may occur in the <update exchange statement>. They are then modified to the replacement value for the referenced exchange only.

4. If an <update exchange statement> bypasses a keyword that was included in the <define exchange statement> its associated variables remain unchanged.

5. A databus word or parameter changed in an <update exchange statement> retains its new value for all subsequent bus transactions until another <update exchange statement> references it or the <do exchange statement> execution is terminated.

6. Execution of an <update exchange statement> will employ the new variable values in the next regularly scheduled relevant <databus transaction> as defined in the <do exchange statement>.

7. The optional <when field> defines the point in time when the <exchange> is to be updated.

13.3.4.2 Rules applicable to an <update exchange configuration statement> that references a <protocol> label

1. The <exchange configuration> shall have been made active by a preceding <enable exchange configuration statement> and <do exchange statement> with a PROCEED field.

2. BUS-PARAMETER and PROTOCOL-PARAMETER may only occur if the relevant BUS- PARAMETER or PROTOCOL-PARAMETER field entries in the corresponding <establish protocol statement> were established with both UPDATABLE and PROTOCOL attributes. They are updated to the new value for all exchanges that did not contain a superseding value for the particular parameter in the corresponding <define exchange statement>.

3. The set of parameters listed in the <update exchange configuration statement> will replace the equivalent parameters if they were previously set by the <enable exchange configuration statement>.

4. Bus and protocol parameters changed in an <update exchange configuration statement> retain their new value for all subsequent bus transactions until another <update exchange configuration statement> references them or the <do exchange statement> execution is terminated.

13.4 FETCH EXCHANGE-CONFIGURATION statement

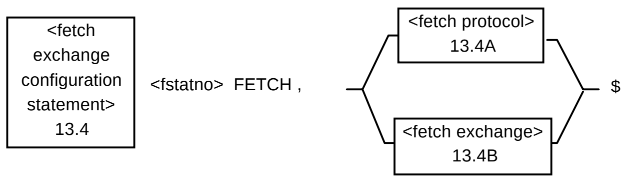
13.4.1 Function

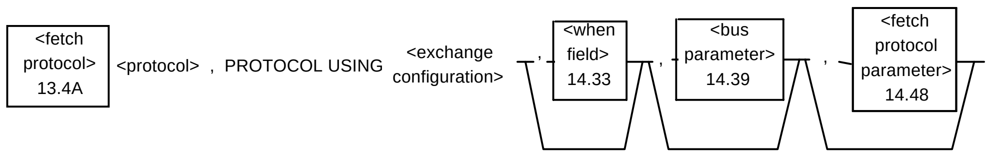
This statement’s function is to fetch data that has been monitored by the test equipment as a result of a previous <do exchange statement> with the PROCEED option selected. FETCH transfers the monitored values into named variables specified in the corresponding field in the <fetch exchange configuration statement>.

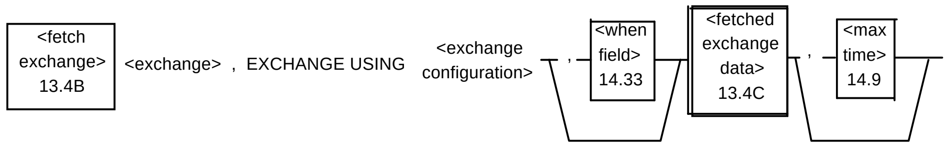
13.4.2 Formal syntax

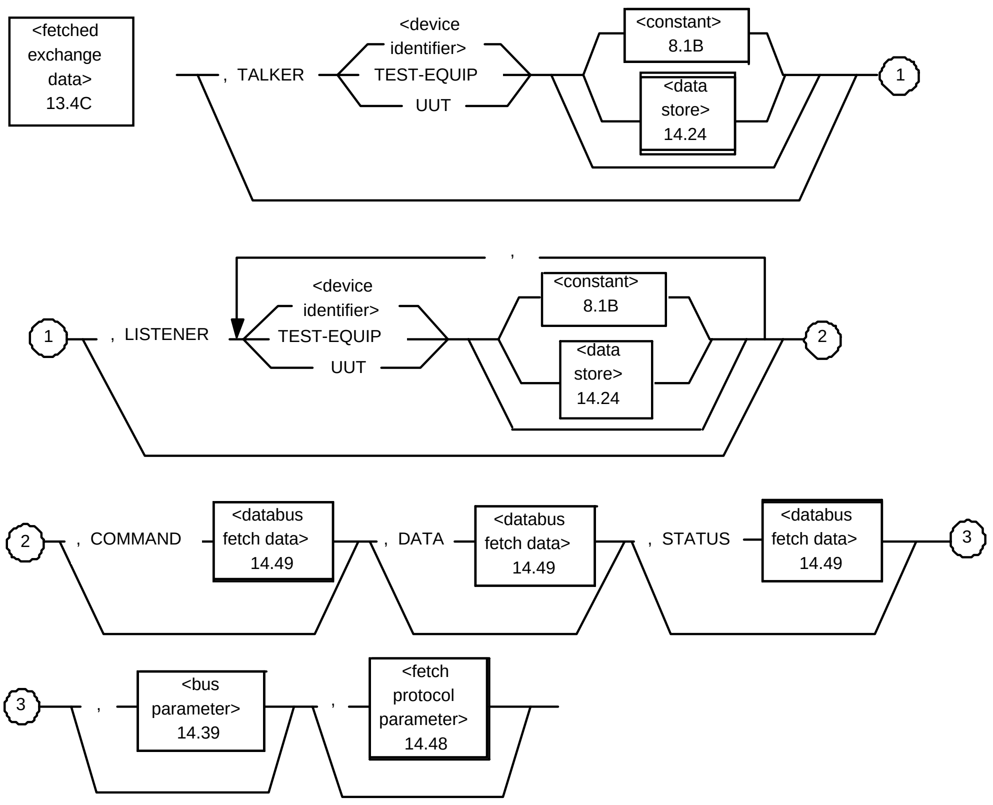
Reference fetch-exchange-configuration-statement

13.4.3 Syntax diagram









13.4.4 Rules

13.4.4.1 Rules applicable to a <fetch exchange configuration statement> that references an <exchange> label

1. The <exchange configuration> shall have been made active by a preceding <enable exchange configuration statement>.

2. The <exchange> label shall be one that was defined previously in a <define exchange statement>. It shall be currently active following a reference in a <do exchange statement> with the PROCEED option.

3. Only those fields specified as FETCHABLE in the associated <establish protocol statement> may be fetched. For COMMAND, DATA, and STATUS information, the corresponding TEST- EQUIP-MONITOR field shall be specified in the <establish protocol statement>. For <bus parameter>s and <protocol parameter>s each specified parameter to be read shall be established with both the FETCHABLE and EXCHANGE attributes and the fetched value shall be consistent with the established type in the associated <establish protocol statement>.

4. The COMMAND, DATA, and STATUS information that is extracted by this statement and is not sent by the test equipment shall be identified in the TEST-EQUIP-MONITOR field for the <exchange> in the associated <do exchange statement>.

13.4.4.2 Rules applicable to a <fetch exchange statement> that references a <protocol> label

1. The <exchange configuration> shall have been made active by a preceding <enable exchange configuration statement>.

2. Only those fields specified in the associated <establish protocol statement> may be fetched by the <fetch protocol statement>. Each specific <bus parameter> or <protocol parameter> to be read shall be established with both the FETCHABLE and PROTOCOL attributes in the associated <establish protocol statement>. The fetched value shall be consistent with the type in the <establish protocol statement>.

13.4.5 Notes and examples

The associated data transfer may not have preceded the initiation of the <fetch exchange configuration statement>, the optional <max time> field establishes a maximum delay waiting for the completion.

13.5 ENABLE EXCHANGE-CONFIGURATION statement

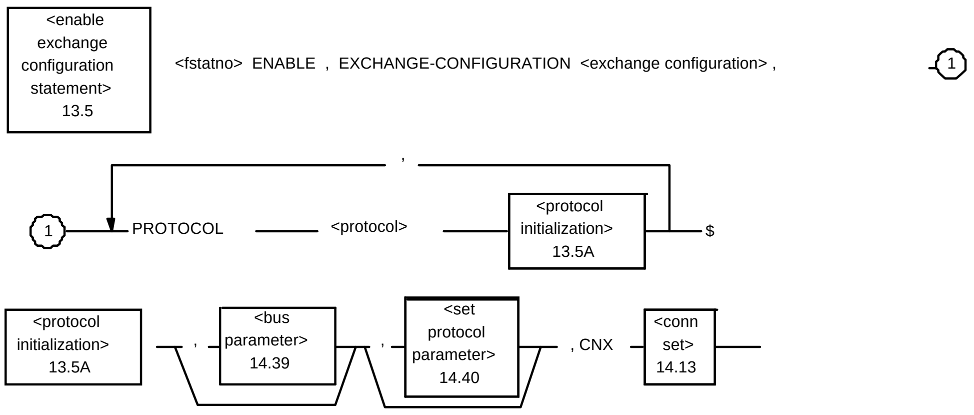
13.5.1 Function

This statement’s function is to enable a set of <protocol>s and to initialize them for the subsequent <do exchange statement>, <update exchange configuration statement>s, and <fetch exchange configuration statement>s and to connect them to the correct UUT pins.

13.5.2 Formal syntax

Reference enable-exchange-configuration-statement

13.5.3 Syntax diagram



<enable exchange configuration statement> 13.5
<fstatno> ENABLE , EXCHANGE-CONFIGURATION <exchange configuration> ,
1
,
1
PROTOCOL
<protocol>
<protocol initialization> 13.5A
$
<protocol initialization> 13.5A
,
<bus parameter> 14.39
,
<set protocol parameter> 14.40
, CNX
<conn set> 14.13

13.5.4 Rules

1. The <exchange configuration> shall be one defined in a <define exchange configuration statement>. It shall not be currently active as the result of a previous <enable exchange configuration statement>.

2. If a specific bus parameter is not specified for a particular <protocol> in the <enable exchange configuration statement>, the databus is initialized with standard bus parameters for that parameter.

3. If a specific protocol parameter is not specified for a particular <protocol> in the <enable exchange configuration statement>, the databus is initialized with standard <protocol parameter>s for that parameter.

4. Each referenced <protocol> can occur only once in the statement and shall be included in the referenced <exchange configuration>.

5. Any <bus parameter> specified shall be defined in the corresponding <establish protocol statement> with both UPDATABLE and PROTOCOL attributes. The <dim>, data <type>, and data value shall be consistent with the <bus parameter> definition in the <establish protocol statement>.

6. Any <protocol parameter> specified shall be defined in the corresponding <establish protocol statement> with both UPDATABLE and PROTOCOL attributes. The usage of the <protocol parameter> shall be consistent with the usage defined in the <establish protocol statement>.

7. In the CNX field each <pin descriptor> shall have appeared in the CNX field of the <establish protocol statement>.

8. A <protocol> becomes active when it is referenced in an <enable exchange configuration statement> and remains active until the associated <exchange configuration> is referenced in a <disable exchange configuration statement>.

13.6 CONNECT EXCHANGE-CONFIGURATION statement

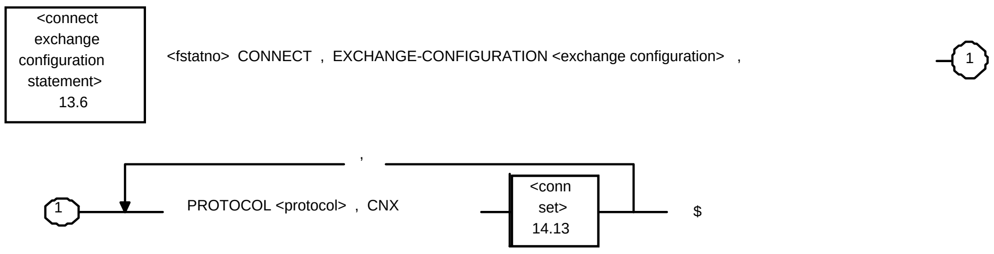
13.6.1 Function

This statement's function is to re-connect databus <protocol>s that previously have been made active by an <enable exchange configuration statement> and disconnected by a subsequent <disconnect exchange configuration statement>.

13.6.2 Formal syntax

Reference connect-exchange-configuration-statement

13.6.3 Syntax diagram


<connect exchange configuration statement> 13.6
<fstatno> CONNECT , EXCHANGE-CONFIGURATION <exchange configuration> ,
1
1
,
PROTOCOL <protocol> , CNX
<conn set> 14.13
$

13.6.4 Rules

1. A <connect exchange configuration statement> shall reference an active <exchange configuration>.

13.6.5 Notes and examples

The <connect exchange configuration statement> is included to facilitate tests on a databus having many potential connections to a UUT that require the databus to be repeatedly connected and disconnected to the UUT pins.

13.7 DISCONNECT EXCHANGE-CONFIGURATION statement

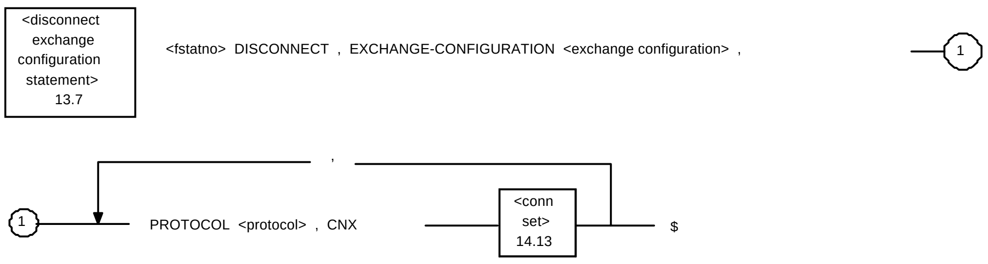
13.7.1 Function

This statement's function is to disconnect databus <protocol>s that previously have been made active by an <enable exchange configuration statement>.

13.7.2 Formal syntax

Reference disconnect-exchange-configuration-statement

13.7.3 Syntax diagram


<disconnect exchange configuration statement> 13.7
<fstatno> DISCONNECT , EXCHANGE-CONFIGURATION <exchange configuration> ,
1
1
,
PROTOCOL <protocol> , CNX
<conn set> 14.13
$

13.7.4 Rules

A <disconnect exchange configuration statement> shall reference an active <exchange configuration> that is currently connected to at least one UUT pin.

13.7.5 Notes and examples

The <disconnect exchange configuration statement> is included to facilitate tests on a databus having many potential connections to a UUT that require the databus to be repeatedly connected and disconnected to the UUT pins.

13.8 DISABLE EXCHANGE-CONFIGURATION statement

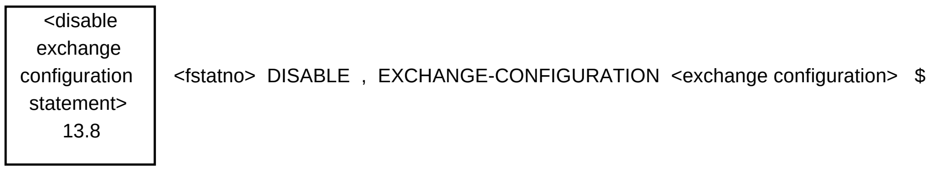
13.8.1 Function

This statement's function is to render an active <exchange configuration> inactive.

13.8.2 Formal syntax

Reference disable-exchange-configuration-statement

13.8.3 Syntax diagram


<disable exchange configuration statement> 13.8
<fstatno> DISABLE , EXCHANGE-CONFIGURATION <exchange configuration> $

13.8.4 Rules

1. The <exchange configuration> shall be currently active.

2. The <exchange configuration> will be deactivated as a result of the <disable exchange configuration statement>.

3. Once a <protocol> has been referenced by an <enable exchange configuration statement> it cannot be made active as a facility in a different <exchange configuration> until the original <exchange configuration> has been rendered inactive by a <disable exchange configuration statement>.

4. A REMOVE, ALL statement will disable all enabled <exchange configuration>s.

14.0 Field and subfield definition

14.1 <statement characteristics>

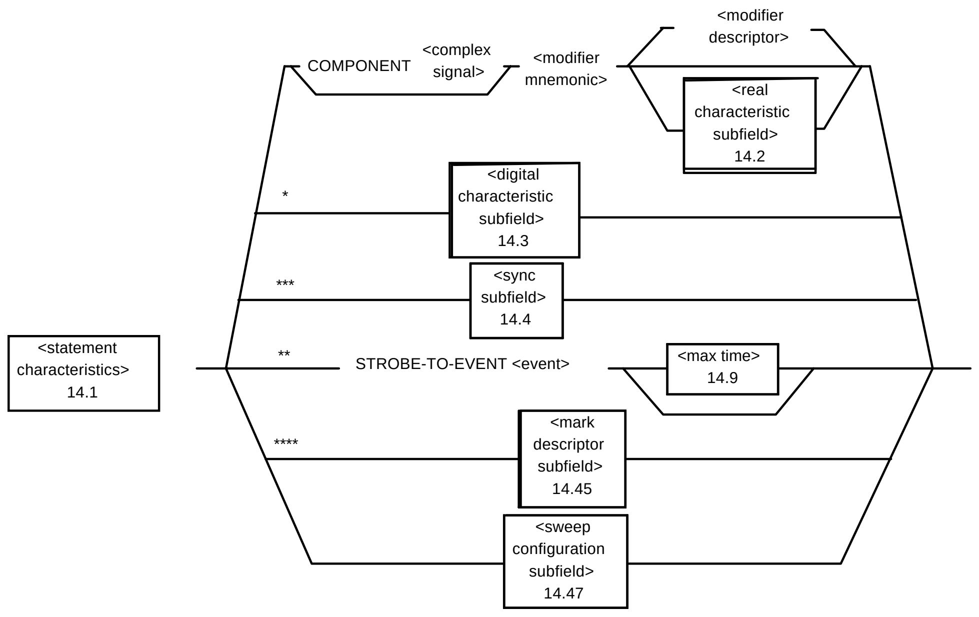
14.1.1 Function

Listed with each noun in clause 16 is a list of the characteristics for which a specific value may be assigned. A separate statement characteristic field is required for each characteristic used, but every characteristic listed with the noun need not be included.

14.1.2 Formal syntax

Reference statement characteristics

14.1.3 Syntax diagram



<statement characteristics> 14.1
COMPONENT
<complex signal>
<modifier mnemonic>
<modifier descriptor>
<real characteristic subfield> 14.2
*
<digital characteristic subfield> 14.3
***
<sync subfield> 14.4
**
STROBE-TO-EVENT <event>
<max time> 14.9
****
<mark descriptor subfield> 14.45
<sweep configuration subfield> 14.47

NOTES

* This branch is used for LOGIC-type nouns.
** This branch is used only for SENSOR statements.
*** This branch is used only for SOURCE statements.
**** This branch is used only for COMPLEX SIGNAL.

14.1.4 Rules

1. A sensor statement cannot imply the application of an accompanying load. Loads shall be connected and removed by load statements.

2. A <modifier descriptor>, <real characteristic subfield>, or <digital characteristic subfield> is included to indicate the required value of the statement characteristic.

3. A <modifier descriptor> shall be appropriate for the modifier and shall be used in accordance with the rules for the modifier in clause 17 of this specification. The <modifier descriptor> may be a <modifier descriptor name> which has been introduced in an EXTEND statement.

4. STROBE TO EVENT indicates the referred event at which the instrument reading is to be initiated.

5. The keyword COMPONENT is only permitted when the entry in the <noun field> is a <complex signal> and it is desired to employ a modifier associated with one of the COMPONENTS specified in the <define complex signal structure>.

6. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the definition for the referenced component in the <complex signal>.

14.1.5 Notes and examples

The following example shows how to specify that a measuring device does not draw more than 0.1mA from the UUT:

```
120510 MEASURE, (VOLTAGE INTO 'MGW'), DC SIGNAL,
         CURRENT LIMIT_TO MAX 0.1 MA,... $
```

Alternatively, the prohibition of implied loads can be illustrated by an illegal example, as follows:

```
120510 MEASURE, (VOLTAGE INTO 'MGW'), DC SIGNAL,
         CURRENT 2.0A, ... $   (illegal)
```

The user is attempting to connect a load that draws 2 A when measuring a DC voltage. To do this legally, the user shall APPLY the load and then MEASURE the voltage.

14.2 <real characteristic subfield>

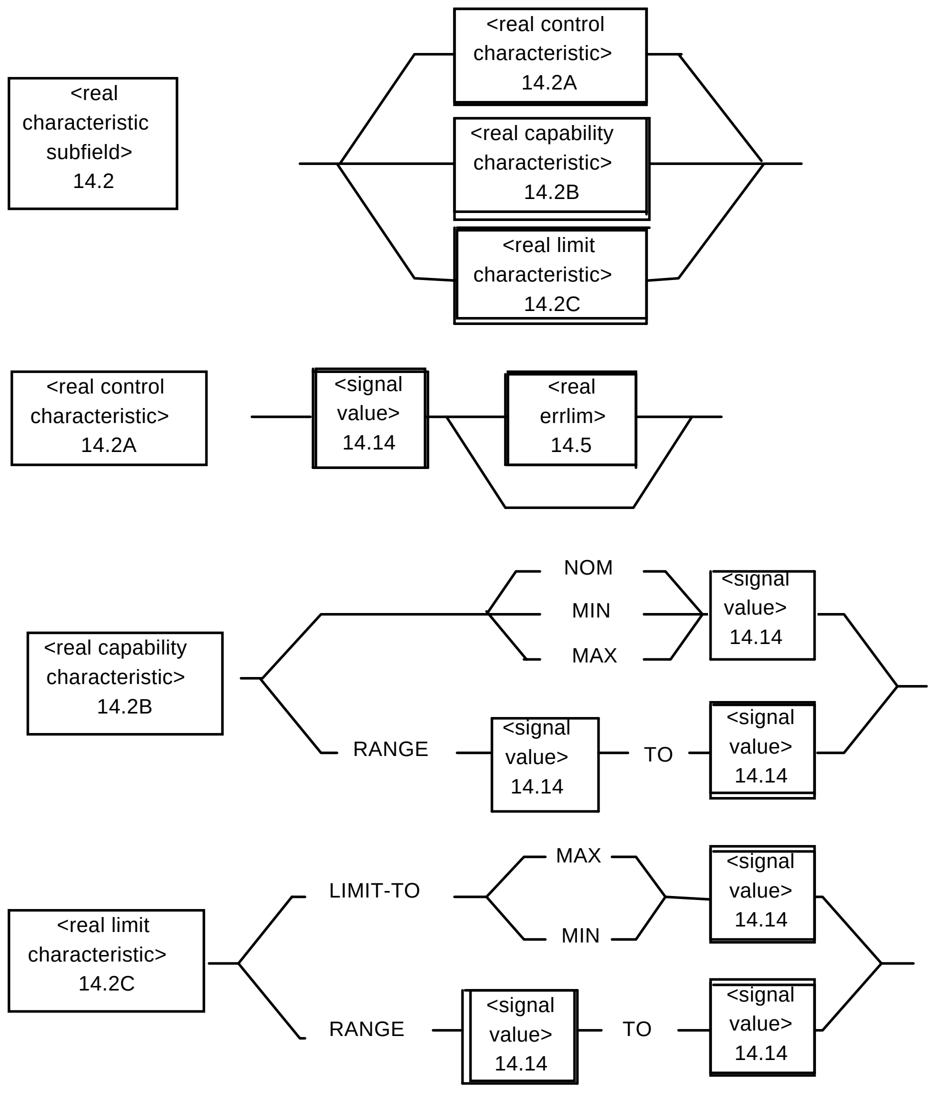
14.2.1 Function

A <real characteristic subfield> specifies values or a value range for noun characteristics expressible in real numbers.

14.2.2 Formal syntax

Reference real characteristic subfield

14.2.3 Syntax diagram


<real characteristic subfield> 14.2
<real control characteristic> 14.2A
<real capability characteristic> 14.2B
<real limit characteristic> 14.2C
<real control characteristic> 14.2A
<signal value> 14.14
<real errlim> 14.5
<real capability characteristic> 14.2B
NOM
MIN
MAX
<signal value> 14.14
RANGE
<signal value> 14.14
TO
<signal value> 14.14
<real limit characteristic> 14.2C
LIMIT-TO
MAX
MIN
<signal value> 14.14
RANGE
<signal value> 14.14
TO
<signal value> 14.14

14.2.4 Rules

1. In sensor statements, the mnemonic for the noun modifier that is the same as that of the measured characteristic shall be included in the <statement characteristics> field, except for those specifically excluded by subclauses in clauses 16 and 17. This mnemonic shall be followed by a <real capability characteristic> format of the <real characteristic subfield>.

2. A noun modifier mnemonic that is followed by a <real control characteristic> format of the <real characteristic subfield> specifies that the test station shall control the value of the parameter identified by the mnemonic and shall set it to the specified value.

3. A noun modifier mnemonic that is followed by a <real capability characteristic> format of the <real characteristic subfield> specifies that the test station does not control the value of the parameter identified by the mnemonic but is able to implement the action specified in the statement for any value of it that is within the specified range.

4. A noun modifier mnemonic that is followed by a <real limit characteristic> format of the <real characteristic subfield> specifies that the test station does not control the specific value of the parameter identified by the mnemonic but actively restricts that value to be within the specified range.

5. A value may be indicated by a label that refers to a value established by another statement. Such a label shall be succeeded by a MAX, MIN, or RANGE subfield that establishes the permissible range of values for the label.

6. A dimensional unit identifier appropriate to the mnemonic is written following numeric values.

7. Permissible accuracies of controlled parameters are specified using <real errlim> within the <real control characteristic> format of the <real characteristic subfield>. For parameters that identify a capacity or limit, ranges are specified using MAX, MIN, or RANGE before the first value.

14.3 <digital characteristic subfield>

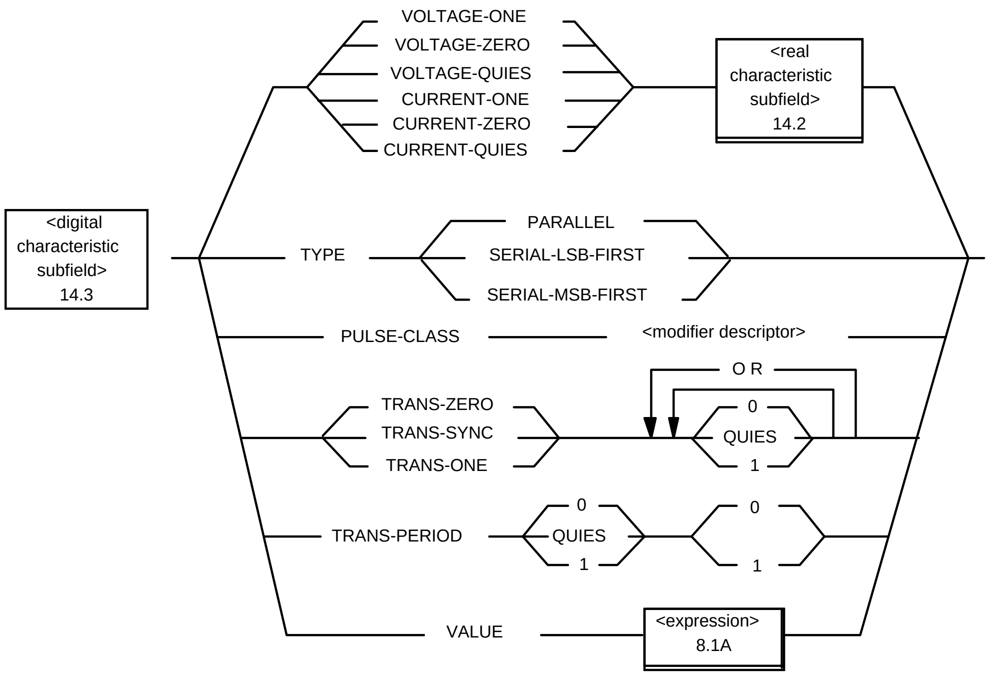
14.3.1 Function

A <digital characteristic subfield> specifies the analog form of a digital or logical signal.

14.3.2 Formal syntax

Reference digital characteristic subfield

14.3.3 Syntax diagram



14.4 <sync subfield>

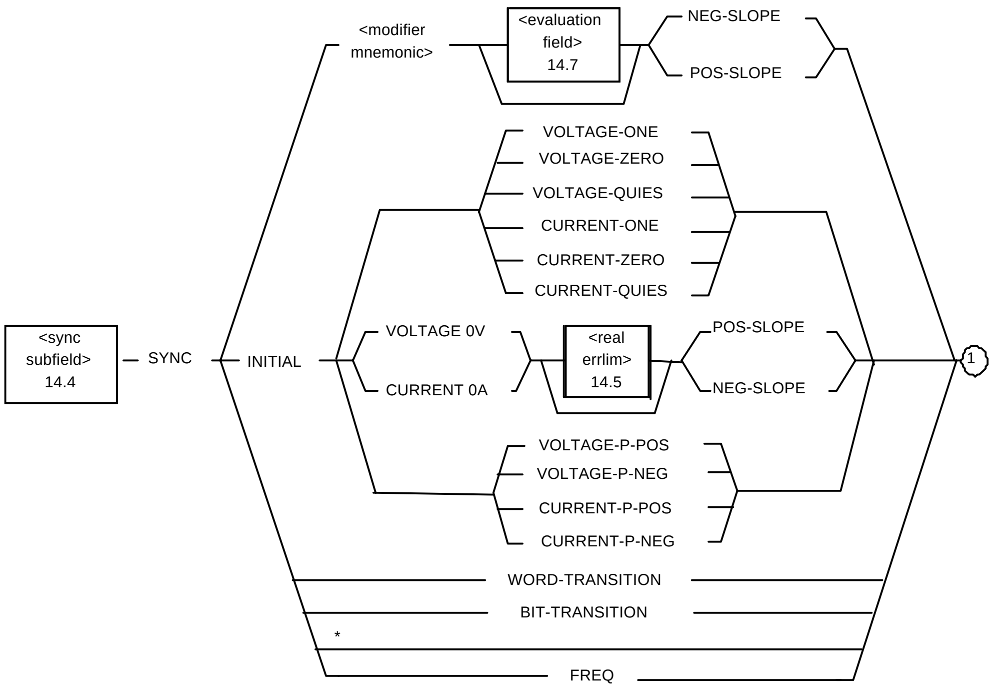
14.4.1 Function

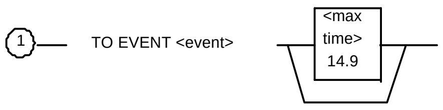
The <sync subfield> function is to indicate that the waveform is synchronized to an <event>.

14.4.2 Formal syntax

Reference sync subfield

14.4.3 Syntax diagram





NOTE

* This branch is allowed with the noun WAVEFORM, with which noun it shall be used once and only once, and within the <sweep configuration subfield> of any signal noun being swept.

14.4.4 Rules

1. The use of FREQ to indicate frequency reference synchronization specifies that a signal defined by a statement that includes a <sync subfield> is in phase-locked synchronization with a frequency defined by a repetitive event referenced by an <event> name.

2. Initial coincidence synchronization covers other forms of synchronization where initial conditions need to be specified. For these forms, a statement defines a point on a signal that is to be the first change from a quiescent value at a point in time defined by an <event> and after which the signal continues as defined without further interference from the <event>.

3. Synchronization occurs at the coincidence of specific points on the synchronized signal and the event; these are identified respectively as the synchronized point and the reference point.

4. The synchronized point of specific VOLTAGE or CURRENT waveforms can be defined as having an INITIAL value that is the value of the synchronized signal at the point of synchronization. There are three forms of INITIAL value synchronization:

a) The point at which a digital logic signal starts a specific logic state, either the logic 0, 1, or quiescent state.

b) The point at which the waveform crosses the zero value either in a POS_SLOPE or a NEG_SLOPE direction.

c) The point at which the waveform is at a positive or negative peak value.

5. The synchronized point of digital data transfers is defined as the point at which to initiate each bit or word transfer synchronous to the reference point(s).

a) BIT_TRANSITION is used in the general case for digital data, such as SERIAL, to specify that the start of each bit position is the synchronized point.

b) WORD_TRANSITION is used for digital data to specify that the start of each data word is the synchronized point. WORD_TRANSITION is used primarily when a SERIAL data transfer of multiple words is to be synchronized into two separate reference signals, one defining the SERIAL bit sync (SYNC BIT_TRANSITION) and one defining the SERIAL word sync (SYNC WORD_TRANSITION). PARALLEL data transfers of multiple words may use either BIT_TRANSITION or WORD_TRANSITION to specify the same synchronized point.

c) If the synchronized signal represents a single event (such as the transmission of one PARALLEL data word), that event is synchronous with the first occurrence of the reference point. If the synchronized signal represents a periodic event (such as the bits of a SERIAL data word), each successive event (bit) is synchronous with repeated occurrences of the reference condition.

6. The synchronized point of the signal WAVEFORM is the initial value of the waveform. For SOURCE signals the initial value synchronized is the first value specified in the STIM <array range> of waveform amplitudes.

7. When the SYNC subfield is used in a SOURCE signal statement to reference a UUT signal or event, the optional MAX_TIME field of that statement should be used to specify an escape mechanism in case the reference signal or event fails to occur (e.g., due to a UUT failure).

8. The synchronized point of a <sweep configuration subfield> of a signal is defined as the point at which to initiate the sweep of the specified signal parameter. A repetitive reference <event> will trigger successive sweeps or steps of the signal parameter depending on the type of sweep specified.

14.5 <real errlim> (real error limit)

14.5.1 Function

A <decimal number> is included to indicate the required accuracy of the test equipment stimulus or measurement function, as observed at the UUT terminals. The accuracy subfield begins with the word ERRLMT and is followed by an expression of the allowable algebraic deviation. In the ERRLMT subfield, the <dim> is the same dimension identifier from 15.8 as was used for the basic characteristics except that PC (which 15.8 refers to as a ratio) only indicates an accuracy requirement that is a calculated percentage of the desired characteristic value or measured value (see 14.5.5). The + value is written first in the sample formats, but the order of writing + and - values may be reversed in an actual statement. The following are the acceptable formats for accuracy expressions, x and y are decimal or floating point decimal values.

14.5.1.1 General Form

ERRLMT + x dim - y dim

ERRLMT + x PC - y PC

14.5.1.2 Symmetrical form

When the limits are symmetrical, the following notations may be used.

ERRLMT +- x dim

ERRLMT +- x PC

NOTE When writing the plus and minus signs, the signs shall be written with zero spaces separating the signs and with the plus sign preceding the negative sign.

14.5.1.3 Mixed form

A combination of dim and calculated PC limits may be expressed as follows.

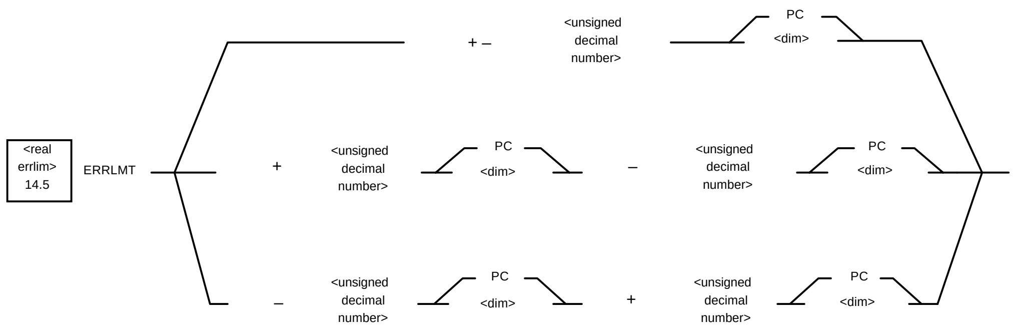
ERRLMT + x dim - y PC

ERRLMT + x PC - y dim

14.5.2 Formal syntax

Reference real errlim

14.5.3 Syntax diagram


<real errlim> 14.5
ERRLMT
+ –
<unsigned decimal number>
PC
<dim>
+
–
<unsigned decimal number>
PC
<dim>
<unsigned decimal number>
PC
<dim>
–
+
<unsigned decimal number>
PC
<dim>
<unsigned decimal number>
PC
<dim>

14.5.4 Rules

1. When no accuracy subfield is included in a <statement characteristics> field, no special control is exercised over the ATE accuracy for this statement. The specified value of this characteristic is established by the inherent accuracy of the ATE system.

2. All accuracy requirements should be stated in terms of actual UUT needs as measured at the UUT terminals and should not take into account any ATE equipment errors.

14.5.5 Notes and examples

Note that PC as used in the accuracy subfield has a different connotation than PC as the absolute dimension of some characteristic such as OVERSHOOT of PULSED AC. In the accuracy subfield, PC describes a calculated percentage of the described value. VOLTAGE 10V ERRLMT + 1V - 5 PC means that values of voltage from 9.5-11V are acceptable. The same applies to characteristics with PC as the basic dimension. Taking the OVERSHOOT example, OVERSHOOT 10 PC ERRLMT +- 5 PC means that values from 9.5PC to 10.5PC are acceptable. (It does not mean that 5-15 percent are the acceptable values.)

14.6 <measured characteristic>

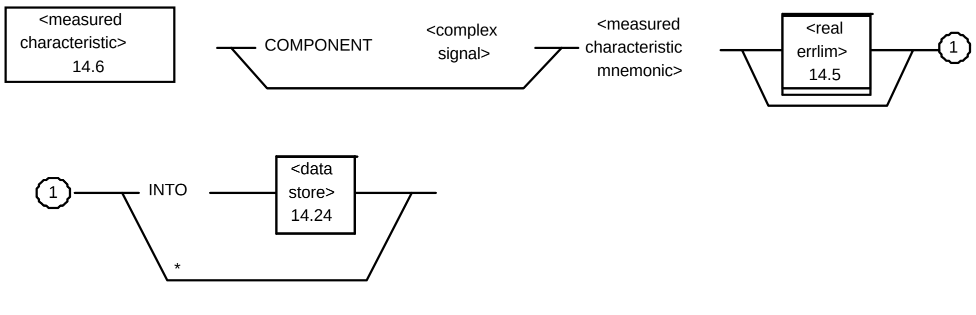
14.6.1 Function

A <measured characteristic> field is included in sensor type statements to specify which of the available characteristics (of those listed with the noun in clause 16 or those introduced by <extend statement>s or COMPLEX SIGNAL statements) is to be evaluated by the sensor function. The field is positioned between the verb and the noun in the statement and consists of an optional COMPONENT <complex signal> subfield and one of the modifier mnemonics (from the list associated with the noun), which is followed by an optional error limit subfield and is followed by an optional INTO subfield containing a <data store>. If present, the error limit subfield indicates the required accuracy to which the reading shall be taken. The <data store> is preceded by the connective INTO. It indicates that the result is to be stored in the <data store> indicated.

14.6.2 Formal syntax

Reference measured characteristic

14.6.3 Syntax diagram



NOTE

* Only legal when the measured values will be stored in a RESP <array range> list.

14.6.4 Rules

1. Ranging for the <measured characteristic>. In all sensor type statements (MEASURE, VERIFY, etc.), the explicit <measured characteristic> shall be ranged (i.e., sized) by including within the statement one of the following:

a) a maximum value (i.e., a MAX subfield for positive values or a MIN subfield for negative values).

b) a range for the value (i.e., RANGE subfield).

If more than one of the listed ranging options is incorporated in the statement or an error limit is included in the <measured characteristic> field, the same dimensional units shall be used throughout.

2. The scale of measured values following the execution of the sensor type statement for the value in the <data store> that is referenced in the <measured characteristic> field, if any, will be at the scale implied by the dimensional units used in the MAX or MIN subfield or the RANGE subfield.

3. The <measured characteristic> (digital). The digital modifiers and the specifically defined analog modifiers that can be used as a SENSOR <measured characteristic> are identified in clause 17 (Noun Modifiers). Other analog modifiers can be used for measuring analog characteristics of the digital signals in accordance with established analog rules.

4. For all analog measurements, each sensor type statement shall define the range of values to be covered by the <measured characteristic> (see rule 2). When the <measured characteristic> is VALUE, this requirement is met automatically by defining the digital word length in the DECLARE statement.

5. The measured value and its type, as determined from the type entry associated with the <measured characteristic> as listed in clause 16, shall be assignment compatible with the <data store> following the connective INTO.

6. The keyword COMPONENT is only permitted for a COMPLEX SIGNAL noun.

7. When the keyword COMPONENT is used, the <measured characteristic mnemonic> shall be specified by an identical <modifier mnemonic> in the definition for the referenced component in the <complex signal>.

14.6.5 Notes and examples

1. In the example MEASURE, (FREQ INTO 'MIDBAND'), AC SIGNAL,... the general type of signal is an AC signal and the characteristic that will be measured is its frequency.

2. Examples of how the dimensions in the associated MAX, MIN, or RANGE subfield affect the scale of 'MIDBAND':

a) ... , FREQ RANGE 1 KHZ TO 30 KHZ,...

if the value of the frequency is 12,000 Hz, the value in 'MIDBAND' will be 12.

b) VERIFY, (FREQ INTO 'MIDBAND'), AC SIGNAL,
   UL 37.5 KHZ LL 33 KHZ, ...

FREQ MAX 40 KHZ,...

if the value of frequency is 36,240 Hz, the value in 'MIDBAND' will be 36.24.

c) VERIFY, (FREQ INTO 'MIDBAND'), AC SIGNAL,

UL 37.5 KHZ LL 33 KHZ, ... ,

FREQ MAX 40000 HZ, ...

This statement is incorrect since the dimension in the MAX subfield is different from the dimension given in the evaluation field.

14.7 <evaluation field>

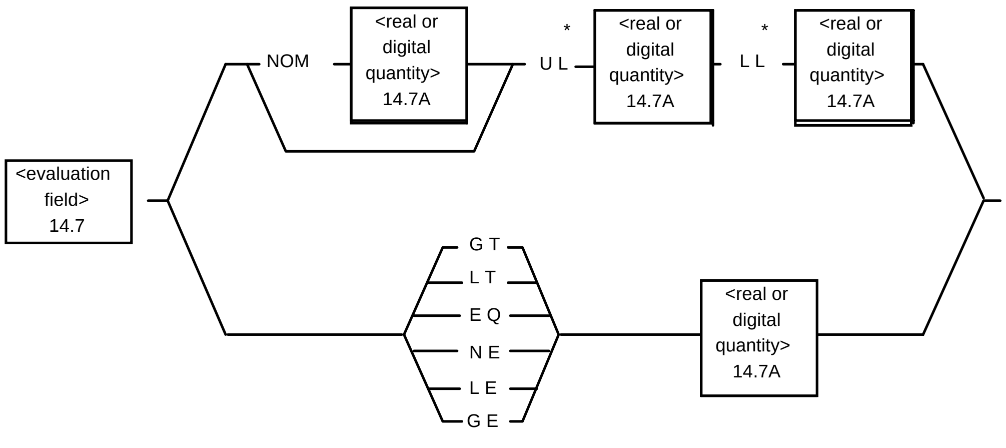
14.7.1 Function

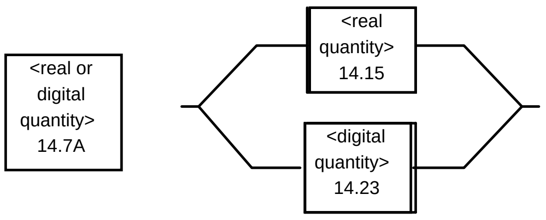
An <evaluation field> is included in statements such as VERIFY to express the values of some program variable. Depending on the format of the <evaluation field>, one or more conditions are established that may be acted upon in an IF or WHILE statement. Only the COMPARE and VERIFY statements set the condition, HI, LO, GO, and NOGO, as specified in table 14-1.

14.7.2 Formal syntax

Reference evaluation field

14.7.3 Syntax diagram


NOM
<real or digital quantity> 14.7A
*
U L
<real or digital quantity> 14.7A
*
L L
<real or digital quantity> 14.7A
<evaluation field> 14.7
G T
L T
E Q
N E
L E
G E
<real or digital quantity> 14.7A



NOTE

* UL and LL may be interchanged.

14.7.4 Rules

1. <evaluation field> Relationships

Table 14-1 lists the possible formats for <evaluation field> in the first column. The next column lists the possible relationships between a program variable (v) and the numbers in the <evaluation field> (x, y, and z). The last column lists the condition that is established as TRUE by each relationship in the middle column. All other conditions are set to FALSE.

The following abbreviations are used in the evaluation field:

UL	upper limit	GT	greater than
LL	lower limit	LT	less than
NOM	nominal value	EQ	equal to
GE	greater than or equal to	NE	not equal to
LE	less than or equal to		

Evaluation field format	value of variable (v)	conditions established by a VERIFY or COMPARE statement
UL y <dim> LL z <dim> (or)	v > y	HI & NOGO
NOM x <dim> UL y <dim>	y >= v >= z	GO
LL z <dim>	v < z	LO & NOGO
GT x <dim>	v > x	GO
	v <= x	LO & NOGO
LT x <dim>	v < x	GO
	v >= x	HI & NOGO
EQ x <dim>	v = x	GO
	v _ x	NOGO
NE x <dim>	v _ x	GO
	v = x	NOGO
GE x <dim>	v >= x	GO
	v < x	NOGO & LO
LE x <dim>	v > x	NOGO & HI
	v <= x	GO

Table 14-1. Evaluation field relationships

As the words imply, algebraic magnitude determines which limit is identified as UL and which is LL (example: NOM - 5V UL - 4V LL - 6V). However, there is an exception for mechanical angles around 0 DEG. If negative angles are used, the algebraic magnitude convention is not violated (NOM 0 DEG UL 2 DEG LL - 2 DEG); but in some systems, such as compass indicators, it is important to refer to 1 DEG minus 2 DEG as 359 DEG. Therefore, an exception to the algebraic magnitude convention in evaluation limits is made. The <evaluation field> for mechanical angles with a pass-band around 0 degrees may use either a negative angle as the LL or the negative angle plus 360 DEG. Thus,

```
NOM 1 DEG UL 3 DEG LL -2 DEG
```

is the same as the equally acceptable

```
NOM 1 DEG UL 3 DEG LL 358 DEG
```

for mechanical angles around zero degrees.

2. UL, LL, GE, LE, GT, LT, EQ, and NE are relational operators. They separate the quantity v to be evaluated from the evaluation limit x, y, or z in table 14-1. During the execution of the <evaluation field>, the GO, NOGO, HI, and LO flags are set according to table 14-1.

NOM is not a relational operator, it is only a designating keyword for a nominal value rather than a limit.

3. Relationship between HI, LO, GO, and NOGO:

```
GO    = NOT NOGO
NOGO  = HI OR LO
```

For example, setting NOGO to FALSE will automatically set GO to TRUE, but will not set HI or LO flags.

4. Dimensional unit restrictions

All dimensional units appearing in an <evaluation field> shall be identical. For example, if V appears with NOM, V shall also appear with UL and LL, not MV or KV. In addition, if the <evaluation field> appears in a sensor type statement, and if a maximum or minimum value or a range is given in the <statement characteristics> for the <measured characteristic>, the dimensional unit in the MAX, MIN, or RANGE subfield shall be identical to the dimensional units in the <evaluation field>.

14.7.5 Notes and examples

1. Evaluation field for digital numbers

The <evaluation field> adheres to the rules established for the evaluation of analog quantities subject to the number interpretations described below. Digital evaluation limits may be expressed in any of the established number forms: binary, octal, or hexadecimal digits.

2. Don't cares

Often when evaluating the digital response from a UUT, not all of the bits, of the response word are important to that test. These unimportant bits or rather, the uncertainty associated with these bits, shall be removed from the UUT response word before evaluating the response against a reference bit pattern.

3. Categories of digital numbers

The syntax diagrams and rules provide for the use of binary, octal, hexadecimal, or decimal numbers in the evaluation field as either a nominal value or a comparison limit. To fully appreciate the use of this field, three types of digital words are considered, as follows.

a) Pattern. A digital word may contain a string of bits having no numerical significance, but may be simply a collection of binary information. This type of digital word might come from a digital storage UUT (a tape unit or a core memory) or from testing a collection of logic gates. Generally, the evaluation criteria is that the digital bit pattern shall match exactly some reference pattern. The EQ option of the <evaluation field> (with the reference bit pattern expressed in B", O", or X" form) is normally used for this purpose.

b) Digital Number. The bits in a digital word may represent an actual number in binary, BCD, etc., form (for example, a binary address from an address decoder in the UUT). Generally, this number shall also have an exact value for a good UUT. For these cases, the EQ condition is normally used.

c) Analog Equivalent. A digital number may be the digital representation of a parameter that is basically an analog variable (for example, the digitized output of an air data transducer). For these cases, the normal analog-oriented evaluation options (i.e., UL, LL, GT, LT, GE, LE) are generally used, with the evaluation limit generally expressed as a literal digital number. Frequently, the digital number output from such an analog-to-digital UUT does not equal the parameter in engineering units, but differs by some scale factor. This scale factor conversion is not built into the digital <evaluation field>. That is, only pure bit strings can be placed in the evaluation field; needed scale factor conversions shall be done or programmed by the test programmers. For example, if the output from a digital altitude transducer is a pure binary number, BNR of 7 bits, with a scale factor of 'Q' feet/quantum, the test programmer has three options, as follows:

1) Program a conversion from the binary UUT output to engineering units via

```
151015 CALCULATE,
                         'FT' = DEC (BNR, 'WORKAREA', 7) * 'Q' $
```

and then use an IF THEN, or WHILE THEN statement with limits given in feet.

2) Program one or two computations to convert the limits given in engineering units into pure BNR form,

```
161115 CALCULATE,
                 'UL_BN' = DIG (BNR, 'UL-FT',7,0) $
161120 CALCULATE, 'LL_BN' = DIG (BNR, 'LL-FT'/'Q',7,0) $
```

and then use an IF THEN or WHILE THEN statement using the limits as strings of bits.

3) Perform the computations of 2) by hand and enter the results as digital numbers directly in the evaluation of an IF THEN or WHILE THEN statement.

4. Real numbers

There is an infinite continuum of real numbers. Any digital computer has a finite number of different binary patterns to represent this series of real numbers. Thus, a group of nearly equal real numbers will have the same binary representation, and, as far as the computer is concerned, are equal. Unfortunately different computers treat different partially overlapping groups as identical. Thus, two real numbers that are considered to be equal by one computer may be unequal in the opinion of another. Because of these possible anomalies, it is recommended that EQ and NE not be used with decimal numbers.

14.8 <eval statement characteristics> (evaluation statement characteristics)

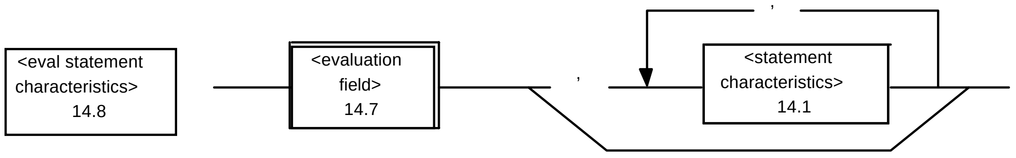
14.8.1 Function

An <eval statement characteristics> field defines the fields usable in statements that include it.

14.8.2 Formal syntax

Reference eval statement characteristics

14.8.3 Syntax diagram



14.9 <max time>

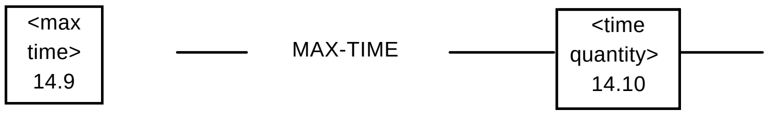
14.9.1 Function

In those statements whose syntax specifies that a <max time> field applies, there is a possibility that the tester could be given an impossible requirement such as waiting for a UUT measurement that will never occur because of a defective UUT. If the tester does not successfully execute the required operation within the specified time, the tester will establish a MAX_TIME condition that may be tested and acted upon by a subsequent statement.

14.9.2 Formal syntax

Reference max_time

14.9.3 Syntax diagram



14.10 <time quantity>

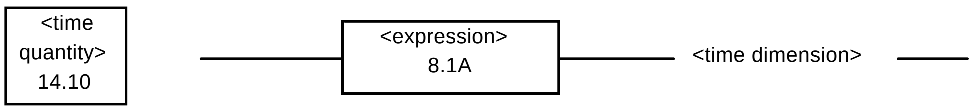
14.10.1 Function

The <time quantity> function is to associate a valid time dimension with an <expression>.

14.10.2 Formal syntax

Reference time quantity.

14.10.3 Syntax diagram



14.10.4 Rules

1. The <expression> shall evaluate to a <decimal number>.

14.11 <condition>

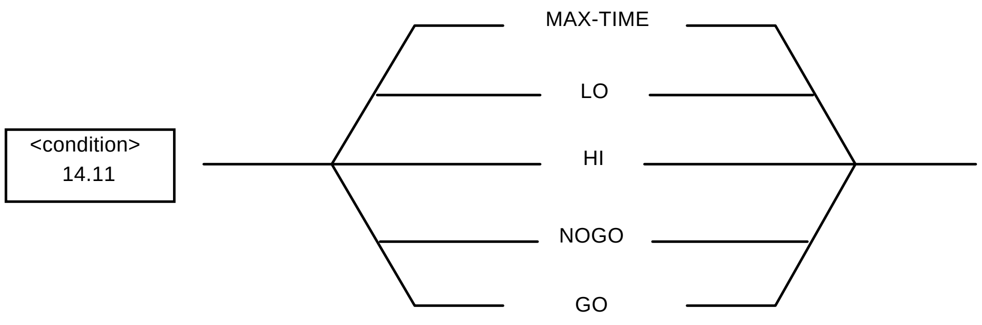
14.11.1 Function

A <condition> is used to make available the status of a test or the results of a measurement comparison. For IF THEN and WHILE THEN statements it is used to determine statements in the procedure that are executed depending on the GO, NOGO, HI, LO, or MAX_TIME condition values.

14.11.2 Formal syntax

Reference condition

14.11.3 Syntax diagram



14.11.4 Rules

Within an expression, a <condition> behaves like a <variable> of type BOOLEAN.

14.12 <conn> (connection field)

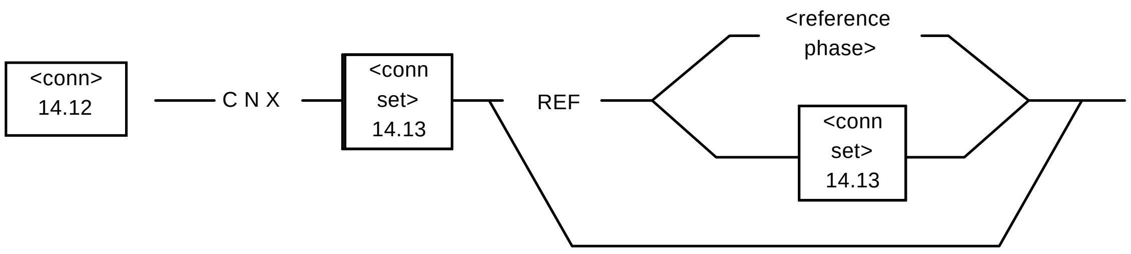
14.12.1 Function

The last field in <identify signal based event statement> source, sensor and load statements specifies the UUT points to which the described function is to be attached. The field is separated from preceding fields by a comma and the word CNX. Generally, connector designations assigned by the UUT manufacturer are used to describe the attachment points (examples: J1-2, P2-AA for electrical connectors; and AIRSPEED, STATIC, PORT-2, FITTING A for pneumatic connectors). Most functions, such as DC signals and synchro signals, have multiple leads that shall be attached to the correct UUT points. The pin descriptors (such as HI and LO, or X, Y, and Z, etc.) are provided in the language to identify the signal port to which the <connection> written immediately following the pin descriptors are to be connected.

14.12.2 Formal syntax

Reference conn

14.12.3 Syntax diagram



14.12.4 Rules

A second set of connectors may be included in the <conn> field preceded by the word REF. This capability is used for signals that shall be defined relative to some reference signal, such as a phase-sensitive or a time-related function. The stated UUT points define the location of the reference signal. When the reference signal is one phase of a multi-phase source that is wholly internal to the test equipment, and, therefore, cannot be identified by UUT connections, it may be identified in the CNX field by a <reference phase> identifier.

14.13 <conn set> (connection set)

14.13.1 Function

All words and characters that are used in the <conn> field, except the manufacturer's pin names, are included in 14.13.1-14.13.5.

14.13.1.1 <pin descriptor>

In any syntax diagram in which <pin descriptor> occurs, a reference may be made to any of the following reserved words or to a <pin descriptor name> that has been introduced in an EXTEND statement.

HI <pin descriptor> denoting "hot" side of a two-wire circuit. The HI side has the same polarity as the magnitude field of the statement when measured with respect to the LO side. HI may also designate the side of the circuit that is most isolated (highest impedance) from the UUT low potential.

LO <pin descriptor> denoting "cold" or return side of a two-wire circuit. The LO side is the reference for the function described in the statement. LO may also designate the side of the circuit that is least isolated (lowest impedance) from the UUT ground reference. (Note that the HI and LO connections conform to a different magnitude convention than do the UL/LL, LT/GT, and IF HI/IF LO statement elements where algebraic magnitude is determining. That is, to put DC of -8 V with respect to -6 V on the UUT, -8 V goes with HI and - 6 V goes with LO).

VIA <pin descriptor> denoting

a) A terminal for a series type connection (e.g., current flow).

b) A port for the transmission of electro-magnetic signals (e.g., waveguide).

c) A port for fluid type signals (e.g., manometric, hydraulic). In all cases, flow to the UUT is arbitrarily designated positive. In case a), above (current flow), the source or sensor should be considered as being connected between the referenced UUT terminal(s) and all other already existing connections to the referenced UUT terminal(s), without causing any temporary interruption to the flow. It should be noted that a previous statement should have provided a path for the return flow.

In case b) (electro-magnetic waves), negative power-ratio quantities may be written (-8 dBm) as a statement characteristic. However, the negative sign should not be construed as being an indication of the direction of power transfer.

SCREEN	<pin descriptor> denoting a terminal on the UUT to which an electro-static shield around the signal conductors is to be connected.
GUARD	<pin descriptor> denoting a terminal on the UUT to which the guard terminal, if any, of the sensor device is to be connected in order to reduce the effect of common mode voltages between the LO side of the UUT signal and the chassis ground of the sensor.
A,B,C	Phase connection <pin descriptor>s. These <pin descriptor>s can be used to describe various phase configurations as shown in the following examples:

a) Single phase: A,B, or C and N

b) Two phase: A,B, and N

c) Three phase delta: A,B,C

d) Three phase wye: A,B,C, and N

N	Neutral <pin descriptor> where used with A, B, and C
X,Y,Z	<pin descriptor> set for three-wire synchro stator connections or to designate the axis of an orthogonal coordinate frame for mechanical rotation.
S1,S2,S3,S4	Four-wire resolver stator connections <pin descriptor> set.
R1,R2,R3,R4	Synchro Resolver rotor connections <pin descriptor> set.
TRUE	<pin descriptor> for the <connection> that carries the "true" signal for a differential digital signal. Used in place of HI.
COMPL	<pin descriptor> for the <connection> that carries the complement signal for a differential digital signal. Used in place of HI. (The connections for a three wire differential digital signal are written using TRUE, COMPL, and LO. For a two wire signal, HI, and LO may be used).
TO	<pin descriptor> for non-electrical signals. It is used when only a single connection is required (irrespective of the flow to or from the UUT). It is also used when two connections are required to define the connection that carries flow to the UUT (see FROM).
FROM	<pin descriptor> for non-electrical signals. It is used when two connections are required and defines the connection that carries flow from the UUT. When the parameter representing flow can assume positive and negative values, a positive value represents flow to the UUT through the TO connection; a negative value represents flow to the UUT through the FROM connection.

14.13.1.2 Reference phases

This section lists all the <reference phase> identifiers that can be employed in the REF subfield.

PHASE–A PHASE–B PHASE–C	Indicates that the reference voltage is between the phase stated and neutral.
PHASE–BC PHASE–CB PHASE–CA PHASE–BA PHASE–AB PHASE–AC	The first letter is the high side of the referenced voltage and the second is the low side.

14.13.1.3 Special connections

COMMON	<connection> to indicate that the lead of the measurement, stimulus, or load device is to be connected to COMMON.
EARTH	<connection> to indicate that the lead of the stimulus, load, or measurement device is to be connected to the low potential reference.
ATMOS	<connection> to indicate that the line is connected to the surrounding atmosphere.

14.13.1.4 Reserved words

The following C/ATLAS words and their use are reserved for the noted purposes:

CNX	First entry in every <conn> field.
REF	Prefix for the UUT connections of the reference signal for phase critical functions.

14.13.1.5 Digital UUT connections

All logic signals, being ordinary analog pulses or DC levels, follow the standard analog conventions for the UUT connections except for the following special definitions for differential signals and for DATA bit sequence and HI/LO pin sequence.

14.13.1.5.1 Connections for SERIAL DATA

Two-wire SERIAL DATA connections are written using the standard HI/LO pin descriptors. Three-wire electrical differential signals use the pin descriptors TRUE, COMPL, and LO. The bits of a SERIAL DATA signal appear at the digital interface in the sequence specified in the TYPE field for a LOGIC DATA signal, that is, either the least-significant or the most-significant bit first.

14.13.1.5.2 Connections for PARALLEL DATA

For two-wire PARALLEL DATA signals, the number of HI connections shall be the same as the number of bits in the parallel digital word. Three possible alternatives exist for the LO connections.

1) One circuit return (LO connection) may be provided for all HI connections.
2) One circuit return may be provided for each HI connection.
3) Individual circuit returns may be grouped within the UUT and one LO connection provided at the UUT interface for each group of HI connections.

Entries in the CNX field are written as follows:

1. For cases 1) and 2), all HI connections are written first, each separated by at least one space, in a sequence that corresponds to the left-to-right bit sequence defined in the value

bit string. (That is, the Most-Significant-Bit (MSB) of the DIGITAL data is transmitted over the pin immediately following the HI keyword.) The LO connection(s) is (are) written next. If individual UUT signal returns are provided, they are written in the same sequence as the HI connections. (That is, corresponding HI and LO connections represent the two sides of one electrical circuit.)

2. For case 3), the HI connections are written in groups such that all members of one group are in adjacent positions. The overall sequence of the HI connections shall also conform to the bit sequence, as for cases 1) and 2). (Note that these two conditions may be mutually exclusive if the data sequence in the value bit stream is not properly planned.) The LO connections for each group are written immediately following the last HI in that group.

Thus, for the three cases above,

a) CNX HI J1-1 J1-2 J1-3 J1-4

    LO J1-5

b) CNX HI J1-1 J1-2 J1-3 J1-4

    LO J1-11 J1-12 J1-13 J1-14

c) CNX HI J1-1 J1-2 J1-3 LO J1-10

    HI J1-4 J1-5 J1-6 LO J1-11

14.13.1.5.3 Electrical differential signals

The input or output for UUT digital DATA signals may require transmission in an electrical differential form. (As illustrated below, this differential form places a digital signal (called TRUE) on one line and its complement (called COMPL) on another line). These connections are written as follows for a PARALLEL DATA signal:

1. Pin descriptors are TRUE, COMPL, and LO.

2. The number of COMPL connections equals the number of TRUE connections. Three options for the number of LO connections exist.

a) One circuit return (LO connection) may be provided for all high side connections.

b) One circuit return may be provided for each high side connection.

c) Individual circuit returns may be grouped within the UUT and one LO connection provided at the UUT interface for each group of high side connections.

3. For cases a) and b), connections are written in the following sequence: all TRUE connections, all COMPL connections, all LO connections.

a) CNX TRUE J1-1 J1-2 J1-3 COMPL J1-11 J1-12 J1-13

    LO J1-10

b) CNX TRUE J1-1 J1-2 J1-3 COMPL J1-11 J1-12 J1-13

    LO J1-21 J1-22 J1-23 J1-31 J1-32 J1-33

4. For case c), the TRUE, COMPL, and LO connections for each group are written together. Thus, for cases a) and c)

a) CNX TRUE J1-1 J1-2 J1-3 COMPL J1-11 J1-12 J1-13

    LO J1-10

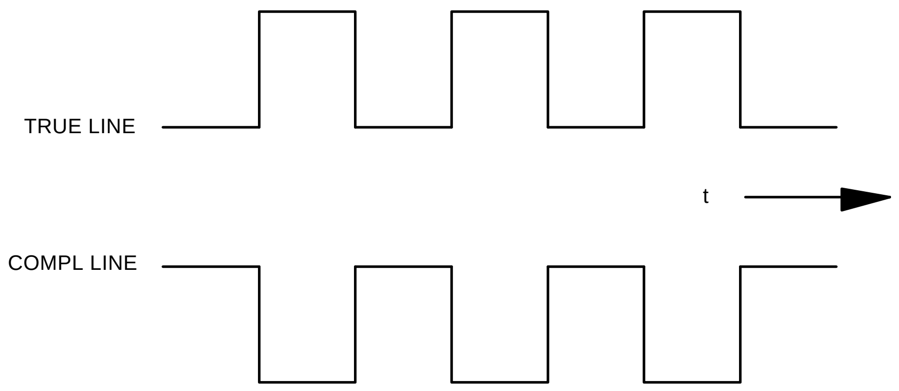
c) CNX TRUE J1-1 J1-2 J1-3 COMPL J1-11 J1-12 J1-13

    LO J1-21

TRUE J1-4 J1-5 J1-6 COMPL J1-14 J1-15 J1-16

LO J1-22

Floating differential signals can be defined by omitting the LO connection. In some cases, differential connections may be treated as two-wire HI/LO circuits. However, the validity of this method shall be determined for each individual application or logic family. The electrical differential signal for a serial binary signal having a VALUE of 101010 with an NRZ pulse code is shown below.



This complement function can be implemented within the test system either by hardware (by providing an inversion of the TRUE line) or by software (by forming the complement bit pattern and placing it on a second TRUE line).

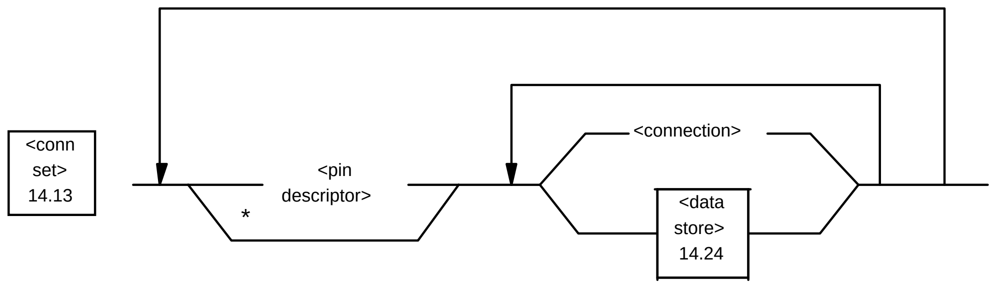
14.13.1.6 Non-electrical connections

The input or output of non-electrical signals to or from the UUT may require the connection of pipes, mechanical linkages, etc. These may be single lines or linkages, or may be pairs of lines, one carrying signal flow to the UUT and one carrying signal flow from the UUT. The pin descriptors for a pair of non-electrical connections are TO (for the connection that normally carries signal flow to the UUT) and FROM (for the connection that normally carries signal flow from the UUT). If multiple connections are written after the pin descriptor, all specified UUT points shall be connected together. For pipes, a manifold is necessary; for mechanical or other linkages, all UUT points shall be tied together.

14.13.2 Formal syntax

Reference conn set

14.13.3 Syntax diagram



NOTE

* This bypass only applies to the nouns such as COMMON, EARTH, and SHORT where a <pin descriptor> is not meaningful.

14.13.4 Rules

The connector designations on the UUT are referred to in C/ATLAS by the term <connection> and are in general assigned arbitrarily by the manufacturer. However, there are certain minor restrictions.

a) A <connection> is represented by an arbitrary sequence of C/ATLAS characters excluding non-printing characters, blanks, apostrophe, currency symbol, comma, left and right parenthesis and equal sign (=). The slash character (/) indicates that the next character, which shall be a <letter>, is lower case (e.g., J/A 2=Ja 2). The slash may not be used for any other purpose in a <connection>.

b) Certain unique character strings cannot be used as connection designations since they have a particular meaning in C/ATLAS. They are listed in 14.13.1.1 through 14.13.1.4.

c) No more than the first 16 characters, excluding any slashes (/) are significant.

d) The term <connection> includes also the special connections listed in 14.13.1.3 (COMMON, EARTH, ATMOS).

14.13.5 Notes and examples

1. For a SOURCE statement

a) For parallel data:

```
STIMULATE, ONE, ON 'DIG' CNX  HI J1-1 LO J1-11,

ZERO, ON 'DIG' CNX  HI J1-2 J1-3

          LO J1-12 J1-13 $
```

the bit string B'100' will appear on the UUT interface. The binary 1 will appear at J1-1 with its circuit return on J1-11.

b) For serial data assume a DEFINE CONFIGURATION structure with data specified as SERIAL_MSB_FIRST and a label 'STIM' previously DECLAREd as a STRING(12) OF BIT containing an octal value of O '4000'.

```
STIMULATE, 'STIM', ON 'DIGA' CNX HI J1-1 LO J1-10 $
```

The most significant bit, which is 1, will appear first; the least significant bit will appear last.

2. The SENSOR statements for the above examples could be, respectively

a) SENSE, (VALUE INTO 'RESP'),

```
ON 'DIGB' CNX  HI J1-1  J1-2  J1-3
LO J1-11 J1-12 J1-13 $
```

'RESP' must have been previously DECLAREd as a variable at least as wide as a STRING(3) OF BIT

b) SENSE, (VALUE INTO 'ANS'),

```
ON 'DIGC' CNX  HI J1-1
LO J1-10 $
```

'ANS' must have been previously DECLAREd as a variable at least as wide as a STRING(12) OF BIT

The bit sequence remains as in example 1, that is, if the UUT output is B'100', the binary 1 is transmitted on lines J1-1 and J1-11 as in example 2a); if the output is O'4000', the binary 1 is transmitted first as in example 2b).

3. For a SENSOR statement, the above examples are valid if the VALUE field is omitted. The bit sequence remains as above, that is, if the UUT output is B'100' with example 1a), the binary 1 is transmitted on lines J1-1 and J1-11; if the output is O'4000' with example 1b), the binary 1 is transmitted first.

14.14 <signal value>

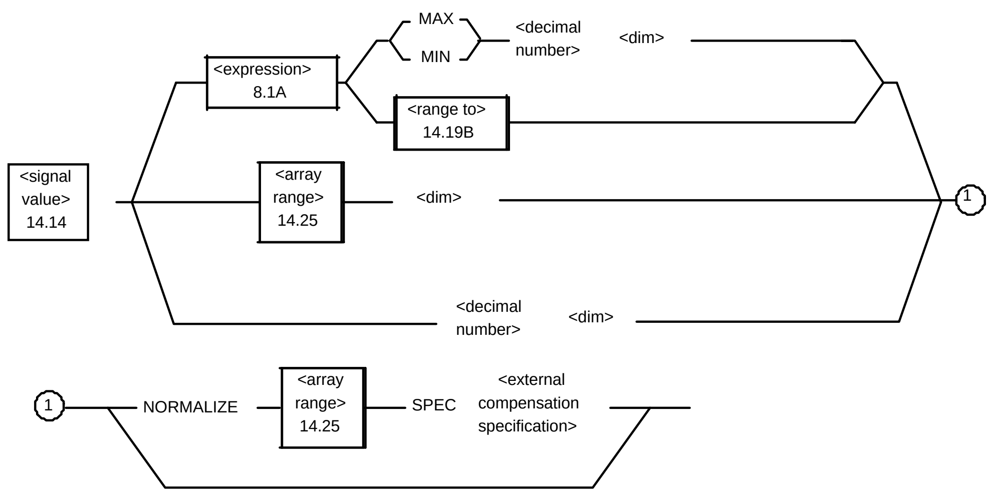
14.14.1 Function

The <signal value> function is to identify where the value of a noun modifier can be found, its permitted range of values, and if and where it is to retrieve its path loss compensation data.

14.14.2 Formal syntax

Reference signal value

14.14.3 Syntax diagram



14.14.4 Rules

If NORMALIZE is present, it indicates the location of the permitted values of path loss compensation data. How that data is to be applied to the <signal value> is specified in the '<external compensation specification>'.

14.15 <real quantity>

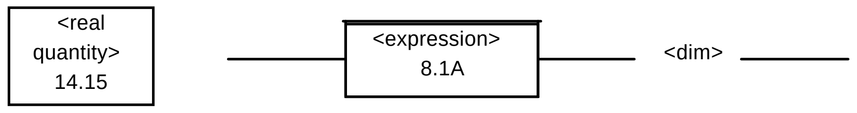
14.15.1 Function

The <real quantity> function is to associate a valid dimension with an <expression>.

14.15.2 Formal syntax

Reference real quantity

14.15.3 Syntax diagram



14.15.4 Rules

The <expression> shall evaluate to a <decimal number>.

14.16 <index>

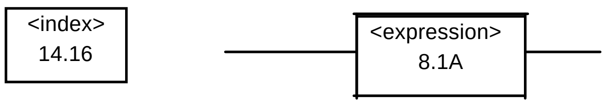
14.16.1 Function

The <index> function is to specify each dimension of an array element.

14.16.2 Formal syntax

Reference index

14.16.3 Syntax diagram



14.16.4 Rules

The type of <expression> shall be the same as the type of the <index> as defined in the associated structure declaration.

14.17 <noun field>

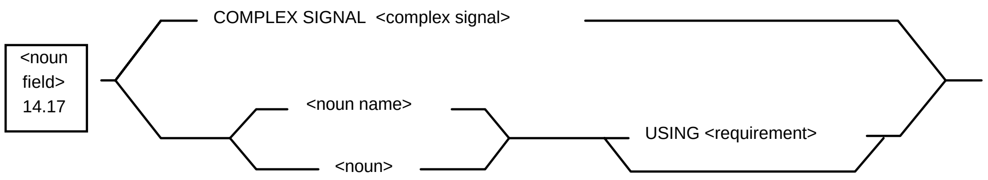
14.17.1 Function

A <noun field> contains a signal type description optionally followed by a virtual resource name to be used.

14.17.2 Formal syntax

Reference noun field

14.17.3 Syntax diagram



14.17.4 Rules

The <requirement> label, when written, shall have been previously established in a <require statement>. The <noun> or <noun name> shall be the same as the <noun> or <noun name> used in the <require statement>.

14.18 <dimensioned number>

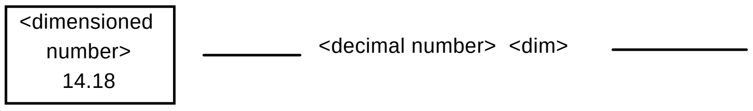
14.18.1 Function

A <dimensioned number> is a <decimal number> or a <long decimal number> followed by a valid dimension.

14.18.2 Formal syntax

Reference dimensioned number

14.18.3 Syntax diagram



14.19 <require control>

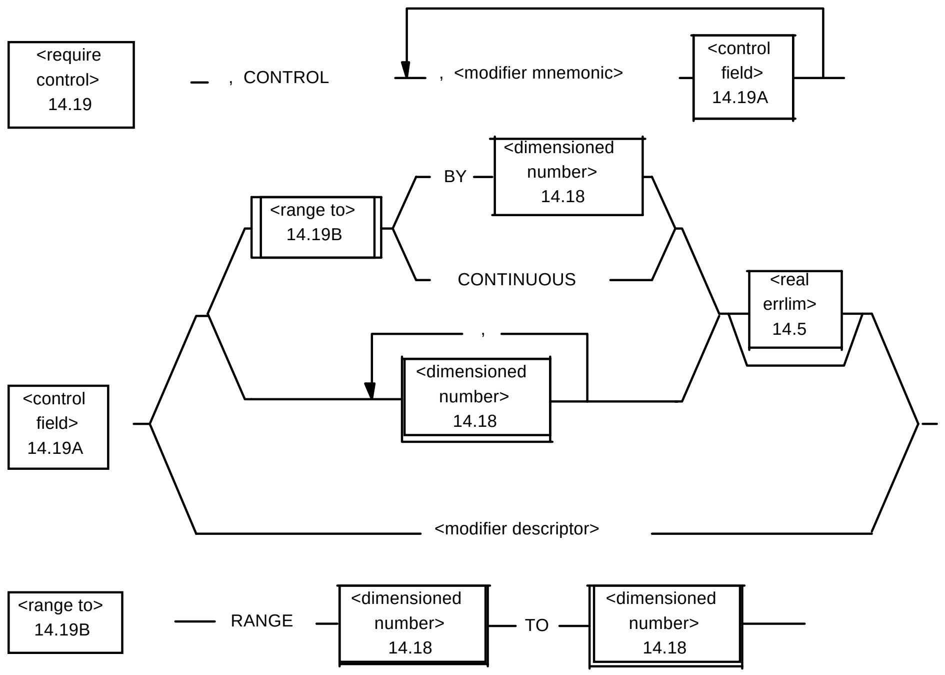
14.19.1 Function

The <require control> field of a <require statement> specifies those characteristics of the test capacity that shall be controlled by the resource to execute the <atlas test program>.

14.19.2 Formal syntax

Reference require control

14.19.3 Syntax diagram



14.20 <require capability>

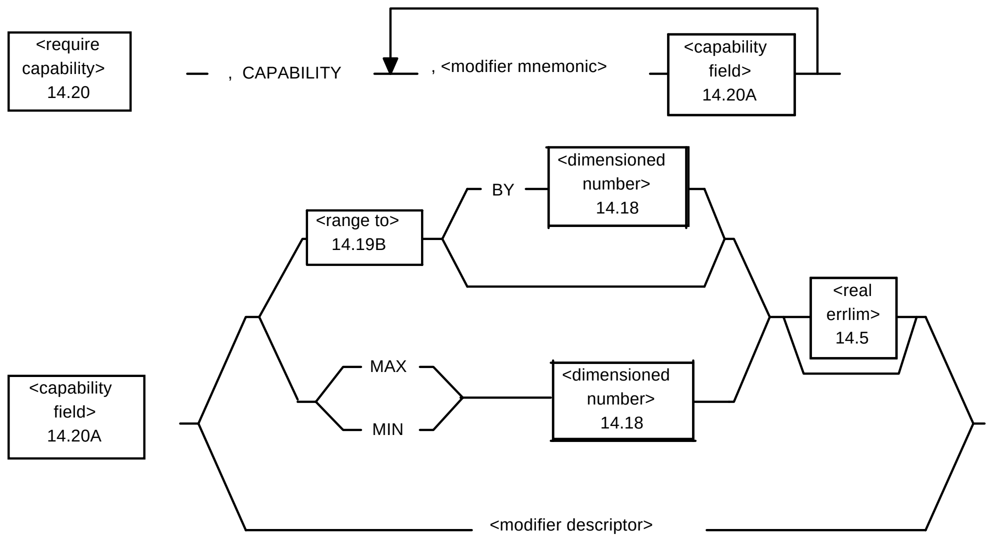
14.20.1 Function

The <require capability> field of a <require statement> specifies those characteristics of the test capacity that are not required to be controlled or limited by the resource to execute the <atlas test program>.

14.20.2 Formal syntax

Reference require capability

14.20.3 Syntax diagram


<require capability> 14.20
, CAPABILITY
, <modifier mnemonic>
<capability field> 14.20A
<capability field> 14.20A
<range to> 14.19B
BY
<dimensioned number> 14.18
MAX
MIN
<dimensioned number> 14.18
<real errlim> 14.5
<modifier descriptor>

14.21 <require limit>

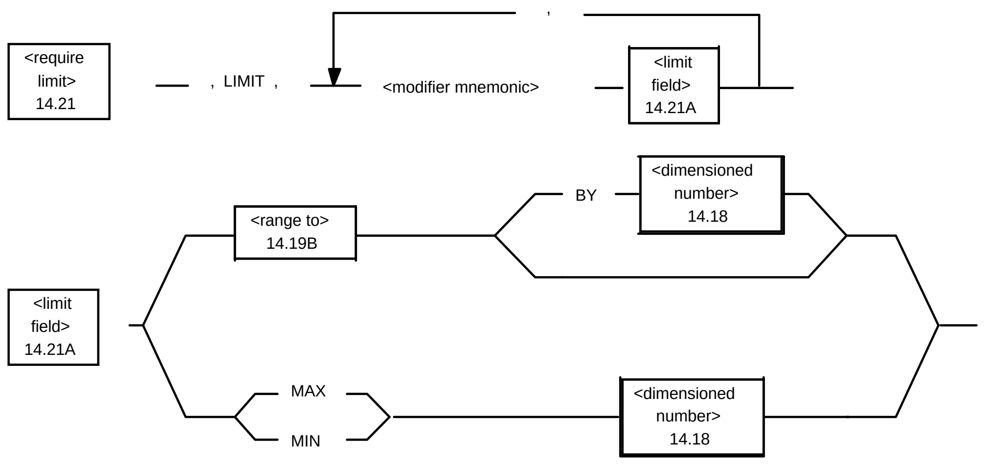
14.21.1 Function

The <require limit> field of a <require statement> specifies those characteristics of a signal oriented test capacity that shall be limited (constrained) to safeguard the UUT from malfunction of either itself or the test resource during the execution of the <atlas test program>.

14.21.2 Formal syntax

Reference require limit

14.21.3 Syntax diagram



14.22 <require cnx>

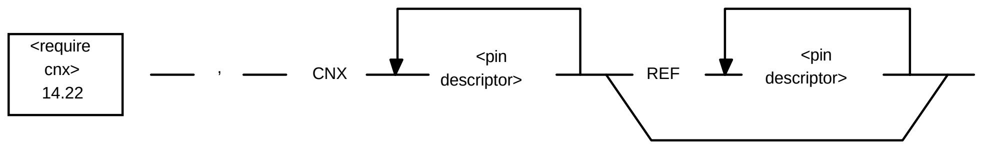
14.22.1 Function

The <require cnx> field lists the <pin descriptor>s associated with the connection points of the virtual resource associated with <requirement>.

14.22.2 Formal syntax

Reference require cnx

14.22.3 Syntax diagram



14.23 <digital quantity>

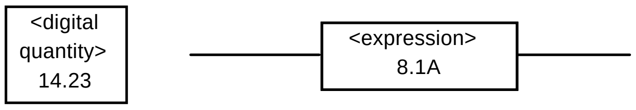
14.23.1 Function

The <digital quantity> function is to identify an expression that shall evaluate to a digital number.

14.23.2 Formal syntax

Reference digital quantity

14.23.3 Syntax diagram



14.24 <data store>

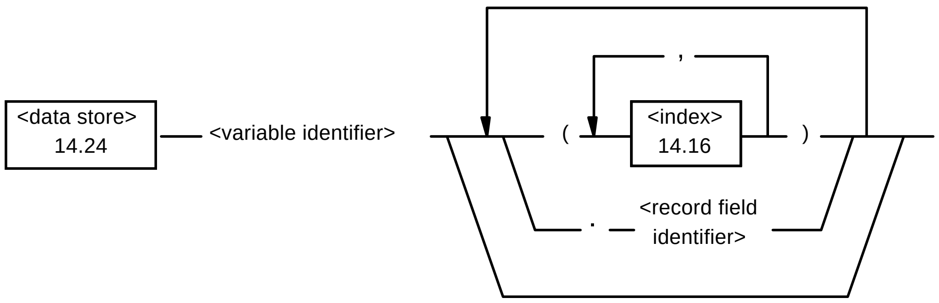
14.24.1 Function

The <data store> function is to specify a simple variable, or an element of an array, or strings, or a record field; or to specify an entire array, string, or record.

14.24.2 Formal syntax

Reference data store

14.24.3 Syntax diagram



14.24.4 Rules

1. If the <variable identifier> label is followed by a parenthesized <index>, then an element of an ARRAY or STRING is being addressed.

2. To reference an ARRAY element, the number of indices used in the reference shall match the number of indices in the declaration of the ARRAY.

3. If the <variable identifier> label is followed by ".", then the first label is the name of a RECORD and the subsequent label is the name of a <record field> within the RECORD.

14.25 <array range>

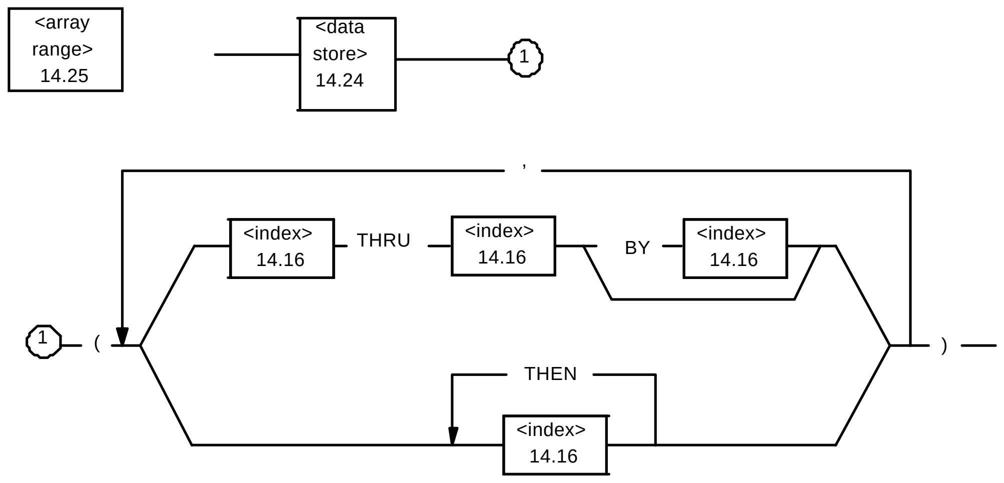
14.25.1 Function

The <array range> function is to specify a range or series of discrete indices of an ARRAY.

14.25.2 Formal syntax

Reference array range

14.25.3 Syntax diagram



14.25.4 Rules

1. The <data store> shall represent a variable or element of a variable of type ARRAY.

2. If the syntactical form <array range> is M THRU N BY J, then a sequence of array elements is specified, where M defines the first array index, N defines the final index, and J defines the increment to be used to compute successive indices.

3. If the BY subfield is omitted, the increment is assumed to be 1.

4. The syntactical form of <array range>, (N1 THEN N2 THEN...Nn) specifies a series of discrete array elements in the given order.

5. If the <data store> refers to a multiple dimensioned array, the index specifications for each of the multiple dimensions are read from left to right and are separated by commas. Array elements are so identified in row major order, such that the last written index or indices vary most rapidly and the first written index or indices varies more slowly in the conventional manner.

14.25.5 Notes and examples

The following is an example of the use of <array range> for a three dimensional array. The array was declared in the preamble with the following statement:

```
DECLARE, VARIABLE,
'VALUE MATRIX' IS ARRAY
(1 THRU 10, 1 THRU 10, 1 THRU 10) OF DECIMAL $
```

The following reference to 'VALUE MATRIX' has been taken out of the context of a complete C/ATLAS statement:

```
... 'VALUE MATRIX' (1 THEN 3, 2 THRU 4, 1 THRU 3 BY 2) ...
```

The following array elements would be accessed by this statement, in the order specified:

```
(1,2,1) (1,2,3) (1,3,1) (1,3,3) (1,4,1) (1,4,3) (3,2,1) ...
```

14.26 <parameter>

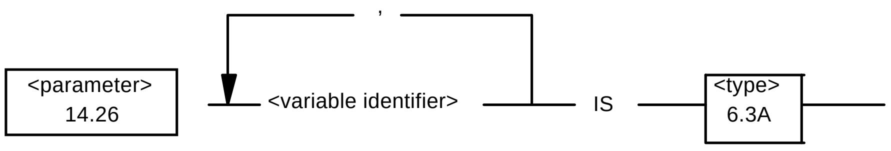
14.26.1 Function

A <parameter> is a <variable identifier> local to the body of a <define procedure structure> used for communication between the calling level and the PROCEDURE.

14.26.2 Formal syntax

Reference parameter

14.26.3 Syntax diagram



14.26.4 Rules

When a PROCEDURE is performed or a <pre-defined function> is invoked, the arguments associated with the invocation shall be assignment compatible with the <parameter> type.

14.27 <gate field>

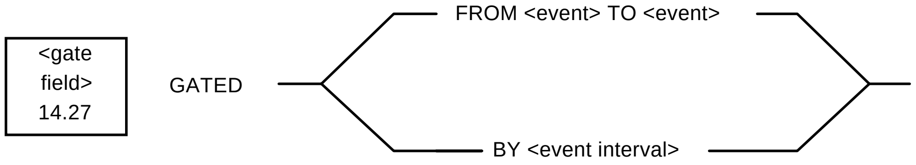
14.27.1 Function

The <gate field> function is to state the time-interval during which a signal is applied or sensed.

14.27.2 Formal syntax

Reference gate field

14.27.3 Syntax diagram



14.27.4 Rules

1. A <gate field> specifies a time interval bounded either by two different events during which a signal is applied or sensed, or by an <event interval>.

2. Stimulus signals assume the quiescent state outside the gate interval.

14.27.5 Notes and examples

The <gate field> permits a particular portion of a waveform to be selected as the subject for the measure statement containing the <gate field>. For instance if the signal is a pulse train and it is required to measure the width of the third pulse, the inclusion of the following <gate field> would permit this to be achieved:

```
  . . .
IDENTIFY, EVENT 'TRANSITION'
  AS (VOLTAGE_INST),PULSED DC TRAIN, EQ 3.0V,
  VOLTAGE_INST RANGE 0V TO 5V,
  CNX HI J1 LO J2 $
IDENTIFY, EVENT 'A' AS 4 OCCURRENCES
  OF 'TRANSITION' $
IDENTIFY, EVENT 'B' AS 6 OCCURRENCES
  OF 'TRANSITION' $
IDENTIFY, EVENT 'C'
  AS 10.0 USEC AFTER 'A' $
IDENTIFY, EVENT 'D'
  AS 10.0 USEC AFTER 'B' $
  . . .
MEASURE, (PULSE_WIDTH INTO 'PWA'), PULSED DC,
  PULSE_WIDTH MAX 2.0 MSEC,
  GATED FROM 'C' TO 'D',
  CNX HI J1 LO J2 $
  . . .
```

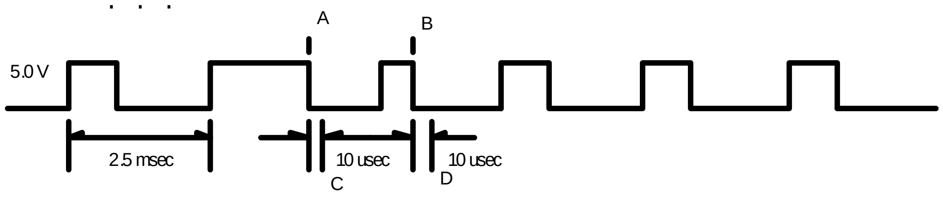


Figure 14-1. Example 6 pulse train

14.28 <stim rate>

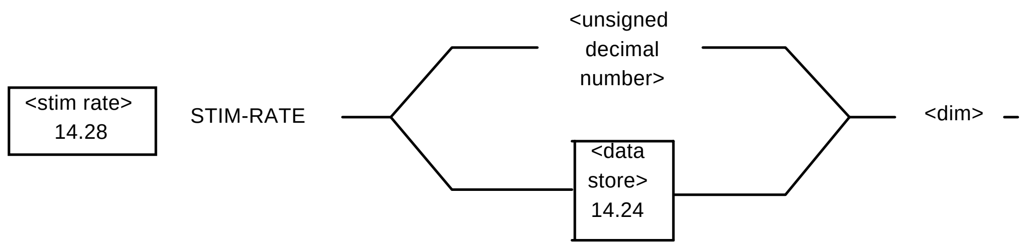
14.28.1 Function

The <stim rate> function is to specify the rate at which the critical actions of successive STIMULATE statements shall be executed.

14.28.2 Formal syntax

Reference stim rate

14.28.3 Syntax diagram



14.29 <sense rate>

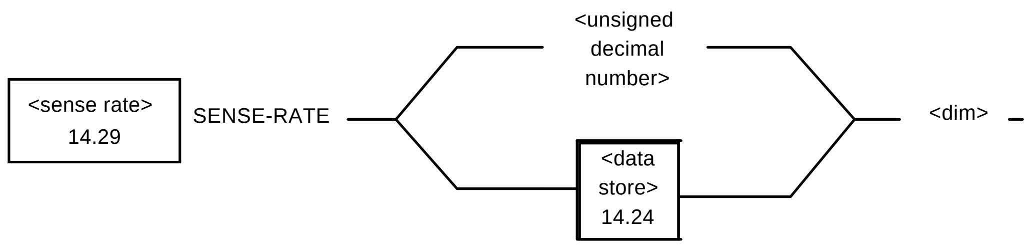
14.29.1 Function

The <sense rate> function is to specify the rate at which the critical actions of successive SENSE or PROVE statements shall be executed.

14.29.2 Formal syntax

Reference sense rate

14.29.3 Syntax diagram



14.30 <sense delay>

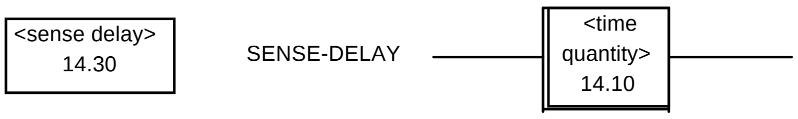
14.30.1 Function

To specify the precise time delay from the critical action of a STIMULATE statement at which the critical action of a following SENSE or PROVE statement shall occur.

14.30.2 Formal syntax

Reference sense delay

14.30.3 Syntax diagram



14.31 <stim event>

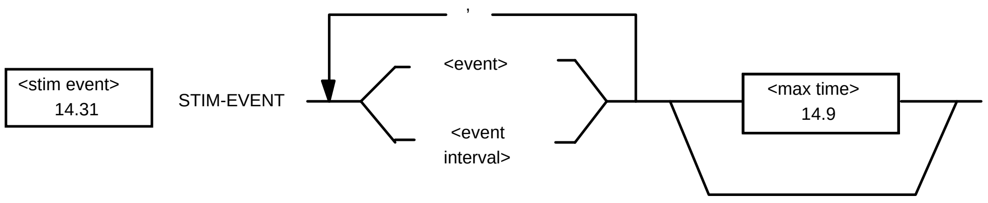
14.31.1 Function

The <stim event> function is to establish the precise time based upon successive event or event interval occurrences at which the critical actions of successive STIMULATE statements shall be executed.

14.31.2 Formal syntax

Reference stim event

14.31.3 Syntax diagram



14.31.4 Rules

1. If a single event is specified, the referencing <stimulate statement>s are activated at every occurrence of the event.

2. If a single <event interval> is specified, the referencing <stimulate statement>s are activated at the beginning of every occurrence of the <event interval> (i.e., at the occurrence of the <event> which generates the start of the <event interval>).

3. If a list of <event>s/<event interval>s is specified, the referencing <stimulate statement>s are activated as follows:

Starting with the occurrence of the first <event> as specified (or the first <event> that generated the <event interval>) followed by the next occurrence of the second specified element.

4. When an <event interval> is used, the corresponding <digital source> shall have LEVEL_LOGIC_QUIES defined. At the end of the <event interval> the corresponding <digital source> goes to LEVEL_LOGIC_QUIES.

5. An active <event interval> will not inhibit the action of a following <event>.

14.32 <sense event>

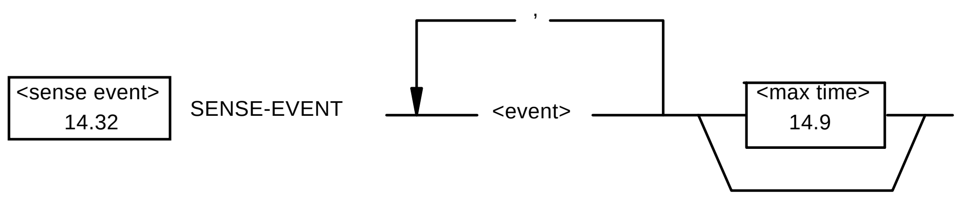
14.32.1 Function

The <sense event> function is to establish the precise time based upon successive event occurrence at which the critical action of successive SENSE or PROVE statements shall be executed.

14.32.2 Formal syntax

Reference sense event

14.32.3 Syntax diagram



14.32.4 Rules

The <sense event> generates a sequence of events.

a) If a single event is specified the sequence of events is every occurrence of this event.

b) If a list of events is specified the sequence of events is as follows:

Starting with the occurrence of the first specified event in the list, followed by the next occurrence of the second specified element, etc.

14.33 <when field>

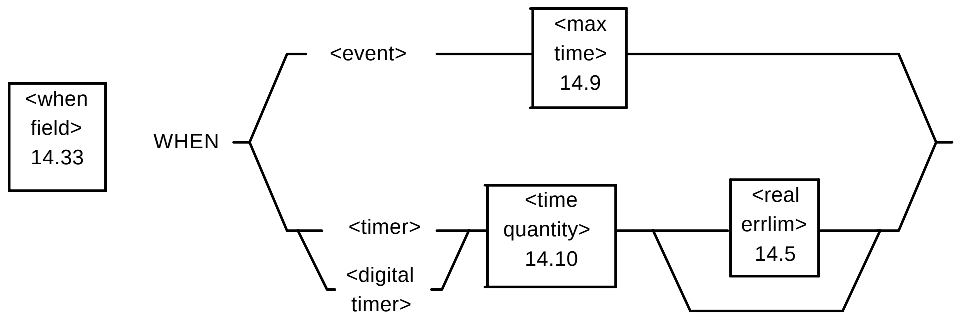
14.33.1 Function

The <when field> function is to specify the precise instant at which an action or set of actions shall be executed.

14.33.2 Formal syntax

Reference when field

14.33.3 Syntax diagram



14.33.4 Rules

1. WHEN <event> causes the instant of execution to be the first new occurrence of the referenced event.

2. The <digital timer> attribute can only be used within a <do timed digital body>. Additionally, the <digital timer> in the <when field> shall be identical to one of the <do timed digital statement> <digital timer>s.

14.34 <digital source characteristics>

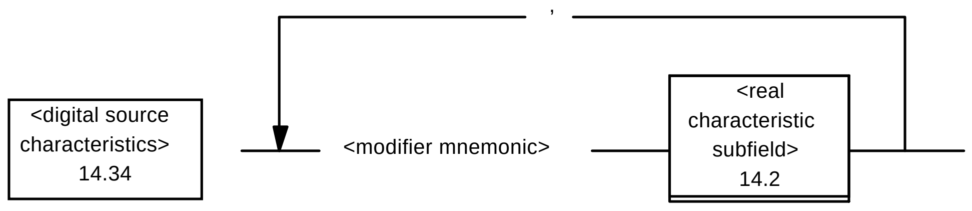
14.34.1 Function

A subfield of a <define digital source statement> specifying the physical characteristics and corresponding to driven logic symbols.

14.34.2 Formal syntax

Reference digital source characteristics

14.34.3 Syntax diagram



14.34.4 Rules

1. The <modifier mnemonic> chosen shall be applicable to the <noun> specified in the associated <digital source field>.

2. When used following LOGIC_HIZ to specify the high impedance condition of the test

system, the direction of leakage current flow is specified as follows:

a) Current flowing from the test system as a positive current

b) Current flowing into the test system as a negative current

14.35 <digital sensor characteristics>

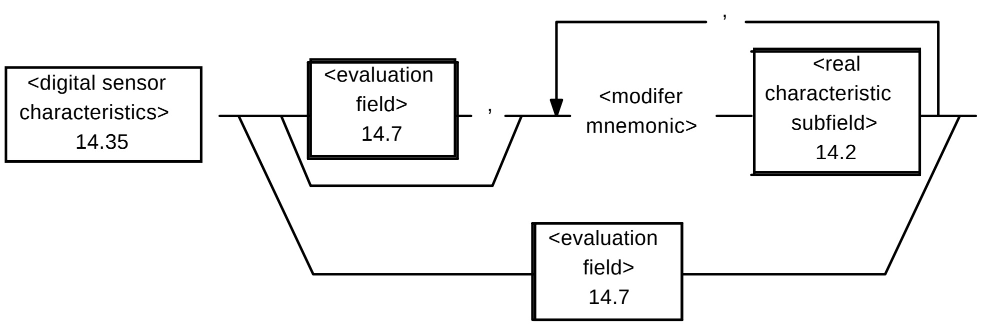
14.35.1 Function

This function is a subfield of <define digital sensor statement> specifying the physical characteristics corresponding to sensed logic symbols.

14.35.2 Formal syntax

Reference digital sensor characteristics

14.35.3 Syntax diagram


<digital sensor characteristics> 14.35
<evaluation field> 14.7
,
,
<modifer mnemonic>
<real characteristic subfield> 14.2
<evaluation field> 14.7

14.35.4 Rules

The <modifier mnemonic> chosen shall be applicable to the <noun> specified in the associated <digital sensor statement>.

14.36 <on field>

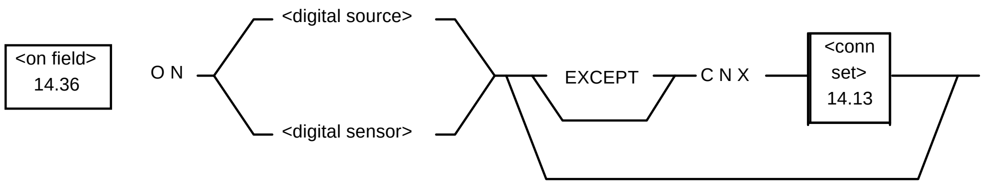
14.36.1 Function

The <on field> is used in STIMULATE, SENSE and PROVE statements (or their compounds) to define the UUT connections these statements are to operate on. The <on field> allows digital testing to be performed using all, or a subset, of the connections specified in a defined DIGITAL CONFIGURATION.

14.36.2 Formal syntax

Reference on field

14.36.3 Syntax diagram



14.36.4 Rules

1. The <connection>s specified or implied (i.e., by using EXCEPT, or by using variables in <conn set> shall be from those DEFINEd in <digital source> or <digital sensor> statements of an enabled <digital configuration>.

2. If a CNX field is not included, digital testing is performed on all <connection>s associated with the specified <digital source>, <digital sensor>, or <digital configuration>.

3. If a CNX field is included, digital testing is performed only on the specified <connection>s. <connection>s not specified will not be changed by STIMULATE statements and will not be sensed by SENSE statements.

4. If EXCEPT is used, digital testing is performed on all <connection>s associated with the specified <digital source>, <digital sensor>, or <digital configuration> except for those given in the following CNX field.

14.37 <exchange expression>

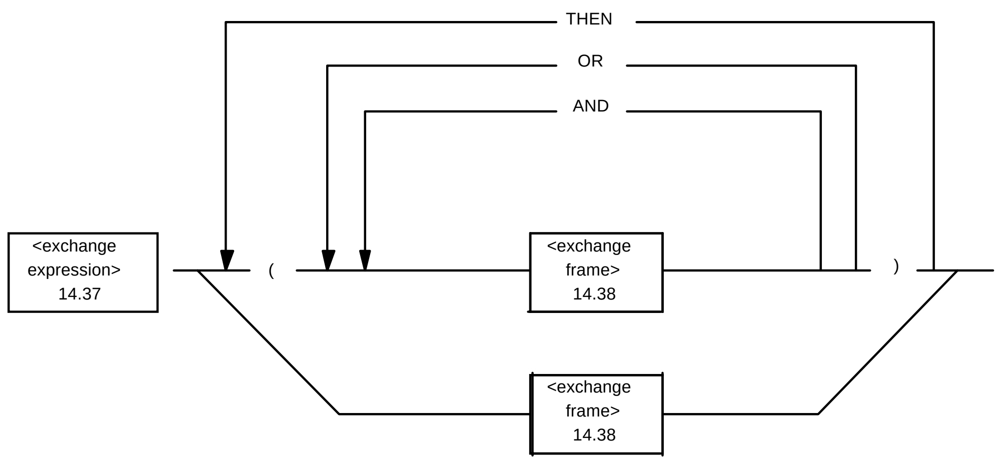
14.37.1 Function

An <exchange expression> is used to describe the inter-relationships between <exchange frame>s.

14.37.2 Formal syntax

Reference exchange expression

14.37.3 Syntax diagram



14.37.4 Rules

1. For multiple <exchange frame>s, the THEN connective specifies that the <exchange frame> before the THEN will be terminated before starting the next <exchange frame>, the AND connective specifies that all <exchange frame>s so connected will be acted upon, the OR connective specifies that only one of the <exchange frame>s will be acted on.

2. The AND relationship between <exchange frame>s indicates that, in the case of bus contention/conflict, priority is invoked (whether implicit or explicit). Priority can be explicitly expressed by exercising the PRIORITY field in the exchange frame definition. Lower priority numbers have the higher priority. The highest possible priority is PRIORITY 0. That is, the <exchange frame> with the highest priority is serviced; the servicing of any lower priority <exchange frame> is delayed.

3. The OR relationship indicates, in the case of bus contention/conflict, that priority is also invoked; however, the lower priority <exchange frame>s are not serviced for that particular interval of time.

4. The THEN relationship is used to separate sequential <exchange frame>s.

5. Parentheses are to be used to group or enclose <exchange frame>s linked by the AND, and OR, relationships.

6. AND and OR do not imply normal boolean operators. They may not be mixed between the parentheses which separate sequential <exchange frame>s. e.g., (A AND B AND C) THEN (D OR E OR F) is valid, but (A AND B OR C) is not.

14.37.5 Notes and examples

See 13.2.

14.38 <exchange frame>

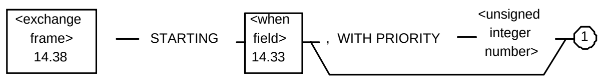
14.38.1 Function

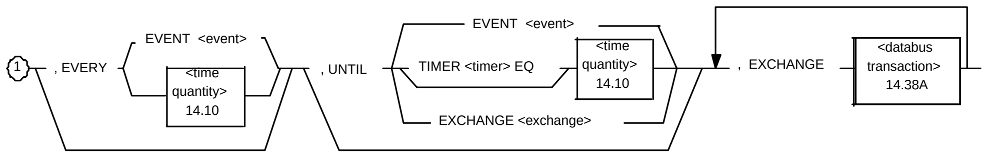
An <exchange expression> uses an <exchange frame> to describe a sequence of exchanges on a databus.

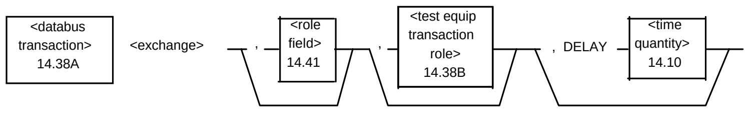
14.38.2 Formal syntax

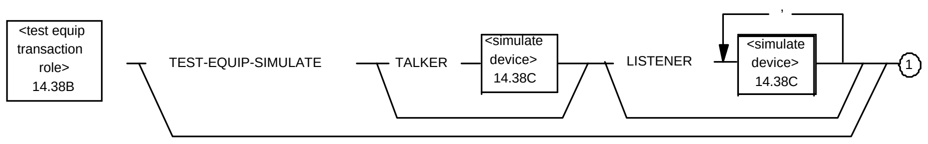
Reference exchange frame

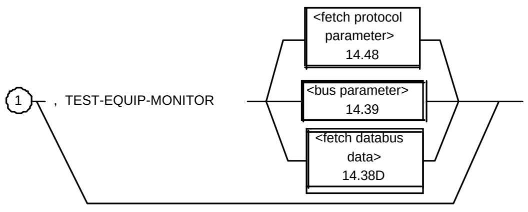
14.38.3 Syntax diagram

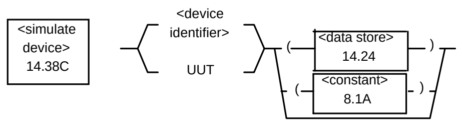


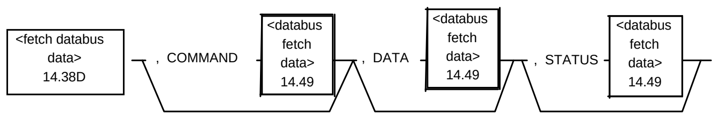



<databus transaction> 14.38A
<exchange>
,
<role field> 14.41
,
<test equip transaction role> 14.38B
, DELAY
<time quantity> 14.10








<fetch databus data> 14.38D
, COMMAND
<databus fetch data> 14.49
, DATA
<databus fetch data> 14.49
, STATUS
<databus fetch data> 14.49

14.38.4 Rules

1. The <exchange frame> describes, in effect, an independent "operation" within a <do exchange statement>. Each <exchange frame> may contain a starting condition, a priority level, an EVERY condition, and an UNTIL condition. Each <exchange frame> contains one or more <exchange>s linked by an optional time delay. The <exchange frame> can optionally specify the test equipment role during the <exchange frame>. It also specifies if the test equipment simulates some devices or monitors some data on a specific exchange.

2. The <exchange frame> includes a PRIORITY field that determines the numerical priority accorded to this <exchange frame> structure compared to the other <exchange frame> structures in the same <do exchange statement>. Each <exchange frame> within a <do exchange statement> shall have a different priority. Lower priority numbers have the higher priority. In the event of a conflict, the highest priority has precedence. The highest possible priority is PRIORITY zero. The PRIORITY field may be omitted only if the <do exchange statement> includes neither AND or OR connectives between <exchange frame>s.

3. The UNTIL <exchange> is to be used when a specific exchange is required to determine the end of an <exchange frame>.

4. The entry in the <role field> represents the function or role that the test equipment is to assume for the single <exchange>. It supersedes any role defined by the <do exchange statement> as described in 13.2 for this single <exchange>. If the <do exchange statement> contains no <role field>, each <exchange frame> shall specify the role of the test equipment.

5. The TEST_EQUIP_SIMULATE labels shall be valid talker or listener <device identifier name>s for the specified <exchange> as defined by the associated <define exchange statement>. The type of the <expression>s shall be the same as the <type> of the <data store> or <constant> in the TALKER or LISTENER field of the <define exchange statement>. Any COMMAND, DATA, or STATUS information that is to be transmitted by the test equipment will be extracted from the corresponding field of the <define exchange statement>.

6. The TEST_EQUIP_MONITOR field is used to identify bus data that is to be monitored by the test equipment. For COMMAND, DATA, and STATUS information, it specifies the <data store>s defining which bus data items are to be monitored. The <type>s of the <data store>s shall be either the same <type>s as in the COMMAND, DATA, or STATUS field of the <define exchange statement>.

7. If the WAIT option is specified in the <do exchange statement>, the specified variables are updated during the execution of the <do exchange statement>. If the destination is an <array range>, successive array elements receive successive values of the parameter up to the limit defined by the <array range> indices. Such a list may have an associated <data store> introduced by COUNT_INTO. If it does, the start of the execution of the <do exchange statement> sets the <data store> to zero. Each time a new value of that monitored parameter is stored in the array, the <data store> is incremented.

8. If the PROCEED option is specified, the monitored data is retained by the test equipment, and the specified variables are NOT updated. An <array range> in this case serves only to identify the number of consecutive bus words that are to be monitored by the

test equipment. The COUNT_INTO branch is not used. The monitored data shall be extracted using a subsequent <fetch exchange configuration statement>, and will be returned to the variables specified in the <fetch exchange configuration statement>.

9. If the TEST_EQUIP_MONITOR field is used to monitor <bus parameter>s or <protocol parameter>s, then each of them shall be established in the <establish protocol statement> with the FETCHABLE and EXCHANGE attributes.

10. The <file> path in <databus fetch data> can only be used following the keyword DATA.

11. DELAY specifies the time from the initiation of the <exchange> preceding the DELAY to the initiation of the <exchange> following the DELAY. The user shall ensure that adequate time exists to execute the exchange. The DELAY branch shall not occur following the final <exchange>.

14.39 <bus parameter>

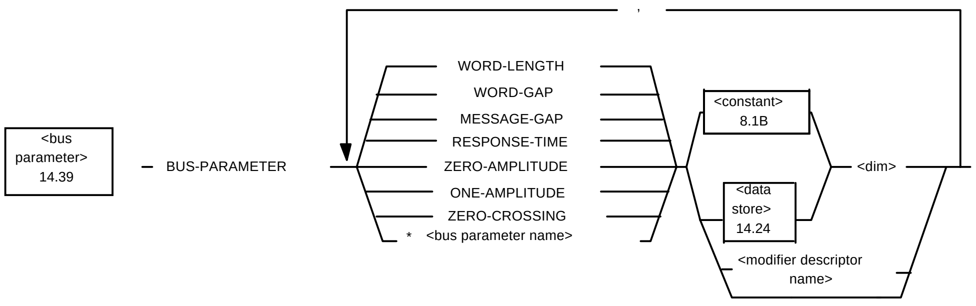
14.39.1 Function

This field explicitly specifies any bus parameter, including deviations from the referenced standard databus, that are required to be employed in the tests.

14.39.2 Formal syntax

Reference bus parameter

14.39.3 Syntax diagram



NOTE

* The name shall be one that has been introduced by an <extend statement> of the associated type.

14.39.4 Rules

1. A standard bus parameter is stated only when several have been set to deviations from the standard and a subset of these are to resume their standard values.

2. All entries that are used in this field shall be defined in the <establish bus parameter>

section of the <establish protocol statement>. The <constant> branch may be used only when a value for the bus parameter is to be set, (i.e., in a <define exchange statement>, an <enable exchange configuration statement>, or an <update exchange configuration statement>). The <data store> branch can be used in the same way as a <constant> above, but in addition, it can be used as the destination for the <bus parameter> value that can be read from the test equipment in a <do exchange statement> or <fetch exchange configuration statement>. The type of the <constant> or <data store> shall be the same <type> as the <bus parameter range> in the <establish protocol statement>, and its value, if specified, must be within the specified range.

3. The following standard entries with the listed general meaning may appear in the field. In accordance with rule 2, each one used shall be precisely defined by the <external bus specification>s.

a) WORD_LENGTH The length of a digital word transferred via the referenced bus.

b) WORD_GAP The length of time between the transmission of successive words in one message; this may be specified as bit times or a time quantity.

c) MESSAGE_GAP The length of time between the last bit transmitted in a complete message to the first bit transmitted in the next successive message to be transmitted on the same bus.

d) RESPONSE_TIME The length of time between the last bit transmitted in a message word that requests a receiver's response and the first bit transmitted of the response word(s) from that receiver. Both the request word(s) and the response word(s) may be defined as elements of a single message format.

e) ZERO_AMPLITUDE, ONE_AMPLITUDE The signal levels, respectively, of the logic level zero component and logic level one component of the bit coding for this bus.

f) ZERO_CROSSING The point in time, expressed as a percentage of the bit duration, at which the bit transition crosses the zero potential for logic trans-coding of data on this bus.

4. If none of the standard entries is suitable to define a bus failure mode that is required in the test, a <bus parameter name> may be included. The <bus parameter name> shall be defined in an <extend statement> and referenced in the corresponding field of the <establish protocol statement>.

14.40 <set protocol parameter>

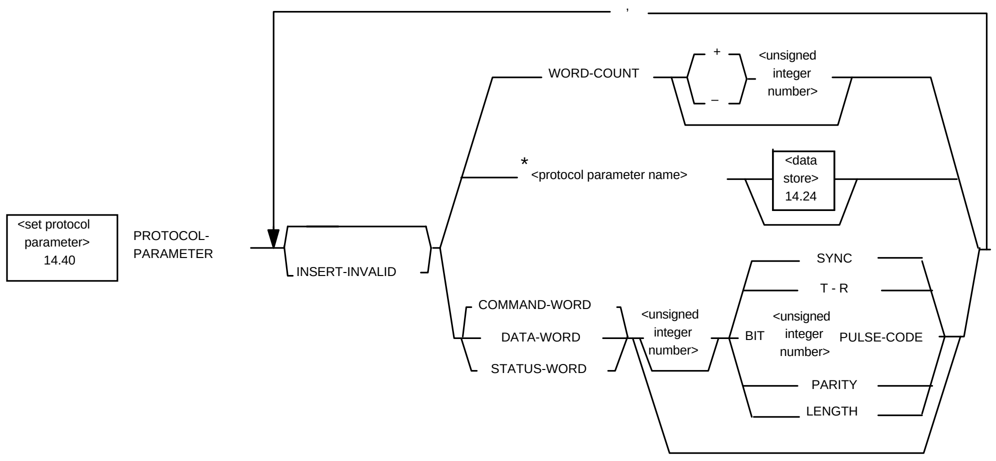
14.40.1 Function

This field specifies the bus deviations from the referenced standard databus that are required to be employed in the tests.

14.40.2 Formal syntax

Reference set protocol parameter

14.40.3 Syntax diagram



<set protocol parameter> 14.40
PROTOCOL-PARAMETER
INSERT-INVALID
WORD-COUNT
+
–
<unsigned integer number>
*
<protocol parameter name>
<data store> 14.24
COMMAND-WORD
DATA-WORD
STATUS-WORD
<unsigned integer number>
SYNC
T - R
BIT
<unsigned integer number>
PULSE-CODE
PARITY
LENGTH

NOTE

* The name shall be one that has been introduced by an <extend statement> of the associated type.

14.40.4 Rules

1. The INSERT_INVALID option on this field enables a <protocol parameter> to be changed to the non-standard value which may be an invalid value outside the range of values permitted by the databus specification only if the FAULT -TEST option was used in the corresponding <establish protocol statement>. The precise specification of the invalid parameter and its values shall be defined in the external bus specifications that are referenced in the associated <establish protocol statement>.

2. The bypass of the INSERT -INVALID option in the field enables a <protocol parameter> to be changed to its standard value. The exact specification of the parameter and its standard value shall be defined in the <external bus specification>, which are referenced in the associated <establish protocol statement>. In this case, no value can be specified for the parameter and the bypass around the associated value field shall be taken (WORD_COUNT and <protocol parameter name> branches).

3. The following standard entries with the listed general meaning may appear in the field. Each one used shall be precisely defined by the <external bus specification>s referenced in the associated <establish protocol statement>.

a) WORD_COUNT The legal word count can be changed in a message by a specified positive or negative increment.

b) COMMAND_WORD, STATUS_WORD, DATA _WORD The <unsigned integer number> following the keyword shall be 1 (one) except where the corresponding COMMAND, DATA, and STATUS <type> definition in the corresponding <exchange model> is a RECORD, in which case the number identifies the corresponding <record element> of the RECORD. In each case, the element can be rendered invalid by including faulty transmit/receive coding, faulty sync coding, invalid word parity, invalid word length or an invalid pulse code on a particular bit in the element. These are indicated respectively by T -R, SYNC, PARITY, LENGTH, and BIT <unsigned integer number> PULSE_CODE. The value of the <unsigned integer number> between BIT and PULSE_CODE shall be less than or equal to the string length of the corresponding record element if the element is of STRING <type>, otherwise it shall be defined in the <external bus specification> documents referenced in the associated <establish protocol statement>.

4. If none of the standard entries is suitable to define a protocol failure mode on the bus that is required in a test, a <protocol parameter name> may be included. The <protocol parameter name> shall be defined in an <extend statement> and referenced in the corresponding field of the <establish protocol statement>.

5. The <protocol parameter name> shall have been established as valid for the databus in the associated <establish protocol statement>. For it to occur in an <update exchange configuration statement>, it shall have been defined as UPDATABLE for that category of statement. It is the only form of protocol parameter than can occur in an <update exchange configuration statement> or an <enable exchange configuration statement>.

14.41 <role field>

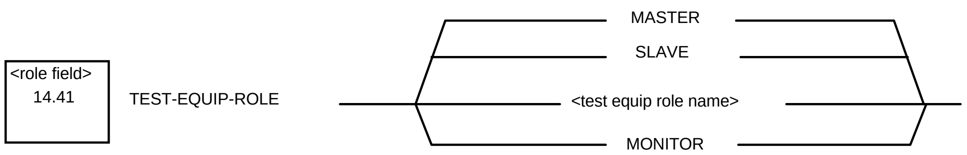
14.41.1 Function

This field specifies the role that the test equipment is to take during an exchange on a databus.

14.41.2 Formal syntax

Reference role field

14.41.3 Syntax diagram



14.41.4 Rules

1. The entry in the TEST_EQUIP_ROLE field shall be defined by the specifications referenced by the associated <establish protocol statement>. It represents the function or role that the test equipment is to assume.

2. The general meaning of the standardized entries for the TEST_EQUIP_ROLE field are as follows:

MASTER	The test equipment is in control of the databus and determines the activity on it.
SLAVE	The UUT controls the databus and the test equipment acts as instructed by the UUT.

MONITOR	The test equipment generates none of the traffic on the databus. It plays a passive role in which it monitors the traffic and selects particular databus transactions for saving and future analysis.

14.42 <command field>

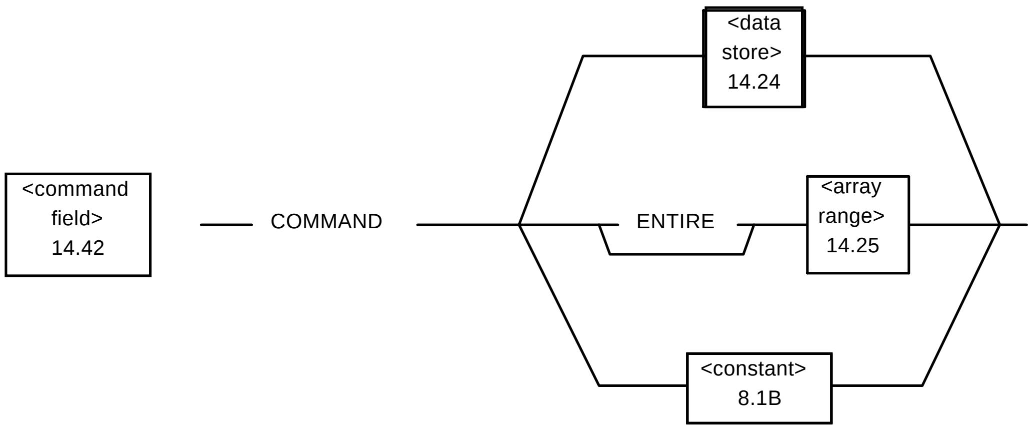
14.42.1 Function

This field specifies the command words to be transmitted on the databus when the exchange is made.

14.42.2 Formal syntax

Reference command field

14.42.3 Syntax diagram



14.42.4 Rules

1. A COMMAND field in a <define exchange statement> or an <update exchange statement> may only have an explicit definition of the command data. Consequently, the <data store> and <array range> fields are evaluated at run time when the <do exchange statement> and <update exchange statement> are started.

2. The type of the <data store>, <constant>, and the basic type of the <array range> in all cases shall be identical to the type defined in the relevant <exchange model> from the <establish protocol statement>.

3. If the <array range> branch is used with the ENTIRE word, this indicates that all the specified words in the <array range> are sent every time the exchange is invoked. Otherwise, consecutive transmissions of the <exchange> on the bus will take their values from successive elements of the specified <array range> on an auto incrementing basis.

4. The <record field identifier> path in the <data store> field is allowed in the <update exchange statement>, but not in the <define exchange statement>. This indicates that only the specified 'record field' will be updated.

14.43 <data field>

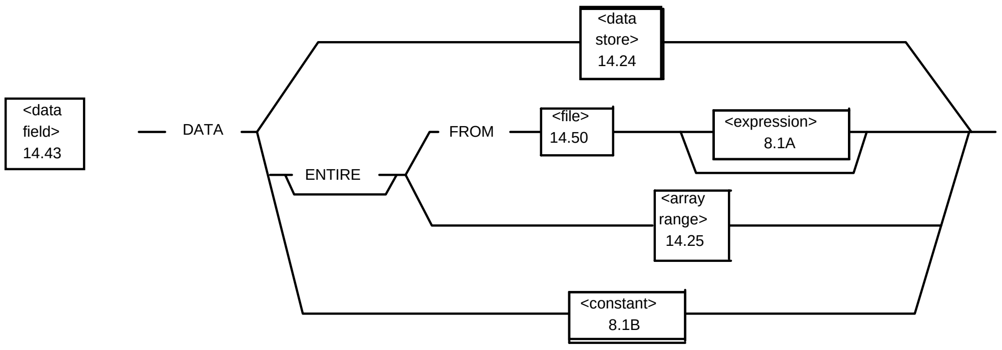
14.43.1 Function

This field specifies the data words to be transmitted on the databus when the exchange is made.

14.43.2 Formal syntax

Reference data field

14.43.3 Syntax diagram



14.43.4 Rules

1. A DATA field in a <define exchange statement> or <update exchange statement> may have only an explicit definition of the data. Consequently, the <data store> and <array range> fields are evaluated at run time when the <do exchange statement> and <update exchange statement> are started.

2. The type of the <data store>, <constant>, and the basic type of the <array range> shall be identical to the type defined in the relevant <exchange model> from the <establish protocol statement>.

3. If the <array range> or <file> branch is used with the ENTIRE word, this indicates that all the specified words in the <array range> are sent every time the exchange is invoked. Otherwise, consecutive transmissions of the <exchange> on the bus will take their values from successive elements of the specified <array range> or <file> on an auto incrementing basis. For rules regarding the use of the optional file position <expression>, see 9.2.5.4.

4. The <record field identifier> path in the <data store> field is allowed in the <update exchange statement> but not in the <define exchange statement>. This indicates that only the specified 'record field' will be updated.

5. The <record field identifier> syntax is allowed in the <update exchange statement> but not in the <define exchange statement>. This indicates that only the specified 'record field' will be updated.

14.44 <status field>

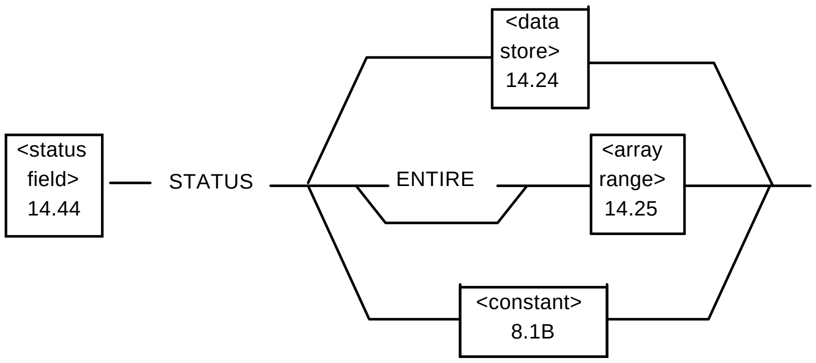
14.44.1 Function

This field specifies the status words to be transmitted on the databus when the exchange is made.

14.44.2 Formal syntax

Reference status field

14.44.3 Syntax diagram



14.44.4 Rules

1. A STATUS field in a <define exchange statement> or <update exchange statement> may have only an explicit definition of the data. Consequently, the <data store> and <array range> fields are evaluated at run time when the <do exchange statement> and <update exchange statement> are started.

2. The type of the <data store>, <constant>, and the basic type of the <array range> shall be identical to the type defined in the relevant <exchange model> from the <establish protocol statement>.

3. If the <array range> branch is used with the ENTIRE word, this indicates that all the specified words in the <array range> are sent every time the exchange is invoked. Otherwise, consecutive transmissions of the <exchange> on the bus will take their values from successive elements of the specified <array range> on an auto incrementing basis.

4. The <record field identifier> path in the <data store> field is allowed in the <update exchange statement> but not in the <define exchange statement>. This indicates that only the specified 'record field' will be updated.

14.45 <mark descriptor subfield>

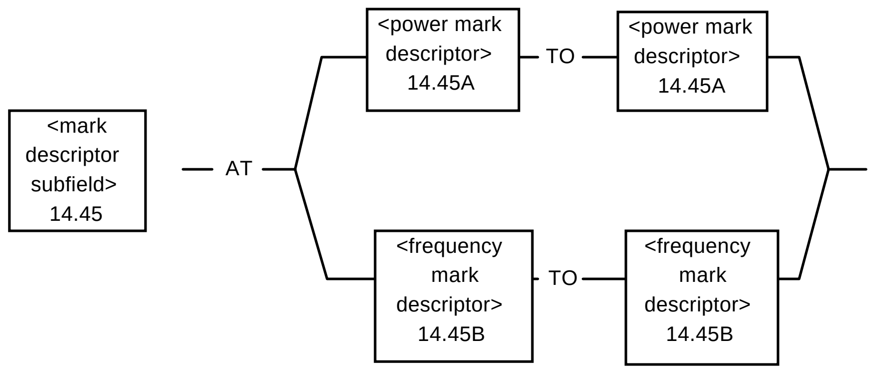
14.45.1 Function

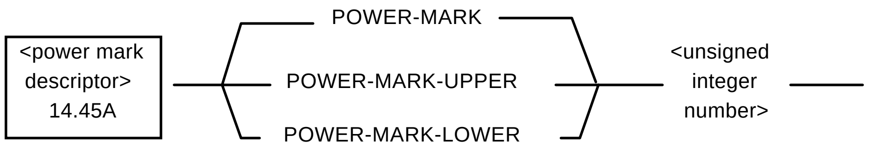
The POWER-MARK and FREQ-MARK <statement characteristics> allow special measurement capabilities for the <measured characteristic mnemonic>s SPECTRUM, SIGNAL-SEARCH, FREQ-DELTA, and POWER-DELTA used with a complex signal.

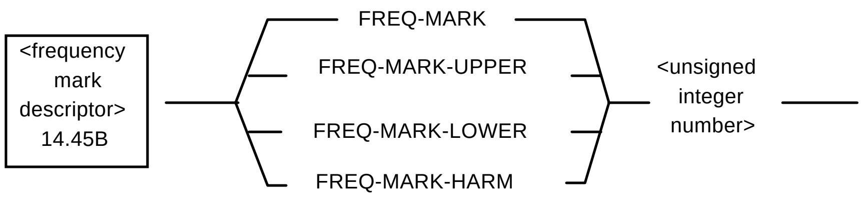
14.45.2 Formal syntax

Reference mark descriptor subfield

14.45.3 Syntax diagram


<mark descriptor subfield> 14.45
AT
<power mark descriptor> 14.45A
TO
<power mark descriptor> 14.45A
<frequency mark descriptor> 14.45B
TO
<frequency mark descriptor> 14.45B





14.45.4 Rules

1. Whenever the <mark descriptor subfield> is used, the COMPLEX SIGNAL <statement characteristics> shall also include the following:

FREQ-RANGE

RESOLUTION-BANDWIDTH

POWER MAX

2. The definitions for the mark descriptions are given in 17.7.

3. Whenever a measurement is made of SPECTRUM or SIGNAL-SEARCH, two values are returned in an array. This array shall be specified as a <statement characteristics> of the COMPLEX SIGNAL using the modifier RESP, and shall indicate that a list of two elements is to be returned.

4. The resultant response returned for SPECTRUM is a (frequency, power) pair of values representing the differences of the frequency and power marks defined. These values are calculated as follows:

FREQ diff. = FREQ (at the first <power mark descriptor>) - FREQ at the second <power mark descriptor>).

POWER diff. = POWER (at the first <power mark descriptor>) - POWER (at the second <power mark descriptor>).

The resultant response returned for SIGNAL-SEARCH is a (frequency, power) pair of values representing the differences of the frequency and power at the two frequency marks specified. These values are calculated as follows:

> FREQ diff. = FREQ (at the first <frequency mark descriptor>) - FREQ (at the second <frequency mark descriptor>)
>
> POWER diff. = POWER (at the first <frequency mark descriptor>) - POWER (at the second <frequency mark descriptor>)

5. The resultant response returned for POWER-DELTA is a power difference value representing the difference of the power at the two power marks specified, as follows:

> POWER-DELTA = POWER (at the first <power mark descriptor>) - POWER (at the second <power mark descriptor>)

The resultant response returned for FREQ-DELTA is a frequency difference value representing the difference of the frequency at the two frequency marks specified, as follows:

FREQ-DELTA = FREQ (at the first frequency mark descriptor>) - FREQ (at the second mark descriptor>)

6. POWER_MARK, FREQ_MARK shall be used only when SPECTRUM, SIGNAL SEARCH, FREQ_DELTA, or POWER_DELTA are the <measured characteristic mnemonic>.

14.45.5 Notes and examples

The use of the MARK_DESCRIPTOR subfield allows salient characteristics of the complex signals to be analyzed without the need for operator intervention. The mark descriptors represent place holders that describe the attribute of the signal at that point. The MARK_DESCRIPTORS themselves do not return a value. The result of the <measured characteristic> establishes the values needed. The power/ amplitude and frequency are separate domains. For explicit typing, the POWER-DELTA requires the POWER-MARK and the FREQ-DELTA requires the FREQ-MARK.

When power mark descriptors are used with SPECTRUM, two array values representing the difference in frequency and power as established by the power mark descriptors are obtained.

When power mark descriptors are used with POWER-DELTA, one value representing the difference in power as established by the power mark descriptors is obtained.

When frequency mark descriptors are used with SIGNAL-SEARCH, two array values representing the difference in frequency and power as established by the frequency mark descriptors are obtained.

When frequency mark descriptors are used with SIGNAL-DELTA, one value representing the difference in frequency as established by the frequency mark descriptors is obtained.

POWER-DELTA and SIGNAL-DELTA are single value measurement characteristics and can be used with a verify statement.

1. FREQ-MARK

   FREQ-MARK 0 represents the frequency position at the peak power (POWER-MARK 0) position within FREQ-RANGE. FREQ-MARK n, where (n = 0,1,2,...) represents the nth indicated frequency at the nth indicated decreasing power position from FREQ-MARK 0 (POWER-MARK 0) within FREQ-RANGE and can be above or below the frequency of FREQ-MARK 0 (increasing n indicates decreasing power peaks).

2. FREQ-MARK-HARM

   Has integer values from 1 to n representing harmonic number, where (n = 1,2,...) represents the power positions ordered by harmonic from the reference FREQ-MARK 0. This may not end up on a power peak, it is whatever the power is at that harmonic.

3. FREQ-MARK-LOWER

   Has integer values from 1 to n, where (n = 1,2,...) represents the next nth indicated signal peak from POWER-PEAK 0, irrespective of power magnitude above signal threshold in decreasing order of frequency.

4. FREQ-MARK-UPPER

   Has integer values from 1 to n, where (n = 1,2,...) represents the next nth indicated signal peak from POWER-PEAK 0, irrespective of power magnitude above signal threshold in increasing order of frequency.

5. POWER-MARK

   POWER-MARK 0 represents the peak power position within the frequency range of a sweep. POWER-MARK m represents the next mth indicated decreasing power position from POWER-MARK 0 within the frequency range of a sweep and can be above or below the frequency of POWER-MARK 0 (increasing m indicates decreasing power peaks).

6. POWER-MARK-LOWER

POWER-MARK n represents the next nth indicated decreasing power position from POWER-MARK 0 within the frequency range of sweep at a frequency below that of POWER-MARK 0 (increasing n indicates decreasing power peaks).

7. POWER-MARK-UPPER

POWER-MARK n represents the next nth indicated decreasing power position from POWER-MARK 0 within the frequency range of sweep at a frequency above that of POWER-MARK 0 (increasing n indicates decreasing power peaks).

14.46 <proportionality subfield>

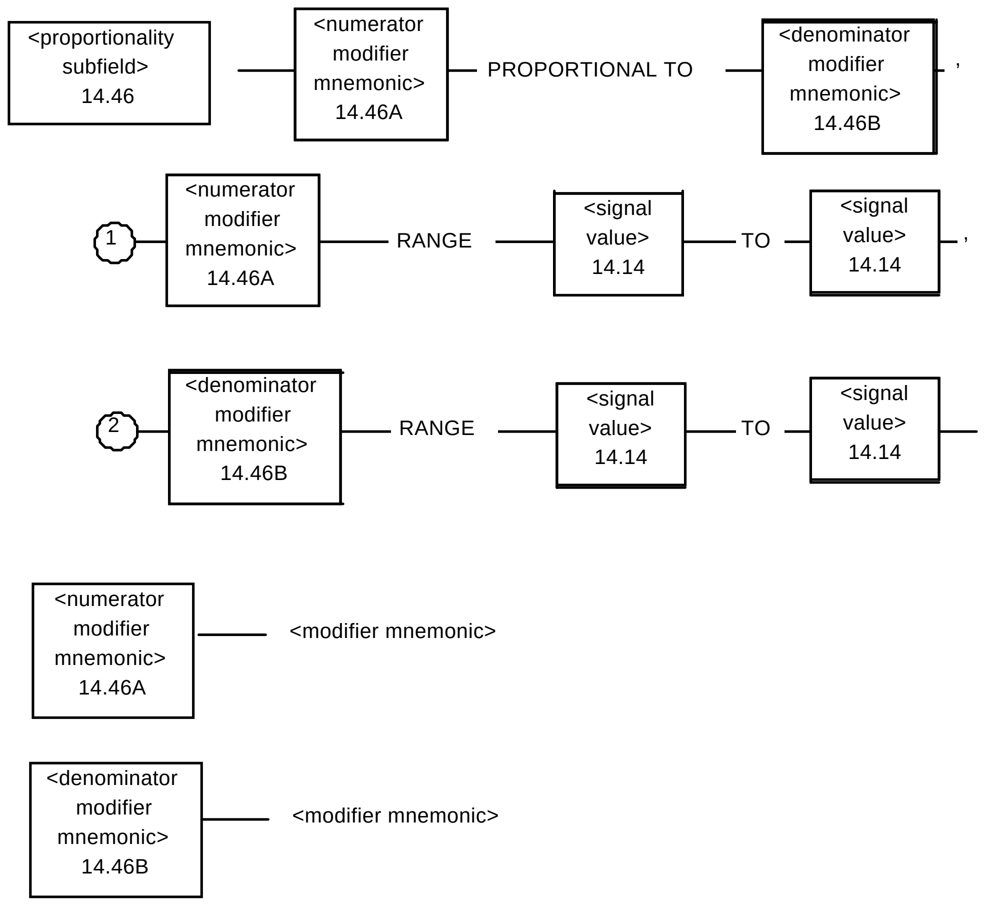
14.46.1 Function

The <proportionality subfield> is used to describe a sweep action of a COMPLEX SIGNAL where the characteristic being swept is to be controlled as a linear function proportional to a characteristic of a separate sweep control signal.

14.46.2 Formal syntax

Reference proportionality subfield

14.46.3 Syntax diagram


<proportionality subfield> 14.46
<numerator modifier mnemonic> 14.46A
PROPORTIONAL TO
<denominator modifier mnemonic> 14.46B
,
1
1
<numerator modifier mnemonic> 14.46A
RANGE
<signal value> 14.14
TO
<signal value> 14.14
,
2
2
<denominator modifier mnemonic> 14.46B
RANGE
<signal value> 14.14
TO
<signal value> 14.14
<numerator modifier mnemonic> 14.46A
<modifier mnemonic>
<denominator modifier mnemonic> 14.46B
<modifier mnemonic>

14.46.4 Rules

1. This subfield is used for the SWEEP complex function when a separate sweep control signal input is used to control the sweep actions.

2. The sweeping action defined by this subfield is a linear function, proportional to a parameter of the sweep control signal.

3. The numerator notation refers to the COMPONENT signal that is being swept by the complex function, and the <numerator modifier mnemonic> identifies the signal's characteristic which is to be swept the numerator of the proportionality. The denominator refers to the SWEEP-CONTROL signal of the complex function and the <denominator modifier mnemonic> identifies the signal characteristic which controls the sweep and is the denominator of proportionality.

14.47 <sweep configuration subfield>

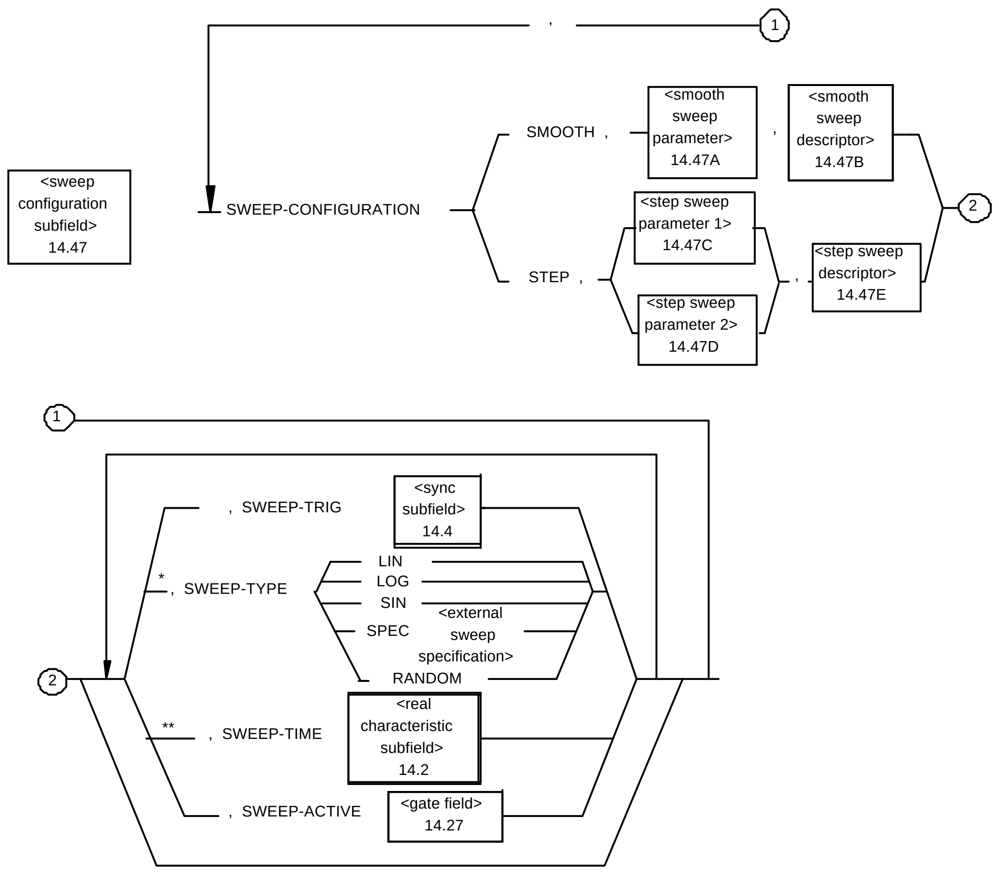
14.47.1 Function

The <sweep configuration subfield> is used in any signal source or sensor statement where a detailed description is required for the sweeping action of one signal parameter in accordance with the sweep characteristics specified within this subfield, based on a specific timing or event sequence description.

14.47.2 Formal syntax

Reference sweep configuration subfield

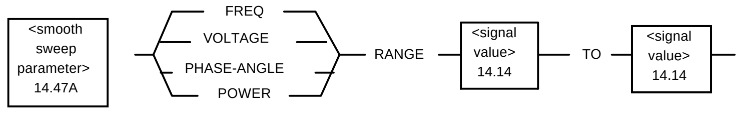
14.47.3 Syntax diagram

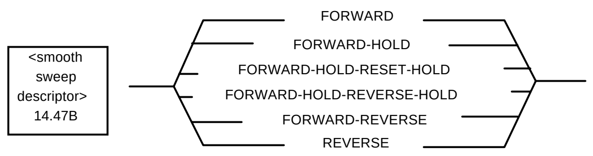

,
1
<sweep configuration subfield> 14.47
SWEEP-CONFIGURATION
SMOOTH ,
<smooth sweep parameter> 14.47A
,
<smooth sweep descriptor> 14.47B
2
STEP ,
<step sweep parameter 1> 14.47C
<step sweep parameter 2> 14.47D
,
<step sweep descriptor> 14.47E
1
, SWEEP-TRIG
<sync subfield> 14.4
* , SWEEP-TYPE
LIN
LOG
SIN
SPEC
<external sweep specification>
RANDOM
2
** , SWEEP-TIME
<real characteristic subfield> 14.2
, SWEEP-ACTIVE
<gate field> 14.27

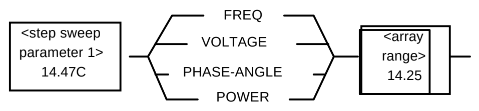
NOTES

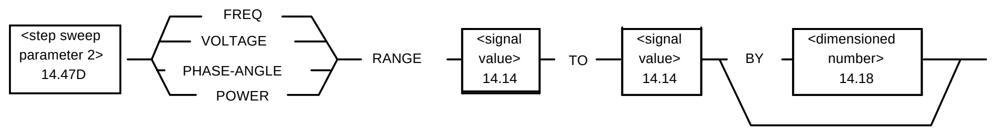
* Branch applicable only with SMOOTH configuration.

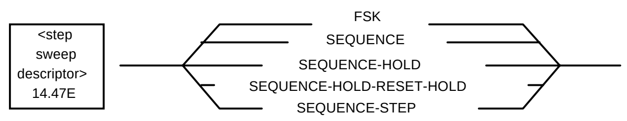
** Branch not applicable with SEQUENCE-STEP and FSK.











14.47.4 Rules

1. The <sweep configuration subfield> appears with any analog signal type that may be swept when no separate sweep control signal is provided to control the sweeping action.

2. <modifier mnemonic>s appearing in the <sweep configuration subfield> may not be specified outside this subfield as <statement characteristics>.

3. A <sweep configuration subfield> shall be present for each component of the signal that is to be swept.

4. The SWEEP-TRIG and the SWEEP-ACTIVE branches describe timing and synchronization that are applicable just to the sweeping action of the signal parameter. These control only the specified sweeping actions and do not control the basic signal being measured or applied.

5. The SWEEP-TRIG branch identifies the trigger to be used to control the timing of the sweep action. The effect of this trigger depends upon the type of sweep, as defined in the sweep descriptor subfields. Prior to the first trigger that is to start the sweep, if the signal is active the signal's sweep parameter is set to the first sweep value specified the reset value of the sweep parameter.

a) The SWEEP-TRIG branch is required with FSK, SEQUENCE-STEP and all -HOLD sweep sequences.

b) The SWEEP-TRIG branch is optional with all other sweep sequences. When SWEEP-TRIG is not specified, the sweep action will repeat continuously following reset of the sweep parameter to its first specified value at the end of each sweep.

6. The SWEEP-TIME branch defines the period in time over which the specified sweep is to be accomplished, following the start of the sweep action. If not specified, the sweep time is test system implementation dependent. SWEEP-TIME is not allowed for FSK and SEQUENCE-STEP.

7. The SWEEP-ACTIVE branch identifies a period of time during which the sweep actions specified are to be continued. Outside of this time period the sweep parameter is to be reset to the first value specified of the sweep range.

8. The following definitions apply:

SMOOTH The sweep is continuous and the changes in it are also continuous.

STEP A sweep with discrete changes.

SWEEP-CONFIGURATION This is a keyword used to indicate the beginning of a <sweep configuration subfield>. Sweeping is the traversing of a range of values of a quantity for the purpose of delineating, sampling, or controlling another quantity. Examples of swept quantities are the displacement of a scanning spot on the screen of a cathode-ray tube, and the frequency of a wave. Unless otherwise specified, a linear time function is implied; but the sweep may also vary in some other controlled and desirable manner. There are two types of SWEEP-CONFIGURATIONs that may be described: SMOOTH and STEP.

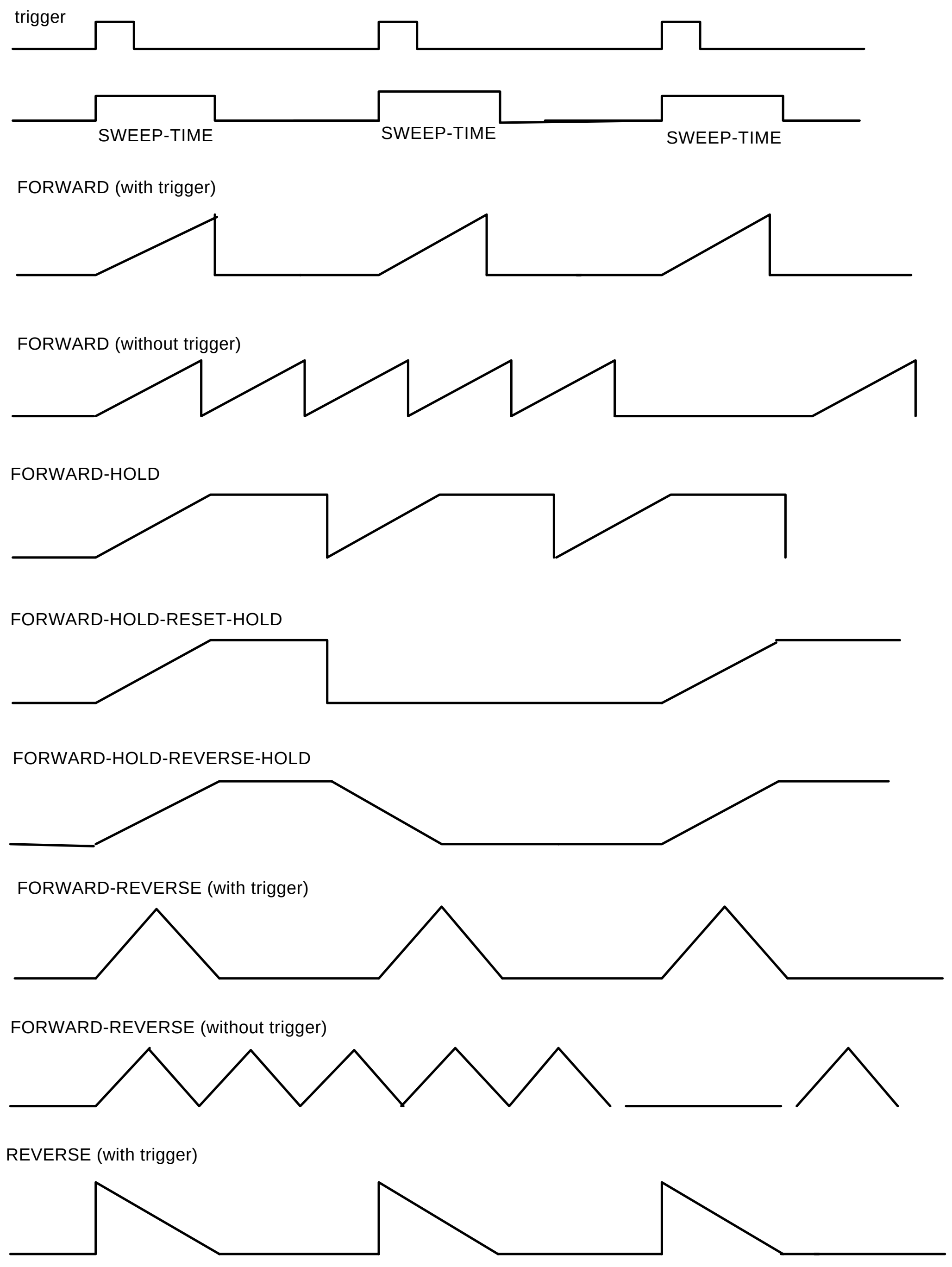
SWEEP-TYPE This mnemonic is the function in time in which a sweep is to change and shall contain one of the following values:

value	definition
LIN	linear
SIN	sine function
RANDOM	random function
LOG	log function
<external sweep specification>	

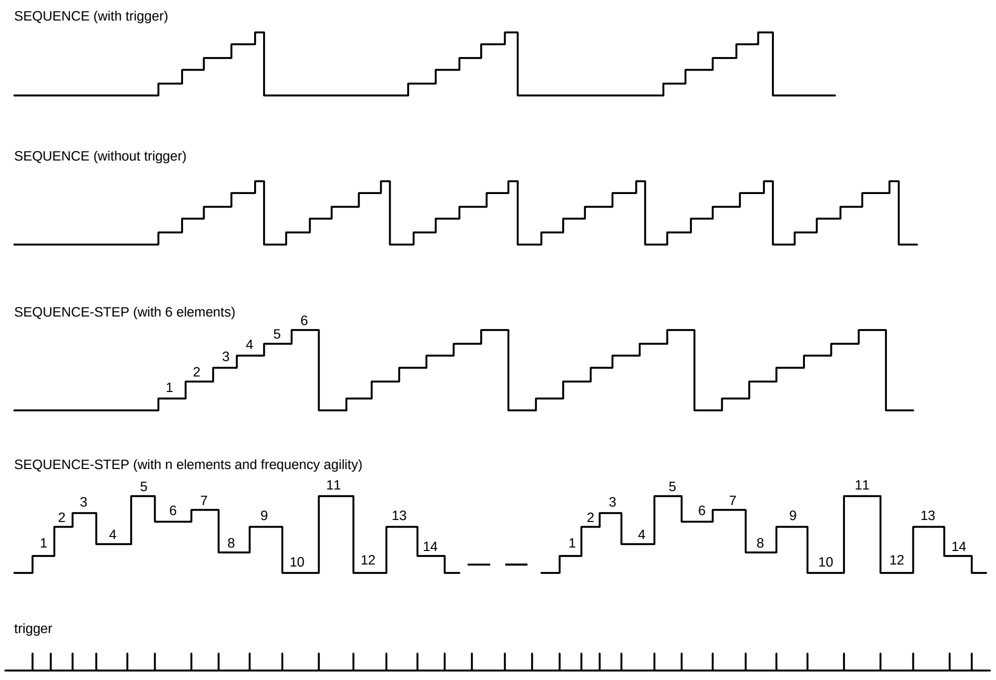
9. The <smooth sweep descriptor>s are defined in 17.8.

14.47.5 Notes and examples

1. <smooth sweep descriptor>



2. <step sweep descriptor>


SEQUENCE (with trigger)
SEQUENCE (without trigger)
SEQUENCE-STEP (with 6 elements)
1
2
3
4
5
6
SEQUENCE-STEP (with n elements and frequency agility)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
1
2
3
4
5
6
7
8
9
10
11
12
13
14
trigger

14.48 <fetch protocol parameter>

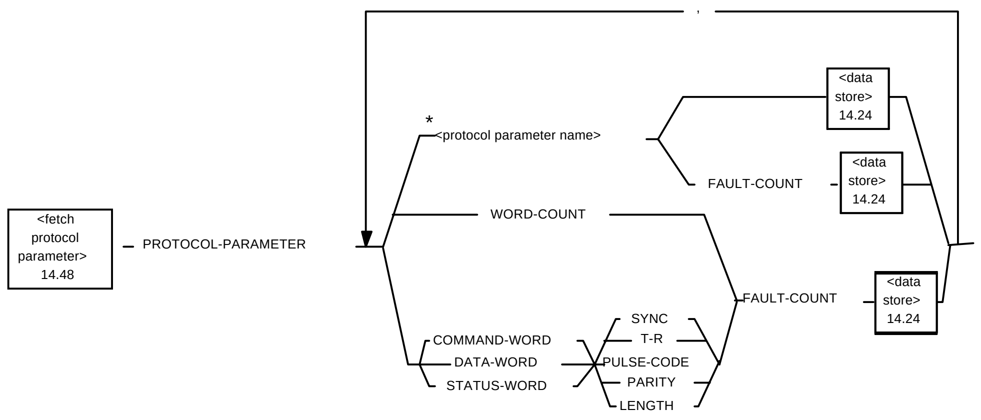
14.48.1 Function

This field provides the facility to extract from the databus test equipment the value or fault count for a given protocol parameter during the tests.

14.48.2 Formal syntax

Reference fetch protocol parameter

14.48.3 Syntax diagram



NOTE

* The name shall be one that has been introduced by an <extend statement> of the associated type.

14.48.4 Rules

1. All entries that are used in this field shall be fully defined and typed in the <establish protocol parameter> field of the associated <establish protocol statement>. Any <data store>s that follow FAULT_COUNT shall be of <type> INTEGER. For <protocol parameter name> the type of the <data store> shall be either the same <type> as in the <establish protocol statement> or the <data store> shall be a reference to a <record field> or <array element> with the same <type> as that employed in the <establish protocol statement>.

2. The following standard entries with the listed general meaning may appear in the field. Each one used shall be precisely defined by the <external bus specification>s referenced in the associated <establish protocol statement>.

a) WORD_COUNT Enables the extraction of a count of the total number of word count errors for a given <exchange> or <protocol> during the operation of a <do exchange statement>.

b) COMMAND_WORD, DATA_WORD, and STATUS_WORD Enables the extraction of a count of the total number of occurrences of a specified type of error on the specified word for a given exchange or <protocol> during the operation of a <do exchange statement>. The following predefined error types can be specified: faulty transmit/receive coding, faulty sync coding, invalid word parity, invalid word length, or an invalid pulse code in the word. These are indicated respectively by T_R, SYNC, PARITY, LENGTH, and PULSE_CODE. The required parameters shall be identified as FETCHABLE in the associated <establish protocol statement>.

3. If none of the standard entries is suitable to define a protocol parameter failure mode that is required in a test, a <protocol parameter name> may be included. The <protocol parameter name> shall be defined in an <extend statement> and referenced in the corresponding field of the <establish protocol statement>. The actual value of the parameter

(<data store> branch) or a count of the errors detected for that parameter (FAULT_COUNT branch) may be extracted depending on the associated <extend statement> and <establish protocol statement> definitions.

4. <protocol parameter name>s shall have been established as valid for the databus in the associated <establish protocol statement>. For it to occur in a <fetch exchange configuration statement> it shall have been defined as FETCHABLE for that category of statement. It is the only form of protocol parameter that can occur in a <fetch exchange configuration statement>, and then only if it has been established as monitored by the test equipment in the <do exchange statement>.

14.49 <databus fetch data>

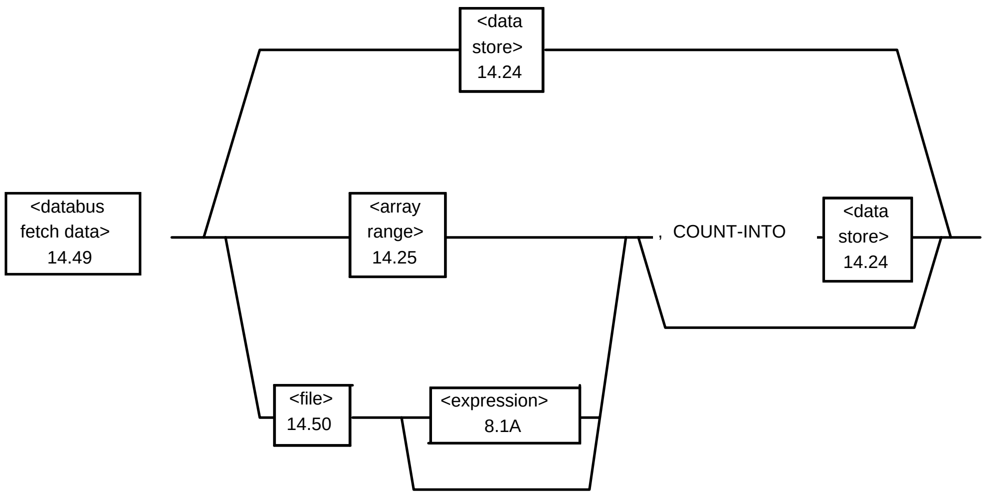
14.49.1 Function

This field specifies where the COMMAND, DATA, and STATUS data are stored when it is used in the <fetch exchange configuration statement> and the <do exchange statement>.

14.49.2 Formal syntax

Reference databus fetch data

14.49.3 Syntax diagram



14.49.4 Rules

1. A <databus fetch data> field is used in a <fetch exchange statement> field or an <exchange monitor> field to specify the destination of monitored/fetched data for the exchange. The <data store> and <array range> fields are filled at run time when the <do exchange statement> and <fetch exchange configuration statement> are completed.

2. The type of the <data store> and the basic type of the <array range> shall be identical to the type defined in the relevant <exchange model> from the <establish protocol statement>.

3. If the <array range> or <file> branch is used, this indicates that consecutive transmissions of the specific <exchange> on the bus will put their values into successive elements of the specified <array range> on an auto incrementing basis. The COUNT_INTO field is used to specify a destination for count of fetched/monitored elements stored in the <array range> or <file>. For rules regarding the use of the optional file position <expression>, see 8.1.4.2.

4. The <record field identifier> path in the <data store> field is allowed in the <update exchange statement> but not in the <define exchange statement>. This indicates that only the specified 'record field' will be updated.

14.50 <file>

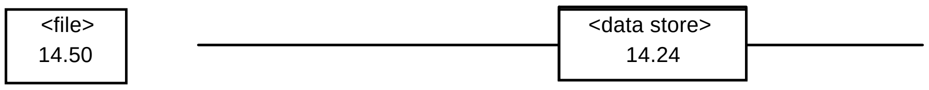
14.50.1 Function

The <file> function is to associate a valid file with a <data store>.

14.50.2 Formal syntax

Reference file

14.50.3 Syntax diagram



14.50.4 Rules

The <data store> shall be of type FILE.

15.0 Language component specification

15.1 Authorized characters

15.1.1 Function

All characters in table 15-2, Seven-bit ASCII code, in 15.9 are authorized for use in C/ATLAS statements.

15.1.2 Formal syntax

Reference 18.6.12 Basic Symbols and Character Sets

15.1.3 Syntax diagram

(not applicable)

15.1.4 Rules

1. Applicability of the C/ATLAS characters to specific character sets is specified in 15.10. Otherwise, applicability is specified by the syntactical definition of each C/ATLAS statement.

2. Lower case letters in UUT connector names may be represented in C/ATLAS connection fields either by lower case letters or by an upper case letter immediately preceded by a slash symbol (/) (see also 14.13.4).

3. The minus sign and hyphen are represented by the same character and will be interpreted according to the context of the statement.

4. Non-printing characters may be used in <character string> or <message text> and shall be represented by the symbols shown in table 15-2, enclosed in backslash symbols. If the backslash symbol itself is to be included, it shall be enclosed in backslash symbols.

15.2 Number representations

The following conventions will be used to express values in the language:

a) A decimal integer number is written as a string of characters beginning with an optional plus or minus sign, followed by one or more of the digits 0 through 9. A number written in this manner is of <base type> INTEGER.

b) A decimal fixed-point number is written as a string of characters beginning with an optional plus or minus sign followed by one or more digits 0 through 9, followed by a decimal point followed by one or more digits 0 through 9. A number written in this manner that is within the range and precision for DECIMAL numbers (see item g), below) is of <base type> DECIMAL; else it is of <base type> LONG_DECIMAL.

c) A decimal floating-point number is written in the form

±n.mE±p

where n, m, and p are numerical strings. Numbers allowed in the string for n, m, or p are 0 through 9. If m is 0 the decimal point and the 0 may be omitted. A number written in this manner that is within the range for DECIMAL numbers (see item g), below) is of <base type> DECIMAL; else it is of <base type> LONG_DECIMAL. The signs preceding n and p may be omitted if positive.

d) A binary number is written with the letter B followed by the appropriate string of numbers 0 and 1 enclosed in single quotes. A number written in this manner is of <base type> BIT if the string contains exactly one 0 or 1; else it is of <structured type> STRING OF BIT.

e) An octal number is written with a letter O followed by a string of characters 0 through 7 enclosed within single quotes. A number written in this manner is of <structured type> STRING OF BIT.

f) A hexadecimal number is written with a letter X followed by a string of the characters 0 through 9 or A through F enclosed within single quotes. A number written in this manner is of <structured type> STRING OF BIT.

g) Precision and magnitude of decimal, long decimal, and integer data types. The machine representation of decimal, long decimal and integer data types is implementation dependent. All implementations shall represent C/ATLAS <decimal number>s to a precision of at least 6 significant decimal digits over a magnitude range of at least $\pm 10^{\pm 38}$. All implementations shall represent C/ATLAS <long decimal number>s to a precision of at least 15 significant decimal digits over a magnitude range of at least $\pm 10^{\pm 307}$. C/ATLAS INTEGER implementations shall represent all integers in the range of at least ±32,767.

15.2.1 <decimal number>

15.2.1.1 Definition

A <decimal number> is a decimal integer number [see 15.2, item a)] or a decimal fixed-point number [15.2, item c)] or a decimal floating-point number [15.2, item c)].

15.2.1.2 Formal syntax

Reference decimal number

15.2.2 <unsigned decimal number>

15.2.2.1 Definition

An <unsigned decimal number> is a <decimal number> constrained to be written without a preceding plus or minus sign (+ or -).

15.2.2.2 Formal syntax

Reference integer number

15.2.3 <unsigned integer number>

15.2.3.1 Definition

An <unsigned integer number> is an integer number [15.2, item a)] constrained to be written without a preceding plus or minus sign (+ or -).

15.2.3.2 Formal syntax

Reference unsigned decimal integer

15.2.4 <digital number>

15.2.4.1 Definition

A <digital number> is a binary number [15.2, item d)] or an octal number [15.2, item e)] or a hexadecimal number [15.2, item f)].

15.2.4.2 Formal syntax

Reference digital number

15.2.5 <long decimal number>

15.2.5.1 Definition

A <long decimal number> is a decimal fixed-point number [15.2, item b)] or a decimal floating-point number [15.2, item c)] which is beyond the range or precision for DECIMAL [15.2, item g)].

15.3 Flag and statement numbers

15.3.1 <fstatno>

15.3.1.1 Function

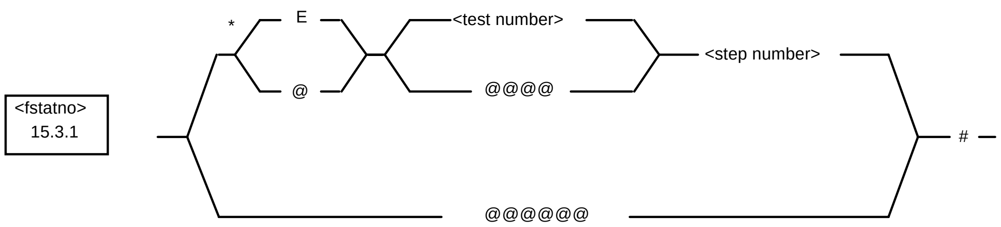
The flag and statement number field, <fstatno>, is used to identify C/ATLAS statements and structures.

15.3.1.2 Formal syntax

Reference fstatno

15.3.1.3 Syntax diagram

In the following diagram, all allowable blanks are represented by the @ or the # characters (see 15.4).



where,

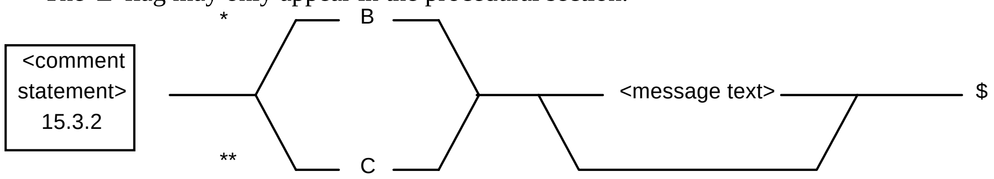
<test number> is exactly four decimal digits

<step number> is exactly two decimal digits

@ represents only the blank space

# represents one or more blank spaces

NOTE
* The 'E' flag may only appear in the procedural section.



NOTES
* A 'B' flagged statement must be succeeded by a non-null statement number in the <fstatno> of the statement immediately following. Furthermore, the statement immediately following must be one of the <main procedural statements>s.
** A 'C' flagged statement may appear between any other two C/ATLAS statements.

15.3.2 Flag

15.3.2.1 Function

A flag other than space designates a special function statement.

15.3.2.2 Formal syntax

(Not applicable)

15.3.2.3 Syntax diagram

(Not applicable)

15.3.2.4 Rules

1. The first character of each statement is the flag. It is located in the first column of the first row of each statement. The flag shall be one of the following characters, B, E, C, or space.

2. A flag may be written as a separate statement immediately preceding the statement to which it applies. This will allow more than one flag to be applied to a single statement.

3. An 'E' flag indicates entry points where the test conditions are completely stated and are not dependent on previous tests in any way. These are the only points where it is permissible to begin part of the complete test program.

Since the test must be valid starting at the 'E' flag with no signals attached to the UUT, with no SETUP or other single action verbs having been executed, and with no storage spaces having had data assigned to them, it is normally good practice to immediately precede the 'E' flagged statement by either a GO TO statement or a REMOVE, ALL statement or to make the 'E' flagged statement a REMOVE, ALL. It is strongly recommended that a comment be inserted immediately following the BEGIN, ATLAS PROGRAM statement that lists all the 'E' flagged statements and indicates the section of the test procedure that starts at the 'E' flag.

4. A 'B' flag indicates the destination statement of a GO TO statement elsewhere in the procedure. Characters following the 'B' flag and preceding the statement terminator are commentary and are ignored during translation. The 'B' flag shall always be written as a separate statement, and may be followed by commentary to indicate the origin statement of a program branch.

5. A 'C' flag indicates that information in that statement is to be ignored during translation. Characters in a statement flagged 'C' will appear only at the compiler input level as an aid to readability of the procedure.

15.3.3 <statement number>

15.3.3.1 Function

<statement number> is the number identifier of a statement in the C/ATLAS program.

15.3.3.2 Formal syntax

Reference statement number

15.3.3.3 Syntax diagram

(not applicable)

15.3.3.4 Rules

1. The second field is the <statement number>. This field is six characters long and provides a facility for a reference designator to be included in the program statement. The first four digits are called the <test number> and the remaining two digits are called the <step number>.

2. Each program step that is required to be referenced by another statement in the program shall have a <statement number>, as shall all 'E' flagged statements unless the 'E' flag is the whole statement, in which case the statement at the end of the flag series shall have a <statement number>. Other statements may be assigned a <statement number> that shall exceed the preceding number used; it is not necessary to use the next higher number.

3. A <statement number> is not allowed in the field following a 'B' or 'C' flag, but the statement following the 'B' flagged statement shall have a <statement number>.

4. If the <test number> of a numbered statement is the same as a previous statement, the first four characters of the <statement number> may be left blank.

15.3.3.5 Notes and examples

The <test number> and <step number> designations within the <statement number> are arbitrarily assigned for convenience in referencing test statements. The method of assigning the numbers is left to the discretion of the test program writer with the previously mentioned restrictions.

15.4 Blank space requirements

Blank spaces are used to improve readability, as subfield separators, and to separate words within some language elements. For consistent machine processing of C/ATLAS, one or more blank spaces are required between any two language elements except as specified below. A language element is any number, word, character, or subfield that is defined as a separate item in this standard. Also, one and only one blank space is required between words within any language element that contains two or more words. For example, WAIT FOR is a language element with two words. One blank space is required between the two words WAIT and FOR. Blank spaces adjacent to all special characters that are not part of a language element (such as comma, +, -, or $) and between alphabetic and numeric subfields are optional.

15.5 <label>

C/ATLAS uses <label>s as identifiers for the following. These label types are defined in the clauses listed below.

<constant identifier> (<constant>)	6.3, 15.11.2
<enumeration element>	6.3, 15.11.5
<signal>	6.5, 15.11.9
<program name>	4.1, 15.10.7
<module name>	4.1, 6.8, 15.10.6
<parameter>	6.5, 6.6, 15.6.4
<procedure>	6.6, 15.11.8
<record field identifier>	6.3, 15.11.6
<requirement>	6.7, 15.6.8
<type identifier> (<type>)	6.3, 15.11.3
<variable identifier>	6.3, 15.11.4
<timer>	6.10, 15.11.3
<event>	6.11, 6.12, 6.15, 15.11.10

<event interval>	6.13, 15.11.12
<event indicator>	6.14, 15.11.11
<configuration>	6.16, 15.11.16
<digital sensor>	6.16, 15.11.18
<digital source>	6.16, 15.11.17
<protocol>	6.18, 15.11.14
<exchange>	6.19, 15.11.15
<digital timing>	6.20, 15.11.27
<databus parameter label>	15.11.28
<device identifier>	6.19, 15.11.28.1
<digital timer>	6.10, 15.11.50
<exchange configuration>	6.22
<boolean variable>	6.16

1. A unique character string, called <label>, is required in certain statements to provide an identifier for subsequent reference to the labeled information within the current program. A <label> may be selected at the writer's convenience, and if properly selected will improve readability of the procedure. A <label> is formed by any combination of the characters set forth in 15.1.1, except non-printing symbols, the currency symbol, and parenthesis marks. The first character and the last character of each <label> shall be the apostrophe, which may not be used as any other character in the <label>. There is no limit to the length of the <label>, but only the first sixteen characters, excluding the apostrophes and any blank spaces, are considered significant in establishing the unique pattern of the <label>. At least one non-blank character shall be used. The case of the letters is significant.

2. Although user created <label>s are delimited by apostrophes, predeclared and predefined <label>s are written without apostrophes. Specifically, these <label>s are as follows:

a) The <pre-defined function>s of 8.1.4.4.

b) The <pre-declared type>s and ENUMERATION <constant>s of 6.3.4.7.

c) The boolean condition flags HI, LO, GO, NOGO, and MAX_TIME.

15.5.1 Notes and examples

The three examples below illustrate the 16 significant character feature.

'HIGH VOLTAGE SOURCE NR. 1' and 'HIGH VOLTAGE SOURCE NR. 2' are not unique.

'NR1. HIGH VOLTAGE SOURCE' and 'NR2. HIGH VOLTAGE SOURCE' are unique.

'HARDWIRE' and 'HARDWARE' are unique.

The following example illustrates how proper selection of a <label> can provide additional useful information in a statement.

```
    123456  CALCULATE, 'DOG' = 'WOLF' + 'TRAINING' $
```
or
```
    123456  CALCULATE, 'FOX' = 'DOG' + 'DRINK' $
```

15.6 Retention of labeled information

Labels are used throughout the language to represent different types of information. The retention period varies during compilation or execution of a test program depending on the type of the <label>. Generally, labeled information is retained until the same <label> is assigned to different information.

15.6.1 Scope of labels and identifiers

C/ATLAS has three different levels of access to the contents of a label or an identifier.

1. Labels specifically designated as GLOBAL/EXTERNAL have the widest scope throughout a combination of an <atlas program structure> (3.1.1) and one or more <atlas module structure>s (3.1.2) or <non-atlas module structure>s (3.1.3). This is a truly global scope at test execution of separately compiled modules whose executable code is linked. In the structure or module where the label is completely or centrally defined, the GLOBAL attribute is used. In the remaining structure or modules of the combination jointly using this label, the EXTERNAL attribute is attached to the <label>. For the <procedure> label, the EXTERNAL attribute is accompanied by a short form of the procedure.

2. The next lower level of <label> visibility is obtained when the scope of a <label> is limited by the extent of an <atlas program structure> or <atlas module structure>. This is the level of scoping that was implicitly considered global within a compilable program unit before the module and object code linkage concepts were introduced into C/ATLAS. This visibility of a <label> is still achieved by a simple DECLARE, DEFINE, IDENTIFY, or REQUIRE statement in the <program preamble structure>, the <module preamble structure>, or by an equivalently visible declaration in a <non-atlas module structure>.

3. The lowest level of a <label> visibility is obtained, when a <label> declaration appears in a DECLARE statement located within the <local preamble structure> (6.2), within a <procedure body>, or as a formal parameter in the DEFINE <procedure> statement. Such a <label> is only recognized within this procedure.

15.6.2 Retention of measured sensor values

Information assigned to a <label> is available for reuse in subsequent statements until the end of the test specification or until the same label is assigned other information. Information assigned to a local <label> within a <procedure> is only available within that procedure. A local <data store> can be made available outside the procedure by assigning it to a parameter label that appears in the RESULT section of the DEFINE statement or by assigning it to a global <label>.

15.6.3 <label> for DEFINE and IDENTIFY statement

DEFINE and IDENTIFY statements establish and label segments of a <complete atlas test program structure> so that the atlas program writer can write a segment the first time and use only a label where the segment is used subsequently in the <complete atlas test program structure>. DEFINE and IDENTIFY statements with the attribute EXTERNAL establish a label for segments of a C/ATLAS test program that are defined outside of the actual C/ATLAS program or C/ATLAS module.

15.6.4 <parameter>

When a <procedure> is defined, it is convenient to designate certain values as variables so that the definition can be used in several subsequent places in the program with different actual values. This is accomplished by writing a <parameter> in place of the value in the <procedure body> and listing the labels in the formal parameter list. Parameter labels are local to the <procedure>. A label used as a <parameter> for one procedure may be used as a <parameter> label for another procedure or as any other type of label outside the procedure without ambiguity.

A parameter <label>, when used in place of the magnitude of a signal value in a signal-oriented statement, shall also have a RANGE sub-field to indicate the expected magnitude of the variable. This additional information need not be given in non-signal-oriented statements (e.g., OUTPUT, CALCULATE, etc.).

15.6.5 <data store>

Information, including values of measurements, calculated values, and data entered via an INPUT statement, can be placed and held in storage spaces identified by <data store> labels. By use of these labels in the proper contexts, their information can be retrieved for use. Each <data store> may hold one value if it is not identified as having a <structured type>. Such <data store>s contain one data value of type <base type> or a type defined from these types. Each <data store> that is defined as <structured type> may contain more than one value, in accordance with data typing rules of 6.3.4.8.

Depending on its declaration, a <data store> label may designate a single value stored, or all elements of a RECORD, ARRAY, or STRING.

15.6.6 Condition identifiers

C/ATLAS includes a number of GLOBAL condition identifiers that are implicitly defined and are of BOOLEAN type. They are called GO, NOGO, HI, LO, and MAX_TIME, and they are never delimited by single quotes. These condition identifiers have values that are set in specific C/ATLAS statement contexts. The GO, NOGO, HI, and LO condition identifiers are set to TRUE or FALSE as defined in 14.8 <evaluation field>. The MAX_TIME condition identifier is set as defined in 14.9 <max time>.

The initial values of these boolean type identifiers are:

GO	=	TRUE
NOGO	=	FALSE
HI	=	FALSE
LO	=	FALSE
MAX_TIME	=	FALSE

Any of these condition identifiers may be referenced by using the <condition> field (14.11). Note that references are made by writing the condition identifier without apostrophes. Whenever any condition identifier is referenced, it will return the value TRUE or FALSE as set by the last <evaluation field>, <input statement>, or <max time> field executed.

15.6.7 Other variables ( )

In <signal> definitions, variable numbers and character strings are designated by a parentheses pair ( ). These variables have the same effect as those designated with <parameter> in DEFINE procedures. In <signal> definitions, the number of spaces between the parentheses pairs is not significant.

A <define signal statement> or <define procedure structure> may have certain portions of the statements left as variable. These items are like buckets that are filled by the procedural statement that uses the <signal> definitions, and by <parameter> in <procedure> definitions. The <parameter> labels are local to DEFINE <procedure>. In other words, a number passed to a <parameter> (e.g., 'CAT') in a particular <procedure> will not displace information in another <parameter> named 'CAT' in another <procedure>, or a <value> named 'CAT' in a <procedural structure>.

When a <parameter> is used in a <define procedure structure> or parentheses are used in a <define signal statement> to represent variable quantities, they are written with dimensions. Dimensions can not be a variable in a DEFINE <signal> (see 6.5.4). Constants or values are given with the arguments in the PERFORM statement or, in the case of a DEFINE <signal>, as arguments of the procedural statement that reference the defined <signal>.

15.6.8 <requirement>

A label within a <require statement> that identifies a virtual test resource. The virtual resource may or may not correspond to a real test resource.

15.6.9 <exchange>

An <exchange> label is used to reference a particular exchange that has been defined in a previous <define exchange statement>.

15.6.10 <complex signal>

A <complex signal> label is used to reference a particular COMPLEX SIGNAL that has been defined in a previous <define complex signal structure>.

15.7 Range subfield

When a characteristic is designated variable in a DEFINE <signal> statement, a range minor field may be included to state the lowest and highest values that will be assigned to that characteristic in subsequent statements. These values are separated by the word TO and follow the word RANGE in this subfield (example: VOLTAGE ( ) RANGE 5V TO 50V). Use of the range field with a sensor function implies the total capability of all scales of the measurement device.

15.7.1 Range requirement for <data store> or <parameter>

A <data store> or <parameter> label that is used in place of a signal in a signal oriented statement shall have a MAX, MIN, or RANGE subfield to indicate the expected magnitudes of the variable. This additional information need not be given in non-signal-oriented statements (e.g., OUTPUT, CALCULATE, etc.).

15.8 Dimensions

The vocabulary includes dimensional units that are used in the modifier fields to fix the units of the value stated. Dimensional units are always included with the modifier value. The authorized dimensional units are listed and defined in table 15-1. The use of units conforming to ISO 1000 : 1992 is preferred. These are indicated by "Yes" in the column headed ISO 1000. Some other units are recognized under ISO 1000 for special use. These are indicated by a "Perm" in the column headed ISO 1000.

In any syntax diagram in which <dim> occurs, a reference may be made to any of these dimensional units or to a <dim name> that has been introduced in an EXTEND statement. Where no dimension is shown in the X10• column, the <dim> subfield is empty. In all other columns, if no dimension occurs, the multiplier is prohibited, [e.g., if it were necessary to refer to 3 megavolts, this could be written as +3E+3KV or +3E+6V, but not as +3MV (which means 3 millivolts, not 3 megavolts].

15.8.1 Units of rate of change

Units of rate of change or acceleration may be formed from any of the units listed in the table. Rate units are written by adding the slash character and appropriate units of time to the basic unit from the table (example: FT/SEC, V/MSEC, or REV/MIN.) Acceleration units are formed by adding another slash and unit of time to the rate convention (example: DEG/SEC/SEC, FT/SEC/SEC). The following special cases of angular rate may be used:

RPS	revolutions per second in lieu of REV/SEC
RPM	revolutions per minute in lieu of REV/MIN
RPH	revolutions per hour in lieu of REV/HR

Table 15-1. Table of authorized dimensions

Quantity	Basic units	ISO 1000	Relationship	X10^{0}	X10^{3}	X10^{6}	X10^{9}	X10^{-3}	X10^{-6}	X10^{-9}	X10^{-12}
Angle, plane	degree	No	1 DEG = π/180 RAD	DEG							
	radian	Yes	SI supplied unit	RAD				MRAD	URAD		
	revolution	No	1 REV = 2π RAD	REV							
Angle, solid	steradian	No	SI supplied unit	SR				MSR			
Burst length	cycles	No		CYCLES							
	pulses	No		PULSES							
Capacitance	farad	Yes	1F = 1 C/V			(NOT USED)					
		No	1FD = 1F	FD					UFD	NFD	PFD
Charge	coulomb	Yes	1C = 1A–SEC	C	KC				UC	NC	
Conductance	siemens	Yes	SI basic unit	S							
Current	ampere	Yes	SI basic unit	A	KA			MA	UA	NA	
Digital bit rate	bits per second	N/A		BITS/ SEC	KBITS/ SEC	MBITS/ SEC					
Digital word length	digit, character or bit	N/A		DIGITS, CHAR, BITS							
Digital word rate	words per second	N/A		WORDS/ SEC	KWORDS/ SEC	MWORDS/ SEC					
Distance, wavelength	meter	Yes	SI basic unit	M	KM			MM	UM	NM	
	inch	No	1 IN = 25.4 MM	IN							

Table 15-1. Table of authorized dimensions (continued)

Quantity	Basic units	ISO 1000	Relationship	$X10^{0}$	$X10^{3}$	$X10^{6}$	$X10^{9}$	$X10^{-3}$	$X10^{-6}$	$X10^{-9}$	$X10^{-12}$
	foot	No	1 FT = 12 IN	FT							
	statute mile	No	1 S-MILE = 5280 FT	S-MILE							
	nautical mile	No	1 N-MILE = 1825 M	N-MILE							
Energy, work, heat	joule	Yes	1 J = 1 N-M	J	KJ			MJ			
	electron volt	Perm	1 EV = 1.6 E-19J	EV	KEV	MEV					
Events	counts	N/A		TIMES							
Flux, magnetic	weber	Yes	1 WB = 1 V-SEC	WB				MWB			
Flux density	tesla	Yes	1 T = 1 WB/M–M	T				MT	UT	GAM	
Force	newton	Yes	1 N = 1KG–M /SEC–SEC	N	KN			MN	UN		
Frequency	hertz	Yes	1 HZ = 1/SEC	HZ	KHZ	MHZ	GHZ				
	pulses per sec	No		PPS	KPPS						
Illuminance	lux	Yes	1 LX = 1LM/M–M	LX							
Inductance	henry	Yes	1 H = 1 WB/A	H				MH	UH	NH	PH
Luminance	nit	No		NT							
Luminous flux	lumen	Yes	1 LM = 1 CD–SR	LM							

Table 15-1. Table of authorized dimensions (continued)

Quantity	Basic units	ISO 1000	Relationship	X10^{0}	X10^{3}	X10^{6}	X10^{9}	X10^{-3}	X10^{-6}	X10^{-9}	X10^{-12}
Luminous intensity	candela	Yes	SI basic Unit	CD							
Mass	kilogram	Yes	SI basic Unit	KG				G	MG	UG	
Power	watt	Yes	1 W = 1J/SEC	W	KW			MW	UW		
Pressure	pascal	Yes	1 PA = 1 N/M–M	PA	KPA			MPA	UPA		
	millibar	Perm	1 MB = 100 PA	MB							
	millimeters of mercury	No		MMHG							
	inches of mercury	No		INHG							
Time delay	second	No		SEC				MSEC	USEC	NSEC	
Pulse	pulse	No		PULSES							
Phase shift	degree	No		DEG							
Ratio	decibel	Perm		DB							
	percent	—		PC							
	dimensionless	—									
Resistance	ohm	—	1 OHM = 1 V/A	OHM	KOHM	MOHM					
Susceptance	siemens	Yes	SI basic Unit	S							
Temperature	degree kelvin	Yes	SI basic Unit (°K)								
		No	1 DEGK = 1° K	DEGK							

Table 15-1. Table of authorized dimensions (continued)

Quantity	Basic units	ISO 1000	Relationship	$X10^0$	$X10^3$	$X10^6$	$X10^9$	$X10^{-3}$	$X10^{-6}$	$X10^{-9}$	$X10^{-12}$
Temperature (continued)	degree centigrade	No		DEGC							
	degree farenheit	No		DEGF							
Time	second	Yes	SI basic unit (S)			(NOT USED)		MSEC	USEC	NSEC	PSEC
		No	1 SEC = 1S	SEC							
	minute	Perm	1 MIN = 60 SEC	MIN							
	hour	Perm	1 HR = 60 MIN			(NOT USED)					
		No	1 HR = 1 H	HR							
Torque	newton-meter	Yes	1 N-M = 1N M	N-M							
Velocity, linear	meters per second	Yes	SI derived unit (1 M/S)			(NOT USED)					
		No	1 M/SEC = 1 M/S	M/SEC							
	feet per second	No		FT/SEC							
	knot	No	1 KT = 0.5 M/SEC	KT							
	mach no.	No		MACH							
Velocity, angular	radian per second	Yes	SI derived unit (1 RAD/S)			(NOT USED)					
		No	1 RAD/SEC = 1 RAD/S	RAD/SEC							
Voltage	volt	Yes	1 V = 1 J/C	V	KV			MV	UV		
Volume	liter	Perm	1 L = 0.001 M-M-M	L				ML			

15.8.2 Units

Units that are a product of others (e.g., torque which is newton-metre), shall be formed from the units listed in the table. Units comprising a product shall be separated by a hyphen. Thus the newton-metre dimension is written as N_M.

15.8.3 Power dimensions

Power may also be expressed in decibels above a specified power level according to the following convention:

DBM is decibels above one milliwatt.

DBW is decibels above one watt.

DBK is decibels above one kilowatt.

15.8.4 ISO characters

1. ISO 1000 lower-case letters are replaced by uppercase letters.

2. The following are used in lieu of adopting special characters for the ISO 1000 symbol:

Ω	OHM
°	DEG
'	MIN
"	SEC

3. Units are selected over the range most frequently encountered in testing, such that conflict will not exist for mega vs. milli prefixes. That is, either the milli or mega prefix (but not both) are specified for some dimensions.

15.8.5 <dim>

Any noun-modifier dimensional unit. <dim> contains the dimension of the quantity referenced. If the quantity is dimensionless, <dim> is empty (see also clause 16).

15.8.6 <freq dimension>

HZ, MHZ, KHZ, GHZ

15.8.7 <time dimension>

HR, MIN, SEC, MSEC, USEC, NSEC, PSEC.

15.9 Character code definitions

The C/ATLAS digital language has the ability to define alphanumeric and control characters, coded in various standard forms, that shall be transmitted to a UUT as stimuli or read from a UUT as a response. Existing standard definitions for these codes are used in C/ATLAS.

15.9.1 ASCII (American Standard Code for Information Interchange)

A 7-bit ASCII code is defined in approved USA standards.

The ASCII codes are reproduced in table 15-2. In the ASCII code system, bit positions are defined as follows:

Table 15-2. Seven-bit ASCII code

b7 b6 b5					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b 4	b 3	b 2	b 1	COLUMN ROW	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	'	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	–	o	DEL

NOTES

1. Reference: ANSI X.3.4-1986 (R 1992).
2. The binary code sequence is as follows:

b	b	b	b	b	b	b	b
8	7	6	5	4	3	2	1
MSB							LSB

Control characters (such as, ASCII EOT or NUL) may be written in C/ATLAS by enclosing the character in backslash symbols (See 15.1.4, rule 4).

15.9.2 ISO (International Organization for Standardization)

The ISO codes are shown in table 15-3.

Table 15-3. Seven-bit ISO code

	b7	0	0	0	0	1	1	1	1
	b6	0	0	1	1	0	0	1	1
	b5	0	1	0	1	0	1	0	1
b4 b3 b2 b1	COLUMN / ROW	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	TC7 (DLE)	SP	0	@	P	'	p
0 0 0 1	1	TC1 (SOH)	DC1	!	1	A	Q	a	q
0 0 1 0	2	TC2 (STX)	DC2	"	2	B	R	b	r
0 0 1 1	3	TC3 (ETX)	DC3	#	3	C	S	c	s
0 1 0 0	4	TC4 (EOT)	DC4	$	4	D	T	d	t
0 1 0 1	5	TC5 (ENQ)	TC8 (NAK)	%	5	E	U	e	u
0 1 1 0	6	TC6 (ACK)	TC9 (SYN)	&	6	F	V	f	v
0 1 1 1	7	BEL (ETB)	TC10	'	7	G	W	g	w
1 0 0 0	8	FE0 (BS)	CAN	(	8	H	X	h	x
1 0 0 1	9	FE1 (HT)	EM	)	9	I	Y	i	y
1 0 1 0	10	FE2 (LF)	SUB	*	:	J	Z	j	z
1 0 1 1	11	FE3 (VT)	ESC	+	;	K	[	k	{
1 1 0 0	12	FE4 (FF)	IS4 (FS)	,	<	L	\	l	|
1 1 0 1	13	FE5 (CR)	IS3 (GS)	-	=	M	]	m	}
1 1 1 0	14	SO	IS2 (RS)	.	>	N	^	n	~
1 1 1 1	15	SI	IS1 (RS)	/	?	O	_	o	DEL

NOTES

1. Reference: ISO/IEC 646:1991.

15.10 Special character sets

15.10.1 <character string>

<character string> is any contiguous sequence of characters meeting the following conditions: The first character is an upper-case letter C, the first non-blank character following the first character is an apostrophe, the last character is an apostrophe, and there are exactly two apostrophes.

15.10.2 <message text>

Any sequence of characters except the currency symbol.

15.10.3 <letter>

One of the upper case alphabetical characters.

15.10.4 <digit>

One of the numerical characters 0 through 9.

15.10.5 <modifier descriptor>

<modifier descriptor> is used to indicate one of a set of possible classes for a modifier. It is constructed from a <letter> or <decimal digit> that may be followed by any number of <letter>s, <decimal digit>s, or hyphens (-). The permitted <modifier descriptor>s are listed in clause 17 in the section for the associated modifier.

15.10.6 <module name>

Any sequence of characters except the comma, currency symbol, and non-printing characters. The first and last characters shall be apostrophes, and there are exactly two apostrophes in the name.

15.10.7 <program name>

Any sequence of characters except the comma, currency symbol, and non-printing characters. The first and last characters shall be apostrophes, and there are exactly two apostrophes in the name.

15.10.8 <extend label>

An <extend label> is any sequence of characters that consists of the following:

a) Upper case letters: A through Z

b) Numerals: 0 through 9

c) The hyphen (minus sign) '-'

The first character shall be an upper case letter. <noun name>, <modifier name>, <modifier descriptor name>, <pin descriptor name>, and <dim name> are all <extend label>s.

There is no limit to the length of an <extend label>, but only the first sixteen characters are considered significant in establishing the unique pattern of the <extend label>.

15.11 Miscellaneous language elements

15.11.1 <reference phase>

One of the reference phase identifiers listed in 14.13.1.2.

15.11.2 <constant identifier>

A <label> that is established by a <declare statement> (see 6.3) to represent a <constant>.

15.11.3 <type identifier>

A <label> that is established by a <declare statement> (see 6.3) to represent a data type.

15.11.4 <variable identifier>

A <label> that is established by a <declare statement> (see 6.3) to represent a variable of DECLAREd <type>.

15.11.5 <enumeration element>

One of a set of constants of an ENUMERATION data type as established by the corresponding <declare statement> (see 6.3).

15.11.6 <record field identifier>

A <label> that represents the field within the <structured type> RECORD that is established by a <declare statement> (see 6.3).

15.11.7 <modifier mnemonic>

The mnemonic for a noun-modifier listed in clause 17.

15.11.8 <procedure>

A <label> that is established by a <define procedure statement> (see 6.6.2) to identify a sequence of statements to be executed.

15.11.9 <signal>

A <label> that is established by a <define signal statement> (see 6.5).

15.11.10 <event>

A <label> that is established by an <identify signal based event statement> (see 6.11), an <identify event based event statement> (see 6.12), or an <identify time based event statement> (see 6.15).

15.11.11 <event indicator>

A <label> that is established by an <identify event indicator statement> (see 6.14).

15.11.12 <event interval>

A <label> that is established by an <identify event interval statement> (see 6.13).

15.11.13 <timer>

A <label> that is established by an <identify timer statement> (see 6.10).

15.11.14 <protocol>

A <label> that is established by an <establish protocol statement> (see 6.18).

15.11.15 <exchange>

A <label> that is established by a <define exchange statement> (see 6.19).

15.11.16 <configuration>

A <label> that is established by a <define digital configuration statement> (see 6.16.2).

15.11.17 <digital source>

A <label> that is established by a <define digital source statement> (see 16.16.3).

15.11.18 <digital sensor>

A <label> that is established by a <define digital sensor statement> (see 6.16.4).

15.11.19 <dim name>

An <extend label> (see 15.10.8) that is introduced in the <mod dim> field of the <extend statement> (6.17).

15.11.20 (deleted)

15.11.21 <noun>

Any of the nouns listed in clause 16.

15.11.22 <noun name>

An <extend label> (see 15.10.8) that is introduced in the NOUN field of an <extend statement> (Section 6.17).

15.11.23 <pin descriptor>

See 14.13.1.1.

15.11.24 <pin descriptor name>

An <extend label> (see 15.10.8) that is introduced in the <extend cnx> field of an <extend statement> (6.17).

15.11.25 <modifier name>

An <extend label> (see 15.10.8) that is introduced in the <modifier definition> field (see 6.17A in the syntax diagram in 6.17.3) of an <extend statement> (see 6.17).

15.11.26 <signal oriented verb>

One of the verbs included in clauses 11, 6.5, 6.7, 6.11, 6.16.3 and 6.16.4.

15.11.27 <digital timing>

see <label> that is established by a <define digital timing statement> (see 6.20.1).

15.11.28 <databus parameter label>

A <label> associated with one of the aspects of a databus that are defined in detail by the <external bus specification> set referenced by an <establish protocol statement>. A <databus parameter label> is one of the following (15.11.28.1-15.11.28.5):

15.11.28.1 <device identifier>

A <label> identifying a device that is normally attached to the databus and which may act as a TALKER, a LISTENER, or as both at different times.

15.11.28.2 <test equip role name>

An <extend label> identifying a role that the test equipment can perform during a databus test that is different from the three standardized roles and is introduced in an <extend statement>.

15.11.28.3 <bus parameter name>

An <extend label> identifying a parametric mode of the databus that is additional to the seven standardized parametric modes and is introduced in an <extend statement>.

15.11.28.4 <protocol parameter name>

An <extend label> identifying a protocol mode on a databus that is additional to the sixteen standardized protocol modes and is introduced in an <extend statement>.

15.11.28.5 <bus mode name>

An <extend label> identifying a databus operational mode that is not one of the standard modes defined in 6.20 and is introduced in an <extend statement>.

15.11.29 <external bus specification>

A specification, not written in C/ATLAS, that defines aspects of a databus, its protocol, failure modes, and the system in which it operates. The information included in the specification is essential to a complete understanding of the databus operation and consequently of the test. Complete information may require reference to two or more <external bus specification>s.

15.11.30 <modifier descriptor name>

An <extend label> (see 15.10.8) that is introduced in the <mod descr info> field (see 6.17G in the syntax diagram in 6.17.3) of an <extend statement> (see 6.17).

15.11.31 <denominator modifier mnemonic>

An identification of the signal characteristic that controls the sweep and is the denominator of proportionality.

15.11.32 <external compensation specification>

A <message text> that identifies a specific reference document specifying how the path loss compensation data is to be applied to the <signal value>.

15.11.33 <external complex function specification>

A <message text> that identifies a specific reference document containing the type and specification of the complex function of a COMPLEX SIGNAL definition.

15.11.34 <external signal conditioning specification>

A <message text> that identifies a specific reference document containing the identification of other signal conditioning parameters.

15.11.35 <external signal name specification>

A <message text> that identifies a specific reference document containing the identification of other standard signal types.

15.11.36 <external sweep specification>

A <message text> that identifies a specific reference document containing a detailed description of the sweeping action of one signal parameter in accordance with the sweep characteristics specified, based on specific timing or event sequence descriptions.

15.11.37 <frequency mark descriptor>

One of the <statement characteristics> listed in 14.45B in the syntax diagram in 14.45.3.

15.11.38 <function characteristic mnemonic>

The <signal characteristic>s applicable to the <function mnemonic> as defined in 6.21.4.4.

15.11.39 <function mnemonic>

The type of function applied to the component signals to generate the COMPLEX SIGNAL.

15.11.40 <measured characteristic mnemomic>

A <signal characteristic>, within the available characteristics of those listed with the noun in clause 16, that is to be evaluated by a sensor function.

15.11.41 <numerator modifier mnemonic>

An identification of the <signal characteristic>s that controls the sweep and is the numerator of proportionality.

15.11.42 <power mark descriptor>

One of the <statement characteristics> listed in 14.45A in the syntax diagram in 14.45.3.

15.11.43 <signal conditioning mnemonic>

A description of the type of conditioning to be applied to the COMPLEX SIGNAL. The types of conditioning and their legal characteristics are defined in 6.21.6.4.

15.11.44 <single action verb>

One of the verbs included in 11.2.

15.11.45 <smooth sweep descriptor>

One of the <statement characteristics> listed in 14.47.

15.11.46 <smooth sweep parameter>

One of the <statement characteristics> listed in 14.47.

15.11.47 <step sweep descriptor>

One of the <statement characteristics> listed in 14.47.

15.11.48 <step sweep parameter 1>

One of the <statement characteristics> listed in 14.47.

15.11.49 <step sweep parameter 2>

One of the <statement characteristics> listed in 14.47.

15.11.50 <digital timer>

A <label> that is established by an <identify timer statement> (see 6.10).

15.11.51 <exchange configuration>

A <label> that is introduced by a <define exchange configuration statement>.

15.11.52 <boolean variable>

A <label> associated with a variable of <type> BOOLEAN that is used in a <define digital source statement>. It is set to TRUE when the ILLEGAL_STATE_INDICATOR relating to either a digital source or a digital sensor indicates that its associated source or sensor signal value is not as specified.

16.0 Nouns and their modifiers

The following is the list of nouns that may be used, in general, with any source, sensor, or load statement. Each noun is listed in its C/ATLAS form and is followed by its definition and a list of noun modifiers that may be used with that noun. For each modifier, its C/ATLAS form (or mnemonic) is given along with the allowed dimensions and other information as noted below. Definitions of the modifiers may be found in clause 17. Where meaningful, modifiers that are constructed from a mnemonic by adding prefixes and suffixes in the manner specified in clause 17 may be used with those nouns having that mnemonic in its noun modifier set.

a) Verb Dependence

A major concern is damage to banks of relays or electric switches in the routing matrix between ATE resources and the ATE interface connections. The single action verb illustration in figure 11-5 is based on the interactions between UUTs and virtual ATE resources. The state diagrams for sources and loads in figures 11-1 and 11-2 illustrate a complete set of allowable states and transitions between states.

In order to have a definitive sequence of actions for APPLY and REMOVE statements, the nouns have been divided into two general classes: source nouns and load nouns. For the APPLY and REMOVE statements, a separate sequence of actions is specified for each class of nouns and these sequences are described using separate sequences of single action verbs.

1) Source nouns

All NOUNS not listed in 2) below except TIME INTERVAL

2) Load nouns

IMPEDANCE	16.20
SHORT	16.39
COMMON	16.6
EARTH	16.12
LOGIC LOAD	16.24

b) Modifier Type

The Modifier Type defines the type of information contained in the modifier field and the associated modifier field format, or syntax. The modifier lists in this clause show the modifier type in terms of a Type Code, which is defined in table 16-1.

Table 16-1. Modifier type codes and typical formats

Item	Modifier type	Type code	Examples
1	Numeric		
1.1	Decimal (or real)	R	VOLTAGE 5.5 V, CURRENT MAX 2.0 A
1.2	Integer	I	INDEX 5, BURST-COUNT 5 PULSES
1.3	Digital (bit or char)	B	STIM B'101100'
1.4	Decimal (or real) range	RR	FREQ-WINDOW RANGE 1.0 HZ TO 1.0 KHZ
2	Array reference		
2.1	Decimal (or real)	RA	STIM 'WAVEFORM' (1,2,3)
2.2	Integer	IA	PROVE,..., ERROR-INDEX 'X' (2 THEN 5 THEN 6)
2.3	Digital (bit or char)	BA	STIM 'PATTERN' (1 THRU 5)
3	Modifier descriptor	MD	MODE ACTIVE
4	Mnemonic only	MO	PULSE-SPECT

c) Modifier usage

Nouns are generally applicable to both UUT stimulus and response functions. For example, their modifiers may be used in the <require control>, <require capability>, or <require limit> clauses, as the measured characteristic mnemonic in the <require statement>, as statement characteristics, and as the measured characteristic mnemonic in the test statements.

Usage options for individual modifiers are shown in the modifier lists of this clause by the following usage codes:

STIMULUS (S) Modifier applicable for UUT stimulus functions. Valid when the associated noun is used with SOURCE resource types, except as specifically noted.

RESPONSE (R) Modifier applicable as statement characteristics for UUT response functions. Valid when the associated noun is used with SENSOR resource types, except as specifically noted.

MEASUREMENT (M) Modifier applicable as a measured characteristic. Valid when the associated noun is used with SENSOR resource types, except as specifically noted.

d) Dimensions and basic units

The modifier lists in this clause specify dimensions for each modifier in terms of one or more quantities, which corresponds to one or more families of basic units. Entries in the dimensions column are either a basic unit or a quantity entry as listed in table 15-1. Any of the associated basic units are valid and any of their associated dimensional mnemonics are permitted. For example, if the basic unit of hertz is listed, then the following mnemonics are permitted: HZ, KHZ, MHZ, and GHZ; if the quantity of frequency is listed, then the basic units of hertz and pulses per second are both allowed, and the following mnemonics are permitted: HZ, KHZ, MHZ, GHZ, PPS, and KPPS.

When no dimensions are specified for a given modifier, the units are considered to be null (i.e., dimensionless). Units formed by the division of one basic unit by another are shown in the modifier lists as needed.

16.1 AC SIGNAL (alternating current signal)

16.1.1 Definition

A sinusoidal time-varying electrical potential.

16.1.2 Formal syntax

16.1.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Aging rate	AGE-RATE	I	SRM	Frequency/time
AM component	AM-COMP	R	-RM	Voltage or ratio
Bandwidth	BANDWIDTH	R	SRM	Frequency
Burst length	BURST	I	SR-	Burst length
Crest factor	CREST-FACTOR	R	-RM	Ratio
Current	CURRENT	R	SRM	Current
DC offset	DC-OFFSET	R	SRM	Voltage, Current
Total distortion	DISTORTION	R	-RM	Ratio, voltage, current or power
FM component	FM-COMP	R	-RM	Frequency or ratio
Frequency	FREQ	R	SRM	Frequency
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Harmonic distortion	HARMONICS	R	SRM	Voltage, current, power or ratio
Harmonic phase	HARM-***-PHASE	R	-RM	Angle, plane
Harmonic power	HARM-***-POWER	R	-RM	Power
Harmonic voltage	HARM-***-VOLTAGE	R	-RM	Voltage or ratio
Noise	NOISE	R	SRM	Voltage, current, power or ratio
Noise amplitude	NOISE-AMPL-DENS	R	SRM	Voltage/frequency
Noise power density	NOISE-PWR-DENS	R	SRM	Power/frequency
Non-harmonic distortion	NON-HARMONICS	R	-RM	Voltage, current, power or ratio
Period	PERIOD	R	SRM	Time
Phase angle	PHASE-ANGLE	R	SRM	Angle, plane
Phase jitter	PHASE-JIT	R	-RM	Angle, plane
Power	POWER	R	SRM	Power
Reference frequency	REF-FREQ	R	SRM	Frequency
Reference power	REF-POWER	R	SRM	Ratio, power
Reference voltage	REF-VOLT	R	SRM	Voltage
Standing wave ratio	SWR	R	SRM	Ratio
Delta-conn.	THREE-PHASE-DELTA	MO	SR-	(None)
Wye-conn.	THREE-PHASE-WYE	MO	SR-	(None)
Voltage	VOLTAGE	R	SRM	Voltage

16.1.4 Rules

1. (***) An <unsigned integer number> shall be included at this point indicating the harmonic which is being specified.

2. If decibel or percent is used, the figure is referenced to the fundamental amplitude.

16.2 ADF (automatic direction finder)

16.2.1 Description

A direction-finder that automatically and continuously provides a measure of the direction of arrival of the received signal.

16.2.2 Formal syntax

16.2.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Burst length	BURST	I	SR-	Burst length
Carrier amplitude	CAR-AMPL	R	SRM	Voltage or power
Carrier frequency	CAR-FREQ	R	SRM	Frequency
Field strength	FIELD-STRENGTH	R	SR-	Voltage/distance
Harmonic distortion	HARMONICS	R	SRM	Voltage or ratio
Modulation	MOD-AMPL	R	SRM	Voltage or ratio
Modulation	MOD-FREQ	R	SRM	Frequency
Noise	NOISE	R	-RM	Voltage or ratio
Non-harmonic distortion	NON-HARMONICS	R	-RM	Voltage or ratio
Relative bearing	REL-BEARING	R	SRM	Angle, plane
Relative bearing rate	REL-BEARING-RATE	R	SRM	Velocity, angular

16.2.4 Rules

16.2.5 Notes and examples

The ADF is a radiogoniometer consisting of a receiver and an antenna. Testing of the receiver requires an antenna simulator emitting a stimulus composed of three signals simulating the three outputs of the antennas (two loop antennas and one sensor antenna), immersed in an electromagnetic field of specified strength, with these three antennas receiving the HF signal with a specified bearing. The governing document is ARINC 570.

```
APPLY, ADF, CAR-FREQ 1 MHZ, FIELD-STRENGTH 3 UV/M,
     MOD-AMPL 30 PC, MOD-FREQ 1000 HZ,
     REL-BEARING 30 DEG,  CNX ... $
```

16.3 AM SIGNAL (amplitude modulation signal)

16.3.1 Definition

A continuous sinusoidal wave (carrier) whose amplitude is varied as a function of the instantaneous value of a second wave (modulating).

16.3.2 Formal syntax

16.3.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Burst length	BURST	I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	S--	Frequency
Carrier amplitude	CAR-AMPL	R	SR-	Ratio, voltage or power
Carrier frequency	CAR-FREQ	R	SR-	Frequency
Carrier harmonic distortion	CAR-HARMONICS	R	SRM	Ratio, voltage, current or power
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Modulation amplitude	MOD-AMPL	R	SRM	Voltage or ratio
Modulation distortion	MOD-DIST	R	-RM	Ratio
Modulation frequency	MOD-FREQ	R	SR-	Frequency
Noise	NOISE	R	SRM	Voltage, power or ratio
Noise amplitude density	NOISE-AMPL-DENS	R	SRM	Voltage/frequency
Noise power density	NOISE-PWR-DENS	R	SRM	Power/frequency
Non-harmonic distortion	NON-HARMONICS	R	-RM	Voltage, current, power or ratio
Phase jitter	PHASE-JIT	R	-RM	Ratio or angle, plane
Power	POWER	R	SRM	Power
Standing wave ratio	SWR	R	SRM	Ratio

16.4 AMBIENT CONDITIONS

16.4.1 Definition

The characteristics of the local environment (for example, temperature, humidity, pressure). See the noun HEAT for UUT applications.

16.4.2 Formal syntax

16.4.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Barometric pressure	BAROMETRIC-PRESS	R	SRM	Pressure
Debris count	DEBRIS-COUNT	R	S--	Events/volume
Debris size	DEBRIS-SIZE	R	S--	Distance
Dew point temperature	DEWPOINT	R	-RM	Temperature
Humidity	HUMIDITY	R	SRM	Mass/volume

continued

Modifier	Mnemonic	Type	Usage	Quantity
Operating temperature	OPER-TEMP	R	SRM	Temperature
Relative humidity	RELATIVE-HUMIDITY	R	SRM	Ratio
Wind direction	RELATIVE-WIND	R	SRM	Angle, plane
Specification temperature	SPEC-TEMP	R	SRM	temperature
Temperature	TEMP	R	SRM	temperature
Capacitance temperature coefficient	TEMP-COEFF-CAP	R	SRM	Ratio/temperature
Current temperature coefficient	TEMP-COEFF-CURRENT	R	SRM	Ratio/temperature
Inductance temperature coefficient	TEMP-COEFF-IND	R	SRM	Ratio/temperature
Reactance temperature coefficient	TEMP-COEFF-REACT	R	SRM	Ratio/temperature
Resistance temperature coefficient	TEMP-COEFF-RES	R	SRM	Ratio/temperature
Voltage temperature coefficient	TEMP-COEFF-VOLT	R	SRM	Ratio/temperature
Wind speed	WIND-SPEED	R	SRM	Velocity, linear or distance/time

16.4.4 Rules

This noun uses the pin descriptor TO and may be followed by the special <connection> ATMOS.

16.4.5 Notes and examples

The following are examples of AMBIENT CONDITIONS statements:

Example 1:

```
111122 MEASURE, (RELATIVE-HUMIDITY INTO 'HUMVAL'),
       AMBIENT CONDITIONS,
       RELATIVE-HUMIDITY RANGE 20 PC TO 100 PC,
       CNX TO CELL-AMB-HUM $
```

Example 2:

```
111123 MEASURE, (WIND-SPEED INTO 'WSPEED'),
       AMBIENT CONDITIONS,
       WIND-SPEED MAX 25 KT,
       CNX TO SPD-SEN $
```

Example 3:

```
101000 APPLY, AMBIENT CONDITIONS,
       TEMP 60 DEGC, RELATIVE-HUMIDITY 80 PC,
       CNX TO ATMOS $
```

16.5 ATC (air traffic control)

16.5.1 Definition

The ATC transponder is an on-board device that is used in conjunction with ground-based radar installations. The governing documents are ARINC 572 and ARINC 711.

The ATC receives information from the radar on a specific frequency (1030 MHz) and responds with coded signals on another frequency (1090 MHz). The radar-generated interrogations are composed of three pulses, called P1, P2, and P3, respectively (see figure 16-1). The normal spacing between P1 and P2 is 2 µs. Normal spacing between P1 and P3 depends on the choice of mode (see rule 3, below).

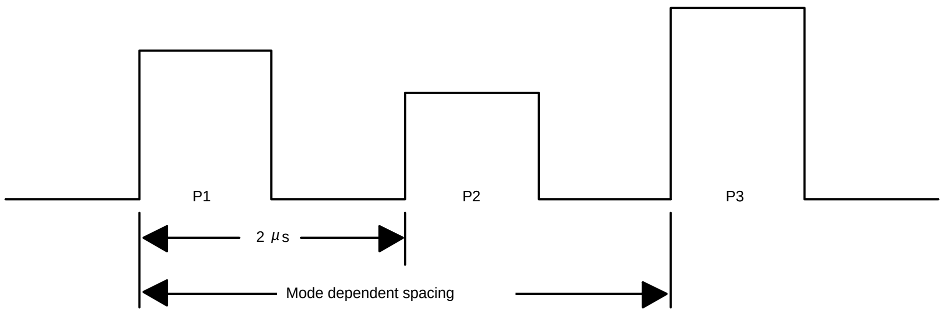
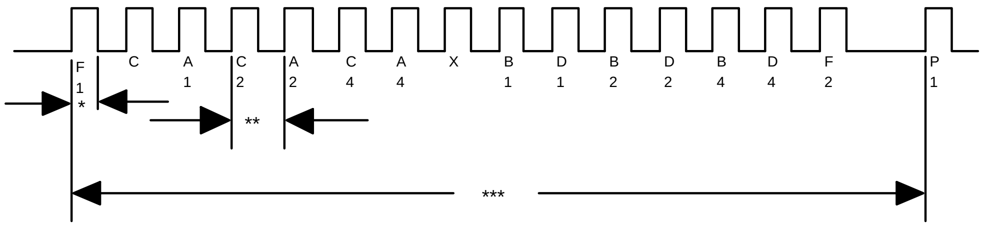


Figure 16-1. ATC interrogation signal

The ATC reply to these interrogations consists of an encoded pulse train, as shown in figure 16-2. Each pulse train consists of a start pulse followed by a given number of data pulses, the number and position of these data pulses (after the start pulse) being selectable on a control panel. A stop pulse is the final pulse in the train. There are 16 pulse positions (including start and stop).



NOTES

*   0.45 µs
**   1.45 µs
*** 24.65 µs

Figure 16-2. ATC reply signal

16.5.2 Formal syntax

16.5.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Carrier amplitude	CAR-AMPL	R	SRM	Voltage or power
Carrier frequency	CAR-FREQ	R	SRM	Frequency
Double interrogation	DBL-INT	R	S--	Time
Fall time	FALL-TIME	R	SRM	Time
Harmonic voltage	HARM-***-VOLTAGE	R	-RM	Voltage or ratio
Interrogation jitter	INT-JITTER	R	S--	Time or ratio
Interrogation rate	INT-RATE	R	S--	Frequency
Mode	MODE	MD	S--	(None)
P3 deviation	P3-DEV	R	S--	Time
P3 level	P3-LEVEL	R	S--	Ratio
Pulse pair droop	PAIR-DROOP	R	-RM	Ratio
Pulse identification	PULSE-IDENT	I	-R-	(None)
Pulse position of	PULSE-POSN-x	R	-RM	Time
Pulse spectrum measurement	PULSE-SPECT	MO	--M	(None)
Pulse spectrum measurement threshold	PULSE-SPECT-THRESHOLD	R	-R-	Voltage
Pulse width	PULSE-WIDTH	R	SRM	Time
Pulses excluded	PULSES-EXCL	MD	S--	(None)
Pulses included in	PULSES-INCL	MD	S--	(None)
Reply efficiency	REPLY-EFF	R	SRM	Ratio
Response list for	RESP	RA	-R-	(None)
Ringing	RINGING	R	SRM	Ratio
Rise time	RISE-TIME	R	SRM	Time
SLS deviation	SLS-DEV	R	S--	Time
SLS level	SLS-LEVEL	R	S--	Ratio
Standing wave ratio	SWR	R	SR-	Ratio
Data word contents	VALUE	B	--M	(None)
Noise	NOISE	R	-RM	Ratio, voltage or power

16.5.4 Rules

1. For modifier MODE, the following modifiers may be included:

PULSES-INCL:	ALL, or i-j-k....., where I, J, K, ...represent integer literals
PULSES-EXCL:	All, or I, J, K, ..., where I, J, K, ...represent integer literals

2. When testing the ATC transponder, modes A, B, C, or D may be activated independently or modes A, B, or D may be interlaced with mode C. The MODE defines the spacing between the P1 and P3 pulses.

3. When MODE is used:

Mod desc.	Mode name or meaning	Pulse P1/P3 spacing
A	Identification	8 +-0.2 µs
B	Identification (non U.S.)	17 +-0.2
C	Altitude	21 +-0.2
D	(Unassigned)	25 +-0.2
A-C	Alternating modes A and C	

4. When SLS-DEV or SLS-LEVEL is used, the side lobe suppression pulse is normally referred to as P2.

16.6 COMMON

16.6.1 Definition

A conducting body used as a common return for an electrical circuit and as an arbitrary zero potential.

16.6.2 Formal syntax

16.6.3 Noun modifier set

There are no modifiers for COMMON.

16.6.4 Rules

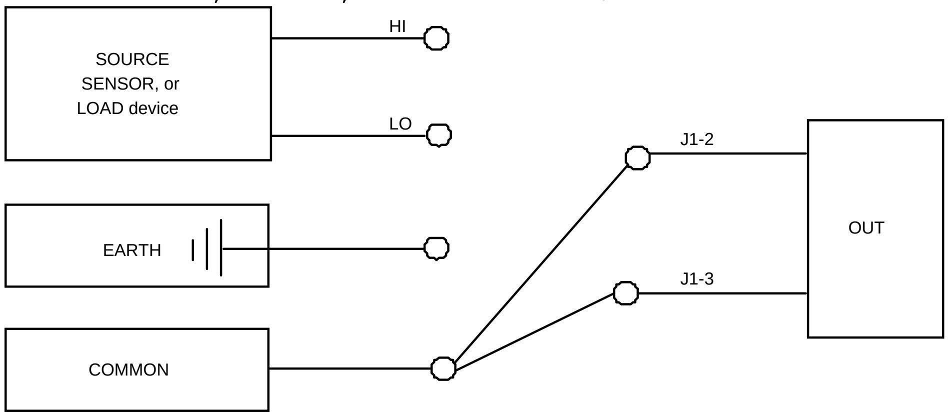
COMMON and EARTH are defined and listed both in the noun list and the <conn set>. COMMON and EARTH are phenomena that exist in the test environment and shall be discretely attached to the UUT in order to have an effect on the UUT. If COMMON or EARTH is used in the noun field of a source or load type statement, it implies that the function is to be attached to the specified UUT pins. If COMMON or EARTH is used in the Connector field of some other source, sensor, or load type function, it implies that the specified load of the ATE device is to be attached to the ATE COMMON or EARTH function.

16.6.5 Notes and examples

The following examples clarify the proper use of EARTH and COMMON. In the diagram associated with each sample statement, the line represents the connection caused by the statement(s).

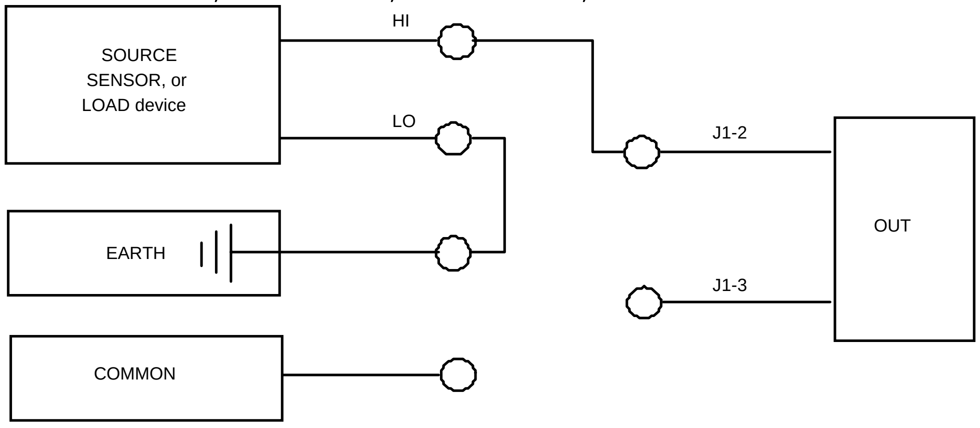
Example 1:

```
APPLY, COMMON, CNX J1-2 J1-3 $
```



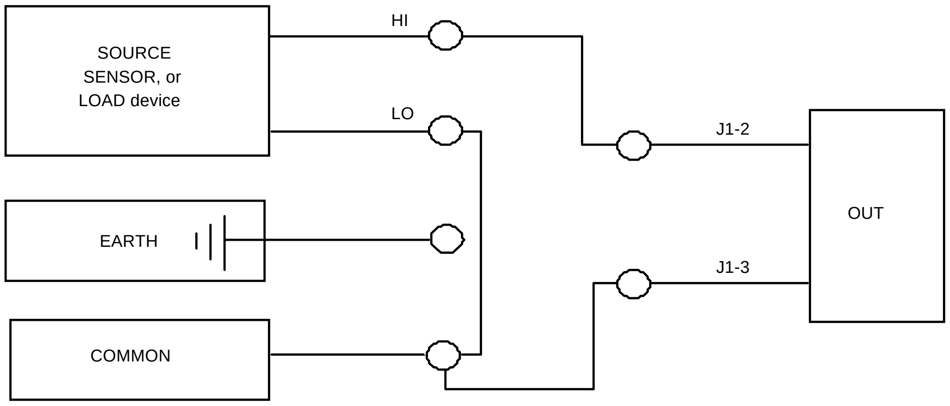
Example 2:

```
APPLY, DC SIGNAL, VOLTAGE 5V, CNX HI J1-2 LO EARTH  $
```



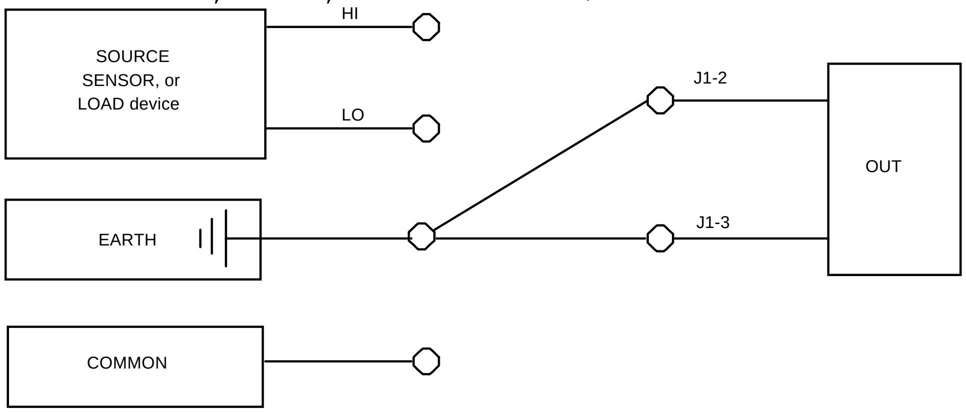
Example 3:

```
        APPLY, COMMON, CNX J1-3 $
        MEASURE, (VOLTAGE INTO 'SAVE'), AC SIGNAL,
             ...CNX HI J1-2 LO COMMON $
```



Example 4:

```
        APPLY, EARTH, CNX J1-2 J1-3 $
```



16.7 COMPLEX SIGNAL

16.7.1 Definition

A COMPLEX SIGNAL is the resultant signal created by operation(s) upon a set of input signals.

16.7.2 Formal syntax

16.7.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Amplitude	AMPL	R	SRM	Voltage
Average sweeps	AVG-SWEEPS	I	-R-	(None)
Bandwidth	BANDWIDTH	RR	-R-	Frequency
Burst length	BURST	I	S--	Burst length
Burst repetition rate	BURST-REP-RATE	R	S--	Frequency
Frequency	FREQ	R	SRM	Frequency
Frequency difference	FREQ-DELTA	R	--M	Frequency
Frequency resolution	FREQ-RESOLUTION	R	SR-	Frequency
Intermediate frequency	INTERMEDIATE-FREQ	R	SR-	Frequency
Marker frequency	MARKER-FREQ	R	SR-	Frequency
Mask	MASK	B/BA	-R-	(None)
Maximum power	MAX-POWER	R	-RM	Power
Maximum sweeps	MAX-SWEEPS	I	-R-	(None)
Maximum voltage	MAX-VOLTAGE	R	--M	Voltage
Minimum power	MIN-POWER	R	--M	Power
Minimum sweeps	MIN-SWEEPS	I	-R-	(None)
Minimum voltage	MIN-VOLTAGE	R	--M	Voltage
Negative sampling	NEG-SAMPLING	MO	-R-	(None)
Noise ampli density	NOISE-AMPL-DENS	R	SRM	Voltage/frequency
Noise power density	NOISE-PWR-DENS	R	SRM	Power/frequency
Pattern correlation	BIT-MATCH-THRESHOLD	I	-R-	Digital length
Peak to peak sampling	PP-SAMPLING	MO	-R-	(None)
Phase jitter	PHASE-JIT	R	-RM	Angle (plane) or ratio
Positive sampling	POS-SAMPLING	MO	-R-	(None)
Power	POWER	R	SRM	Power
Power difference	POWER-DELTA	R	--M	Power
Power ratio	POWER-RATIO	R	--M	Ratio
Power source	POWER-SOURCE	MO	S--	(None)
Reference pattern	REF-PATTERN	B/BA	-R-	(None)
Reference pattern size	REF-PATTERN-SIZE	I	-R-	Digital length
Resolution bandwidth	RESOLUTION-BANDWIDTH	R	SR-	Frequency
Response list	RESP	RA	--M	(None)
Signal search	SIGNAL-SEARCH	MO	--M	(None)
Signal threshold	SIGNAL-THRESHOLD	R	-R-	Power or voltage
Signal to noise ratio	SNR	R	SRM	Ratio
Spectrum	SPECTRUM	MO	--M	(None)
Standing wave ratio	SWR	R	SRM	Ratio
Sweep time	SWEEP-TIME	R	-R-	Time
Tune frequency	TUNE-FREQ	R	SR-	Frequency
Voltage	VOLTAGE	R	SRM	Voltage
Zero power	ZERO-POWER	R	--M	Power

16.7.4 Rules

1. Modifiers of 16.7.3 are applicable to the complete COMPLEX SIGNAL. The characteristics of each of the component signals (e.g., carrier and modulating signals) are specified in the separate SPECIFY statements within a DEFINE COMPLEX SIGNAL structure.

2. When the measured characteristics SPECTRUM or SIGNAL-SEARCH are indicated, the specified number of measured values are stored in the array specified by the RESP modifier. The verb VERIFY cannot be used with these measured characteristics.

16.7.5 Notes and examples

See <define complex signal structure>.

16.8 DC SIGNAL (direct current signal)

16.8.1 Definition

An unvarying electrical potential.

16.8.2 Formal syntax

16.8.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
AC component	AC-COMP	R	SRM	Voltage, current
AC component frequency	AC-COMP-FREQ	R	SRM	Frequency
DC current	CURRENT	R	SRM	Current
Total distortion	DISTORTION	R	-RM	Voltage, ratio or current
Noise	NOISE	R	SRM	Voltage, current, Ratio or power
Power	POWER	R	SRM	Power
Sample width (duration)	SAMPLE-WIDTH	R	-R-	Time
Voltage	VOLTAGE	R	SRM	Voltage

16.8.4 Rules

1. SAMPLE-WIDTH is an optional modifier used to indicate restrictions on the sampling aperture time for dc measurement (for example when sampling a time-varying waveform). Normal usage will be with MAX, to indicate that a longer sampling aperture will "smear" the measurement excessively. Normal usage with MIN refers to the minimum averaging time or filter time constant for a noisy signal.

2. In DC SIGNAL, SOURCE statements, AC-COMP MAX refers to the maximum allowable ripple on the source. The actual value may be anything from 0 to AC-COMP MAX. AC-COMP (no MAX) refers to the actual required value of ripple on the source.

16.9 DISPLACEMENT

16.9.1 Definition

The magnitude of position change. See the noun ROTATION for angle constructs.

16.9.2 Formal syntax

16.9.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Distance	DISTANCE	R	SRM	Distance

16.10 DME (distance measuring equipment)

16.10.1 Definition

A radio aid to air navigation that provides distance information by measuring the time of transmission from an interrogator to a transponder and return.

16.10.2 Formal syntax

16.10.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Carrier amplitude	CAR-AMPL	R	SR-	Voltage or power
Carrier frequency	CAR-FREQ	R	SR-	Frequency
Channel	CHANNEL	MD#	S--	(None)
Fall time	FALL-TIME	R	SRM	Time
Frequency pairing	FREQ-PAIRING	R	S--	Frequency
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Identification signal	IDENT-SIG	MD#	SRM	(None)
Identification signal	IDENT-SIG-FREQ	R	SRM	Frequency
Interrogation jitter	INT-JITTER	R	SRM	Time or ratio
Interrogation rate	INT-RATE	R	-RM	Frequency
Modulation amplitude	MOD-AMPL	R	SRM	Ratio
Pair droop	PAIR-DROOP	R	-RM	Ratio
Pair spacing	PAIR-SPACING	R	-RM	Time
Pulse spectrum meas.	PULSE-SPECT	MO	--M	(None)
Pulse spectrum measurement threshold	PULSE-SPECT-THRESHOLD	R	-R-	Voltage
Pulse width	PULSE-WIDTH	R	SRM	Time
Range echo pulse	RANGE- PULSE-ECHO	R	S--	Distance or time
Range echo pulse power	RANGE- PULSE-DEV	R	S--	Ratio

continued

Modifier	Mnemonic	Type	Usage	Quantity
Reply efficiency	REPLY-EFF	R	SRM	Ratio
Response list name for pulse spectrum meas.	RESP	RA	-R-	(None)
Ringing	RINGING	R	SRM	Ratio
Rise time	RISE-TIME	R	SRM	Time
Slant range	SLANT-RANGE	R	SRM	Distance
Slant range acceleration	SLANT-RANGE-ACCEL	R	SRM	Distance/time/time
Slant range rate of change	SLANT-RANGE-RATE	R	SRM	Distance/time
Squitter pulse distribution	SQTR-DIST	R	S--	Ratio
Squitter pulse rate	SQTR-RATE	R	S--	Frequency
Standing wave ratio	SWR	R	SR-	Ratio

16.10.4 Rules

The # of Modifier Descriptors are as follows:

a) CHANNEL: i-X or i-Y, where i represents an integer literal

b) IDENT-SIG: Station identification letters

16.11 DOPPLER

16.11.1 Definition

A response signal in the form of a band-limited spectrum having a specified frequency offset from a transmitted (source) reference frequency. The offset is proportional to the time rate of change in the effective path length between a transmitter (source) of the reference frequency and the receiver (point of observation).

16.11.2 Formal syntax

16.11.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Burst length	BURST	I	SR-	Burst length
Reference carrier	CAR-FREQ	R	SR-	Frequency
Current	CURRENT	R	SRM	Current
Doppler frequency	DOPPLER-FREQ	R	SR-	Frequency
Doppler bandwidth	DOPPLER-BANDWIDTH	R	SR-	Frequency or ratio
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Power	POWER	R	SRM	Power
Voltage	VOLTAGE	R	SRM	Voltage

16.11.4 Rules

The reference carrier is identified by REF in the CNX field.

16.12 EARTH

16.12.1 Definition

The earth-ground electrical reference.

16.12.2 Formal syntax

16.12.3 Noun modifier set

There are no modifiers for EARTH.

16.12.4 Rules

COMMON and EARTH are defined and listed both in the noun list and the <conn set>. COMMON and EARTH are phenomena that exist in the test environment and shall be discretely attached to the UUT in order to have an effect on the UUT. If COMMON or EARTH is used in the noun field of a source or load type statement, it implies that the function is to be attached to the specified UUT pins. If COMMON or EARTH is used in the connector field of some other source, sensor, or load type function, it implies that the specified load of the ATE device is to be attached to the ATE COMMON or EARTH function.

16.12.5 Notes and examples

See the noun COMMON for examples.

16.13 EM FIELD (electro magnetic field)

16.13.1 Definition

A state produced by current flow in a conductor or by a permanent magnet that can induce a voltage in a second conductor when the state changes or when relative motion occurs.

16.13.2 Formal syntax

16.13.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Flux density	FLUX-DENS	R	SRM	Flux density

16.13.4 Rules

For the modifier FLUX-DENS, either of the suffixes -QUAD or -IN-PHASE may be used.

16.14 EVENTS

16.14.1 Definition

An event is identified as that instant in time when a specified set of conditions is satisfied.

16.14.2 Formal syntax

16.14.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Count	COUNT	I	-RM	Events
Slope, negative	NEG-SLOPE	MO	-R-	(None)
Noise	NOISE	R	-R-	Ratio or voltage
Slope, positive	POS-SLOPE	MO	-R-	(None)
Time	TIME	R	-R-	Time
Trigger	TRIG	R	-R-	Voltage
Voltage	VOLTAGE	R	-R-	Voltage

16.14.4 Rules

1. This single statement method for determining the number of events in a given time period requires that the following items of information be supplied:

a) The maximum TIME taken to observe the events.

b) Voltage level that shall be present to indicate an event (TRIG xV).

c) The slope of the signal, either NEG-SLOPE or POS-SLOPE, which indicates the direction of change of the triggering voltage.

d) An optional VOLTAGE MAX field may be added for voltage magnitude scaling.

2. The modifier TRIG used in conjunction with other optional modifiers (e.g., POS-SLOPE or NEG- SLOPE) defines an event.

16.14.5 Notes and examples

The number of pulse counts over a period of time may be measured by making use of the EVENTS noun.

```
MEASURE, (COUNT INTO 'SMW'), EVENTS,
  COUNT MAX 2052 TIMES, VOLTAGE MAX 2V,
  TRIG 2 MV, POS-SLOPE, CNX HI J1 LO J2 $
```

16.15 FLUID SIGNAL

16.15.1 Definition

A signal transmitted to or from the UUT by means of a fluid.

16.15.2 Formal syntax

16.15.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Debris count	DEBRIS-COUNT	R	S--	Events/volume
Debris size	DEBRIS-SIZE	R	S--	Distance
Dew point temperature	DEWPOINT	R	-RM	Temperature
Fluid type	FLUID-TYPE	MD#	SR-	(None)
Mass flow rate	MASS-FLOW	R	SRM	Mass/time
Absolute pressure	P-PRESS-A*	R	SRM	Pressure
Gauge pressure	P-PRESS-G*	R	SRM	Pressure
Rate of change of pressure	ts-PRESS-RATE*	R	S--	Pressure/time
Specific gravity	SPEC-GRAV	R	SRM	(None)
Temperature	ts-TEMP	R	SRM	Temperature
Volume flow	VOLUME-FLOW	R	SRM	Volume/time

16.15.4 Rules

1. This noun uses only the pin descriptors TO or TO and FROM.

2. The # of Modifier Descriptors for FLUID-TYPE are as follows:

AIR, FUEL, GAS, N2, OIL, WATER, DTD-n, MIL-n where n represents an integer.

3. (*) The following prefixes (P) may be used (see 17.2):

a) TOTAL

b) STATIC

c) none (assume STATIC)

4. The specification of both a pressure and a pressure rate in a single stimulus statement places restrictions on the manner in which the specified values are achieved. See rule 2 under the noun ROTATION.

16.15.5 Notes and examples

The following are example FLUID SIGNAL statements:

Example 1: To measure the compressor inlet pressure

```
000401 MEASURE, (TOTAL-PRESS-A INTO 'MPW'), FLUID SIGNAL,
          FLUID-TYPE AIR, TOTAL-PRESS-A MAX 30.00 INHG,
          CNX TO PT2 $
```

Example 2: To measure the fuel flow from the fuel control

```
000402 MEASURE, (MASS-FLOW INTO 'MPW'), FLUID SIGNAL,
          FLUID-TYPE FUEL, MASS-FLOW MAX 10000 MG/SEC,
          CNX TO WF $
```

Example 3: To measure exhaust gas temperature

```
000403 MEASURE, (TOTAL-TEMP INTO 'MPW'), FLUID SIGNAL,
          FLUID-TYPE GAS, TOTAL-TEMP MAX 775 DEGK,
          CNX TO TT7 $
```

Example 4: To apply engine starter air pressure

```
 000404 APPLY, FLUID SIGNAL, FLUID-TYPE AIR,
           PRESS-G 250 INHG,
           CNX TO START-AIR-INPUT $
```

16.16 FM SIGNAL (frequency modulation signal)

16.16.1 Definition

A continuous sinusoidal wave (carrier) generated when the frequency of one wave is varied in accordance with the amplitude of another wave (modulating).

16.16.2 Formal syntax

16.16.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Burst length	BURST	I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	S--	Frequency
Carrier amplitude	CAR-AMPL	R	SR-	Voltage, power or ratio
Carrier frequency	CAR-FREQ	R	SRM	Frequency
Carrier harmonic distortion	CAR-HARMONICS	R	SRM	Voltage, power or ratio
Deviation ratio (FM index)	FREQ-DEV	R	SRM	Ratio
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Modulation amplitude	MOD-AMPL	R	SRM	Voltage or ratio
Modulation distortion	MOD-DIST	R	-RM	Ratio, voltage or power
Modulation frequency	MOD-FREQ	R	SRM	Frequency
Noise	NOISE	R	SRM	Voltage, power or ratio
Noise amplitude density	NOISE-AMPL-DENS	R	SRM	Voltage/frequency
Noise power density	NOISE-PWR-DENS	R	SRM	Power/frequency
Non-harmonic distortion	NON-HARMONICS	R	-RM	Voltage, power or ratio
Power	POWER	R	SRM	Power
Standing wave ratio	SWR	R	SRM	Ratio

16.17 HEAT

16.17.1 Definition

The molecular kinetic energy condition of a body or bodies that determines the transfer of heat to or from the UUT. See the noun AMBIENT CONDITIONS for environmental applications.

16.17.2 Formal syntax

16.17.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Temperature	TEMP	R	SRM	Temperature

16.18 IFF (identification, friend or foe)

16.18.1 Definition

A complex signal used for automatic identification between two stations.

16.18.2 Formal syntax

16.18.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Carrier amplitude (P1 pulse)	CAR-AMPL	R	SRM	Voltage or power
Carrier frequency	CAR-FREQ	R	SRM	Frequency
Double interrogation	DBL-INT	R	S--	Time
Fall time	FALL-TIME	R	SRM	Time
Harmonic voltage	HARM-***-VOLTAGE	R	-RM	Voltage or ratio
Interrogation jitter	INT-JITTER	R	S--	Time or ratio
Interrogation rate	INT-RATE	R	S--	Frequency
Mode	MODE	MD#	S--	(None)
P3 deviation	P3-DEV	R	S--	Time
P3 level	P3-LEVEL	R	S--	Ratio or power
Pulse pair droop	PAIR-DROOP	R	-RM	Ratio
Antenna power differential	POWER-DIFF	R	SRM	Ratio
Pulse identification	PULSE-IDENT	I	-R-	(None)
Pulse position of pulse x	PULSE-POSN-x	R	-RM	Time
Pulse spectrum measurement	PULSE-SPECT	MO	--M	(None)
Pulse spectrum measurement threshold	PULSE-SPECT-THRESHOLD	R	-R-	Voltage
Pulse width	PULSE-WIDTH	R	SRM	Time
Pulses included	PULSES-INCL	MD#	S--	(None)
Pulses excluded	PULSES-EXCL	MD#	S--	(None)
Reply efficiency	REPLY-EFF	R	SRM	Ratio
Response list name for pulse spectrum measurement	RESP	RA	-R-	(None)
Ringing	RINGING	R	SRM	Ratio
Rise time	RISE-TIME	R	SRM	Time
Side lobe suppression deviation	SLS-DEV	R	S--	Time
Side lobe suppression level	SLS-LEVEL	R	S--	Ratio
Standing wave ratio	SWR	R	SR-	Ratio
Data word contents or value	VALUE	B	--M	(None)
Noise	NOISE	R	-RM	Voltage, power or ratio

16.18.4 Rules

1. When testing the IFF transponder, only modes 1, 2, 3-A, 3-C, 4, or C are permissible. The mode defines the spacing between P1 and P3 pulses.

2. When MODE 3-C is specified, it implies testing is to be done alternately in mode 3-A and mode C, respectively.

3. The P3 level is referenced to the amplitude of P1.

4. For modifier MODE, the following modifiers may be included:

PULSES-INCL:	ALL, or i-j-k....., where I, J, K, ...represent integer literals
PULSES-EXCL:	All, or I, J, K, ..., where I, J, K, ...represent integer literals

5. When MODE is used:

Mod desc.	Mode name or meaning	Pulse P1/P3 spacing
1	Traffic, beacon assist and general information	3 +-0.2 µs
2	Personal identification	5 +-0.2
3-A	Traffic identification	8 +-0.2
C	Altitude	21 +-0.2
4	Secure IFF	Special purpose
3-C	Alternating modes 3-A and C	

6. When SLS-DEV or SLS-LEVEL are used

a) For Modes 1, 2, 3-A, and C, the Side Lobe Suppression pulse is referred to as P2.

b) For Mode 4, the Side Lobe Suppression pulse is referred to as P5.

7. (***) An <unsigned integer number> shall be included at this point indicating the harmonic that is being specified.

16.18.5 Notes and examples

1. IFF (identification, friend or foe) is an airborne transponder system used for identification in an air traffic control system. A ground station interrogates the aircraft with a pulse coded signal at 1030 MHz and, if properly interrogated, the aircraft replies with a pulse coded signal at 1090 MHz. The governing document is MIL-A-28826.

2. All of the interrogation pulses are transmitted by a directional antenna except the sidelobe suppression pulse, which is radiated by an omnidirectional antenna. This permits discrimination of sidelobes of the interrogating antenna. When replying to a Mode 1, 2, or 3 interrogation, the IFF provides information by means of the specific pulses included between two "framing" pulses. Combined modes 3 and C provide normal mode 3 information interleaved with mode C information derived from the aircraft air data computer. A mode 4 reply only occurs if the IFF has been properly interrogated by an enciphered pulse-code string.

3. Special modes of operation include identity, which adds an extra pulse to the reply for 20

seconds when the IDENT switch is actuated on the aircraft control panel, and Emergency, which transmits a unique sequence of pulse strings.

16.19 ILS (instrument landing system)

16.19.1 Definition

An internationally adopted instrument landing system for aircraft, consisting of a VHF localizer, a UHF glide slope, and 75 MHz markers as defined by ARINC Specification 578.

16.19.2 Formal syntax

16.19.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Carrier amplitude	CAR-AMPL	R	S--	Voltage or power
Carrier frequency	CAR-FREQ	R	S--	Frequency
Difference in depth of modulation	DDM	R	S--	(None)
Dominant modulating signal	DOMINANT-MOD- SIG	R	S--	Frequency
Frequency pairing	FREQ-PAIRING	R	S--	Frequency
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Subsystem identifier	GLIDE-SLOPE	MO	S--	(None)
Harmonic distortion	HARMONICS	R	-RM	Voltage, power or ratio
150 Hz signal	HI-MOD-FREQ	R	S--	Frequency
Identification signal	IDENT-SIG	MD#	S--	(None)
Identification signal frequency	IDENT-SIG-FREQ	R	S--	Frequency
Modulation of identification signal	IDENT-SIG-MOD	R	S--	Ratio
90 Hz signal	LO-MOD-FREQ	R	S--	Frequency
Subsystem identifier	LOCALIZER	MO	S--	(None)
Subsystem identifier	MARKER-BEACON	MO	S--	(None)
Mean modulation of the 90 and 150 Hz signals	MEAN-MOD	R	-RM	Ratio
Modulation amplitude	MOD-AMPL	R	SRM	Ratio
Modulating distortion	MOD-DIST	R	-RM	Voltage, power or ratio
Non-harmonic distortion	NON-HARMONICS	R	-RM	Voltage, power or ratio
150/90 Hz signal phase angle	PHASE-ANGLE	R	S--	Angle, plane

16.19.4 Rules

1. The modifier PHASE-ANGLE is the angle between the nominally coincident positive zero- crossings of the 150 Hz modulation signal and the 90 Hz modulating signal, and is expressed in degrees of the 150 Hz signal. The angle is positive when the 150 Hz signal leads the 90 Hz signal.

2. The # of Modifier Descriptors are as follows:

IDENT-SIG: Station identification letters

16.19.5 Notes and examples

A continuous-wave amplitude-modulated horizontally-polarized fixed-beam system, aids aircraft approaching a runway. This system is comprised of the following, which are explained in detail below.

a) A UHF glide slope to give elevation guidance about an optimum approach angle, with an operating frequency in the band 328.6-335.4 MHz.

b) A VHF localizer to give azimuth guidance with reference to the runway center line, with an operating frequency in the band 108.0-112.0 MHz.

c) Two or three marker beacons operating at 75 MHz to give range with reference to the touch-down point.

d) High intensity strobe lights to provide visual guidance toward touch-down.

NOTE The governing reference document is ARINC 578.

The glide slope

The glide slope is the vertical guidance portion of the ILS. At present, 40 glide-slope channels exist with 150 kHz channel separation in the frequency range from 328.6-335.4 MHz. The carrier is amplitude modulated at 90 and 150 Hz in a spatial pattern with the 90 Hz modulation predominant when the airplane is above the glide path and the 150 Hz modulation predominant if the airplane is below the glide path. If the aircraft is positioned exactly on the glide slope, the ILS receiver will receive an RF carrier where the 90 Hz and the 150 Hz modulation depths are exactly the same (each nominally 40 percent). The two antennas, the carrier antenna, and the sideband antenna that send out the same RF signal are arranged so that the 90 Hz sideband increases linearly with increasing glide-slope angle, while the 150 Hz sideband increases linearly with decreasing glide-slope angle; both of them are equal at the nominal glide-slope angle. The deviation from the glide slope is, therefore, a function of the difference in depth of modulation (DDM).

The localizer

The localizer is the lateral guidance portion of the ILS. Forty channels are at present provided from 108.1-111.9 MHz, each localizer channel being paired with a glide slope channel. The localizer course is aligned with the projected runway center line. As before, the carrier is modulated with a 90 Hz and 150 Hz tone in a spatial pattern that makes the 90 Hz modulation predominant when the aircraft is to the left of the course, and the 150 Hz modulation predominant when the aircraft is to the right of the course. Similarly, the antenna array is arranged so that the total peak envelope modulation remains constant. The 90 Hz modulation increases linearly with the angle deviation from the runway center line towards the left, and the 150 Hz modulation increases linearly towards the right. At the center of the runway, the modulation at 90 Hz is 20 percent and the modulation at 150 Hz is 20 percent, resulting in a peak modulation of 40 percent nominal.

The localizer carrier contains a Morse coded signal identifying the runway and approach direction and also may carry a ground-to-air radiotelephone communication channel.

A variation of the above is a two frequency carrier system for each glide slope and localizer. The two frequencies are centered about the nominal center frequency for the specific channel and are 4-32 kHz apart for glide slope, 5-14 kHz apart for localizer.

Marker beacons

The outer marker is approximately 2 W at 75 MHz, modulated 95 percent at 400 Hz. It is located 4-7 miles from the end of runway where the glide slope intersects the procedure turn altitude +-50 ft vertically. It radiates a fan shaped pattern vertically and normal to the localizer and actuates a marker receiver when the aircraft passes through.

The middle marker is a second fan marker similar to the outer marker. It is located approximately 3500 ft from the ILS approach end of the runway and modulated at 1300 Hz.

The inner marker, when used for category II approaches (operation down to 30 m (or 100 ft) and 400 m (or 1300 ft) visibility), intercepts the glide path at about the 100 ft height to mark the overshoot decision point if the runway still is not visible. This marker is recognized by its 3000 Hz modulation.

Approach lights

Approach lights are high-intensity strobe lights sequenced to direct the aircraft toward touch-down. They are a visual aid only.

16.20 IMPEDANCE

16.20.1 Definition

An electrical quantity determined by the voltage/current ratio in an electrical circuit Z = E(T)/I(T).

16.20.2 Formal syntax

16.20.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Capacitance	CAP	R	SRM	Capacitance
Conductance	CONDUCTANCE	R	-RM	Conductance
Current limit	CURRENT-LMT	R	-R-	Current
Dissipation factor	DISS-FACTOR	R	SRM	Ratio
Frequency	FREQ	R	SR-	Frequency
Inductance	IND	R	SRM	Inductance
Power limit	PWR-LMT	R	-R-	Power

continued

Modifier	Mnemonic	Type	Usage	Quantity
Q Factor	Q	R	SRM	Ratio
Reactance	REACTANCE	R	-RM	Resistance
Reference voltage	REF-VOLT	R	-R-	Voltage
Resistance	RES	R	SRM	Resistance
Susceptance	SUSCEPTANCE	R	-RM	Susceptance
Standing wave ratio	SWR	R	SRM	Ratio
Voltage limit	VOLT-LMT	R	-R-	Voltage

16.20.4 Rules

If two or more of the RES, CAP, or IND modifier fields are employed, they indicate a collection of series impedances. If parallel components are to be connected to the same UUT pins, two or more statements shall be employed.

16.21 LIGHT

16.21.1 Definition

Radiation capable of stimulating the eye to produce visual sensation.

16.21.2 Formal syntax

16.21.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Luminous efficiency	EFF	R	SRM	Ratio
Luminous efficacy	EFFICACY	R	SRM	Luminous flux/power
Illumination	ILLUM	R	SRM	Illuminance
Luminance	LUMINANCE	R	SRM	Luminance
Luminous flux	LUM-FLUX	R	SRM	Luminous flux
Luminous intensity	LUM-INT	R	SRM	Luminous intensity
Luminance temperature	LUM-TEMP	R	SRM	Temperature
Wavelength	WAVE-LENGTH	R	SRM	Wavelength

16.22 LOGIC CONTROL

16.22.1 Definition

An analog signal transmitted to or from the UUT over a single signal path or a multi-bit parallel digital signal used to convey control information for digital testing. LOGIC CONTROL signals may be single pulses, pulse trains, or dc levels. They may be used to control the passing, sequencing, or timing of data, including such functions as enable, disable, ready, data-received, etc.

16.22.2 Formal syntax

16.22.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Burst length	BURST	I	SR-	Burst length
Complement	COMPL	R	SR-	Voltage
Current	CURRENT	R	SR-	Current
Current logic 1	CURRENT-ONE	R	SR-	Current
Current logic 0	CURRENT-ZERO	R	SR-	Current
DC offset	DC-OFFSET	R	SR-	Voltage
Droop	DROOP	R	SR-	Ratio
Duty cycle	DUTY-CYCLE	R	SR-	Ratio
Fall time	FALL-TIME	R	-R-	Time
Noise	NOISE	R	SR-	Voltage, current, power or ratio
Overshoot	OVERSHOOT	R	SR-	Ratio
Pulse period	PERIOD	R	SRM	Time
Power	POWER	R	SR-	Power
Preshoot	PRESHOOT	R	SR-	Ratio
Pulse repetition frequency	PRF	R	SRM	Frequency
Pulse width	PULSE-WIDTH	R	SR-	Time
Ringing	RINGING	R	SR-	Ratio
Rise time	RISE-TIME	R	-R-	Time
Corner rounding	ROUNDING	R	SR-	Ratio
Bit skew	SKEW-TIME	R	SR-	Time
Time jitter	TIME-JIT	R	SR-	Time
True	TRUE	R	SR-	Voltage
Undershoot	UNDERSHOOT	R	SR-	Ratio
Control word contents	VALUE	B/BA	S-M	(None)
Voltage	VOLTAGE	R	SR-	Voltage
Voltage logic 1	VOLTAGE-ONE	R	SR-	Voltage
Voltage logic 0	VOLTAGE-ZERO	R	SR-	Voltage
Control word length	WORD-LENGTH	I	S--	Digital word length

16.22.4 Rules

1. The applicability of each modifier to SOURCE or SENSOR type statements and the applicability of each modifier mnemonic as a measured characteristic mnemonic is specified in table 16-2.

2. The following rules apply to the use of PERIOD in digital test statements.

a) The PERIOD modifier is used only as either a statement characteristic or a measured characteristic. PERIOD is required for all periodic SOURCE signals; optional for other cases.

b) For single-word parallel transfer, this modifier is undefined.

3. The following rules apply to PULSE-WIDTH.

a) PULSE-WIDTH is required for nonclocked parallel data and optional for clocked parallel data.

b) Optional for LOGIC CONTROL.

4. The following rules apply to SKEW-TIME.

a) Optional with parallel data. Cannot be used as a SENSOR measured characteristic.

b) Standard C/ATLAS modifier form, generally expressed as a MAX value.

c) For SOURCE, this modifier defines the maximum permissible skew time. For SENSOR, it defines the maximum skew time expected from the UUT.

Table 16-2. Summary of digital modifier usage

	MODIFIERS FOR STATEMENT CHARACTERISTICS (1)							
NOUNS	FOR SOURCE STATEMENTS				FOR SENSOR STATEMENTS			
MODIFIERS	LOGIC REF	LOGIC CONT'L	PAR DATA	SERIAL DATA	LOGIC REF	LOGIC CONT'L	PAR DATA	SERIAL DATA
TYPE	NA	NA	R	R	NA	NA	R	R
PULSE-CLASS	NA	NA	NA	R	NA	NA	NA	R
TRANS-ONE, -ZERO, -PERIOD	NA	NA	NA	(3)	NA	NA	NA	(3)
VALUE	NA	NA	R	R	NA	NA	NA	NA
VOLTAGE-ONE, -ZERO, -QUIES (2)	R (7)	R (7)	R	R	O	O	R	R
CURRENT-ONE, -ZERO, QUIES (2)	O (8)	O (8)	O	O	O	O	O	O
FREQ-ONE, -ZERO, -QUIES (2)	NA	NA	(5)	(5)	NA	NA	(5)	(5)
PRF, PERIOD	R (6)	R (6)	NA	NA	O	O	NA	NA
DELAY (7)	O	O	O	O	O	O	O	O
WORD-LENGTH	NA	NA	(5)	(5)	NA	NA	(5)	(5)
SKEW-TIME	NA	NA	O	NA	NA	NA	O	NA
PULSE-WIDTH	R	O	O	NA	O	O	O	NA
BIT-RATE	NA	NA	NA	O	NA	NA	NA	O
WORD-RATE	NA	NA	O	O	NA	NA	O	O

NOTES

1) Code: R-modifier required, O-modifier use is optional, NA-modifier not used.

2) Voltage and current modifiers: Usage shown for a voltage source as the signal driver.

3) TRANS modifiers: Required for all BIP SOURCE SERIAL DATA statements.

4) Voltage and current magnitude measurements are handled as normal pulse or dc measurements.

5) FREQ modifiers: Required for all waveforms that carry binary data via frequency modulation.

6) Either PRF or PERIOD is required for periodic waveforms.

7) Does not apply to VOLTAGE-QUIES.

8) Does not apply to CURRENT-QUIES.

Table 16-2. Summary of digital modifier usage (continued)

NOUNS / MODIFIERS	MEASURED CAHRACTERISTIC (1) FOR SENSOR STATEMENTS			
	LOGIC REF	LOGIC CONT'L	PAR DATA	SERIAL DATA
TYPE	NA	NA	NA	NA
PULSE-CLASS	NA	NA	NA	NA
TRANS-ONE, -ZERO, -PERIOD	NA	NA	O	O
VALUE	NA (4)	NA(4)	NA	NA
VOLTAGE-ONE, -ZERO, -QUIES (2)	NA (4)	NA (4)	NA	NA
CURRENT-ONE, -ZERO, -QUIES (2)	NA	NA	NA	NA
FREQ-ONE, -ZERO, -QUIES (2)	NA	NA	NA	NA
PRF, PERIOD	O	O	NA	NA
DELAY	NA	NA	NA	NA
SKEW-TIME	NA	NA	NA	NA
PULSE-WIDTH	NA	NA	NA	NA
BIT-RATE	NA	NA	NA	NA
WORD-RATE	NA	NA	NA	NA

NOTES
1) Code: R-modifier required, O-modifier use is optional, NA-modifier not used.
2) Voltage and current modifiers: Usage shown for a voltage source as the signal driver.
3) TRANS modifiers: Required for all BIP SOURCE SERIAL DATA statements.
4) Voltage and current magnitude measurements are handled as normal pulse or dc measurements.
5) FREQ modifiers: Required for all waveforms that carry binary data via frequency modulation.
6) Either PRF or PERIOD is required for periodic waveforms.
7) Does not apply to VOLTAGE-QUIES.
8) Does not apply to CURRENT-QUIES.

16.23 LOGIC DATA

16.23.1 Definition

A binary digital signal containing more than one binary digit (bit) and used to convey data to or from a UUT. All bits may appear simultaneously at the digital interface, each on a separate circuit (i.e., in parallel form) or they may appear sequentially over a single circuit (i.e., in serial form). These bits may represent either a true binary number (as used in binary arithmetic or to represent an analog quantity) or just a collection of bits having no intrinsic value of their own. LOGIC DATA signals may be single pulses, pulse trains, or dc levels.

16.23.2 Formal syntax

16.23.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Bit-rate	BIT-RATE	R	SR-	Digital bit rate
Burst length	BURST	I	SR-	Burst length
Complement	COMPL	R	SR-	Voltage
Current	CURRENT	R	SR-	Current
Current, logic 1	CURRENT-ONE	R	SR-	Current
Current, quiescent	CURRENT-QUIES	R	SR-	Current
Current, logic 0	CURRENT-ZERO	R	SR-	Current
DC offset	DC-OFFSET	R	SR-	Voltage
Droop	DROOP	R	SR-	Ratio
Duty cycle	DUTY-CYCLE	R	SR-	Ratio
Fall time	FALL-TIME	R	-R-	Time
Frequency, logic 1	FREQ-ONE	R	SR-	Frequency
Frequency, quiescent	FREQ-QUIES	R	SR-	Frequency
Frequency, logic 0	FREQ-ZERO	R	SR-	Frequency
Noise	NOISE	R	SRM	Voltage, ratio or current
Overshoot	OVERSHOOT	R	SR-	Ratio
Power	POWER	R	SR-	Power
Preshoot	PRESHOOT	R	SR-	Ratio
Pulse repetition frequency	PRF	R	SR-	Frequency
Pulse class or code	PULSE-CLASS	MD	SR-	(None)
Pulse width	PULSE-WIDTH	R	SR-	Time
Ringing	RINGING	R	SR-	Ratio
Rise time	RISE-TIME	R	-R-	Time
Corner rounding	ROUNDING	R	SR-	Ratio
Bit skew	SKEW-TIME	R	SR-	Time
Time jitter	TIME-JIT	R	SR-	Time
Transition for logic 1	TRANS-ONE	MD	SR-	(None)
Initial transition	TRANS-PERIOD	MD	SR-	(None)
Transition for logic 0	TRANS-ZERO	MD	SR-	(None)
True	TRUE	R	SR-	Voltage
Digital signal type	TYPE	MD	SR-	(None)
Undershoot	UNDERSHOOT	R	SR-	Ratio
Data word contents or value	VALUE	B/BA	S-M	(None)
Voltage	VOLTAGE	R	SR-	Voltage
Voltage, logic 1	VOLTAGE-ONE	R	SR-	Voltage
Voltage, quiescent	VOLTAGE-QUIES	R	SR-	Voltage
Voltage, logic 0	VOLTAGE-ZERO	R	SR-	Voltage
Word length	WORD-LENGTH	I	S--	Digital word length
Word-rate	WORD-RATE	R	SR-	Digital word rate

16.23.4 Rules

16.23.4.1 General

1. CURRENT and VOLTAGE shall be used when binary data is coded in other than current or voltage amplitudes.

2. PULSE-CLASS and TIME-JIT are used for serial data only.

3. SKEW-TIME and PULSE-WIDTH are used for parallel data only.

4. The applicability of each modifier to SOURCE or SENSOR type statements and the applicability of each modifier mnemonic as a measured characteristic mnemonic is specified in table 16-2.

5. If a single <sync subfield> appears in the <rate/data> field

a) For SERIAL data if no BIT-RATE field is present, the <sync subfield> defines the <rate/bit>.

b) For PARALLEL data, the <sync subfield> defines the <rate/word>.

The <rate/bit> modifier is applicable to serial data only. BIT-RATE and WORD-RATE can only be used together for a serial signal comprising more than one word where the WORD-RATE specifies the word spacing in the serial stream. When two sync-subfields are written to express the requirements of the <rate/bit> and <rate/word> element of the <rate/data> subfield, the first sync- subfield relates to bit synchronization, and the second relates to word synchronization.

6. VALUE, when used as a measured characteristic, is not repeated as a statement characteristic element.

7. The following rules apply to PULSE-WIDTH when used with the noun LOGIC DATA.

a) Either PULSE-WIDTH or WORD-RATE is required for nonclocked parallel data; PULSE-WIDTH is optional for clocked parallel data.

b) The pulse width for SERIAL DATA is implicit in the PULSE-CLASS type.

8. The modifiers TRANS-ZERO/-ONE/-PERIOD are required for all SERIAL DATA signals of the PULSE-CLASS BIP or PULSE-CLASS MIP.

9. Form for BIP.

a) TRANS-ONE xy, TRANS-ZERO xy

b) TRANS-ONE x1y1 OR x2y2, TRANS-ZERO x1y1 OR x2y2

c) TRANS-PERIOD wy

where x and y = 0 or 1

w = Q, 0, or 1

10. Each bit period of a BIP waveform is divided into two equal sub-periods. A transition in voltage may or may not occur at the start of each bit period. A second transition may or may not occur at the half-period point. The binary data is coded into the waveform in terms of this second transition and the voltage levels in the two sub-periods.

11. Form for MIP.

a) TRANS-ONE . . .xyz. . .

TRANS-ZERO . . .xyz . . .

b) TRANS-ONE x1 y1 z1. . .OR x2 y2 z2. . .

c) TRANS-PERIOD wy

where:

x, y and z = 0 or 1 (or Q for systems with 3 significant voltage levels, e.g., Gina Bus)

w = Q, 0, or 1

12. Each bit period of a MIP waveform is divided into three or more equal sub-periods. A transition in voltage level may or may not occur at the start of each bit period. A second transition may or may not occur at the start of each sub-period in the bit period. The binary data is coded into the waveform in terms of these transitions and the voltage levels in the sub-periods.

13. The TRANS-ONE and TRANS-ZERO modifiers define, respectively, the binary 1 and binary 0 data conditions. The integers x, y and z describe the voltage levels (or pulse amplitudes) in the sub- periods, respectively, according to the following conventions:

a) 1 Indicates an amplitude of VOLTAGE-ONE.

b) 0 Indicates an amplitude of VOLTAGE-ZERO.

c) Q Indicates an amplitude of VOLTAGE-QUIES.

14. If the binary data is always represented by the same sequence of voltage levels, form 'a' is used. If each binary digit is represented by either of two sets of voltage levels, form 'b' is used.

15. The TRANS-PERIOD modifier, when used, indicates that there is always a voltage transition at the start of each period. The entries for w and y define the direction of the initial waveform transition as follows:

a) w defines the voltage level just prior to this initial transition.

b) y defines the voltage level immediately after this initial transition.

c) Q indicates an amplitude of VOLTAGE-QUIES.

16. For certain waveforms, the TRANS-ONE/-ZERO modifiers are not sufficient to fully define the waveform. This uncertainty is eliminated by use of the TRANS-PERIOD modifier. Although its use is essential only when an uncertainty exists without it, its use is recommended whenever such a transition does take place. Thus, in figure 6.1 the modifier is required for waveforms (d) and (e). See examples given for the modifier PULSE-CLASS.

17. The modifier TYPE is required for all LOGIC DATA statements, for both SOURCE and SENSOR. TYPE cannot be used as a measured characteristic in SENSOR statements. In LOGIC DATA statements the TYPE subfield has the form:

a) TYPE PARALLEL

b) TYPE SERIAL-LSB-FIRST

c) TYPE SERIAL-MSB-FIRST

For a PARALLEL signal, all bits appear concurrently at the digital interface, each on a separate electrical circuit. For a SERIAL signal, all bits appear at the digital interface sequentially over a single electrical circuit. For a SERIAL signal, this modifier also defines the sequence of bits at the interface as follows:

SERIAL-LSB-FIRST	The least significant bit appears first.
SERIAL-MSB-FIRST	The most significant bit appears first.

All digital representations show the LSB in the right-most bit position.

18. The modifier VALUE is required for all source DATA statements. It can be used as the measured characteristic for sensor DATA statements. In LOGIC DATA statements the VALUE subfield has the form:

VALUE N Where N is an expression.

N may be expressed in any of the standard C/ATLAS forms (such as, B'1011', X'5E', etc.). All numbers used for N are treated as dimensionless quantities. Integer numbers are right justified; fractional numbers are left justified. Only <value> type labels may be used with VALUE.

The following are examples of VALUE:

```
LOGIC DATA, TYPE SERIAL-MSB-FIRST, VALUE B'100000'

LOGIC DATA, TYPE SERIAL-MSB-FIRST, VALUE 32

LOGIC DATA, TYPE SERIAL-MSB-FIRST, VALUE O'40'
```

Each causes a bit pattern of 100000 to be placed in the digital word being described. The binary 1 will be the first of these bits to appear at the digital interface.

19. The following requirements apply to VOLTAGE-ONE and VOLTAGE-ZERO:

a) Required for all logic signals that are driven from voltage sources
b) Optional for all logic signals that are driven from current sources
c) Cannot be used as a SENSOR measured characteristic

20. The following rules apply to all logic signals:

a) VOLTAGE-ONE and VOLTAGE-ZERO are used with logic signal waveforms that have only two logic states. VOLTAGE-ONE, VOLTAGE-ZERO, and VOLTAGE-QUIES are used with waveforms that have two logic states plus a quiescent level. These groups are referred to as the "applicable set."
b) For a SOURCE waveform driven from a voltage source, together with the associated tolerances given in ERRLMT, MIN, or MAX subfields, define the permissible range of voltage levels to be applied to the UUT. If no tolerances are given, no special control is exercised by the test system over the accuracy of the voltage levels.
c) For a SENSOR waveform driven from a voltage source, these modifiers, together with the associated tolerances or boundary limits given in RANGE, MAX, or MIN subfields, define the range of voltages from the UUT that shall be accepted by the test system.
d) For either SOURCE or SENSOR waveforms driven from a current source, these modifiers define the RANGE, MAX, or MIN values of voltage associated with the corresponding driving logic currents. The same conventions used with analog signals are employed.
e) For differential digital signals carried on a three-wire circuit, these modifiers define the voltage from the TRUE line to the COMPL line.

21. The following rules apply to all LOGIC DATA signals driven from voltage sources except bi-phase SERIAL DATA:

a) VOLTAGE-ONE and VOLTAGE-ZERO define, respectively, the nominal voltages at the test system/UUT interface that represent a binary 1 and a binary 0. VOLTAGE-QUIES defines the third voltage level when one exists. These voltages may be either DC levels or pulse amplitudes.

b) For SENSOR, the VOLTAGE-ONE and VOLTAGE-ZERO modifiers, together with their associated tolerances or boundary limits, define the range of voltages from the UUT that shall be recognized by the test system as a binary 1 or a binary 0, respectively. (Note that this operation implies a test system evaluation). If voltage tolerances or limits are not defined, the threshold between the binary 1 and 0 logic levels is determined by the inherent design of the test system.

22. The following rules apply to all bi-phase (BIP) SERIAL data driven from voltage sources:

a) The concept of a binary 1 or 0 applied to the pulse amplitudes has no intrinsic meaning as the binary data is not carried by the pulse amplitudes. However, for convenience, these modifiers are redefined and used.

b) VOLTAGE-ONE and VOLTAGE-ZERO define the nominal values of the two voltage states. VOLTAGE-QUIES represents the "no signal" or quiescent voltage value.

23. The following applies to all non-bi-phase LOGIC DATA SENSOR waveforms driven from a voltage source:

a) The VOLTAGE-ONE, VOLTAGE-ZERO modifiers, when specified in terms of a MIN, MAX, or RANGE, imply a test measurement and evaluation. That is, the test system will react to the UUT output voltage amplitude and the stated tolerance or threshold limits that define the binary 1 and 0 ranges and classify the logic signal as binary 1 or 0.

b) If the UUT voltage falls outside both the binary 1 and 0 ranges, the binary interpretation is determined by the test system implementation, but generally is one of the following:

   1) The binary digit is interpreted as a third state, undefined.

   2) The binary digit is interpreted as either a binary 1 or 0, depending on the test system implementation.

24. The binary 1 state need not necessarily be represented by the higher potential. Thus,

a) VOLTAGE-ONE 5 V, VOLTAGE-ZERO 0 V

b) VOLTAGE-ONE 0 V, VOLTAGE-ZERO 5 V

c) VOLTAGE-ONE 5 V, VOLTAGE-ZERO -1 V

25. The process of determining the 1 or 0 status of each bit in a UUT response, given the VOLTAGE-ONE, VOLTAGE-ZERO values, is a digital problem. This process does not involve taking an amplitude "measurement." The measurement being taken is the determination of the 1, 0 bit pattern. The actual "measurement" of these analog pulse amplitudes is an analog problem. Accordingly, the conventional modifiers for PULSED DC are used.

26. The use of the suffix DIFF and the modifiers TRUE and COMPL normally result in the statement carrying redundant information. It is recommended that, since the intelligence is carried by the voltage difference, the suffix DIFF is always used with either a MAX or MIN qualifier.

27. Either WORD-RATE or PULSE-WIDTH is required for non-clocked PARALLEL data.

16.23.4.2 Rules for modifier suffixes for differential digital signals

1. The VOLTAGE-ONE and VOLTAGE-ZERO modifiers optionally may have a suffix added to them to permit description of differential digital signals. In addition, the optional modifiers TRUE and COMPL may be used.

2. The suffix DIFF is used to indicate that the modifier defines the differential voltage as a signed value on the TRUE line with respect to the COMPL line when the data has the indicated logic value (i.e., voltage TRUE minus voltage COMPL).

3. The modifier TRUE, when used, indicates that the modifier defines the absolute voltage level (with respect to LO) that occurs at the TRUE terminal when the data has the indicated logic value.

4. The modifier COMPL, when used, indicates that the modifier defines the absolute voltage (with respect to LO) that occurs at the COMPL terminal when the data has the indicated logic value.

5. The modifiers TRUE and COMPL can only be used in a three-wire system.

16.23.5 Notes and examples

1. For a SOURCE to define ranges of 4.5-5.5 V for a binary 1 and 0.5-1.5 V for binary 0

   VOLTAGE-ONE 5 V ERRLMT + -0.5 V,

   VOLTAGE-ZERO 1 V ERRLMT + -0.5 V

2. For a SENSOR to define the same range as in 1)

   VOLTAGE-ONE RANGE 4.5V TO 5.5V,

   VOLTAGE-ZERO RANGE 0.5V TO 1.5V

3. For a SENSOR to define the following ranges: 3.0 to + infinity V for binary 1, 2.0 to - infinity V for binary 0, and 2-3 V not defined:

   VOLTAGE-ONE MIN 3.0 V,

   VOLTAGE-ZERO MAX 2.0 V

4. For a SOURCE

   TYPE SERIAL-MSB-FIRST, PULSE-CLASS NRZ,

   VALUE B'101101', VOLTAGE-ONE 4.5 V ERRLMT + -0.5 V,

   VOLTAGE-ZERO 0.5 V ERRLMT + -0.5 V

The associated voltage ranges and a possible pulse train are shown in the following figure:

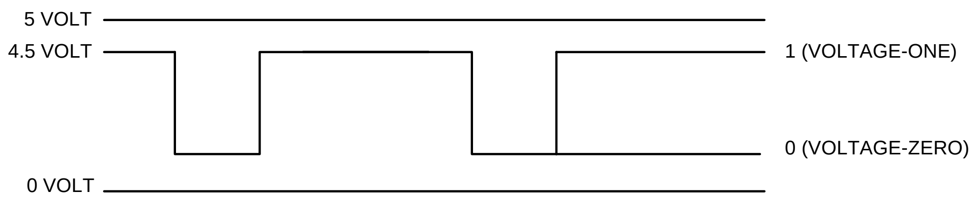


Figure 16-3. Defined voltage ranges

5. For a SENSOR

```
   TYPE SERIAL-LSB-FIRST,  PULSE-CLASS NRZ,
   VOLTAGE-ONE RANGE 4.0V TO 5.0V,
   VOLTAGE-ZERO RANGE 0V TO 1.0V
```

These ranges defined for the binary 0 and 1 are the same as shown in 4), above.

16.24 LOGIC LOAD

16.24.1 Definition

A nonlinear load used with logic circuits which, itself, contains an active logic circuit. The LOGIC LOAD is characterized by a nonlinear input impedance and a nonlinear voltage/current characteristic. The magnitude of the load is specified in terms of UNIT-LOADS.

16.24.2 Formal syntax

16.24.3 Noun modifier set

There are no modifiers for LOGIC LOAD.

16.25 LOGIC REFERENCE

16.25.1 Definition

A binary digital signal similar to a LOGIC CONTROL. The LOGIC REFERENCE is a pulse train used as a timing or phase reference or for gating other digital signals. It may originate in the UUT or in the test system. For the latter case, the signal may exist only within the test system or it may also be applied to the UUT.

16.25.2 Formal syntax

16.25.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Burst length	BURST	I	SR-	Burst length
Complement	COMPL	R	SR-	Voltage
Current	CURRENT	R	SR-	Current
Current logic 1	CURRENT-ONE	R	SR-	Current
Current logic 0	CURRENT-ZERO	R	SR-	Current
DC offset	DC-OFFSET	R	SR-	Voltage
Droop	DROOP	R	SR-	Ratio
Duty cycle	DUTY-CYCLE	R	SR-	Ratio
Fall time	FALL-TIME	R	-R-	Time
Noise	NOISE	R	SRM	Voltage, current, power or ratio
Overshoot	OVERSHOOT	R	SR-	Ratio
Pulse period	PERIOD	R	SRM	Time
Power	POWER	R	SR-	Power
Preshoot	PRESHOOT	R	SR-	Ratio
Pulse repetition frequency	PRF	R	SRM	Frequency
Pulse width	PULSE-WIDTH	R	SR-	Time
Ringing	RINGING	R	SR-	Ratio
Rise time	RISE-TIME	R	-R-	Time
Corner rounding	ROUNDING	R	SR-	Ratio
Time jitter	TIME-JIT	R	SR-	Time
True	TRUE	R	SR-	Voltage
Undershoot	UNDERSHOOT	R	SR-	Ratio
Voltage	VOLTAGE	R	SR-	Voltage
Voltage logic 1	VOLTAGE-ONE	R	SR-	Voltage
Voltage logic 0	VOLTAGE-ZERO	R	SR-	Voltage

16.25.4 Rules

1. The applicability of each modifier to SOURCE or SENSOR type statements and the applicability of each modifier mnemonic as a measured characteristic mnemonic is specified in table 16-2.

2. The following rules apply to the use of PERIOD in digital testing statements.

a) The PERIOD modifier is used as either a statement characteristic or a measured characteristic. PERIOD is required for all periodic SOURCE signals; optional for other cases.

b) For single-word parallel transfer, this modifier is undefined.

3. The following rules apply to PULSE-WIDTH.

a) Required as a SOURCE; optional for a SENSOR.
b) PULSE-WIDTH is optional for clocked parallel data.
c) The pulse width for SERIAL DATA is implicit in the PULSE-CLASS type and the PRF of the LOGIC REFERENCE.

16.26 MANOMETRIC

16.26.1 Definition

A term denoting the characteristics of air at low pressure and negligible flow rate.

16.26.2 Formal syntax

16.26.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Altitude	ALT	R	SRM	Distance
Altitude Rate	ALT-RATE	R	SRM	Distance/time
Atmosphere	ATMOS	MD	SR-	(None)
Indicated airspeed	IAS	R	SRM	Distance/time
Pressure absolute	PRESS-A	R	SRM	Pressure
Pressure gauge	PRESS-G	R	SRM	Pressure
Pressure oscillation amplitude	PRESS-OSC-AMPL	R	S--	Pressure
Pressure oscillation frequency	PRESS-OSC-FREQ	R	S--	Frequency
Slew rate	SLEW-RATE	R	S--	Pressure/time
True airspeed	TAS	R	SRM	Distance/time or velocity, linear

16.26.4 Rules

Modifier usage shall be as follows:

a) Only one of the modifiers ALT, TAS, IAS, PRESS-A, PRESS-G may occur in any one statement.
b) If ALT, ALT-RATE, TAS, or IAS are used, then ATMOS shall also be used.
c) If ALT-RATE is used, then ALT shall also be used.
d) If PRESS-A, PRESS-G, TAS, or IAS are used, then the connection field shall include a reference subfield referring to the reference pressure PRESS-A connection on the UUT.

16.27 PAM (pulse amplitude modulation)

16.27.1 Definition

A signal in which a pulsed AC signal (carrier) is caused to depart from its unmodulated level by an amount proportional to the instantaneous value of the modulating signal.

16.27.2 Formal syntax

16.27.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Burst length	BURST	I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	SR-	Frequency
Carrier frequency	CAR-FREQ	R	SR-	Frequency
Droop	DROOP	R	SRM	Ratio
Fall time	FALL-TIME	R	SRM	Time
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Modulation amplitude	MOD-AMPL	R	SRM	Ratio
Modulation frequency	MOD-FREQ	R	SRM	Frequency
Noise	NOISE	R	-RM	Voltage, power or ratio
Overshoot	OVERSHOOT	R	SRM	Ratio
Period	PERIOD	R	SRM	Time
Preshoot	PRESHOOT	R	SRM	Ratio
Pulse repetition frequency	PRF	R	SRM	Frequency
Pulse width	PULSE-WIDTH	R	SRM	Time
Peak pulse amplitude	P-AMPL	R	-RM	Voltage, ratio or power
Ringing	RINGING	R	SRM	Ratio
Rise time	RISE-TIME	R	SRM	Time
Corner rounding	ROUNDING	R	SRM	Ratio
Undershoot	UNDERSHOOT	R	SRM	Ratio

16.28 PM SIGNAL (phase modulated signal)

16.28.1 Definition

A continuous sinusoidal wave (carrier) whose phase is varied in accordance with the amplitude of another wave.

16.28.2 Formal syntax

16.28.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity*
Burst length	BURST	I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	SR-	Frequency
Carrier amplitude	CAR-AMPL	R	SRM	Voltage
Carrier frequency	CAR-FREQ	R	SRM	Frequency
Carrier harmonic distortion	CAR-HARMONICS	R	SRM	Ratio, voltage or power
Modulation amplitude	MOD-AMPL	R	SRM	Voltage or ratio
Modulation distortion	MOD-DIST	R	-RM	Voltage or ratio

continued

Modifier	Mnemonic	Type	Usage	Quantity*
Modulation frequency	MOD-FREQ	R	SRM	Frequency
Noise	NOISE	R	-RM	Ratio, voltage or power
Non-Harmonic distortion	NON-HARMONICS	R	-RM	Voltage, power or ratio
Period	PERIOD	R	SRM	Time
Phase deviation	PHASE-DEV	R	SRM	Angle, plane
Power	POWER	R	SRM	Power

NOTE
* Decibel and percent values are referenced to the unmodulated carrier amplitude.

16.29 PULSED AC (pulsed alternating current signal)

16.29.1 Definition

An EMF characterized by short duration periods of AC (sinusoidal) electrical potential.

16.29.2 Formal syntax

16.29.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Bandwidth	BANDWIDTH	R	SR-	Frequency
Burst length	BURST	I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	SR-	Frequency
Carrier frequency	CAR-FREQ	R	SRM	Frequency
Carrier phase	CAR-PHASE	R	SRM	Angle, plane
Current	CURRENT	R	SRM	Current
DC offset	DC-OFFSET	R	SRM	Voltage
Droop	DROOP	R	SRM	Ratio
Fall time	FALL-TIME	R	SRM	Time
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Noise	NOISE	R	-RM	Ratio,voltage, current or power
Overshoot	OVERSHOOT	R	SRM	Ratio
Period	PERIOD	R	SRM	Time
Phase jitter	PHASE-JIT	R	SRM	Angle, plane
Power	POWER	R	SRM	Power
Preshoot	PRESHOOT	R	SRM	Ratio
Pulse repetition frequency	PRF	R	SRM	Frequency
Pulse width	PULSE-WIDTH	R	SRM	Time
Ringing	RINGING	R	SRM	Ratio

continued				
Modifier	Mnemonic	Type	Usage	Quantity
Rise time	RISE-TIME	R	SRM	Time
Corner rounding	ROUNDING	R	SRM	Ratio
Standing wave ratio	SWR	R	SRM	Ratio
Time jitter	TIME-JIT	R	SRM	Time
Undershoot	UNDERSHOOT	R	SRM	Ratio
Voltage	VOLTAGE	R	SRM	Voltage

16.30 PULSED AC TRAIN

16.30.1 Definition

An EMF characterized by a train of different short duration periods of AC sinusoidal electrical potential.

16.30.2 Formal syntax

16.30.3 Noun modifier set

Modifier	Mnemonic	Local scope	Type	Usage	Quantity
Bandwidth	BANDWIDTH		R	SR-	Frequency
Burst length	BURST		I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	SR-		Frequency
Carrier frequency	CAR-FREQ		R	SRM	Frequency
Current	CURRENT	X	R	SRM	Current
DC offset	DC-OFFSET		R	SRM	Voltage
Droop	DROOP	X	R	-RM	Ratio
Fall time	FALL-TIME	X	R	SRM	Time
Noise	NOISE		R	-RM	Ratio, voltage, current or power
Overshoot	OVERSHOOT	X	R	-RM	Ratio
Period	PERIOD		R	SRM	Time
Power	POWER		R	SRM	Power
Preshoot	PRESHOOT	X	R	-RM	Ratio
Pulse repetition frequency	PRF		R	SRM	Frequency
Pulse width	PULSE-WIDTH	X	R	SRM	Time
Ringing	RINGING	X	R	-RM	Ratio
Rise time	RISE-TIME	X	R	SRM	Time
Corner rounding	ROUNDING	X	R	-RM	Ratio
Pulse spacing	SPACING	X	R	SRM	Time
Undershoot	UNDERSHOOT	X	R	-RM	Ratio
Voltage	VOLTAGE	X	R	SRM	Voltage

16.30.4 Rules

1. The scope of certain modifiers can be specified as global (i.e., they apply to each pulse in the pulse train) or they can be specified as local (i.e., they apply only to the specified single pulse or pulse pair). Local modifiers are identified by the pulse identifier suffix (a -P followed by an <unsigned integer number>) that defines the number of the pulse being described. Pulses are numbered in order of their occurrence in time with the first pulse being number one.

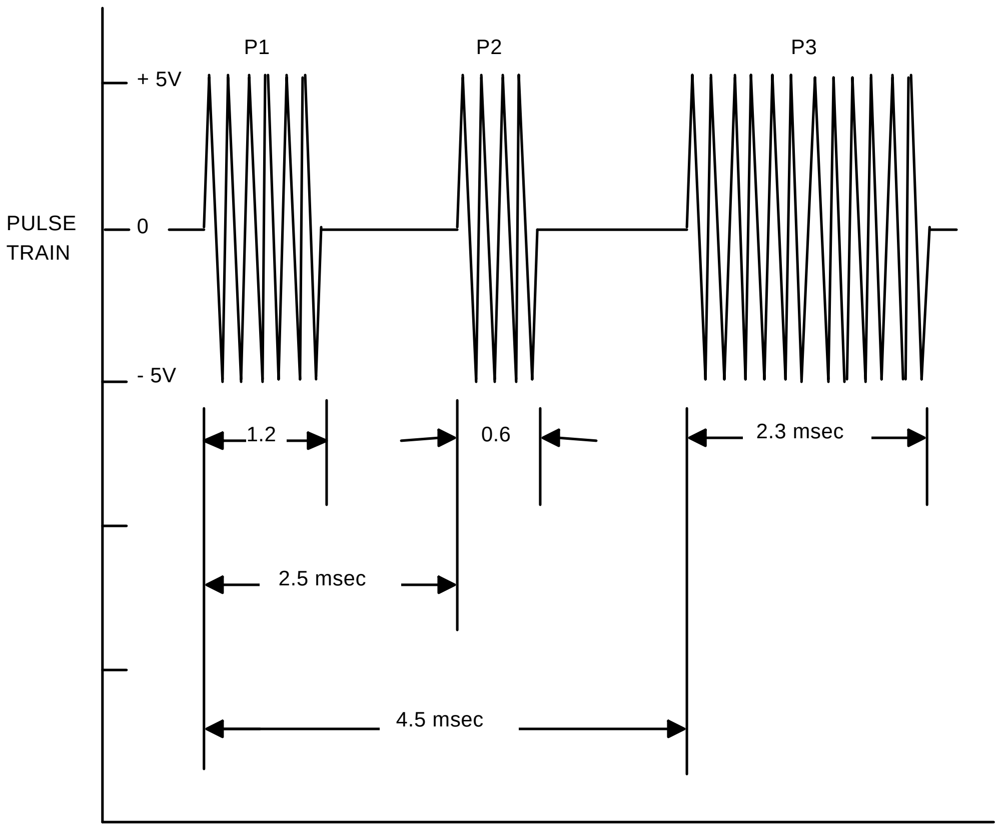
2. All modifiers except SPACING may be used with global scope. Those modifiers having an X in the local scope column may be used with local scope. SPACING is used only with local scope. See Pulse Identifier Suffix for additional rules.

3. Spacing between pulses in the pulse train is specified as the time between either the initial or final slope of any two pulses.

4. The amplitude of pulses that have a common overlap in time are algebraically added during the time of overlap.

16.30.5 Notes and examples

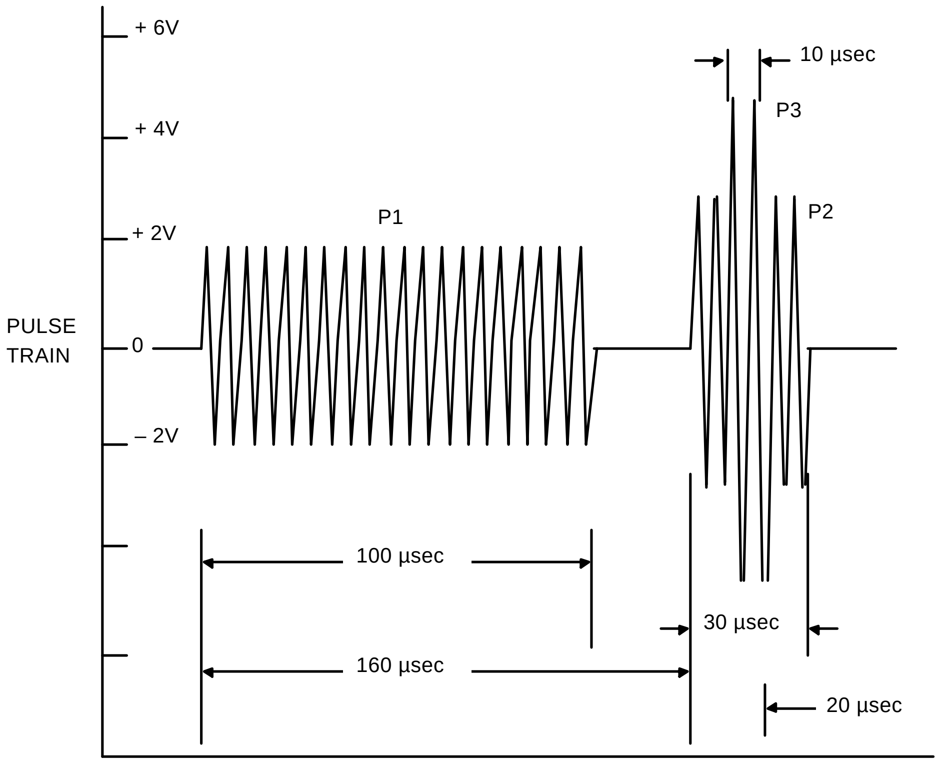
Example 1.



The pulse train shown above is defined by three single pulses. This signal may be described by the following statement:

```
 100050 SETUP, PULSED AC TRAIN,
           CAR-FREQ 1 GHZ,
           VOLTAGE-P 5 V,
           RISE-TIME 100 NSEC,
           PULSE-WIDTH-P1 1.2 MSEC,
           PULSE-WIDTH-P2 0.6 MSEC,
           PULSE-WIDTH-P3 2.3 MSEC,
           SPACING-P1-POS-P2-POS 2.5 MSEC,
           SPACING-P1-POS-P3-POS 4.5 MSEC, .... $
```

Example 2.



```
 100100 DEFINE, 'PULSED AC TRAIN EXAMPLE', SENSOR, PULSED AC
TRAIN,
           CAR-FREQ 1060 MHZ,
           VOLTAGE-P-P1 2 V,
           PULSE-WIDTH-P1 100 USEC,
           VOLTAGE-P-P2 3 V,
           PULSE-WIDTH-P2 30 USEC,
           VOLTAGE-P-P3 2 V,
           PULSE-WIDTH-P3 10 USEC,
           SPACING-P1-POS-P2-POS 160 USEC,
           SPACING-P2-POS-P3-NEG 20 USEC, ..... $
```

16.31 PULSED DC (pulsed direct current signal)

16.31.1 Definition

An EMF characterized by short duration periods of DC electrical potential.

16.31.2 Formal syntax

16.31.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Current	CURRENT	R	SRM	Current
DC offset	DC-OFFSET	R	SRM	Voltage
Droop	DROOP	R	-RM	Ratio
Duty cycle	DUTY-CYCLE	R	SRM	Ratio
Fall time	FALL-TIME	R	SRM	Time
Noise	NOISE	R	-RM	Ratio,voltage, current or power
Overshoot	OVERSHOOT	R	-RM	Ratio
Period	PERIOD	R	SRM	Time
Power	POWER	R	SRM	Power
Preshoot	PRESHOOT	R	-RM	Ratio
Pulse repetition frequency	PRF	R	SRM	Frequency
Pulse width	PULSE-WIDTH	R	SRM	Time
Ringing	RINGING	R	-RM	Ratio
Rise time	RISE-TIME	R	SRM	Time
Corner rounding	ROUNDING	R	-RM	Ratio
Time jitter	TIME-JIT	R	SRM	Time
Undershoot	UNDERSHOOT	R	-RM	Ratio
Voltage	VOLTAGE	R	SRM	Voltage

16.32 PULSED DC TRAIN

16.32.1 Definition

An EMF characterized by a train of different short duration periods of DC electrical potential.

16.32.2 Formal syntax

16.32.3 Noun modifier set

Modifier	Mnemonic	Local Scope	Type	Usage	Quantity
Burst length	BURST		I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE		R	SR-	Frequency
Current	CURRENT	X	R	SRM	Current
DC offset	DC-OFFSET		R	SRM	Voltage
Droop	DROOP	X	R	-RM	Ratio
Fall time	FALL-TIME	X	R	SRM	Time
Noise	NOISE		R	-RM	Ratio,voltage, current or power
Overshoot	OVERSHOOT	X	R	-RM	Ratio
Period	PERIOD		R	SRM	Time
Power	POWER		R	SRM	Power
Preshoot	PRESHOOT	X	R	-RM	Ratio
Pulse repetition frequency	PRF		R	SRM	Frequency
Pulse width	PULSE-WIDTH	X	R	SRM	Time
Ringing	RINGING	X	R	-RM	Ratio
Rise time	RISE-TIME	X	R	SRM	Time
Corner rounding	ROUNDING	X	R	-RM	Ratio
Pulse spacing	SPACING	X	R	SRM	Time
Undershoot	UNDERSHOOT	X	R	-RM	Ratio
Voltage	VOLTAGE	X	R	SRM	Voltage

16.32.4 Rules

1. The scope of certain modifiers can be specified as global (i.e., they apply to each pulse in the pulse train) or they can be specified as local (i.e., they apply only to the specified single pulse or pulse pair). Local modifiers are identified by the pulse identifier suffix (a -P followed by an <unsigned integer number>) that defines the number of the pulse being described. Pulses are numbered in order of their occurrence in time with the first pulse being number one.

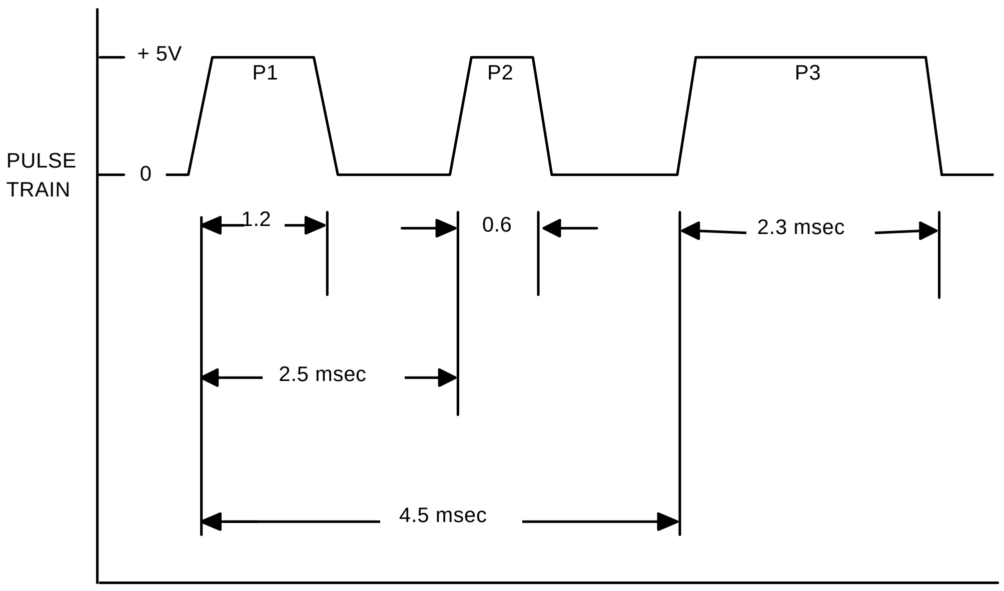
2. All modifiers except SPACING may be used with a global scope. Those modifiers having an X in the local scope column may be used with a local scope. SPACING is used only with local scope. See Pulse Identifier Suffix for additional rules.

3. Spacing between pulses in the pulse train is specified as the time between either the initial or final slope of any two pulses.

4. The amplitude of pulses that have a common overlap in time are algebraically added during the time of overlap.

16.32.5 Notes and examples

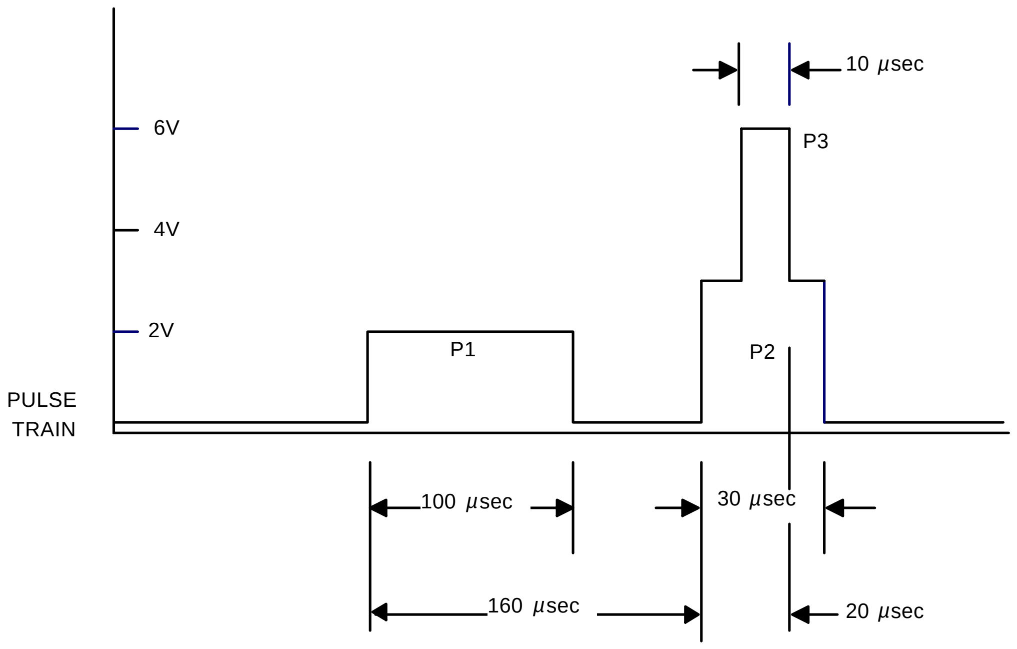
Example 1:



The pulse train shown above is defined by three single pulses. This signal may be described by the following statement:

```
100050 SETUP, PULSED DC TRAIN, VOLTAGE 5 V,
          RISE-TIME 100 NSEC,
          FALL-TIME 100 NSEC,
          PULSE-WIDTH-P1 1.2 MSEC,
          PULSE-WIDTH-P2 0.6 MSEC,
          PULSE-WIDTH-P3 2.3 MSEC,
          SPACING-P1-POS-P2-POS 2.5 MSEC,
          SPACING-P1-POS-P3-POS 4.5 MSEC, ... $
```

Example 2:



```
 100100 DEFINE, 'PULSED DC TRAIN EXAMPLE', SENSOR, PULSED DC
TRAIN,
            VOLTAGE-P1 2V, PULSE-WIDTH-P1 100 USEC,
            VOLTAGE-P2 3V, PULSE-WIDTH-P2 30 USEC,
            VOLTAGE-P3 3V, PULSE-WIDTH-P3 10 USEC,
            SPACING-P1-POS-P2-POS 160 USEC,
            SPACING-P2-POS-P3-NEG 20 USEC, ... $
```

16.33 PULSED DOPPLER

16.33.1 Definition

A PULSED AC wave having a carrier frequency offset (Doppler shift) and a pulse delay from some reference PULSED AC signal.

16.33.2 Formal syntax

16.33.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Bandwidth	BANDWIDTH	R	SR-	Frequency
Burst length	BURST	I	SR-	Burst length
Carrier frequency (reference)	CAR-FREQ	R	SRM	Frequency
Current	CURRENT	R	SRM	Current
Doppler shift	DOPPLER-SHIFT	R	-RM	Frequency
Duty cycle	DUTY-CYCLE	R	SR-	Ratio
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Period	PERIOD	R	SRM	Time
Phase angle	PHASE-ANGLE	R	S--	Angle, plane
Power	POWER	R	SRM	Power
Pulse repetition frequency	PRF	R	SRM	Frequency
Voltage	VOLTAGE	R	SRM	Voltage

16.33.4 Rules

The reference signal is identified via the CNX REF field.

16.34 RADAR SIGNAL

16.34.1 Definition

A simulated radar range produced by delaying an output pulse from a transmitter before returning the pulse to a receiver.

16.34.2 Formal syntax

16.34.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Pulse power attenuation	ATTEN	R	S--	Ratio
Ratio of returned pulse quantity to transmitted pulse quantity	REPLY-EFF	R	S--	Ratio
Range of the simulated target	TARGET-RANGE	R	S--	Distance
Rate of change of range rate	TARGET-RANGE-ACCEL	R	S--	Distance target/ time/time
Rate of change of target range	TARGET-RANGE-RATE	R	S--	Distance/time

16.35 RAMP SIGNAL

16.35.1 Definition

A periodic waveform whose instantaneous value varies alternately and linearly between two specified values (initial and alternate). The interval required to change from the initial value to the alternate value shall not be equal to the interval to change from the alternate value to the initial value.

16.35.2 Formal syntax

16.35.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Bandwidth	BANDWIDTH	R	SR-	Frequency
Burst length	BURST	I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	SR-	Frequency
Current (AC)	CURRENT	R	SRM	Current
DC offset	DC-OFFSET	R	SRM	Voltage
Fall time	FALL-TIME	R	SRM	Time
Frequency	FREQ	R	SRM	Frequency
Negative slope	NEG-SLOPE	R	SRM	Voltage/time
Noise	NOISE	R	-RM	Ratio, voltage or current
Non-linearity	NON-LIN	R	SRM	Ratio
Peak degeneration	PEAK-DEGEN	R	SRM	Ratio
Period	PERIOD	R	SRM	Time
Phase jitter	PHASE-JIT	R	SRM	Angle, plane
Positive slope	POS-SLOPE	R	SRM	Voltage/time
Rise time	RISE-TIME	R	SRM	Time
Time asymmetry	TIME-ASYM	R	SRM	Ratio
Voltage	VOLTAGE	R	SRM	Voltage

16.36 RANDOM NOISE

16.36.1 Definition

Transient disturbances occurring unpredictably, except in a statistical sense.

16.36.2 Formal syntax

16.36.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Bandwidth	BANDWIDTH	R	SR-	Frequency
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Noise amplitude density	NOISE-AMPL-DENS	R	SRM	Voltage/frequency
Noise power density	NOISE-PWR-DENS	R	SRM	Power/frequency
Voltage	VOLTAGE	R	SRM	Voltage

16.37 RESOLVER

16.37.1 Definition

Two AC sinewave voltages whose relationships of amplitude represent the rotation shaft position of an electro-mechanical transducer.

16.37.2 Formal syntax

16.37.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Angle	ANGLE	R	SRM	Angle, plane
Angle rate	ANGLE-RATE	R	SRM	Angle, plane/ time
Bandwidth	BANDWIDTH	R	SR-	Frequency
Frequency	FREQ	R	SRM	Frequency
Harmonic distortion	HARMONICS	R	SRM	Voltage or ratio
Noise	NOISE	R	SRM	Voltage or ratio
Non-harmonic distortion	NON-HARMONICS	R	-RM	Voltage or ratio
Voltage quadrature	QUAD	R	SRM	Voltage
Voltage	VOLTAGE	R	SRM	Voltage
Zero index	ZERO-INDEX	R	S--	Angle, plane

16.38 ROTATION

16.38.1 Definition

A mathematical indication of the magnitude and direction of a rotation angle, rotation rate, or rotation acceleration of the unit under test or stimuli applied to the unit under test relative to a specific reference frame with a specified origin.

16.38.2 Formal syntax

16.38.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Angle	ANGLE	R	SRM	Angle, plane
Angle acceleration	ANGLE-ACCEL	R	SRM	Angle, plane/ time/time
X angle acceleration	ANGLE-ACCEL-X	R	SRM	Angle, plane/ time/time
Y angle acceleration	ANGLE-ACCEL-Y	R	SRM	Angle, plane/time/time
Z angle acceleration	ANGLE-ACCEL-Z	R	SRM	Angle, plane/ time/time
Angle rate	ANGLE-RATE	R	SRM	Angle, plane/time
X angle rate	ANGLE-RATE-X	R	SRM	Angle, plane/time
Y angle rate	ANGLE-RATE-Y	R	SRM	Angle, plane/time
Z angle rate	ANGLE-RATE-Z	R	SRM	Angle, plane/time
X axis angle	ANGLE-X	R	SRM	Angle, plane

continued

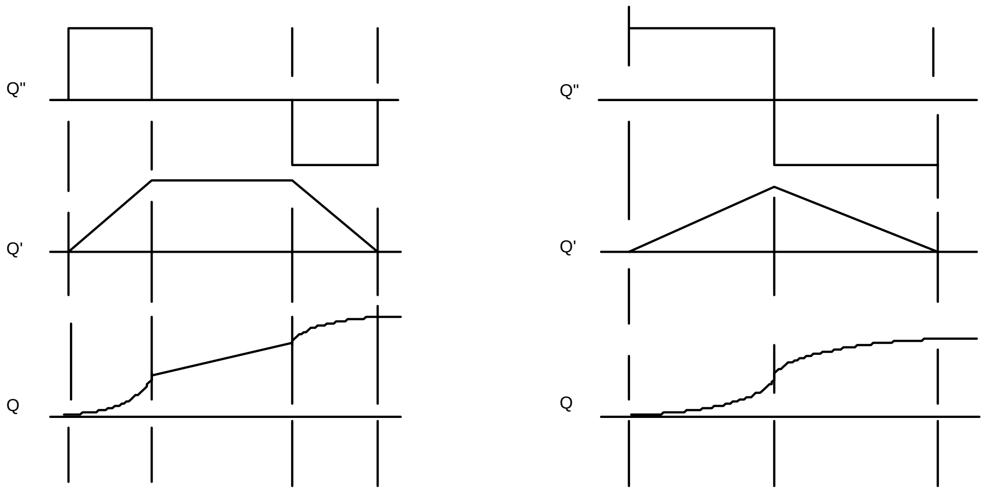
Modifier	Mnemonic	Type	Usage	Quantity
Y axis angle	ANGLE-Y	R	SRM	Angle, plane
Z axis angle	ANGLE-Z	R	SRM	Angle, plane
Earth stabilized coordinate frame	REF-INERTIAL	MO	S--	(None)
UUT stabilized	REF-UUT	MO	S--	(None)

16.38.4 Rules

16.38.4.1 General

1. Rotation angles, rates, and accelerations are specified relative to a reference coordinate frame that is stabilized relative to inertial space (an earth stabilized frame) or to the UUT (for example, an airframe stabilized reference frame). Either the modifier REF-INERTIAL or the modifier REF-UUT should be specified when using the noun ROTATION to avoid ambiguity.

2. The specification in a single statement both of a quantity and its derivative places restrictions on how the value of the quantity is obtained. When a quantity appears with more than one of its derivatives, actions associated with the way in which the process concludes are implied (by physics) beyond those explicit in the statement. For example, consider a case in which a quantity (such as linear or angular displacement), its rate, and its acceleration are all specified. In the following diagrams and statement, an arbitrary quantity is represented by Q or QUAN and its derivatives as Q' or Q".



```
APPLY, ROTATION, QUAN 'A' UNITS, QUAN-RATE 'B' UNITS/SEC,
    QUAN-ACCEL 'C' UNITS/SEC/SEC,. . . . $
```

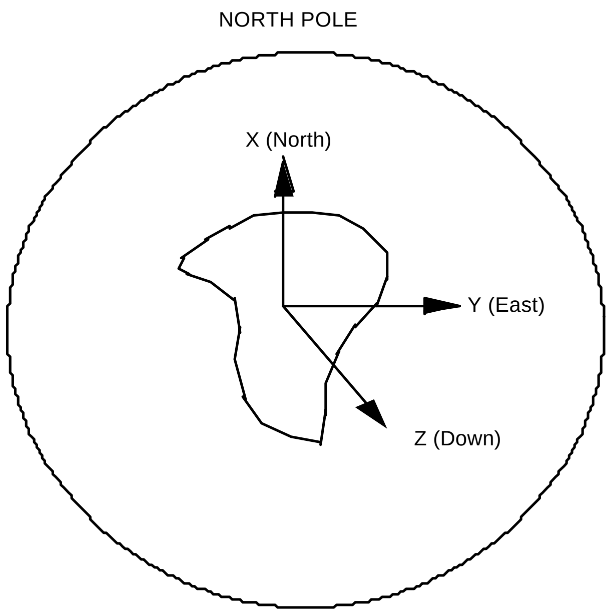
The statement implies that the highest derivative assumes its specified value until the value of the next-lower order derivative is attained, at which time the highest derivative order goes to zero and the next lower order derivative remains constant (if possible).

Conclusion of the process begins at or after the mid-point of travel of the quantity to its specified value. Since only one magnitude can be given for each derivative, the process is inherently symmetrical, and all derivatives reverse sign and/or direction as appropriate. (The simultaneous specification of differing rates of acceleration and deceleration is beyond the scope of this statement).

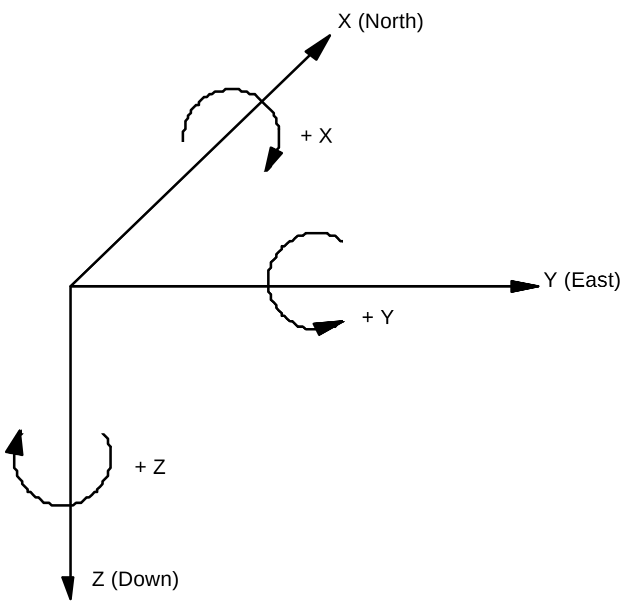
16.38.4.2 Inertially (Earth) stabilized coordinate frame REF-INERTIAL

An inertially stabilized reference frame has the characteristics that its origin may be translated in space, but angular orientation remains fixed relative to a specified object in inertial space. For C/ATLAS purposes, the inertial reference frame is defined as being stabilized relative to the surface the planet EARTH (used also in C/ATLAS to specify a zero potential electrical voltage reference). An aircraft or missile inertial platform is an example of an electromechanical device that is inertially stabilized relative to the surface of the planet Earth.

The inertially stabilized coordinate frame is defined as a three axis orthogonal reference frame with X, Y, and Z axes. The +Z axis is orientated toward the center of the earth and the +X axis toward North. The plane formed by the XY axes is thus parallel to the surface of the earth.



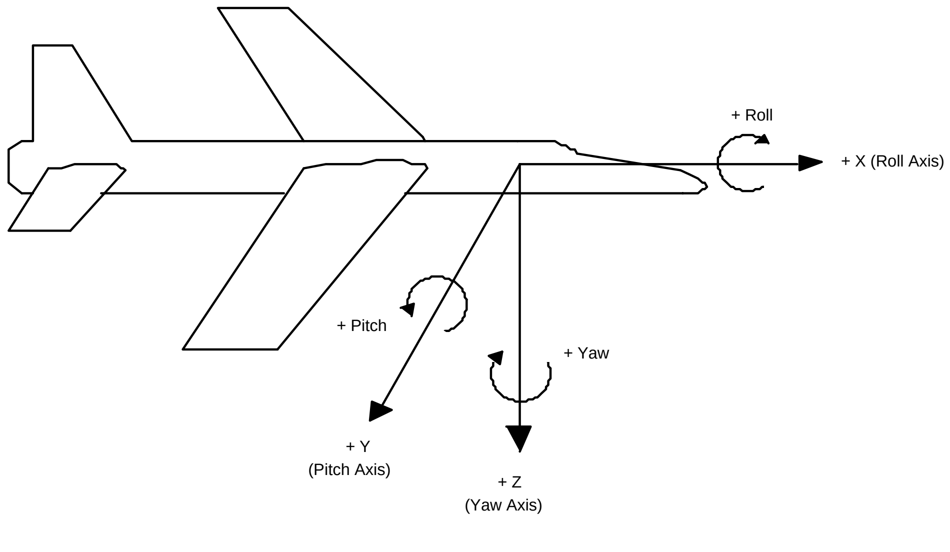
The X, Y, Z axes thus respectively point toward North, East, and Down. Angular rotations, rotational rates, and rotational accelerations are defined as rotations about the X, Y, and Z axes with positive rotation defined by the right hand rule.



The modifier REF-INERTIAL is used to specify ROTATION motion relative to the above defined coordinate frame.

16.38.4.3 Unit under test stabilized coordinate frame REF-UUT

The modifier REF-UUT is used to designate ROTATION relative to a unit under test stabilized coordinate frame. A UUT stabilized reference frame is defined as orthogonal frame affixed to the UUT.



The modifier REF-UUT is used to specify ROTATION motion relative to the above defined UUT stabilized X, Y, Z coordinate reference frame. For example, for an aircraft, X, Y, and Z could be defined as Roll, Pitch, and Yaw, respectively.

16.38.5 Notes and examples

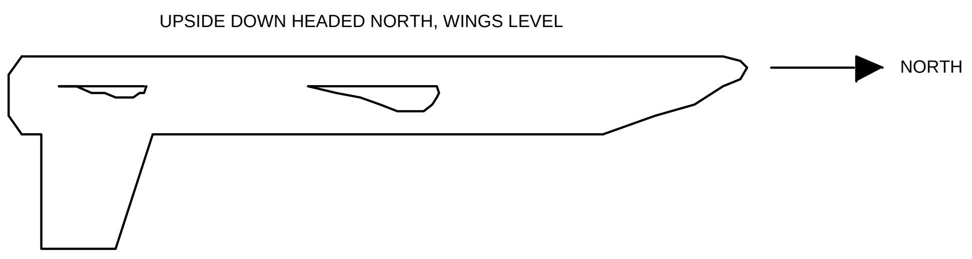
Example 1:

```
 001000 APPLY, ROTATION, REF-INERTIAL, ANGLE-X 0
           DEGREE ERRLMT +-0.1 DEG,
           ANGLE-Y 0 DEG ERRLMT +-0.1 DEG,
           ANGLE-Z 0 DEG ERRLMT +-0.1 DEG,
           CNX X ROLL Y PITCH Z YAW $
```

Statement 001000 will orient the UUT, an airplane, for example, in a wings level attitude headed towards the North Pole.

```
 001001 APPLY, ROTATION, REF-INERTIAL,
           ANGLE-X + 180 DEG ERRLMT +-0.1 DEG,
           ANGLE-RATE-X MAX 10.0 DEG/SEC,
           ANGLE-ACCEL-X MAX 1.0 DEG/SEC/SEC,
           CNX X ROLL $
```

Statement 001001 would cause the airplane to roll 180 degrees in the direction of right wing down with a maximum roll rate of 10 degrees/s and a maximum roll acceleration of 1 degree/s/s. Thus, the airplane would end up pointed north and in an upside down attitude.



Example 2:

Apply a constant rotation rate, accelerate to higher rate, decelerate to rate in opposite direction, decelerate to higher rate in the negative direction, and accelerate to zero.

Note that negative acceleration, often called deceleration, is defined as a rate of change of an angular rate that is becoming less positive with increasing time. For example, to change from a -5 degree/s rotation rate to a -10 degree/s, a negative acceleration rate is applied (i.e., a deceleration). This is easily seen if one recalls the definition of acceleration as:

accel = (velocity at T2 - velocity at T1)/(T2-T1)

where time T2 is later than time T1.

For the above case:

accel = (-10 deg/sec-(-5 deg/sec))/(T2-T1),

which is negative.

```
 090100 APPLY, ROTATION, REF-UUT,
            ANGLE-RATE-Z 5.0 DEG/SEC ERRLMT +-0.1 DEG/SEC,
            CNX Z YAW $

 090101 APPLY, ROTATION, REF-UUT,
            ANGLE-RATE-Z 10.0 DEG/SEC ERRLMT +-0.1 DEG/SEC,
            ANGLE-ACCEL-Z MAX +1.0 DEG/SEC/SEC,
            CNX Z YAW $

 090102 APPLY, ROTATION, REF-UUT,
            ANGLE-RATE-Z -5.0 DEG/SEC ERRLMT +-0.1 DEG/SEC,
            ANGLE-ACCEL-Z MIN -1.0 DEG/SEC/SEC,
            CNX Z YAW $

 090103 APPLY, ROTATION, REF-UUT,
            ANGLE-RATE-Z -10.0 DEG/SEC ERRLMT +-0.1DEG/SEC,
            ANGLE-ACCEL-Z MIN -1.0 DEG/SEC/SEC,
            CNX Z YAW $

 090104 APPLY, ROTATION, REF-UUT,
            ANGLE-RATE-Z 0.0 DEG/SEC ERRLMT +-0.1 DEG/SEC,
            ANGLE-ACCEL-Z MAX 2.0 DEG/SEC/SEC,
            CNX Z YAW $
```

Incorrect Application:

```
 090105 APPLY, ROTATION, REF-UUT,
            ANGLE-RATE-Z 10.0 DEG/SEC ERRLMT +-0.1 DEG/SEC,
            ANGLE-ACCEL-Z MIN -5.0 DEG/SEC/SEC,
            CNX Z YAW $
```

Statement 090100 would cause the UUT to rotate about the center of gravity at 5.0 degree/s (turning right) in the yaw plane. Statement 090101 would increase the rotation rate from 5.0 degree/s to 10.0 degree/s with an acceleration of no more than 1.0 degree/s/s. Statement 090102 would change the rotation rate from +10.0 degree/s to -5.0 degree/s with a deceleration of no more than 1.0 degree/s/s. Statement 090103 would increase the rotation rate in the negative direction (reduce it in the positive direction) from -5.0 degree/s to -10.0 degree/s at a deceleration of no more than 1.0 degree/s/s. Statement 090104 would change the rotation rate from -10.0 degree/s to 0.0 degree/s with an acceleration of no more than 2.0 degree/s. Statement 090105 attempts to establish a rotation 10.0 degree/s, but an acceleration of -5.0 degree/s/s is applied. A positive rotation cannot be obtained via this statement due to the incorrect semantics. (Actual rotat-ion=0)

16.39 SHORT

16.39.1 Definition

A conductor of negligible impedance between two points in an electrical circuit.

16.39.2 Formal syntax

16.39.3 Noun modifier set

No modifiers are used with SHORT.

16.40 SQUARE WAVE

16.40.1 Definition

A periodic wave that alternately assumes one of two fixed values of amplitude for equal lengths of time. The time of transition between the fixed values is relatively small with respect to the period.

16.40.2 Formal syntax

16.40.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Bandwidth	BANDWIDTH	R	SR-	Frequency
Burst length	BURST	I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	-RM	Frequency
Current	CURRENT	R	SRM	Current
DC offset	DC-OFFSET	R	SRM	Voltage
Droop	DROOP	R	-RM	Ratio
Duty cycle	DUTY-CYCLE	R	-RM	Ratio
Fall time	FALL-TIME	R	SRM	Time
Frequency	FREQ	R	SRM	Frequency
Noise	NOISE	R	-RM	Ratio, voltage or current
Overshoot	OVERSHOOT	R	-RM	Ratio
Period	PERIOD	R	-RM	Time
Phase jitter	PHASE-JIT	R	SRM	Angle, plane
Preshoot	PRESHOOT	R	SRM	Ratio
Ringing	RINGING	R	-RM	Ratio
Rise time	RISE-TIME	R	SRM	Time
Corner rounding	ROUNDING	R	-RM	Ratio
Time asymmetry	TIME-ASYM	R	SRM	Ratio
Undershoot	UNDERSHOOT	R	-RM	Ratio
Voltage	VOLTAGE	R	SRM	Voltage

16.41 STEP SIGNAL

16.41.1 Definition

A change of DC electrical potential from one level to another, either positive or negative.

16.41.2 Formal syntax

16.41.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
DC offset	DC-OFFSET	R	SRM	Voltage
Fall time	FALL-TIME	R	SRM	Time
Noise	NOISE	R	-RM	Ratio or voltage
Overshoot	OVERSHOOT	R	-RM	Ratio
Preshoot	PRESHOOT	R	-RM	Ratio
Ringing	RINGING	R	-RM	Ratio
Rise time	RISE-TIME	R	SRM	Time
Settle time	SETTLE-TIME	R	-RM	Time
Voltage	VOLTAGE	R	SRM	Voltage

16.42 SUP CAR SIGNAL (suppressed carrier signal)

16.42.1 Definition

An amplitude modulated signal that causes a phase reversal of the carrier when the amplitude of the modulating signal goes negative, which results in the suppression of the carrier.

16.42.2 Formal syntax

16.42.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Burst length	BURST	I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	SR-	Frequency
Carrier frequency	CAR-FREQ	R	SRM	Frequency
Carrier phase angle relative to signal at CAR-REF HI and LO pins*	CAR-PHASE	R	-RM	Angle, plane
Residual carrier voltage	CAR-RESID	R	SRM	Voltage
Current (RMS carrier current at peak of modulation cycle)	CURRENT	R	SRM	Current
Frequency window	FREQ- WINDOW	RR	-R-	Frequency
Modulation frequency	MOD-FREQ	R	SRM	Frequency
Modulation offset	MOD-OFFSET	R	S--	Voltage
Modulation phase angle relative to signal at MOD-REF HI and LO pins*	MOD-PHASE	R	-RM	Angle, plane
Noise	NOISE	R	-RM	Ratio, voltage or current
Standing wave ratio	SWR	R	SRM	Ratio
Voltage (RMS value of carrier at peak modulation)	VOLTAGE	R	SRM	Voltage

NOTE

* In some cases, the reference signal may also be a SUP CAR signal, providing both CAR-REF and MOD-REF simultaneously. In such cases, CAR-REF and MOD-REF can be combined into a single REF signal.

16.43 SYNCHRO

16.43.1 Definition

Three AC sinewave voltages whose relationships of amplitude represent the rotational shaft position of an electro-mechanical transducer.

16.43.2 Formal syntax

16.43.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Angle	ANGLE	R	SRM	Angle, plane
Angler rate	ANGLE-RATE	R	SRM	Angle, plane/ time
Frequency	FREQ	R	SRM	Frequency
Harmonic distortion	HARMONICS	R	SRM	Voltage or ratio
Noise	NOISE	R	SRM	Voltage or ratio
Non-harmonic distortion	NON-HARMONICS	R	-RM	Voltage or ratio
Phase shift	PHASE-SHIFT	R	SRM	Angle, plane
Voltage quadrature	QUAD	R	SRM	Voltage
Voltage	VOLTAGE	R	SRM	Voltage
Zero index	ZERO-INDEX	R	S--	Angle, plane

16.44 TACAN (tactical air navigation)

16.44.1 Description

A complete UHF polar coordinate (rho-theta) navigation system using pulse techniques. The distance (rho) function operates as distance measuring equipment (DME) and the bearing function is derived by rotating the ground transponder antenna so as to obtain a rotating multilobe pattern for coarse and fine bearing information, as defined in MIL-STD-291B.

16.44.2 Formal syntax

16.44.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Amplitude modulation shift	AM-SHIFT	R	SRM	Angle, plane
Amplitude modulation, coarse	AMPL-MOD-C	R	S--	Ratio
Amplitude modulation, fine	AMPL-MOD-F	R	S--	Ratio
Antenna speed deviation	ANT-SPEED-DEV	R	SRM	Ratio
Burst droop	BURST-DROOP	R	SRM	Ratio
Carrier amplitude	CAR-AMPL	R	SRM	Voltage or power
Carrier frequency	CAR-FREQ	R	SRM	Frequency
Communication channel	CHANNEL	MD#	S--	(None)
Continuous wave level	CW-LEVEL	R	-RM	Voltage, power or ratio

continued

Modifier	Mnemonic	Type	Usage	Quantity
Continuous wave level	CW-LEVEL	R	-RM	Voltage, power or ratio
Fall time (Pulse)	FALL-TIME	R	SRM	Time
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Harmonic voltage	HARM-***-VOLTAGE	R	-RM	Voltage or ratio
Identification signal	IDENT-SIG	MD#	S--	(None)
	IDENT-SIG-EP	R	SRM	Time
	IDENT-SIG-FREQ	R	SRM	Frequency
Interrogation jitter	INT-JITTER	R	SRM	Time or ratio
Interrogation rate	INT-RATE	R	SRM	Frequency
Magnetic bearing	MAG-BEARING	R	SRM	Angle, plane
Magnetic bearing rate	MAG-BEARING-RATE	R	SRM	Angle, plane/time
UUT mode of operation	MODE	MD#	S--	(None)
Pulse pair droop	PAIR-DROOP	R	SRM	Ratio
Pulse pair spacing	PAIR-SPACING	R	SRM	Time
Pulse spectrum meas.	PULSE-SPECT	MO	--M	(None)
Pulse spectrum measurement threshold	PULSE-SPECT-THRESHOLD	R	-R-	Voltage, power or ratio
Pulse width	PULSE-WIDTH	R	SRM	Time
Range echo pulse power deviation	RANGE- PULSE-DEV	R	SR-	Ratio
Range echo pulse	RANGE- PULSE-ECHO	R	SRM	Time or distance
Reference pulses time deviation	REF-PULSES-DEV	R	SRM	Ratio
Reference pulses excluded	REF-PULSES-EXCL	MD#	SR-	(None)
Reference pulses included	REF-PULSES-INCL	MD	SR-	(None)
Reply efficiency	REPLY-EFF	R	SRM	Ratio
Response list name for pulse spec. meas.	RESP	RA	-R-	(None)
Pulse ringing	RINGING	R	SRM	Ratio
Rise time (pulse)	RISE-TIME	R	SRM	Time
Slant range	SLANT-RANGE	R	SRM	Distance
Slant range rate	SLANT-RANGE-RATE	R	SRM	Distance/time
Squitter distribution	SQTR-DIST	R	S--	Ratio
Squitter distribution	SQTR-DIST-***	R	S--	Ratio
Squitter rate	SQTR-RATE	R	SRM	Frequency
Standing wave ratio	SWR	R	SR-	Ratio

16.44.4 Rules

1. Usage of certain modifiers is dependent on the use of other modifiers as shown in table 16-3.

2. The # of Modifier Descriptors are as follows:

For Modifier:

CHANNEL:	An integer followed by the suffix -X or -Y
IDENT-SIG:	Station identification letters
MODE:	RO, GA, AA, or AB and the suffix -I or -N
REF-PULSES-EXCL:	MAIN, AUX, or i-j-k....., where i,j,k,.....represent integer literals

3. AM-SHIFT is used only with AMPL-MOD-C, -F and either MODE RO-N or GA-N.

4. ANT-SPEED-DEV is used only with AMPL-MOD-C, -F and MODE RO-N, GA-N, AA-I, or AB- I.

5. BURST-DROOP is used only with MODE RO-N, GA-N, AA-I, or AB-I.

6. CHANNEL is used with an integer in the range from 1-126 and suffix -X or -Y.

a) MIL-STD-291B considers the TACAN signal to include 252 channels, 126 each in X and Y modes. The complete channel designation includes the X or Y mode designation.

b) CHANNEL is used only in source statements.

c) CHANNEL is not used if CAR-FREQ is used.

7. When used with the noun TACAN, HARM-***-VOLTAGE refers to harmonics of the 15Hz signal. It is used only when AMPL-MOD-C, -F is also used.

8. When INT-JITTER is used, INT-RATE shall also be used and shall be non-zero.

9. When INT-RATE is used, MODE shall be AA-N or AB-N.

10. When MAG-BEARING is used, MODE shall be RO-N, GA-N, or AB-I.

11. When MAG-BEARING is used, the specific direction is toward the ground station.

12. When MAG-BEARING-RATE is used, MODE shall be RO-N, GA-N, or AB-I.

13. When MODE is used:

Mod desc	Mode name or meaning
RO	Receive only
GA	Ground-to-air
AA	Air-to-air
AB	Air-to-air beacon
I	Inverse
N	Normal

14. When SLANT-RANGE is used, MODE shall not be RO-N.

15. When SLANT-RANGE is used, zero range corresponds to a transponder time delay of 62 µsec.

16. When SLANT-RANGE-RATE is used, MODE shall not be RO-N.

17. RANGE-PULSE-DEV is used only for MODE RO-N or GA-N and with RANGE-PULSE- ECHO.

18. RANGE-PULSE-ECHO is used only for MODE RO-N or GA-N.

19. The echo is less than or equal to the true range response.

20. REPLY-EFF may not be used with MODE RO-N.

21. SQTR-DIST-*** and SQTR-RATE may be used only with TACAN MODE RO-N, GA-N, AA- N, or AB-N.

22. (***) An <unsigned integer number> shall be included at this point indicating the harmonic that is being specified for HARM-***-VOLTAGE, and indicates to points within, between, or outside the lines for SQTR-DIST-***.

16.44.5 Notes and examples

The noun TACAN refers to the family of signals used as inputs and outputs with TACAN equipment, as defined in MIL-STD-291B.

The tactical air navigation (TACAN) system utilizes special ground stations and aircraft terminals to provide range and bearing information to aircraft. The ground stations transmit radio-frequency (RF) pulses that are amplitude modulated to provide bearing information. The amplitude modulation (AM) is produced by a parasitic reflector array rotating about the antenna radiating element. The array consists of one 15 Hz and nine 135 Hz reflectors. As the pattern from the 15 Hz reflector passes through magnetic east azimuth, a main reference burst (MRB) is transmitted. As the pattern from the 135 Hz reflectors passes through east, an auxiliary reference burst (ARB) is transmitted, except for the pattern coincident with the 15 Hz pattern. This produces a total of one MRB and eight ARB bursts per antenna rotation. The bursts are distinguished by different pulse spacing. The airborne receiving equipment determines the aircraft bearing from the ground station by measuring elapsed time, first, from the MRB to the 0 degree phase of the 15 Hz component and, second, from the ARB to 0 degree of the 135 Hz component.

Range is determined by the aircraft transmitting an interrogation signal to the ground station and measuring the time delay until it receives a response. Aircraft interrogations are automatically staggered in time to prevent erroneous tracking by one aircraft of the response to another.

A modification to TACAN permits air-to-air operation in addition to air-to-ground. This enables aircraft to range on each other and to form master/slave combinations in which slave aircraft determine range relative to the master aircraft. Another modification, Y

Mode, effectively doubles the number of TACAN RF channels available to provide 252 (126 transmit-receive channel pairs).

The original mode is referred to as X Mode. Some TACAN equipment contains additional capability to operate in modes such as inverse and beacon, as described below. These modes are not defined in MIL-STD-291B.

Some aircraft are equipped with rotating antennas similar to the ground station antenna, but having only a single parasitic reflector (15 Hz). Aircraft using such antennas are said to operate in the inverse mode. They can determine the bearing of an unmodulated transmitter. They also can impose bearing information on transmissions to cooperating aircraft. A variation known as beacon mode transmits random pulse patterns (squitters) during idle periods. An aircraft in inverse mode can determine the bearing to an aircraft in beacon mode. TACAN-equipped aircraft can determine the bearing to an aircraft operating in both beacon and inverse modes simultaneously.

Some TACAN equipment may be able to operate as terminals in a data link. Data link operation is not supported by the existing TACAN modifiers.

Example 1. To apply the signal seen by an airborne receiver transmitter operating in the inverse mode generated by a cooperating aircraft operating in the beacon mode:

```
APPLY, TACAN, REF-PULSES-EXCL 1-2-3-4-5-6-7-8-9-10,
  AMPL-MOD-C 15 PC, CAR-AMPL 100 MW,
  CHANNEL 64-X,
  MODE AB-I, REPLY-EFF 100 PC, INT-RATE 22 HZ,
  SQTR-RATE 2700 HZ, SLANT-RANGE 100 N-MILE,
  MAG-BEARING 80 DEG, CNX .....$
```

For this signal, all reference pulses are deleted and the simulated cooperating aircraft is operating in the inverse-beacon mode. It is necessary to use circumlocution in this instance since the stimulus shall exhibit the composite effects of both a cooperating aircraft and a local spinning antenna.

Table 16-3. TACAN modifier conditional usage

(The symbols X and Z used in the table have the following meaning.)

	"M"	"N"	"P"
"A"	X	X	
"B"	X		Z
"C"		Z	

- Modifier "A" can be used only if modifiers "M" and "N" are also used.

- Modifier "B" can be used only if modifier "M" is used and "P" is not used.

- Modifier "C" cannot be used if modifier "N" is used.

	AMPL - MOD	CAR-FREQ	INT-RATE	MODE RO-N	MODE GA-N	MODE AA-N	MODE AB-N	MODE AA-I	MODE AB-I	PULSE-SPECTRUM	THRES-HOLD	RESP
AM-SHIFT	X							(a)				
AMPL-MOD				(a)	(a)			(a)	(a)			
ANT- SPEED-DEV	X			(a)	(a)			(a)	(a)			
BURST- DROOP				(a)	(a)			(a)	(a)			
CHANNEL		Z										
HARM-***-VOLTAGE	X							(a)				
IDENT-SIG				(b)	(b)	(b)	(b)					
IDENT-SIG-EP				(b)	(b)	(b)	(b)					
INT-JITTER			X	Z								
INT-RATE						X						
MAG- BEARING				(c)	(c)				(c)			
MAG- BEARING-RATE				(c)	(c)				(c)			
PULSE- SPECT											X	X
REPLY-EFF				(d)								
SLANT- RANGE				(d)								
SLANT- RANGE - RATE				(d)								
SQTR-DIST				(b)	(b)	(b)	(b)					
SQTR-RATE				(b)	(b)	(b)	(b)					

NOTES
(a) Shall use MODE RO-N, GA-N, AA-I, or AB-I.
(b) Shall use MODE RO-N, GA-N, AA-N, or AB-N.
(c) Shall use MODE RO-N, GA-N, or AB-I.
(d) Cannot be used with MODE RO-N.

When an aircraft operates in an inverse mode, four different signals are present, as follows:

a) Its own modulated transmission as seen in the adjacent airspace

b) Its own unmodulated transmissions as seen in the cable to its routing antenna

c) The unmodulated transmissions from the cooperating aircraft as seen in the adjacent airspace

d) The modulated transmissions from the cooperating aircraft as seen in its antenna cable

Thus, the signal AM characteristics depend not only on the operating mode of the stations, but on the reference point at which the signals are seen. Similar circumlocutions will often be necessary for other varieties of inverse signal.

Example 2. To apply the signal from a ground station seen by an airborne receiver-transmitter:

```
APPLY, TACAN, REF-PULSES-INCL, IDENT-SIG LAX,
    AMPL-MOD-C 15.0 PC, AMPL-MOD-F 17.0 PC,
    CAR-AMPL 0 DBM,
    CHANNEL 126-X, MODE GA-N, REPLY-EFF 60 PC,
    SQTR-RATE 2700 HZ, SLANT-RANGE 72 N-MILE,
    MAG-BEARING 34 DEG,
    CNX.....$
```

16.45 TIME INTERVAL

16.45.1 Definition

The time between two events.

16.45.2 Formal syntax

16.45.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Time interval	TIME	R	-RM	Time

16.45.4 Rules

See EVENT TIMING.

16.46 TRIANGULAR WAVE SIGNAL

16.46.1 Definition

A periodic waveform whose instantaneous value varies alternately and linearly between two specified values (initial and alternate). The interval required to change from the initial value to the alternate value shall be equal to the interval to change from the alternate value to the initial value.

16.46.2 Formal syntax

16.46.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Bandwidth	BANDWIDTH	R	SRM	Frequency
Burst length	BURST	I	SR-	Burst length
Burst repetition rate	BURST-REP-RATE	R	SR-	Frequency
Current	CURRENT	R	SRM	Current
DC offset	DC-OFFSET	R	SRM	Voltage
Frequency	FREQ	R	SRM	Frequency
Noise	NOISE	R	-RM	Ratio, voltage or current
Non-linearity	NON-LIN	R	SRM	Ratio
Peak degeneration	PEAK-DEGEN	R	SRM	Ratio
Period	PERIOD	R	SRM	Time
Phase jitter	PHASE-JIT	R	SRM	Angle, plane
Time asymmetry	TIME-ASYM	R	SRM	Ratio
Voltage	VOLTAGE	R	SRM	Voltage

16.47 TURBINE ENGINE DATA

16.47.1 Definition

A signal transmitted to or from the UUT conveying information about the performance characteristics of an engine or associated equipment.

16.47.2 Formal syntax

16.47.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Force	FORCE	R	SRM	Force
Force-rate	FORCE-RATE	R	SRM	Force/time
Fuel supply	FUEL-SUPPLY	MD#	S--	(none)
Power level angle	PLA	R	SRM	Angle, plane or ratio
PLA-rate	PLA-RATE	R	S--	Angle, plane/time or ratio/time
Output power	POWER	R	-RM	Power
Rotor speed	ROTOR-SPEED	R	SRM	Angle, plane/time or ratio
Shaft speed	SHAFT-SPEED	R	SRM	Angle, plane/time or ratio
Thrust	THRUST	R	-RM	Force
Torque	TORQUE	R	SRM	Torque

16.47.4 Rules

1. The specification of both a PLA-RATE and a PLA value or of both a FORCE-RATE and a FORCE value in a single stimulus statement places restrictions on the manner in which the specified values may be achieved. See Rule 2 under the noun ROTATION.

2. This noun only uses pin descriptor TO.

3. The # of Modifier descriptors for FUEL-SUPPLY are: ON, OFF.

16.47.5 Notes and examples

The following are examples of various TURBINE ENGINE DATA statements:

Example 1. To apply a simulated rotor speed to a fuel control:

```
000202 APPLY, TURBINE ENGINE DATA,
          ROTOR-SPEED 2000 RPM,
          CNX TO N2 $
```

Example 2. To verify that the accessory shaft is turning at 500 rpm:

```
000203 VERIFY, (SHAFT-SPEED INTO 'MPW'),
           TURBINE ENGINE DATA,
           UL 510 RPM LL 490 RPM,
           SHAFT-SPEED MAX 600 RPM,
           CNX TO ACC-DRIVE $
```

Example 3. To measure the thrust of an engine:

```
000204 MEASURE, (THRUST INTO 'TVAL'),
           TURBINE ENGINE DATA,
           THRUST MAX 40000 N, CNX TO THRUST-SENSOR1 $
```

Example 4. To define a shaft speed virtual resource:

```
000205 REQUIRE, ... , SENSOR, (SHAFT-SPEED),
           TURBINE ENGINE DATA,
           CAPABILITY,
           SHAFT-SPEED RANGE 500 RPM TO 10000 RPM,
           CNX TO ... $
```

16.48 VIBRATION

16.48.1 Definition

A physical characteristic, either applied to or measured on the UUT, that represents a periodic motion at a specified point on the UUT.

16.48.2 Formal syntax

16.48.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Frequency	FREQ	R	SRM	Frequency
Frequency window	FREQ-WINDOW	RR	-R-	Frequency
Amplitude acceleration	VIBRATION-ACCEL	R	SRM	Distance/time/time
Amplitude	VIBRATION-AMPL-x*	R	SRM	Distance
Amplitude, rate of change	VIBRATION-RATE	R	SRM	Distance/time

16.48.4 Rules

1. This noun uses the pin descriptor TO.

2. (*) One of the following suffixes shall be used: P, PP, TRMS.

16.48.5 Notes and examples

Example 1. To measure vibration in terms of amplitude:

```
000601 MEASURE, (VIBRATION-AMPL-PP INTO 'NEW'),
           VIBRATION,
           VIBRATION-AMPL-PP MAX 3.2 MM,
           FREQ-WINDOW RANGE 10 HZ TO 40 HZ,
           CNX TO ACCESS-PORT $
```

Example 2. To apply a vibration to a UUT:

```
000602 APPLY, VIBRATION, VIBRATION-AMPL-P 5 MM,
           FREQ 3 HZ, CNX TO MOUNT-4 $
```

16.49 VOR (VHF omnidirectional radio range)

16.49.1 Description

A navigation aid operating at VHF and providing radial lines of position in any direction as determined by bearing selection within the receiving equipment. It emits a (variable) modulation whose phase relative to a reference modulation is different for each bearing of the receiving point from the station as defined in ARINC Specification 579.

16.49.2 Formal syntax

16.49.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Carrier amplitude	CAR-AMPL	R	S--	Voltage or power
Carrier frequency	CAR-FREQ	R	S--	Frequency
Frequency window	FREQ-WINDOW	R	-R-	Frequency
Harmonic distortion	HARMONICS	R	-RM	Voltage, power or ratio
Identification signal	IDENT-SIG	MD#	S--	(None)
Identification signal Freq. (steady tone)	IDENT-SIG-FREQ	R	S--	Frequency
Identification signal modulation	IDENT-SIG-MOD	R	S--	Ratio
Modulation distortion	MOD-DIST	R	-RM	Voltage or ratio
Non-harmonic distortion	NON-HARMONICS	R	-RM	Voltage, power or ratio
Radial bearing	RADIAL	R	S--	Angle, plane
Radial bearing rate	RADIAL-RATE	R	S--	Angle, plane/Time
Reference phase modulation frequency	REF-PHASE-FREQ	R	S--	Frequency
Subcarrier frequency	SUB-CAR-FREQ	R	S--	Frequency
Subcarrier modulation	SUB-CAR-MOD	R	S--	Ratio
Standing wave ratio	SWR	R	SRM	Ratio
Variable phase modulation frequency	VAR-PHASE-FREQ	R	S--	Frequency
Variable phase modulation amplitude	VAR-PHASE-MOD	R	S--	Ratio

16.49.4 Rules

1. The # of Modifier Descriptors are as follows:

For modifier IDENT-SIG: Station identification letters.

2. RADIAL indicates the magnetic bearing from the ground station.

3. REF-PHASE-FREQ is the frequency of a modulating signal that is used as a reference in determining the phase relationship of another signal.

4. SUB-CAR-FREQ specifies the modulating frequency of the subcarrier.

16.49.5 Notes and examples

1. VOR provides bearing information relative to its location based on magnetic north at the VOR. No range information is provided. The governing document is ARINC 579.

2. The VOR radiates a radio frequency carrier in the band 111.975-117.975 MHz, at 50KHz increments up to 117.950 MHz, with which are associated two separate 30Hz modulations. The phase of one of these modulations is independent of the point of observation (reference phase). The phase of the other modulation (variable phase) is such that, at a point of observation, it differs from the reference phase by an angle equal to the bearing of the point of observation with respect to the VOR.

3. The two separate modulations consist of the following:

a) A subcarrier of 9960 Hz, frequency modulated at 30 Hz, modulating the carrier to a nominal depth of 30%.

1) This 30 Hz component is fixed without respect to azimuth and termed the Reference Phase for conventional VOR.

2) This 30 Hz component varies with azimuth and is termed the Variable Phase for Doppler VOR.

b) A 30 Hz amplitude-modulation component modulating the carrier to a nominal depth of 30%.

1) This 30 Hz component is caused by a rotating antenna producing a change in phase with azimuth and is termed the Variable Phase for conventional VOR.

2) This 30 Hz component, having a constant phase relative to azimuth and a constant amplitude, is radiated omnidirectionally and is termed the Reference Phase for Doppler VOR.

4. Additional modulations that may be present are a communication channel, which may have a peak modulation depth up to 30%, and an identification signal of two or three letters of Morse code. Depth of modulation due to the identification signal is normally 10%; when there is a communications channel, the modulation depth may be up to 20%.

16.50 WAVEFORM

16.50.1 Definition

A time-varying electrical potential with an arbitrary wave shape.

16.50.2 Formal syntax

16.50.3 Noun modifier set

Modifier	Mnemonic	Type	Usage	Quantity
Bandwidth	BANDWIDTH	R	SR-	Frequency
Burst length	BURST	I	S--	Burst length
Current	CURRENT	R	S--	Current
DC offset	DC-OFFSET	R	SR-	Voltage
Fundamental frequency	FREQ	R	SR-	Frequency
Fundamental period	PERIOD	R	SR-	Time
Power	POWER	R	S--	Power
Response list identifier	RESP	RA	-R-	(None)
Measured voltage amplitude samples	SAMPLE	RA	--M	(None)
Sample spacing*	SAMPLE-SPACING	R	-R-	Time

continued

Modifier	Mnemonic	Type	Usage	Quantity
Duration of set of samples*	SAMPLE-TIME	R	-R-	Time
Stimulus list identifier	STIM	RA	S--	(None)
Voltage	VOLTAGE	R	SR-	Voltage
Sample width	SAMPLE-WIDTH	R	-R-	Time
Voltage interpolation	VOLTAGE-RAMPED	MO	S--	(None)
Requirement	VOLTAGE-STEPPED	MO	S--	(None)
Noise	NOISE	R	-R-	Ratio,voltage, power or current

NOTE

* Figure 16-4 illustrates the definition of PERIOD and SAMPLE-TIME with SAMPLE-SPACING.

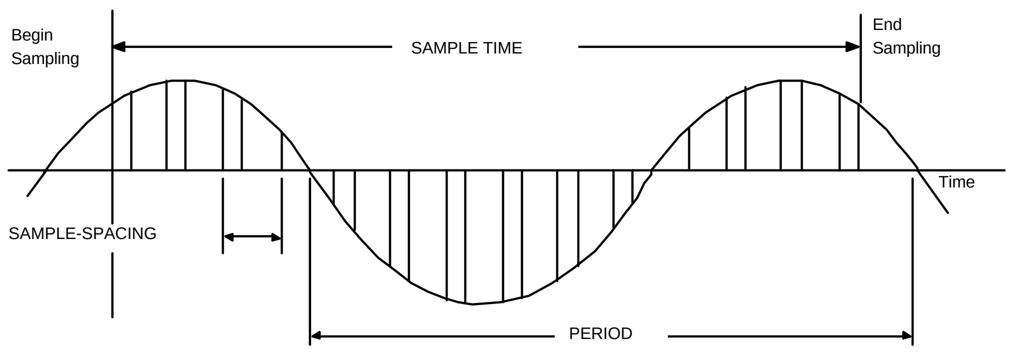


Figure 16-4. Definition of PERIOD and SAMPLE-TIME with SAMPLE-SPACING

16.50.4 Rules

16.50.4.1 Rules applicable to both SOURCE and SENSOR type statements

1. The DC-OFFSET is added to the list of stimulus values in a source statement and is subtracted from the waveform before the samples are stored in a SENSOR statement.

2. The modifier VOLTAGE is used to specify a VOLTAGE RANGE or a VOLTAGE MAX and VOLTAGE MIN pair. In these fields, the voltages referred to are the whole set of instantaneous voltages associated with the STIM or RESP field.

16.50.4.2 Rules applicable to SOURCE type statements

1. The STIM field has the form STIM <array range> and is required in SOURCE statements. This field specifies the storage location and the number of values that define the stimulus waveform.

2. The fundamental frequency (or period) of the waveform (i.e., the repetition frequency of the set of the amplitude numbers) is specified as FREQ (or PERIOD).

3. The time increment between successive amplitude numbers is equal to the PERIOD, or the reciprocal of FREQ, divided by the number of elements in the list range of the STIM field.

16.50.4.3 Rules applicable to SENSOR type statements

1. The measured characteristic SAMPLE causes the specified number of amplitude samples to be taken and stored in the <array range> included in the RESP field.

2. The verb VERIFY cannot be used with SAMPLE.

3. If SAMPLE-SPACING is not specified, the time increment between successive samples is equal to SAMPLE-TIME divided by the number of samples as specified by the number of elements in the list range of the RESP field.

16.50.5 Notes and examples

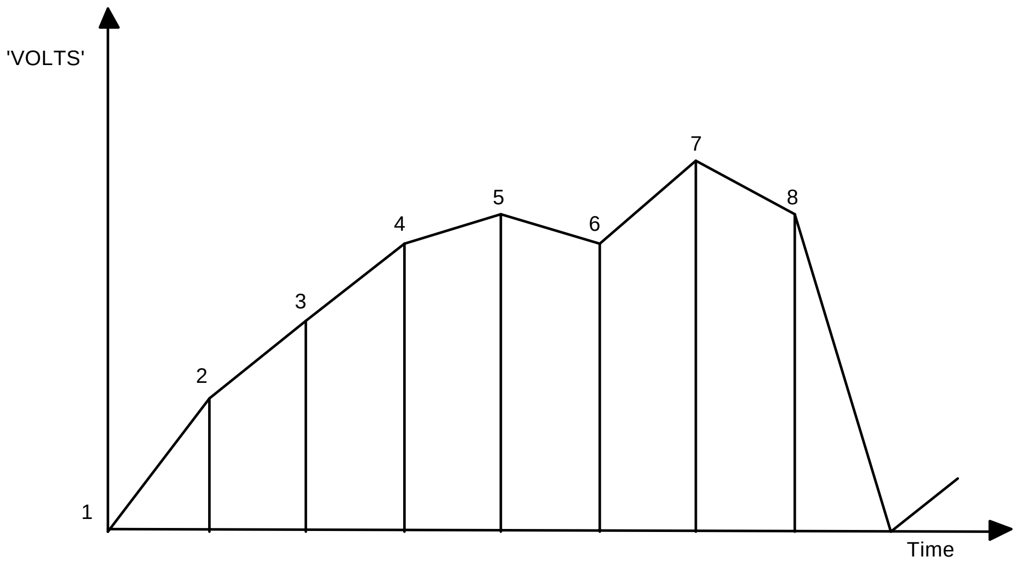


Figure 16-5. Example waveform.

The waveform shown above is defined by the array (0,0.8,1.2,1.6,1.9,1.7,2.4,2.2) at equal time intervals. This can be generated by the following procedural statement:

```
200001 APPLY, WAVEFORM, VOLTAGE-RAMPED,
           STIM 'VOLTS' (1 THRU 8), VOLTAGE RANGE 0V TO 3V,
           FREQ 125KHZ, CNX HI J1-1 LO J1-2 $
```

If the array contains the values 0, 4, 6, 7, 6, 4, 0, -4, -6, -7, -6, -4, then the two waveforms as shown below in figure 16-6 can be generated using the following procedural statements:

```
300001 APPLY, WAVEFORM, VOLTAGE_STEPPED,
           STIM 'VOLTS' (1 THRU 12),
           VOLTAGE RANGE -7V TO 7V, FREQ 125 KHZ,
           CNX HI J1-1 LO J1-2 $
```

```
300002 APPLY, WAVEFORM, VOLTAGE_RAMPED,
           STIM 'VOLTS' (1 THRU 12),
           VOLTAGE RANGE -7V TO 7V, FREQ 125 KHZ,
           CNX HI J1-1 LO J1-2 $
```

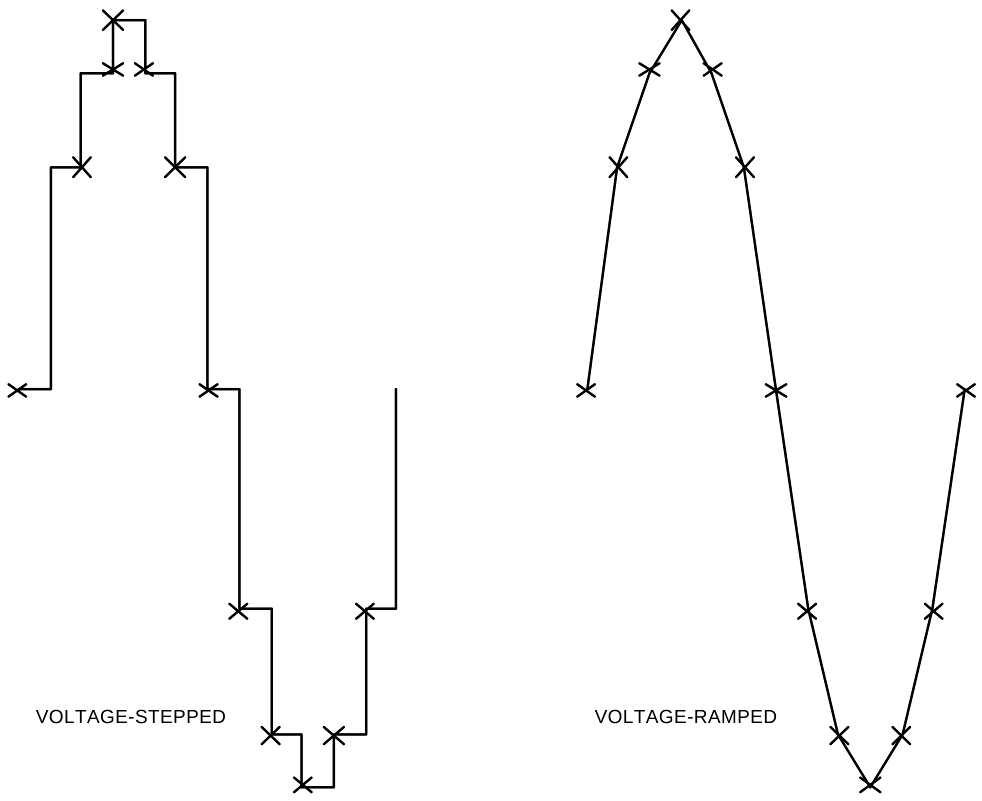


Figure 16-6. Voltage stepped and voltage ramped waveforms

17.0 Noun modifier definitions

This clause lists and defines the noun modifiers used with the nouns identified in clause 16. The modifiers are listed by their reserved C/ATLAS word (i.e., mnemonic). In any syntax diagram in which there is a <modifier mnemonic>, a reference may be made to any of these reserved words or to a <modifier name> that has been introduced in an EXTEND statement.

17.1 Mnemonics for pulse-type signals

A graphical representation of some of the mnemonics used with pulse-type signals is depicted in figure 17-1.

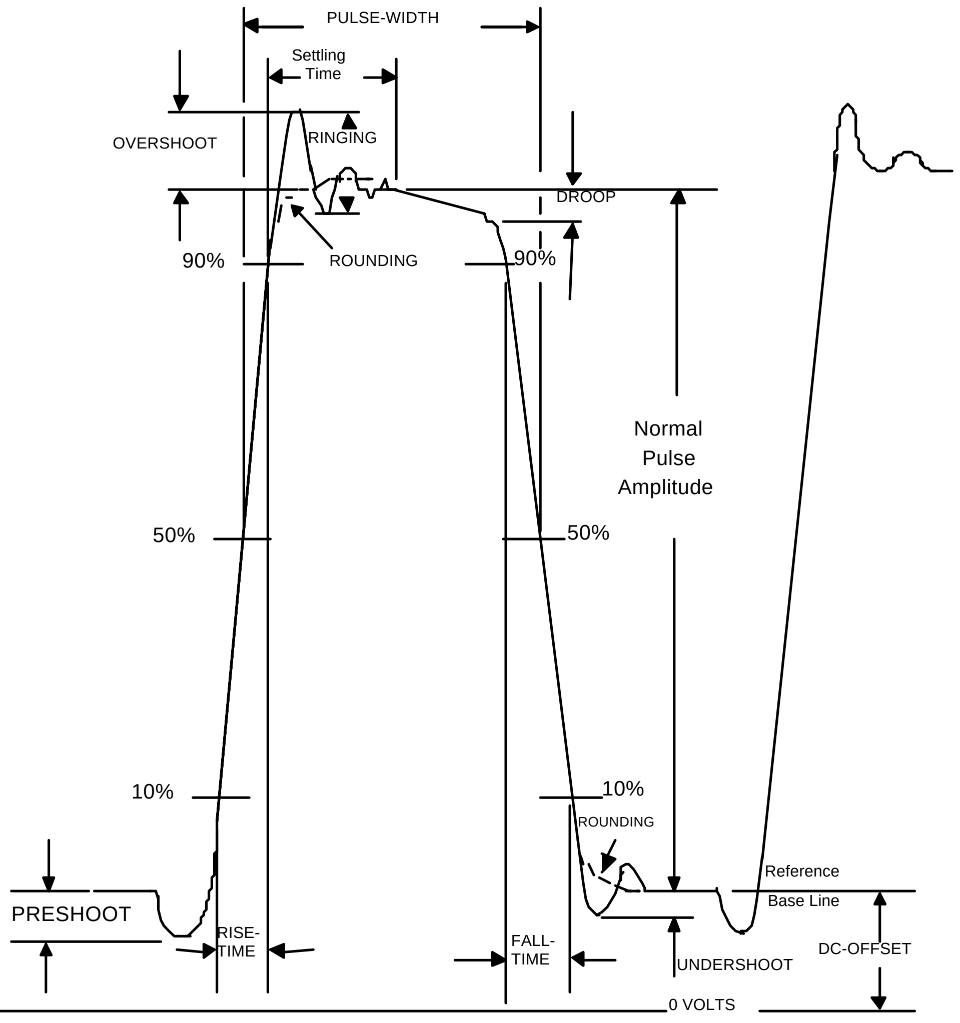


Figure 17-1. Pulse type signal mnemonics graphical representation

17.2 Modifier prefixes and suffixes

Many of the noun-modifier mnemonics may have prefixes or suffixes to further define the type, use, and application of the mnemonics. These prefixes and suffixes are defined in this clause. In general, in the noun-modifier lists in clause 16 and in the modifier definitions given subsequently in this clause, only the basic mnemonic (without prefixes or suffixes) is included.

17.2.1 Pressure and temperature prefixes

The prefixes TOTAL- and STATIC- can be used with the modifiers PRESS-A, PRESS-G, PRESS- RATE, and TEMP in order to indicate the following:

a) TOTAL The pressure or temperature value includes the effects of fluid movement (ram effect).

b) STATIC The pressure or temperature value disregards the effects of fluid movement (ram effect).

When neither prefix is used, the modifier refers to the static pressure or temperature.

Examples:

```
TOTAL-TEMP
TOTAL-PRESS-A
STATIC-PRESS-G
TOTAL-PRESS-RATE
```

17.2.2 Amplitude modifier suffixes

The amplitude modifier suffixes can be used only with the modifiers VOLTAGE, CURRENT, POWER, and NOISE.

17.2.2.1 Voltage and current suffixes

These suffixes, as applied to VOLTAGE and CURRENT, are defined as follows:

The type of parameter may be designated as suffixes to applicable modifier mnemonics. For example, the following suffixes can be applied to the modifier VOLTAGE:

a) VOLTAGE (no suffix) The parameter indicating the rms value.

b) VOLTAGE-TRMS The true rms parameter that indicates an accurate value regardless of the distortion present in the signal.

c) VOLTAGE-AV The parameter indicating the average signal value.

d) VOLTAGE-PP The parameter indicating the peak-to-peak value of a signal. As shown in figure 17-2, this dimension is measured from the positive peak to the negative peak of a signal.

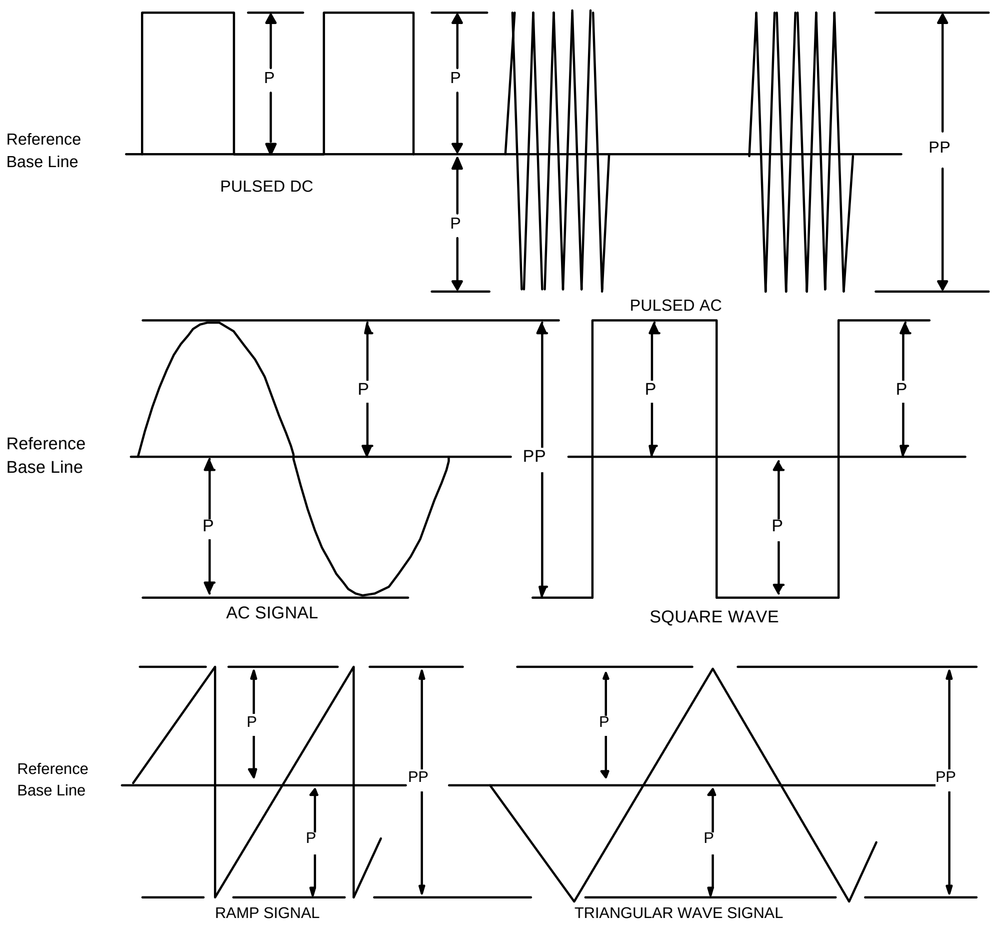


Figure 17-2 Peak and Peak-to-Peak Graphical Representation

e) VOLTAGE-P The parameter indicating the peak value of a signal. As illustrated in figure 17-2, this parameter is measured from the reference base line of a signal to its positive or negative peak. The reference base line of a signal is defined as the signal's zero value level. In practical terms, the level of the reference base line is determined in one of four ways, as follows:

1) It is stated explicitly as a fixed DC-OFFSET without error limit, range, max, or min. The actual value may be determined at execute time and held in a <data store>.

2) Except in the case of PULSED DC, PULSED AC, and SQUARE WAVE, if the DC-OFFSET is not stated then the reference base line is taken as the level at which for half the cycle, the signal is greater than the level and for half the cycle, it is less than the level.

3) In the special case of a SQUARE WAVE, the reference base line is defined as what would be 50% of the normal pulse amplitude if it were a PULSED DC, with an appropriately adjusted reference base line.

4) For PULSED DC or PULSED AC, the reference base line is directly determined as the actual zero value level during the "space" time.

f) VOLTAGE-P-POS, VOLTAGE-P-NEG The parameters indicating the positive peak and the negative peak, respectively. They are normally used for asymmetric waveforms (see figure 17-3).

g) VOLTAGE-IN-PHASE, VOLTAGE-QUAD The parameters indicating the in-phase or quadrature component of voltage. A REF connection field shall be provided for the reference signal. When either of these suffixes is combined with any of the preceding suffixes, these suffixes are written last. For example: VOLTAGE-TRMS-IN-PHASE.

h) VOLTAGE-INST The parameter indicating the instantaneous value of the signal with reference to zero volts (see figure 17-3).

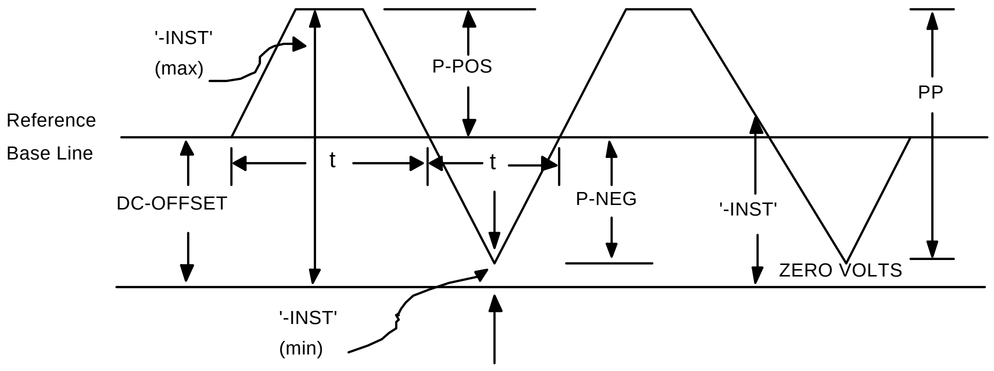


Figure 17-3. VOLTAGE-INST graphical representation

17.2.2.2 Power suffixes

The Amplitude Modifier Suffixes, as applied to the modifier POWER, are defined as follows:

a) POWER-AV The parameter indicating the average value of the power.

b) POWER-P The parameter indicating the peak value of power.

17.2.2.3 Noise suffixes

The amplitude modifier suffixes, as applied to NOISE, are defined as follows:

a) NOISE (no suffix) The true rms value of the noise.

b) NOISE-AV The mean value of the noise.

c) NOISE-P The peak value of the noise.

d) NOISE-PP The peak-to-peak value of the noise.

e) NOISE-TRMS The true rms value of the noise.

17.2.3 Phase identifier suffixes

The following phase identifiers may be used as suffixes with modifiers such as VOLTAGE or CURRENT to form a modifier that refers to a particular phase of a multi-phase signal:

PHASE-A	PHASE-AB	PHASE-AC
PHASE-B	PHASE-BC	PHASE-CB
PHASE-C	PHASE-CA	PHASE-BA

NOTE Use of + or - signs to indicate in-phase or out-of-phase AC, respectively, is permitted. Any of the legal phase identifiers established for the UUT connector field can be used in this phase mnemonic. For example: PHASE-A -180, PHASE-CB 90, etc.

17.2.4 Distance suffixes

The terms X, Y, Z, and R, designating the type of parameter dimensions, are included as suffixes to the modifier mnemonic DISTANCE that applies to the noun DISPLACEMENT. The suffixes X, Y, and Z indicate displacement along each of three orthogonal axes that form a right-handed triad. The suffix R indicates displacement along a radial axis.

Examples:

1. DISTANCE-X, DISTANCE-Y, DISTANCE-Z The parameters indicating linear displacement in each of three orthogonal directions.

2. DISTANCE-R The parameter indicating linear displacement in a radial direction from an origin. This should be associated with ANGLE characteristics.

NOTE DISTANCE without a suffix means that the movement is a scalar quantity.

17.2.5 Angle suffixes

The terms THETA and PHI, designating the direction of a DISPLACEMENT in polar coordinates, are included.

Example:

ANGLE-THETA, ANGLE-PHI The parameters associated with the direction of a polar DISPLACEMENT, which is otherwise designated by DISTANCE-R.

17.2.6 Pulse identifier suffix

The pulse identifier suffix Pxxx is used for correlating a modifier to a single pulse in a pulse train. Each single pulse is identified by an integer, with P1 representing the first pulse in the pulse train. The modifier SPACING uses a suffix of the form -Pxxx-POS-Pyyy-NEG. Pulse identifier suffixes are written last when they are used with other suffixes.

Examples:

```
PULSE-WIDTH-P3  SPACING-P1-POS-P2-POS
VOLTAGE-P-P3    SPACING-P5-POS-P8-NEG
```

17.2.7 Reference identifier suffix

The REF identifier may be used as a suffix with noun modifiers when a REF followed by a <conn set> subfield is used in the <conn> field. When a REF suffix is used with other suffixes, it shall be the last suffix.

<mnemonic> - REF defines the characteristics of the REFERENCE signal.

Example:

```
     MEASURE, (ANGLE INTO 'VAR'), SYNCHRO,
          ANGLE RANGE 0 DEG TO 100 DEG,
          FREQ-REF 400 HZ,
          VOLTAGE-REF 26 V,
          CNX X J1 Y J2 Z J3
          REF HI J6 LO J7 $
```

NOTE In this example, the synchro reference is coming from UUT pins J6 and J7.

17.2.8 Composite vector suffixes

The terms IN-PHASE and QUAD may be used with FLUX-DENS to indicate phase relationships with respect to a specified reference of the energy flow rate per unit of surface area. IN-PHASE represents the component in phase with the specified reference and QUAD represents the component in quadrature with the specified reference.

Examples:

FLUX-DENS-IN-PHASE

FLUX-DENS-QUAD

17.2.9 Quiescent signal suffix

The quiescent signal suffix QUIES is used in source and load statements to define the value to be assumed by a particular signal characteristic, as follows:

a) When a statement includes an initial coincidence synchronization, the <statement characteristics> field that contains a suffix QUIES defines the signal value in the period prior to synchronization. The term initial coincidence synchronization is defined in 14.4.

b) When a statement includes a <gate field>, the <statement characteristics> field that contains a suffix QUIES defines the signal value existing outside the time slot defined by the <gate field>. The <statement characteristics> field that does not include a noun-modifier with the suffix QUIES defines the signal value within the time slot defined by the <gate field>.

When using the suffix QUIES, the following Rules apply:

1. Where appropriate, the suffix QUIES is written immediately following the suffixes defined in 17.2.3 through 17.2.8.

2. For each <statement characteristics> field including a noun-modifier having the suffix QUIES, there shall be another <statement characteristics> field including the same noun-modifier without the suffix QUIES, so that the signal level is specified both during the active and quiescent periods.

17.2.10 Bi-phase digital signal suffixes

<mnemonic>-ONE and <mnemonic>-ZERO are used with bi-phase digital signals to identify the signal levels that are associated with the 1 and 0 levels in a TRANS field. The suffixes can be employed with any mnemonic that carries logic information.

17.2.11 Bandwidth prefixes

A COMPLEX SIGNAL is the resultant signal created by operation(s) upon a set of input signals. Some portion of the resultant signal may exist in specific domains. In such instances, the following prefixes are used:

a) VIDEO Indicates the signal characteristic mode of the portion of the COMPLEX SIGNAL existing in the video domain.

b) RF Indicates the signal characteristic mode of the portion of the COMPLEX SIGNAL existing in the RF domain.

17.3 Digital noun-modifiers

Modifiers used in digital test statements enter the overall statement syntax via one of two routes.

a) Any modifier field that conforms to the standard pattern (i.e., modifier mnemonic followed by a quantity followed by a dimensional identifier), enters via <statement characteristics> (see 14.1) and - <real characteristic subfield> (see 14.2) This includes modifiers defining both digital and analog characteristics of the digital signals, such as VALUE, SKEW-TIME, PRF, etc.

b) Any digital modifier that does not conform to the standard pattern enters via language items <statement characteristics> (see 14.1) and <digital characteristic subfield> (see 14.3) or <sync subfield> (see 14.4)

17.4 Noun modifiers

17.4.1 AC-COMP (AC component)

17.4.1.1 Definition

An AC voltage or current component that is present in a signal other than a defined AC SIGNAL.

17.4.2 AC-COMP-FREQ (AC component frequency)

17.4.2.1 Definition

The frequency of an AC component voltage or a current signal.

17.4.3 AGE-RATE (aging rate)

17.4.3.1 Definition

A change in frequency expressed as a function of time.

17.4.4 ALT (altitude)

17.4.4.1 Definition

The vertical distance above (or below) standard sea level determined by measurement of atmospheric pressure in the vicinity of the object and computation from the height/pressure laws of a standard model atmosphere.

17.4.5 ALT-RATE (altitude rate)

17.4.5.1 Definition

Rate of change in altitude.

17.4.6 AM-COMP (AM component)

17.4.6.1 Definition

The peak-to-peak value of an amplitude-modulation component expressed in volts or as a ratio of this value to the fundamental wave peak amplitude.

17.4.6.2 Rules

AM-COMP shall be expressed as a peak-to-peak value in volts or as a ratio of peak-to-peak amplitude to the carrier-wave peak amplitude.

17.4.7 AM-SHIFT (amplitude modulation shift)

17.4.7.1 Definition

The distance from the positive-slope zero crossing of the 15 Hz AM signal to the nearest positive-slope crossing of the 135 Hz AM signal expressed in degrees, where a cycle of the 15 Hz modulation is 360 degrees.

17.4.7.2 Rules

AM-SHIFT is used only with AMPL-MOD and either MODE RO-N or GA-N.

17.4.8 AMPL-MOD, -C, -F (amplitude modulation; coarse, fine)

17.4.8.1 Definition

Variations in the amplitude of the fundamental signal caused by amplitude variations of some other signal.

17.4.8.2 Rules

1. The suffixes -C and -F are used only with the noun TACAN.

a) These suffixes designate, respectively, the amplitude modulation produced by the 15 Hz signal (coarse modulation) and the 135 Hz signal (fine modulation).

b) AMPL-MOD is used only with MODE RO-N, GA-N, AA-1, or AB-1.

2. The modifier AMPL-MOD (without suffix) is not currently used with any noun.

17.4.9 ANGLE, ANGLE-s

17.4.9.1 Definition

The application or measurement of the angle of rotation of a mathematical vector or mechanical device. Angle rotation about a three axis orthogonal reference frame may be specified. Subclauses 16.38.4.1 through 16.38.5 define the allowable coordinate reference frames and the meaning of various angle modifiers.

17.4.10 ANGLE-ACCEL, ANGLE-ACCEL-w (angle acceleration)

17.4.10.1 Definition

With respect to time, the rate of change in angular rate or the second derivative of angle.

17.4.10.2 Rules

17.4.10.3 Notes and examples

1. Positive acceleration is defined by the right hand rule referenced to the axis of rotation.

2. Subclauses 16.38.4.1 through 16.38.5 define the allowable coordinate reference frames and the meanings of the various angle modifiers.

17.4.11 ANGLE-RATE, ANGLE-RATE-w

17.4.11.1 Definition

With respect to time, the rate of change of angle or the first derivative of angle.

17.4.11.2 Rules

17.4.11.3 Notes and examples

Subclauses 16.38.4.1 through 16.38.5 define the meanings of the various angle modifiers and coordinate frames.

17.4.12 ANT-SPEED-DEV (antenna speed deviation)

17.4.12.1 Definition

As used with the noun TACAN, ANT-SPEED-DEV is the antenna speed deviation from 15 revolutions per second, expressed as a percentage. The modifier causes the repetition rate of the reference bursts and the frequency of the amplitude modulation to vary uniformly.

17.4.12.2 Rules

ANT-SPEED-DEV is used only with AMPL-MOD and MODE RO-N, GA-N, AA-I or AB-I.

17.4.13 ATMOS (standard model atmosphere)

17.4.13.1. Definition

This modifier defines the required standard atmosphere that is to be employed in converting altitudes and airspeeds to absolute and gauge pressures. This modifier uses no units.

17.4.13.2 Rules

17.4.13.3 Notes and examples

1. The ARDC standard atmosphere is defined in specification SURVEY 115, CAMBRIDGE RESEARCH CENTRE.

2. The ICAN standard atmosphere is defined in documents of the INTERNATIONAL COMMITTEE ON AIR NAVIGATION from 1924.

3. The ICAO standard atmosphere is defined in specification STANDARD ATMOSPHERE 1964 DOCUMENT 7488.2.

4. The WADC standard atmosphere is defined in specification WADC TECHNICAL REPORT 54-215.

17.4.14 ATTEN (attenuation)

17.4.14.1 Definition

The ratio between the power returned and the power transmitted.

17.4.15 AVG-SWEEPS

17.4.15.1 Definition

Using swept measurement techniques, this indicates that the measurement of the COMPLEX SIGNAL is to be averaged over a specified number of sweeps.

17.4.16 BANDWIDTH

17.4.16.1 Definition

The difference between the upper and lower frequencies of a frequency selective circuit where 3 dB attenuation occurs.

17.4.17 BAROMETRIC-PRESS (barometric pressure)

17.4.17.1 Definition

The ambient barometric pressure in the test cell.

17.4.17.2 Rules

17.4.17.3 Notes and examples

It is normally expressed in inches of mercury or pounds per square inch.

17.4.18 BIT-MATCH-THRESHOLD

17.4.18.1 Definition

This modifier indicates the upper threshold of the number of words of a digital response array that do not correlate, bit to bit, when being matched against a reference pattern array. A MASK may optionally be specified to eliminate don't care bits from the pattern correlation.

17.4.19 BIT-RATE

17.4.19.1 Definition

The speed at which digital data bits are transmitted.

17.4.20 BURST (Burst Length)

17.4.20.1 Definition

The number of pulses or cycles of a stimulus waveform to be applied. The BURST modifier is applicable to repetitive signals where a requirement might arise for a limited burst of the signal to be injected rather than a continuous signal. A BURST field indicates the number of cycles of a simple signal such as AC, the number of modulation-envelope cycles of a modulated signal, or the number of pulses of a pulsed signal that are to be generated. The BURST signal starts and ends at the reference base level unless a phase angle modifier is employed if appropriate to the particular noun.

17.4.20.2 Rules

1. The critical action of the stimulus statement is the appearance of the first pulse or cycle of the stimulus signal that appears at the UUT. Execution of the next statement starts following this critical action. When the BURST modifier is used with single-action verb

sequences, the pulses or cycles are initiated in accordance with the following two cases affecting BURST, which depend on the availability of an EVENT MONITOR as a trigger mechanism.

a) No EVENT MONITOR available:

   The pulses or cycles are initiated when the BURST source has completed the SETUP action between the CONNECTED and PREPARED state in the SOURCE path of the state diagram in figure 11-1 (A CONNECT action to change from the SET to the PREPARED state in the LOAD path of the state diagram in figure 11-1 is not the proper trigger action sequence because it leads to "hot switching". This sequence may be available in certain older ATE and should not be used).

b) EVENT MONITOR available:

   The pulses or cycles of a BURST source are initiated when triggered by the occurrence of the governing event. This can only happen when the SOURCE has readied the ENABLE, EVENT action in accordance with the state diagram in figure 11-2. A SENSOR for a BURST shall be in the READY state before sensing of a BURST can occur in accordance with the state diagram in figure 11-4.

17.4.21 BURST-DROOP

17.4.21.1 Definition

The ratio of the difference between the peak power of the strongest and weakest pulses in a burst to the average power in the burst expressed in percent or decibels.

17.4.21.2 Rules

1. BURST-DROOP is used only with MODE RO-N, GA-N, AA-I, or AB-I.

17.4.22 BURST-REP-RATE (burst repetition rate)

17.4.22.1 Definition

The average number of bursts per unit time.

17.4.23 CAP (Capacitance)

17.4.23.1 Definition

The ratio of electrical charge (q) to electrical potential (V) in a circuit (C = q/V).

17.4.24 CAR-AMPL (carrier amplitude)

17.4.24.1 Definition

The amplitude of the unmodulated carrier wave.

17.4.25 CAR-FREQ (carrier frequency)

17.4.25.1 Definition

The time-average frequency of the carrier wave signal in the absence of modulation.

17.4.26 CAR-HARMONICS (carrier harmonics)

17.4.26.1 Definition

Any integral multiple of the defined carrier frequency occurring in the carrier signal.

17.4.27 CAR-PHASE (carrier phase)

17.4.27.1 Definition

Phase angle of the carrier relative to the signal at REF HI and REF LO pins.

17.4.28 CAR-RESID (carrier residual)

17.4.28.1 Definition

Voltage of a residual carrier signal.

17.4.29 CHANNEL

17.4.29.1 Definition

Identifier of the frequency or communications channel to be used.

17.4.29.2 Rules

When used with the noun TACAN, CHANNEL is used with an integer in the range from 1 to 126 and suffix -X or -Y.

a) MIL-STD-291B considers the TACAN signal to include 252 channels, 126 each in X and Y modes. The complete channel designation includes the X or Y mode designation.

b) CHANNEL is used only in source statements.

c) CHANNEL is not used if CAR-FREQ is used.

17.4.30 COMPL (complement)

17.4.30.1 Definition

The voltage level of a differential digital signal that represents a complementary value.

17.4.30.2 Rules

17.4.30.3 Notes and examples

See VOLTAGE-ONE

17.4.31 CONDUCTANCE

17.4.31.1 Definition

The magnitude of the real (resistive) part of admittance.

17.4.32 COUNT

17.4.32.1 Definition

Number of events or items.

17.4.33 CREST-FACTOR

17.4.33.1 Definition

The ratio of the peak or maximum value of a signal to the RMS value.

17.4.34 CURRENT

17.4.34.1 Definition

The rate of flow of electrical charge. In the case of DC signals, the rate is unvarying. In the case of AC signals, a distortionless sinusoidal flow rate is assumed.

17.4.35 CURRENT-LMT (current limit)

17.4.35.1 Definition

Maximum safe current applicable without damage.

17.4.36 CURRENT-ONE, CURRENT-ZERO, CURRENT-QUIES

17.4.36.1 Definition

The current associated with the digital logic one, logic zero, and quiescent logic states, respectively.

17.4.36.2 Rules

1. These modifiers are used for signals that have the digital data coded in the voltage or current amplitudes. These modifiers

a) Are optional for all signals produced by voltage sources

b) Are required (including CURRENT-QUIES as applicable) for all SOURCE logic signals and all SENSOR LOGIC DATA signals produced by current sources

c) Cannot be used as a SENSOR measured characteristic

2. Form: Standard C/ATLAS modifier field. Normally includes a MIN or MAX subfield.

3. The following rules apply to all logic signals driven from voltage sources.

a) CURRENT-ONE, CURRENT-ZERO, and CURRENT-QUIES define, respectively, the UUT current requirements associated with VOLTAGE-ONE, VOLTAGE-ZERO, and VOLTAGE-QUIES voltage levels as follows:

   For SOURCE, the maximum and/or minimum current required by a properly functioning UUT.

   For SENSOR, the current limitations of the UUT output.

b) The direction of current flow is shown by the current polarity as follows:

+ sign Normal current flow (i.e., the direction of current flow if the load were a simple resistor).

- sign Reverse current flow (i.e., a current sink).

4. For logic signals driven from current sources, the roles of the VOLTAGE-ONE/VOLTAGE-ZERO/VOLTAGE-QUIES modifiers and the CURRENT-ONE/CURRENT- ZERO/CURRENT-QUIES modifiers become reversed. These current modifiers, then, adhere to the form, use, and rules for voltage sources.

17.4.36.3 Notes and examples

1. For a SOURCE:

```
VOLTAGE-ONE 4.5 V ERRLMT +-0.5 V,
VOLTAGE-ZERO 0.5 V ERRLMT +-0.5 V,
CURRENT-ONE MAX 10 MA, CURRENT-ZERO MAX 100 MA
```

This represents a voltage source stimulus with normal current flow (i.e., flow from the test system to the UUT). When the voltage is in the binary 1 state, up to 10 mA can be drawn by the UUT; for the binary 0 state, up to 100 mA can be drawn.

2. For a SOURCE:

```
VOLTAGE-ONE -4.5 V ERRLMT +-0.5 V,
VOLTAGE-ZERO -0.5 V ERRLMT +-0.5 V,
CURRENT-ONE MAX -100 MA, CURRENT-ZERO MAX -10 MA
```

These current modifiers indicate that the test system voltage source shall act as a current sink and accept positive current from the UUT.

3. For a SENSOR:

```
VOLTAGE-ONE 1.0 V ... , VOLTAGE-ZERO 5.0 V ... ,
CURRENT-ONE MAX 10 MA, CURRENT-ZERO MAX 1 MA
```

The test system shall not draw more than 10 mA and 1 mA, respectively, for the binary 1 and binary 0 states.

4. For a SOURCE:

```
CURRENT-ONE 10.0 MA ERRLMT +-1 MA,
CURRENT-ZERO 1.0 MA ERRLMT +-1 MA,
VOLTAGE-ONE MAX 50 V, VOLTAGE-ZERO MAX 10 V
```

This indicates that the test system signal source shall be a current source that delivers 10.0 mA +- 1.0 mA at a voltage not exceeding 50 V for the binary 1 state and 1.0 mA +- 0.1 mA at a voltage not exceeding 10 V for the binary 0 state.

5. A SOURCE statement for a voltage-source waveform defines a logic signal that is applied to the UUT by the test system. Because the UUT input is an active circuit, current flow for a positive voltage may be either from the test system to the UUT (defined as "Normal" in rule 3b) or from the UUT to the test system (defined as "Reverse" flow). In this latter case the test system is operating as a current sink. A similar situation exists regarding logic signals from the UUT to the test system.

17.4.37 CW-LEVEL (continuous-wave level)

17.4.37.1 Definition

The transmitted output power level between pulse transmissions, expressed in absolute units of Watts, dBm, dBW, or dBk, or in relative units of dB below peak pulse power.

17.4.38 DBL-INT (double interrogation)

17.4.38.1 Definition

The time between the main interrogation pulses and the repeated interrogation pulses.

17.4.38.2 Rules

Use of this modifier specifies that the repeated interrogation pulses are to be present and specifies the time in seconds. If this modifier is omitted, the repeated interrogation pulses are not present.

17.4.39 DC-OFFSET

17.4.39.1 Definition

The DC voltage or current level of a defined reference base line. DC-OFFSET voltage or current is measured from zero volts to reference base line as shown in figure 17-1.

17.4.40 DDM (difference in depth of modulation)

17.4.40.1 Definition

The percentage modulation depth of the larger signal minus the percentage modulation depth of the smaller signal, divided by 100. The modifier uses no units.

17.4.41 DEBRIS-COUNT

17.4.41.1 Definition

The number of foreign particles per unit of volume of a fluid.

17.4.41.2 Rules

17.4.41.3 Notes and examples

Normally expressed as counts/litre.

17.4.42 DEBRIS-SIZE

17.4.42.1 Definition

The average size of foreign particles contained in a fluid.

17.4.42.2 Rules

17.4.42.3 Notes and examples

Normally expressed in units of metres.

17.4.43 DEWPOINT

17.4.43.1. Definition

The temperature at which a gas would be saturated with the current moisture content.

17.4.44 DISS-FACTOR (dissipation factor)

17.4.44.1 Definition

The ratio of the series resistance to the series reactance, expressed as a percentage of the series reactance.

17.4.45 DISTANCE

17.4.45.1 Definition

The measurement of the magnitude of travel or separation in space.

17.4.46 DISTORTION (total distortion)

17.4.46.1 Definition

The total of all or a chosen portion of the distortion elements, such as HARMONICS, NON- HARMONICS, NOISE, and PHASE-JITTER, in a signal.

17.4.47 DOMINANT-MOD-SIG (dominant modulating signal)

17.4.47.1 Definition

This modifier refers to that modulating signal in a multi-modulating-signal system that has the greatest depth of modulation. When used with the noun ILS, this modifier refers to either the 90 Hz or the 150 Hz modulating signal, whichever has the greater depth of modulation (units are Hertz).

17.4.48 DOPPLER-BANDWIDTH

17.4.48.1 Definition

The difference between the upper and lower frequency of the returned signal where the 6dB attenuation occurs.

17.4.49 DOPPLER-FREQ (Doppler frequency)

17.4.49.1 Definition

The received frequency of a signal shifted from the emitted signal by a delta frequency called the doppler shift.

17.4.50 DOPPLER-SHIFT

17.4.50.1 Definition

The magnitude of the change in the observed frequency of a wave due to the Doppler effect.

17.4.51 DROOP

17.4.51.1 Definition

The difference between the normal pulse amplitude and the amplitude to which the instantaneous pulse amplitude sags or falls at the trailing edge of the pulse, expressed as a percentage of the normal pulse amplitude (see figure 17-1).

17.4.52 DUTY-CYCLE

17.4.52.1 Definition

The ratio between the on-time of square wave or pulsed DC signals and the total period.

17.4.53 EFF (efficiency)

17.4.53.1 Definition

The ratio of the output to the input of a conversion process. If used with the noun LIGHT, the ratio of output radiant flux weighted according to the CIE Standard Photometric Observer to the corresponding input radiant flux.

17.4.54 EFFICACY

17.4.54.1 Definition

The quotient of the luminous flux emitted and the power consumed.

17.4.55 FALL-TIME

17.4.55.1 Definition

The time interval during which the instantaneous amplitude of a pulse decreases (falls towards the reference base line) from 90 percent to 10 percent of the normal pulse amplitude (see figure 17-1).

17.4.56 FIELD-STRENGTH

17.4.56.1 Definition

The strength of an electromagnetic field, specified in units of volts per metre.

17.4.57 FLUID-TYPE

17.4.57.1 Definition

The kind of fluid for the noun FLUID SIGNAL.

17.4.58 FLUX-DENS (flux density)

17.4.58.1 Definition

The energy flow rate per unit of surface area.

17.4.59 FM-COMP (frequency modulation component)

17.4.59.1 Definition

The peak-to-peak frequency deviation of a frequency-modulation component that is present in a signal other than an FM SIGNAL.

17.4.60 FORCE

17.4.60.1 Definition

The instantaneous change in momentum of a body produced by or applied to the UUT, expressed in newtons.

17.4.61 FORCE-RATE

17.4.61.1 Definition

The rate at which a force shall be changed from its former value to the value specified by the FORCE modifier, expressed in newtons/second.

17.4.61.2 Rules

This modifier can appear only in a stimulus statement in conjunction with the FORCE modifier.

17.4.62 FREQ (frequency)

17.4.62.1 Definition

The rate at which a periodic electrical function is repeated.

17.4.63 FREQ-DELTA

17.4.63.1 Definition

The algebraic difference of two frequency values, between a complex signal being sensed and a reference signal, or between two specified power or frequency marks of a complex signal being sensed.

17.4.63.2 Rules

The <mark descriptor subfield> shall be used with this modifier in the latter case.

17.4.64 FREQ-DEV

17.4.64.1 Definition

The peak difference between the instantaneous frequency of the modulated wave and the carrier frequency.

17.4.65 FREQ-ONE (frequency-one), FREQ-ZERO (frequency-zero), FREQ-QUIES (frequency quiescent)

17.4.65.1 Definition

The FREQ-ONE, FREQ-ZERO, FREQ-QUIES modifiers define the frequency associated with each logic state.

17.4.65.2 Rules

1. These modifiers are required for all LOGIC DATA signals in which the digital logic is represented by a frequency modulation. They have the form of a standard C/ATLAS modifier field.

2. FREQ-ONE and FREQ-ZERO are used with two-state logic to define the frequency (pulse repetition rate) associated with the binary 1 and binary 0 states, respectively. FREQ-ONE, FREQ-ZERO, and FREQ-QUIES are used with three-state logic to define the frequency associated with each logic state.

17.4.65.3 Notes and examples

The conventional modifiers of VOLTAGE and CURRENT generally are used with frequency modulation logic signals.

17.4.66 FREQ-PAIRING (frequency pairing)

17.4.66.1 Definition

When used with the noun DME, FREQ-PAIRING specifies the navigation receiver frequency selection (which has been universally accepted and paired with DME operating channels and is defined in ICAO Annex 10, Section 3.5).

When used with the noun ILS, FREQ-PAIRING specifies that the glide-slope frequency is paired with the localizer frequency as determined by the frequency pairing table, as defined in ICAO Annex 10, Section 3.1.

17.4.67 FREQ-RESOLUTION (frequency resolution)

17.4.67.1 Definition

The minimum discrete change in frequency.

17.4.68 FREQ-WINDOW (frequency window)

17.4.68.1 Definition

A frequency range. Only signals inside this range contribute to a measurement result.

17.4.68.2 Rules

17.4.68.3 Notes and examples

The parameters NON-HARMONICS, HARMONICS, NOISE-AMPL, etc. may be measured over a frequency range defined by the FREQ-WINDOW modifier. The modifier FREQ or CAR-FREQ defines the frequency of the fundamental from which the harmonics are derived. The modifier BANDWIDTH defines the +3dB to -3dB frequency range of the detector that is normally used to sweep the FREQ-WINDOW to determine the value of the required characteristic. The fundamental itself may, but need not, lie in the FREQ-WINDOW. The FREQ-WINDOW is always used with a RANGE subfield.

```
000200 MEASURE, (NON-HARMONICS INTO 'VAR'), AC SIGNAL,
            FREQ 10 MHZ, VOLTAGE 10 DBM,
            NON-HARMONICS RANGE -60 DBM TO -20 DBM,
            FREQ-WINDOW RANGE 11.0 MHZ TO 30 MHZ,
            BANDWIDTH 100 HZ,
            CNX HI P1-1 LO P1-0 $
```

17.4.69 FUEL-SUPPLY

17.4.69.1 Definition

The state of an on-off valve located on a gas turbine engine fuel control.

17.4.69.2 Rules

This modifier may appear only in a source-type statement.

17.4.69.3 Notes and examples

It has a mechanical linkage to a pilot control and has two positions, ON, meaning the valve is open and fuel may flow, and OFF, meaning that the valve is closed.

17.4.70 GLIDE-SLOPE

17.4.70.1 Definition

This modifier identifies the statement as applying to the vertical guidance subsystem of the ILS system. The modifier uses no units.

17.4.71 HARMONICS (harmonic distortion)

17.4.71.1 Definition

A sinusoidal component of a periodic wave or quantity having a frequency that is an integral multiple of the fundamental frequency.

17.4.72 HARM-***-PHASE (phase of the *** harmonic)

17.4.72.1 Definition

The symbol *** represents an integer, n, that identifies a harmonic component of the signal. HARM-***-PHASE designates the phase of the nth harmonic relative to the fundamental, where n = 1 represents the waveform fundamental component.

17.4.73 HARM-***-POWER (power of the *** harmonic)

17.4.73.1 Definition

The symbol *** represents an integer, n, that identifies a harmonic component of the signal. HARM-***-POWER designates the power of the nth harmonic, where n = 1 represents the waveform fundamental component.

17.4.74 HARM-***-VOLTAGE (voltage of the *** harmonic)

17.4.74.1. Definition

The symbol *** represents an integer, n, that identifies a harmonic component of the signal. HARM-***-VOLTAGE designates the rms voltage amplitude of the nth harmonic, where n = 1 represents the waveform fundamental component.

17.4.74.2 Rules

When used with the noun TACAN, HARM-***-VOLTAGE refers to harmonics of the 15 Hz signal. It is used only when AMPL-MOD is also used.

17.4.75 HI-MOD-FREQ (high modulation frequency)

17.4.75.1 Definition

This modifier establishes a high modulation frequency that is nominally 150 Hz.

17.4.76 HUMIDITY

17.4.76.1 Definition

The absolute water vapor content of air, measured in units of mass of water per volume of air.

17.4.77 IAS (indicated airspeed)

17.4.77.1 Definition

The indicated airspeed that is equivalent to the true forward velocity that corresponds to the required excess of gauge-pressure over absolute pressure.

17.4.78 IDENT-SIG, IDENT-SIG-EP (identification signal)

17.4.78.1 Definition

The IDENT-SIG, modifier identifies and establishes parameters of the identification signal.

17.4.78.2 Rules

1. The following rules apply when an IDENT-SIG is used with the nouns TACAN and DME.

   a) When the IDENT-SIG modifier is omitted, no identification signal is sent.

   b) IDENT-SIG (without any suffix) indicates that the signal is broadcast continually except when the reference bursts take precedence.

   c) IDENT-SIG with a modifier descriptor of one to four letters indicates that the signal is sent every 37.5 s and that the specified letters are the signal transmitted.

   d) The suffix -EP specifies the spacing, in seconds, between the ID pulses and the equalizing pulses. Zeros means an absence of equalizing pulse pairs.

   e) IDENT-SIG and IDENT-SIG-EP are used only with MODE RO-N, GA-N, AA-N, or AB-N. IDENT-SIG-EP is not used with the noun DME.

17.4.79 IDENT-SIG-FREQ (identification signal frequency)

17.4.79.1 Definition

The frequency of the signal used for identification in air navigation systems.

17.4.79.2 Rules

IDENT-SIG-FREQ designates a constant tone at the specified frequency for the identification signal.

17.4.80 IDENT-SIG-MOD (identification signal modulation)

17.4.80.1 Definition

The percentage modulation of a signal containing coded information used for identification in air navigation systems.

17.4.81 ILLUM (illumination)

17.4.81.1 Definition

The quotient of the luminous flux incident on an element of a surface and the area of that element.

17.4.82 IND (inductance)

17.4.82.1 Definition

The ratio of electrical potential to the rate of change of current in an electro-magnetic circuit.

17.4.83 INTERMEDIATE-FREQUENCY

17.4.83.1 Definition

A frequency to which a signal wave is shifted locally as an intermediate step in the conversion process of a transmission or received signal.

17.4.84 INT-JITTER (interrogation jitter)

17.4.84.1 Definition

The random variation of the interrogation rate (INT-RATE) expressed as time or percent of the average time between interrogations.

17.4.84.2 Rules

When INT-JITTER is used INT-RATE shall also be used and shall be non-zero.

17.4.85 INT-RATE (interrogation rate)

17.4.85.1 Definition

INT-RATE specifies the rate at which range interrogations are issued.

17.4.85.2 Rules

When INT-RATE is used, MODE shall be AA-N or AB-N.

17.4.86 LO-MOD-FREQ (low modulation frequency)

17.4.86.1 Definition

This modifier establishes a low modulation frequency that is nominally 90 Hz.

17.4.87 LOCALIZER

17.4.87.1 Definition

This modifier identifies the statement as applying to the localizer subsystem of the ILS system, which is the lateral guidance portion. The modifier uses no units.

17.4.88 LUMINANCE

17.4.88.1 Definition

The measure of the brightness of a surface whether illuminated or self-luminous. In a given direction, it is the quotient of the luminous intensity of an infinitesimal surface element and the area of the element, as projected orthogonally to the given direction expressed in units of luminance (nit).

17.4.89 LUM-FLUX (luminous flux)

17.4.89.1 Definition

The capacity of a light source to produce visual sensation.

17.4.90 LUM-INT (luminous intensity)

17.4.90.1 Definition

The luminous power of a light source.

17.4.91 LUM-TEMP (luminous temperature)

17.4.91.1 Definition

The temperature of a full radiator that has the same luminance as the light source for a given wavelength.

17.4.92 MAG-BEARING (magnetic bearing)

17.4.92.1 Definition

The angle between magnetic north and a specific direction in a clockwise direction. The angle is expressed in degrees.

17.4.92.2 Rules

1. When MAG-BEARING is used, MODE shall be RO-N, GA-N, or AB-I.

2. When MAG-BEARING is used, the specific direction is toward the ground station.

17.4.93 MAG-BEARING-RATE (magnetic bearing rate)

17.4.93.1 Definition

The time rate of change of the magnetic bearing (MAG-BEARING), expressed in degrees per second.

17.4.93.2 Rules

When MAG-BEARING-RATE is used, MODE shall be RO-N, GA-N, or AB-I.

17.4.94 MARKER-BEACON

17.4.94.1 Definition

This modifier identifies the statement as applying to the marker-beacon subsystem of the ILS system, which is a portion of the glide path control. The modifier uses no units.

17.4.95 MARKER-FREQ

17.4.95.1 Definition

Typically used with swept signals, it is the frequency at which a marker signal is to be output.

17.4.96 MASS-FLOW

17.4.96.1 Definition

The mass of a fluid that flows in unit time through the connection specified in the CNX field.

17.4.96.2 Rules

17.4.96.3 Notes and examples

Normally expressed as KG/HR.

17.4.97 MEAN-MOD (mean modulation)

17.4.97.1 Definition

This modifier defines the mean modulation of the two modulating signals, expressed as one- half of the sum of the carrier percentage modulation caused by the 90 Hz (nominal) modulating signal and the carrier percentage modulation caused by the 150 Hz (nominal) modulating signal. For example, if the modulation percentage of the 90 and 150 Hz components are 20 and 30%, the mean modulation is 25%.

17.4.98 MIN-POWER

17.4.98.1 Definition

Using swept measurement techniques, this is the lowest power encountered during a sweep through the frequency range.

17.4.99 MIN-SWEEPS

17.4.99.1 Definition

This modifier indicates a data accumulation technique of a sweep measurement in which the MIN signal values at each frequency step are stored over a specified number of sweeps.

17.4.100 MIN-VOLTAGE

17.4.100.1 Definition

Using swept measurement techniques, this is the lowest voltage encountered during a sweep through a frequency range.

17.4.101 MOD-AMPL (modulation amplitude)

17.4.101.1 Definition

The ratio of the modulation signal amplitude to the carrier wave amplitude, expressed as a percentage of the carrier wave amplitude.

17.4.102 MOD-DIST (modulation distortion)

17.4.102.1 Definition

Distortion introduced into the resultant signal by the modulating process.

17.4.103 MOD-FREQ (modulation frequency)

17.4.103.1 Definition

The frequency of the signal that modulates an electrical carrier wave.

17.4.104 MOD-OFFSET (modulation offset)

17.4.104.1 Definition

The DC offset of the modulating signal as judged by the appearance of a modulated signal (often called "zero offset" in servo applications).

17.4.105 MOD-PHASE (modulation phase)

17.4.105.1 Definition

Modulation phase angle relative to the signal at REF HI and REF LO pins.

17.4.106 MODE

17.4.106.1 Definition

This modifier specifies the operating mode of the UUT. It is used only in source-type statements.

17.4.107 NEG-SAMPLING

17.4.107.1 Definition

Using swept measurement techniques, this indicates that the negative peaks of the signal are to be analyzed in determining the specified measurement.

17.4.108 NEG-SLOPE (negative slope)

17.4.108.1 Definition

The portion of a waveform having a negative first derivative.

17.4.109 NOISE (noise amplitude)

17.4.109.1 Definition

Disturbances superimposed upon a useful signal that tend to obscure its contents.

17.4.110 NOISE-AMPL-DENS (noise amplitude density)

17.4.110.1 Definition

The square root of the average of the squares of all the noise voltages per unit of bandwidth.

17.4.111 NOISE-PWR-DENS (noise power density)

17.4.111.1 Definition

The mean square noise power per unit of bandwidth.

17.4.112 NON-HARMONICS (non-harmonic distortion)

17.4.112.1 Definition

A frequency that is not an integral multiple of the defined fundamental frequency.

17.4.113 NON-LIN (non-linearity)

17.4.113.1 Definition

The maximum instantaneous amplitude variation from a linear waveform, expressed as a percentage of the peak amplitude.

17.4.114 OPER-TEMP (operating temperature)

17.4.114.1 Definition

The temperature range over which a device will operate in a satisfactory manner.

17.4.115 OVERSHOOT

17.4.115.1 Definition

The difference between the normal pulse amplitude and the peak amplitude to which the instantaneous pulse waveform initially rises positively or negatively away from the reference base line, expressed as a percentage of the normal pulse amplitude (see figure 17-1).

17.4.116 P-AMPL (peak pulse amplitude)

17.4.116.1 Definition

Peak voltage amplitude of a modulated pulsed carrier.

17.4.117 P3-DEV (P3 pulse deviation)

17.4.117.1 Definition

This modifier specifies the deviation in time from the nominal position of the P3 pulse, with time greater than nominal expressed as positive deviation.

17.4.118 P3-LEVEL (P3 pulse level)

17.4.118.1 Definition

This modifier specifies the peak power of the P3 pulse, expressed either in decibels relative to the first interrogation pulse or in absolute units (dBm, dBW, or dBk).

17.4.119 PAIR-DROOP (pulse pair droop)

17.4.119.1. Definition

The ratio of peak power of the smaller pulse to the larger pulse in a two-pulse pair.

17.4.120 PAIR-SPACING

17.4.120.1 Definition

The time between the 50% points on the leading edges of two pulses in a pair.

17.4.120.2 Rules

1. When used with the noun TACAN, it applies to the following pulse pairs: identification, equalizing, range interrogations, range responses in ground to-air mode, squitters, and elements in X mode reference bursts.

17.4.121 PEAK-DEGEN (peak degeneration)

17.4.121.1 Definition

Rounding of the corner of a triangular or ramp signal, expressed in percent.

17.4.122 PERIOD

17.4.122.1 Definition

The time between identical points on a periodic waveform. Period is equal to 1/frequency.

17.4.122.2 Rules

The following rules apply to the use of PERIOD in digital test statements.

a) The PERIOD modifier is used only with LOGIC REFERENCE and LOGIC CONTROL nouns as either a statement characteristic or a measured characteristic. PERIOD is required for all periodic SOURCE signals and is optional for other cases.

b) For single-word parallel data transfer, this modifier is undefined.

17.4.123 PHASE-ANGLE

17.4.123.1 Definition

The angular difference, expressed in degrees, between the signal described by the noun and the signal referenced by the REF HI and REF LO pins.

17.4.124 PHASE-DEV (phase deviation)

17.4.124.1 Definition

The deviation ratio of a phase-modulated signal.

17.4.125 PHASE-JIT (phase jitter)

17.4.125.1 Definition

Random, low amplitude oscillations of a signal wave shape along the time axis.

17.4.126 PHASE-SHIFT

17.4.126.1 Definition

The difference in electrical angle between the input voltage and the output voltage of an electrical circuit.

17.4.127 PLA (power lever angle)

17.4.127.1 Definition

This modifier describes the angle of a power lever control on a gas turbine engine. dimensions normally are percent of full rated power or degrees.

17.4.127.2 Rules

17.4.127.3 Notes and examples

A zero value corresponds to zero power from the engine.

17.4.128 PLA-RATE (power lever angle rate)

17.4.128.1 Definition

This modifier specifies the rate at which the power lever shall be changed from its former value to the value specified by the PLA modifier.

17.4.128.2 Rules

This modifier can appear only in a stimulus statement in conjunction with the PLA Modifier.

17.4.128.3 Notes and examples

The rate is normally expressed in units of degrees/second or percent/second.

17.4.129 POS-SAMPLING

17.4.129.1 Definition

Using swept measurement techniques, this indicates that the positive peaks of the signal are to be analyzed in determining the specified measurement.

17.4.130 POS-SLOPE (positive slope)

17.4.130.1 Definition

The portion of a waveform having a positive first derivative.

17.4.131 POWER

17.4.131.1 Definition

The rate at which work is done.

17.4.132 POWER-DELTA

17.4.132.1 Definition

The algebraic difference of two power values, between a complex signal being sensed and a reference signal, or between two specified power or frequency marks of a complex signal being sensed.

17.4.132.2 Rules

The <mark descriptor subfield> shall be used with this modifier in the latter case.

17.4.133 POWER-DIFF (power differential)

17.4.133.1 Definition

This modifier is used with certain avionics navigational UUTs to specify the ratio of the peak power of the lower antenna to the peak power of the upper antenna.

17.4.134 POWER-RATIO

17.4.134.1 Definition

The ratio of two power values, between a complex signal being sensed and a reference signal, or between two specified frequency marks on a complex signal being sensed.

17.4.134.2 Rules

The <mark descriptor subfield> shall be used with this modifier in the latter case.

17.4.135 POWER-SOURCE

17.4.135.1 Definition

Indicates the source of RF power signal (INTernal or EXTernal).

17.4.136 PP-SAMPLING

17.4.136.1 Definition

Using swept measurement techniques in the frequency and time domain, both negative and positive signal peaks are used in detection.

17.4.137 PRESHOOT

17.4.137.1 Definition

The amplitude difference between the reference base line of a pulse and the lowest value to which the pulse waveform falls at the leading edge of the pulse (see figure 17-1).

17.4.138 p-PRESS-A, p-PRESS-A (pressure, absolute)

17.4.138.1. Definition

The physical phenomenon defined as force per unit area in absolute terms. When the modifier is used for a fluid, it may be used with prefixes TOTAL- or STATIC-.

17.4.138.2 Rules

The prefixes TOTAL- and STATIC- may be used to indicate that the effect of fluid motion is to be included (TOTAL-) or not included (STATIC-). When neither prefix is used, the static pressure is referenced (see 17.2).

17.4.139 PRESS-G, p-PRESS-G (pressure, gauge)

17.4.139.1 Definition

The physical phenomenon defined as force per unit area in terms of differential pressure above or below atmospheric pressure. When the modifier is used for a fluid, it may be used with prefixes TOTAL- or STATIC-.

17.4.139.2 Rules

The prefixes TOTAL- and STATIC- can be used to indicate that the effect of fluid motion is to be included (TOTAL-) or not included (STATIC-). When neither prefix is used, the static pressure is referenced (see 17.2).

17.4.140 PRESS-OSC-AMPL (pressure oscillation amplitude)

17.4.140.1 Definition

The peak-to-peak pressure oscillations introduced into a manometric system for test purposes.

17.4.141 PRESS-OSC-FREQ (pressure oscillation frequency)

17.4.141.1 Definition

The frequency of the signal that modulates manometric pressure.

17.4.142 p-PRESS-RATE, p-PRESS-RATE (pressure rate)

17.4.142.1 Definition

The rate of change of pressure, either a gauge or an absolute pressure.

17.4.142.2 Rules

The prefixes TOTAL- and STATIC- can be used to indicate that the effect of fluid motion is to be included (TOTAL-) or not included (STATIC-). When neither prefix is used, the static pressure is referenced (see 17.2).

17.4.143 PRF (pulse repetition frequency)

17.4.143.1 Definition

The time rate at which pulses occur.

17.4.143.2 Rules

The following rules apply to the use of PRF in digital test statements.

a) The PRF modifier is used with LOGIC REFERENCE and LOGIC CONTROL nouns as either a statement characteristic or a measured characteristic. PRF is required for all periodic SOURCE signals and is optional for other cases.

b) For SERIAL DATA signals, the PRF is implicit in the PULSE-CLASS and the PRF of the associated LOGIC REFERENCE.

c) For single-word PARALLEL DATA transfer, this modifier is undefined.

17.4.144 PULSE-CLASS

17.4.144.1 Definition

This modifier identifies the type of pulse coding for serial digital signals.

17.4.144.2 Rules

17.4.144.3 Notes and examples

The various pulse classes are described in 6.16.3.5 and 6.16.3.7.

17.4.145 PULSE-IDENT (pulse identification)

17.4.145.1 Definition

This modifier specifies that the identification pulse is being measured. The identification pulse is appended to the UUT pulse train for a nominal 20 s period whenever manually activated by the pilot. This modifier uses no units.

17.4.146 PULSE-POSN-x (pulse position of pulse x)

17.4.146.1 Definition

This modifier specifies the time between the first pulse and the x-th pulse of the response.

17.4.146.2 Rules

The suffix x has the following forms:

An, Bn, Cn, Dn, X, SPI, m

where:

n = 1,2, ..., 5

m = 1,2, ..., 37

The integer m defines the relative position of the pulse in the pulse train.

17.4.147 PULSE-SPECT (pulse spectrum)

17.4.147.1 Definition

The frequency distribution of the sinusoidal components of the pulse in relative amplitude and relative phase.

17.4.147.2 Rules

1. PULSE-SPECT is used only as a sensor-statement measured characteristic. Modifiers PULSE-SPECT-THRESHOLD and RESP <array> shall also be used.

a) PULSE-SPECT causes the following data to be measured and stored in successive list positions in the RESP <array> for each spectral component measured.
   1) The frequency of the first component.
   2) The amplitude of the first component.
   3) The phase of the first component.
   4) The frequency, amplitude, and phase (in this order) for each successive component measured.
b) Only those spectral components whose amplitude is greater than the specified PULSE-SPECT-THRESHOLD are measured. The maximum number of spectral components measured is equal to 1/3 (array-range), where the array-range is specified in the RESP field. Any unfilled positions in the RESP <array> are set to zero.
c) PULSE-SPECT cannot be used with the verb VERIFY.

17.4.148 PULSE-SPECT-THRESHOLD

17.4.148.1 Definition

The amplitude threshold used in performing a pulse spectrum measurement. Only those spectral components that exceed this threshold are measured.

17.4.149 PULSE-WIDTH

17.4.149.1 Definition

The time interval measured between the 50% amplitude points of the leading and trailing edge of a single pulse (see figure 17-1).

17.4.149.2 Rules for digital applications

1. The following rules apply to PULSE-WIDTH when used with the noun LOGIC DATA, LOGIC CONTROL, or LOGIC REFERENCE.

a) Required for LOGIC REFERENCE as a SOURCE; optional for a SENSOR.

b) Either PULSE-WIDTH or WORD-RATE is required for non-clocked PARALLEL data; PULSE-WIDTH is optional for clocked PARALLEL data.

c) Optional for LOGIC CONTROL.

d) The pulse width for SERIAL DATA is implicit in the PULSE-CLASS type and the PRF of the LOGIC REFERENCE.

17.4.149.3 Notes and examples

Dimensions of PC, HR, and MIN are not allowed.

17.4.150 PULSES-INCL, -EXCL (pulses included, excluded)

17.4.150.1 Definition

The modifier PULSES-INCL specifies the pulses that are included in a pulse train. The modifier PULSES-EXCL specifies the pulses that are excluded from a standard pulse train in the governing document for the noun.

17.4.150.2 Rules

1. Modifier Descriptors are as follows:

a) The word ALL means that all pulses are included in or excluded from the signal.

b) A series of integers having the form i-j-k...., where each integer refers to a specific pulse by number, with pulse number 1 being the first pulse in the pulse train.

17.4.151 PWR-LMT (power limit)

17.4.151.1 Definition

Maximum safe power applicable without damage.

17.4.152 Q (quality factor)

17.4.152.1 Definition

The magnitude of the ratio of reactance to effective series resistance of an inductor.

17.4.153 QUAD (quadrature voltage)

17.4.153.1 Definition

That component of the voltage vector that is at 90 degrees phase to the reference voltage.

17.4.153.2 Rules

17.4.153.3 Notes and examples

This modifier is not to be confused with the -QUAD used as a suffix with modifiers such as VOLTAGE and CURRENT.

17.4.154 RADIAL

17.4.154.1 Definition

The direction from magnetic north taken in a clockwise direction, expressed in degrees.

17.4.154.2 Rules

When used with the noun VOR, RADIAL indicates the magnetic bearing from the ground station.

17.4.155 RADIAL-RATE

17.4.155.1 Definition

The rate of change of the radial (RADIAL) as degrees per second.

17.4.155.2 Rules

The rate of change is considered positive when measured in clockwise direction and negative when measured in counter-clockwise direction.

17.4.156 RANGE-PULSE-DEV (range pulse deviation)

17.4.156.1 Definition

The ratio of the amplitude of the RANGE-PULSE-ECHO to the amplitude of the normal range response, expressed as a positive quantity in percent or decibels.

17.4.156.2 Rules

RANGE-PULSE-DEV is used only for MODE RO-N or GA-N and with RANGE-PULSE-ECHO.

17.4.156.3 Notes and examples

1. Since the echo is always less than or equal to the normal range response, the ratio is expressed as positive decibels.

2. A RANGE-PULSE-ECHO that is 15 dB less than the normal range response is specified as:

RANGE-PULSE-DEV 15 DB

3. A RANGE-PULSE-ECHO equal to the normal range response may be specified as:

RANGE-PULSE-DEV 0 DB

17.4.157 RANGE-PULSE-ECHO

17.4.157.1 Definition

The addition to the test stimulus of a second range pulse response at a specified time delay after the normal range response to stimulate a multi-path condition.

17.4.157.2 Rules

1. RANGE-PULSE-ECHO is used only for MODE RO-N or GA-N.

2. The echo is less than or equal to the true range response.

17.4.157.3 Notes and examples

A RANGE-PULSE-ECHO occurring 10 µs after the normal range response is specified as:

RANGE-PULSE-ECHO 10 USEC

17.4.158 REACTANCE (reactance)

17.4.158.1 Definition

The magnitude of the imaginary (reactive) part of impedance.

17.4.159 REF-FREQ (reference frequency)

17.4.159.1 Definition

The frequency value to which the measurement is offset.

17.4.160 REF-INERTIAL (reference, inertial)

17.4.160.1 Definition

This modifier specifies that the associated angle, angle rate, and/or angle acceleration modifiers refer to an inertial reference frame. This modifier uses no units.

17.4.161 REF-PATTERN

17.4.161.1 Definition

This modifier array specifies the digital reference values to which the UUT digital response values are compared.

17.4.162 REF-PHASE-FREQ (reference phase modulation frequency)

17.4.162.1 Definition

The frequency of a signal that is used as a reference in determining the phase relationship between the reference and another signal.

17.4.163 REF-POWER (reference power)

17.4.163.1 Definition

The power level to which the measurement is offset.

17.4.164 REF-PULSES, -INCL, -EXCL, -DEV (reference pulses)

17.4.164.1 Definition

Pulses that serve as a time or phase reference. The suffix -INCL identifies specific reference pulses to be included in the reference pulse burst. The suffix -EXCL identifies specific pulses to be excluded from the burst. The suffix -DEV specifies the time between elements in a pulse burst in a TACAN signal.

17.4.164.2 Rules

1. The following apply when the noun is TACAN.

   a) The REF-PULSES modifier with suffixes refers to the reference pulse bursts used for bearing computations.

   b) The suffixes -INCL and -EXCL, respectively, designate pulses that are included in or excluded from the signal according to the modifier descriptor used, as follows:

      1) MAIN indicates that all main reference pulses are included or excluded.

      2) AUX indicates that all auxiliary reference pulses are included or included.

3) One or more integers designate specific pulses to be included or excluded. Pulses are numbered with the first pulse in the main burst being number 1 and the last pulse in the eight auxiliary burst having the largest integer, which is equal to the repetition index, n, and is a function of the mode as follows:

For X mode ground bursts: 120

For Y mode ground bursts: 117

For airborne beacon inverse bursts: 10

c) The suffix -DEV describes the time spacing of elements in the bursts, and is followed by a decimal number indicating the deviation from nominal in units of seconds or percent. An "element" of an X mode ground burst is defined as a pair of pulses with the pair spacing of 12 µs. "Elements" of other bursts are defined as single pulses. Nominal element spacing is as follows:

X mode, ground, main: 30 µs

X mode, ground, auxiliary: 24 µs

Y mode, ground, main: 30 µs

Y mode, ground, auxiliary: 15 µs

Beacon-inverse, airborne, X and Y modes (main only): 30 µs

d) This modifier (with suffixes) is used only with

```
PULSE-CODING GX or GY, or
PULSE-CODING AX or AY and MODE AB-I
```

17.4.164.3 Notes and examples

1. An X mode ground signal missing all auxiliary reference pulses would be missing pulses 25-120 and would be specified as:

```
REF-PULSES-EXCL AUX
```

2. Pulse spacing of 15 µs for a Y mode ground main burst is specified as:

```
REF-PULSES-DEV -50 PC
REF-PULSES-DEV 15 USEC
```

17.4.165 REF-UUT (reference, UUT)

17.4.165.1 Definition

Specifies that the associated angle, angle rate, and/or angle acceleration modifiers refer to a UUT reference frame. This modifier uses no units.

17.4.166 REF-VOLT (reference voltage)

17.4.166.1 Definition

The voltage level to which the circuit being measured is referenced.

17.4.167 REF-WORD-LENGTH

17.4.167.1 Definition

This modifier indicates the size of a reference pattern word in terms of the number of bits in one digital word of the REF array.

17.4.168 REL-BEARING (relative bearing)

17.4.168.1 Definition

The angle between one vector, usually the reference direction, and another vector, usually the pointing direction.

17.4.168.2 Rules

The angle is measured in degrees in the clockwise direction.

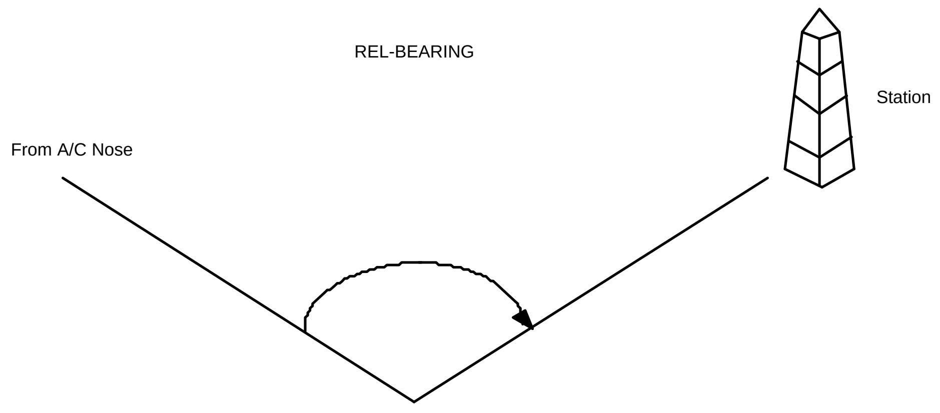


Figure 17-4. Relative bearing

17.4.169 REL-BEARING-RATE (relative bearing rate)

17.4.169.1 Definition

The rate of change of the relative bearing (REL-BEARING), expressed as degrees per second.

17.4.169.2 Rules

The rate of change is considered positive when measured in clockwise direction and negative when measured in counter-clockwise direction.

17.4.170 RELATIVE-HUMIDITY

17.4.170.1 Definition

The water content in the air relative to the amount of water vapor needed to saturate the air at the current temperature.

17.4.171 RELATIVE-WIND

17.4.171.1 Definition

The direction of the wind that is traveling toward the UUT.

17.4.171.2 Rules

17.4.171.3 Notes and examples

Measured in degrees clockwise from the front of the UUT looking at the top of the UUT.

17.4.172 REPLY-EFF (Reply Efficiency)

17.4.172.1. Definition

The ratio between the number of simulator range replies and TACAN range interrogations.

17.4.172.2 Rules

REPLY-EFF may not be used with MODE RO-N.

17.4.173 RES (resistance)

17.4.173.1 Definition

The real portion of an electrical quantity determined by the voltage/current ratio in an electrical circuit.

17.4.174 RESOLUTION-BANDWIDTH

17.4.174.1 Definition

Specifies the instantaneous bandwidth of a sensor.

17.4.175 RESP (response)

17.4.175.1 Definition

Response data storage location. The response data read from the UUT.

17.4.176 RINGING

17.4.176.1 Definition

A distortion in the form of a superimposed damped oscillatory waveform that, when present, usually follows a major transition; measured as the ratio of the maximum peak-to-peak amplitude of the damped oscillation to the normal pulse amplitude (see figure 17-1).

17.4.177 RISE-TIME

17.4.177.1 Definition

The time interval during which the instantaneous amplitude of a pulse or step change increases (positively or negatively from the reference base line) from 10-90% of the normal pulse amplitude (see figure 17-1).

17.4.178 ROTOR-SPEED

17.4.178.1 Definition

The rotational rate of a gas turbine engine rotor.

17.4.178.2 Rules

17.4.178.3 Notes and examples

When the modifier is expressed in percent, the reference value is the 100% rating specified on the engine nameplate.

17.4.179 ROUNDING

17.4.179.1 Definition

Corner rounding on the leading edge of a pulse. The difference between the normal pulse amplitude and the amplitude at which the instantaneous pulse voltage begins to arc toward the normal pulse amplitude, expressed as a percentage of the normal pulse amplitude (see figure 17-1).

17.4.180 SAMPLE

17.4.180.1 Definition

A measured characteristic to specify that a set of waveform samples are to be taken and stored.

17.4.181 SAMPLE-SPACING

17.4.181.1 Definition

The time between the initiation of two consecutive samples.

17.4.182 SAMPLE-TIME

17.4.182.1 Definition

The total time interval over which a set of samples is acquired.

17.4.183 SAMPLE-WIDTH

17.4.183.1 Definition

The time over which a signal characteristic is averaged.

17.4.184 SETTLE-TIME

17.4.184.1 Definition

The time required for a function to stabilize within error limits.

17.4.185 SHAFT-SPEED

17.4.185.1 Definition

The rotational rate of a shaft.

17.4.185.2 Rules

When the modifier is expressed in percent, the reference value is the 100 % rating specified on the engine nameplate.

17.4.186 SIGNAL-SEARCH

17.4.186.1 Definition

Using swept measurement techniques, a specified frequency range of a sweep is searched from the start frequency to the stop frequency and (frequency, power) pair values are collected for each signal occurrence above the SIGNAL-THRESHOLD. The order is in increasing frequency value.

17.4.186.2 Rules

1. RESP shall be used to indicate the number of pairs to be returned and where the number of pairs equals one half of the number of entries of the RESP <array range>.

2. If the <mark descriptor subfield> is used, the frequency and power response values are the algebraic differences of frequency and power between the two specified frequency marks encountered.

17.4.187 SIGNAL-THRESHOLD

17.4.187.1 Definition

The level above which a signal characteristic shall be in order to be classified as a valid signal response.

17.4.188 SKEW-TIME

17.4.188.1 Definition

The elapsed time between the appearance at the digital interface of the earliest and latest bits (pulses) of a PARALLEL DATA word.

17.4.188.2 Rules

1. Optional with PARALLEL DATA. Cannot be used as a SENSOR measured characteristic.

2. Standard C/ATLAS modifier form, generally expressed as a MAX value.

3. For SOURCE, this modifier defines the maximum permissible skew time. For SENSOR, it defines the maximum skew time expected from the UUT.

17.4.189 SLANT-RANGE

17.4.189.1 Definition

The line of sight distance between two points expressed in nautical miles.

17.4.189.2 Rules

When SLANT-RANGE is used, MODE shall not be RO-N.

17.4.189.3 Notes and examples

When SLANT-RANGE is used, zero range corresponds to a transponder time delay of 62 µs.

17.4.190 SLANT-RANGE-ACCEL (range acceleration)

17.4.190.1 Definition

The rate of change of slant range rate (SLANT-RANGE-RATE), expressed in nautical miles per second per second.

17.4.191 SLANT-RANGE-RATE

17.4.191.1 Definition

The rate of change of slant range (SLANT-RANGE), expressed in nautical miles per second or Knots.

17.4.191.2 Rules

When SLANT-RANGE-RATE is used, MODE shall not be RO-N.

17.4.192 SLEW-RATE

17.4.192.1 Definition

The rate of change of pressure signal amplitude per unit time.

17.4.193 SLS-DEV (side-lobe-suppression deviation)

17.4.193.1 Definition

The deviation in time of the position of the side lobe suppression pulse from its normal position, with time greater than nominal expressed as positive deviation.

17.4.194 SLS-LEVEL (side-lobe-suppression level)

17.4.194.1 Definition

The peak power of the side lobe suppression pulse relative to the peak power of the preceding sync pulse(s).

17.4.195 SNR

17.4.195.1 Definition

Signal to noise ratio. The ratio of the value of the signal to that of the noise.

17.4.196 SPACING

17.4.196.1 Definition

The time interval between the 50% amplitude points of any pulse slopes in a pulse train.

17.4.197 SPEC-GRAV (specific gravity)

17.4.197.1 Definition

The ratio of the density of a substance to the density of a reference substance.

17.4.197.2 Rules

17.4.197.3 Notes and Examples

For a liquid, the reference substance is pure water at 60•F (15.6•C). For a gas, the reference substance is air at 32•F (0•C) and 1 atmosphere pressure.

17.4.198 SPECTRUM

17.4.198.1 Definition

Using swept measurement techniques, frequency and power pair values are collected at successive power peaks, in order of decreasing power peaks from the signal's maximum power peak.

17.4.198.2 Rules

1. RESP shall be used to indicate the number of pairs to be returned, where the number of pairs equals 1/2 RESP <array range>.

2. If the <mark descriptor subfield> is used, the measurement values are the algebraic differences of frequency and power between the two specified power marks.

17.4.199 SPEC-TEMP (specification temperature)

17.4.199.1 Definition

The temperature range of values within a device's operating temperature, for which the stated accuracies are valid.

17.4.200 SQTR-DIST, SQTR-DIST-*** (squitter distribution)

17.4.200.1 Definition

This modifier is used to describe the distribution of the spacing between adjacent squitter pulse pairs, expressed as three individual percentages that refer to figure 1 in MIL-STD-291B, as follows:

SQTR-DIST-1 refers to points within the solid lines.

SQTR-DIST-2 refers to points between the solid and dotted lines.

SQTR-DIST-3 refers to points outside the dotted lines.

17.4.200.2 Rules

SQTR-DIST-*** may be used only with TACAN MODE RO-N, GA-N, AA-N, or AB-N.

17.4.201 SQTR-RATE (squitter rate)

17.4.201.1 Definition

This modifier is used to specify the repetition rate of squitter pulse pairs, expressed in Hertz.

17.4.201.2 Rules

SQTR-RATE may be used only with TACAN MODE RO-N, GA-N, AA-N, or AB-N.

17.4.202 STIM (stimulus)

17.4.202.1 Definition

Stimulus data storage location. The stimulus data to be applied to the UUT.

17.4.203 SUB-CAR-FREQ (sub-carrier frequency)

17.4.203.1 Definition

A carrier frequency used to modulate another carrier signal.

17.4.204 SUB-CAR-MOD (sub-carrier modulation)

17.4.204.1 Definition

The proportion of modulation of a carrier frequency that is modulated by a sub-carrier frequency.

17.4.205 SUSCEPTANCE

17.4.205.1 Definition

The magnitude of the imaginary (reactive) part of admittance.

17.4.206 SWEEP-TIME

17.4.206.1 Definition

For swept signals or measurements, the time it takes to sweep across the range of the swept parameter specified.

17.4.207 SWR (standing wave ratio)

17.4.207.1 Definition

The ratio of the maximum to the minimum amplitude of corresponding components of a field (or voltage or current) appearing along the waveguide or transmission line in the direction of propagation.

17.4.208 TARGET-RANGE

17.4.208.1 Definition

The distance between a radar signal source and a target surface.

17.4.209 TARGET-RANGE-ACCEL (target range acceleration)

17.4.209.1 Definition

The rate of change of TARGET-RANGE-RATE.

17.4.210 TARGET-RANGE-RATE

17.4.210.1 Definition

The rate of change of TARGET-RANGE.

17.4.211 TAS (true airspeed)

17.4.211.1 Definition

The equivalent true forward velocity, making due allowance for the absolute pressure that corresponds to the required gauge pressure.

17.4.212 p-TEMP, p-TEMP (temperature)

17.4.212.1 Definition

The molecular kinetic energy condition of a body that determines the transfer of heat to or from other bodies. The TEMP describes the temperature at the point specified in the CNX field. When the modifier is used for a fluid, it may be used with prefixes TOTAL- or STATIC-.

17.4.213 TEMP-COEFF-CAP (capacitance temperature coefficient)

17.4.213.1 Definition

The degradation value, in percent, for each degree the ambient temperature varies from the specified temperature, as applied to the capacitance error limit's function.

17.4.214 TEMP-COEFF-CURRENT (current temperature coefficient)

17.4.214.1 Definition

The degradation value, in percent, for each degree the ambient temperature varies from the specification temperature, as applied to the current error limit's function.

17.4.215 TEMP-COEFF-IND (inductance temperature coefficient)

17.4.215.1 Definition

The degradation value, in percent, for each degree the ambient temperature varies from the specification temperature, as applied to the inductance error limit's function.

17.4.216 TEMP-COEFF-REACT (reactance temperature coefficient)

17.4.216.1 Definition

The degradation value, in percent, for each degree the ambient temperature varies from the specification temperature, as applied to the reactance error limit's function.

17.4.217 TEMP-COEFF-RES (resistance temperature coefficient)

17.4.217.1 Definition

The degradation value, in percent, for each degree the ambient temperature varies from the specification temperature, as applied to the resistance error limit's function.

17.4.218 TEMP-COEFF-VOLT (voltage temperature coefficient)

17.4.218.1 Definition

The degradation value, in percent, for each degree the ambient temperature varies from the specification temperature, as applied to the voltage error limit's function.

17.4.219 THREE-PHASE-DELTA

17.4.219.1 Definition

This modifier specifies that the phases of a three-phase power source are delta-connected. This modifier uses no units.

17.4.220 THREE-PHASE-WYE

17.4.220.1 Definition

This modifier specifies that the phases of a three-phase power source are wye-connected. This modifier uses no units.

17.4.221 THRUST

17.4.221.1 Definition

The force produced by an engine, expressed in newtons.

17.4.222 TIME

17.4.222.1 Definition

The measurement of time using a standard time measurement function as a reference or the time interval over which EVENTS are counted.

17.4.223 TIME-ASYM (time asymmetry)

17.4.223.1 Definition

A measurement of the variation between two adjacent half cycles. It is calculated as one half of the time difference expressed as a percentage of the sum of the times for the two half cycles.

17.4.224 TIME-JIT (time jitter)

17.4.224.1 Definition

The difference between the period of any two adjacent pulses and the average period of all the pulses, expressed in seconds.

17.4.225 TORQUE

17.4.225.1 Definition

The rotational moment of force generated by or applied to the UUT.

17.4.226 TRANS-ONE (transition-one), TRANS-ZERO (transition-zero),

17.4.227 TRANS-PERIOD (transition-period)

17.4.227.1 Definition

The modifiers used to identify a specific type of bi-phase (BIP) or multi-phase (MIP) serial data pulse coding.

17.4.228 TRIG (trigger)

17.4.228.1 Definition

The instantaneous level of a signal at which a trigger is to be generated.

17.4.229 TRUE

17.4.229.1 Definition

The voltage level of a differential digital signal that represents a true value.

17.4.229.2 Rules

17.4.229.3 Notes and examples

See VOLTAGE-ONE.

17.4.230 TUNE-FREQ

17.4.230.1 Definition

The frequency at which a programmable tuned receiver is to be tuned.

17.4.231 TYPE

17.4.231.1 Definition

This modifier is used with the LOGIC DATA noun to define the serial or parallel type of data transmission to or from the UUT.

17.4.232 UNDERSHOOT

17.4.232.1 Definition

The difference between the amplitude of the reference base line of a pulse and the lowest amplitude to which the instantaneous pulse amplitude initially falls following the pulse decay, expressed as a percentage of the normal pulse amplitude (see figure 17-1).

17.4.233 VALUE

17.4.233.1 Definition

The VALUE modifier specifies a digital number or bit pattern.

17.4.233.2 Rules

1. Required for all source LOGIC DATA statements. Can be used as the measured characteristic for sensor LOGIC DATA statements.

2. Form: VALUE N

3. The number of character string N may be expressed in any of the standard C/ATLAS forms (such as B'1011', X'5E', etc.). All numbers used for N are treated as dimensionless quantities. Integer numbers are right justified, fractional numbers are left justified.

17.4.233.3 Notes and examples

1. All illustrations are shown with associated noun.

2. LOGIC DATA, TYPE SERIAL-MSB-FIRST, VALUE B'100000' or VALUE 32 or VALUE O'40'. All three representations for VALUE are valid. Each causes a bit pattern of 100000 to be placed in the digital word being described. The binary 1 will be the first bit to appear at the digital interface.

3. LOGIC DATA, TYPE PARALLEL, VALUE 25 or VALUE X'25' or VALUE X'7E'.

17.4.234 VAR-PHASE-FREQ (variable phase frequency)

17.4.234.1 Definition

The frequency of a signal that is variable in phase from a reference signal. When used with the noun, VOR, this modifier specifies the frequency of the modulating signal that is variable in phase from a reference signal.

17.4.235 VAR-PHASE-MOD (variable phase modulation)

17.4.235.1 Definition

The percentage modulation caused by a modulating signal that is variable in phase from a reference signal.

17.4.236 VIBRATION-ACCEL (vibration acceleration)

17.4.236.1 Definition

The rate of change of VIBRATION-RATE.

17.4.237 VIBRATION-AMPL (vibration amplitude)

17.4.237.1 Definition

The amplitude of a vibration.

17.4.237.2 Rules

17.4.237.3 Notes and Examples

It may be a linear displacement, a velocity, or an acceleration.

17.4.238 VIBRATION-RATE

17.4.238.1 Definition

The time rate of change of VIBRATION-AMPL.

17.4.239 VOLT-LMT (voltage limit)

17.4.239.1 Definition

Maximum safe voltage applicable without damage.

17.4.240 VOLTAGE

17.4.240.1 Definition

An electrical potential. In the case of DC signals, the potential is unvarying. In the case of AC signals, it is the generally used parameter that assumes a distortionless sinusoidal signal for a valid rms value.

17.4.241 VOLTAGE-ONE, VOLTAGE-ZERO, VOLTAGE-QUIES

17.4.241.1 Definition

Voltage levels of logic signal. The VOLTAGE-ONE, VOLTAGE-ZERO, VOLTAGE-QUIES modifiers define the signal levels that correspond to the logic states that exist at the digital interface.

17.4.241.2 Rules

1. These modifiers are used for signals that have the digital data coded in the voltage or cur- rent amplitude. The following requirements apply to "applicable set" of modifiers as defined in Rule 3a):

   a) Required for all SOURCE logic signals and all SENSOR LOGIC DATA signals that are driven from voltage sources.

   b) Optional for all logic signals that are driven from current sources.

   c) Cannot be used as a SENSOR measured characteristic.

2. Form: Standard C/ATLAS modifier field. Normally includes an ERRLMT, MIN, MAX, or RANGE subfield.

3. The following rules apply to all logic signals:

   a) VOLTAGE-ONE and VOLTAGE-ZERO are used with logic signal waveforms that have only two logic states. VOLTAGE-ONE, VOLTAGE-ZERO, and VOLTAGE-QUIES are used with waveforms that have two logic states plus a quiescent level. These groups are referred to as the "applicable set".

   b) For a SOURCE waveform driven from a voltage source, together with the associated tolerances given in ERRLMT, MIN, or MAX subfields, define the permissible range of voltage levels to be applied to the UUT. If no tolerances are given, no special control is exercised by the test system over the accuracy of the voltage levels.

   c) For a SENSOR waveform driven from a voltage source, these modifiers, together with the associated tolerances or boundary limits given in RANGE, MAX, or MIN subfields, define the range of voltages from the UUT that shall be accepted by the test system.

   d) For either SOURCE or SENSOR waveforms driven from a current source, these modifiers define the RANGE, MAX, or MIN values of voltage associated with the corresponding driving logic currents. The same conventions used with analog signals are employed.

e) For differential digital signals carried on a three wire circuit, these modifiers define the voltage from the TRUE line to the COMPL line.

4. The following rules apply to all LOGIC DATA signals driven from voltage sources except bi-phase SERIAL DATA:

a) VOLTAGE-ONE and VOLTAGE-ZERO define, respectively, the nominal voltages at the test system/UUT interface that represent a binary 1 and a binary 0. VOLTAGE-QUIES defines the third voltage level when one exists. These voltages may be either DC levels or pulse amplitudes.

b) For SENSOR, the VOLTAGE-ONE and VOLTAGE-ZERO modifiers, together with their associated tolerances or boundary limits, define the range of voltages from the UUT that shall be recognized by the test system as a binary 1 or a binary 0, respectively. (Note that this operation implies a test system evaluation.) If voltage tolerances or limits are not defined, the threshold is between the binary 1 and 0 logic levels are determined by the inherent design of the test system.

5. The following rules apply to all bi-phase (BIP) SERIAL data driven from voltage sources:

a) The concept of a binary 1 or 0 applied to the pulse amplitudes has no intrinsic meaning as the binary data is not carried by the pulse amplitudes, however, for convenience these modifiers are redefined and used.

b) VOLTAGE-ONE and VOLTAGE-ZERO define the nominal values of the two voltage states, VOLTAGE-QUIES represents the "no signal" or quiescent voltage value.

17.4.241.2.1 Rules for modifier suffixes for differential digital signals

1. The VOLTAGE-ONE and VOLTAGE-ZERO modifiers may optionally have a suffix added to them to permit description of differential digital signals. In addition, the optional modifiers TRUE and COMPL may be used.

2. The suffix DIFF is used to indicate that the modifier defines the differential voltage as a signed value on the TRUE line with respect to the COMPL line when the data has the indicated logic value (i.e., voltage TRUE minus voltage COMPL).

3. The modifier TRUE, when used, indicates that the modifier defines the absolute voltage level (with respect to LO) that occurs at the TRUE terminal when the data has the indicated logic value.

4. The modifier COMPL, when used, indicates that the modifier defines the absolute voltage (with respect to LO) that occurs at the COMPL terminal when the data has the indicated logic value.

5. The modifiers TRUE and COMPL can only be used in a three-wire system.

17.4.241.3 Notes and examples

1. For a SOURCE to define ranges of 4.5-5.5 V for a binary 1 and 0.5-1.5 V for binary 0:

```
VOLTAGE-ONE 5 V ERRLMT +-0.5 V,
VOLTAGE-ZERO 1 V ERRLMT +-0.5 V
```

2. For a SENSOR to define the same range as in 1, above:

```
VOLTAGE-ONE RANGE 4.5 V TO 5.5 V,
VOLTAGE-ZERO RANGE 0.5 V TO 1.5 V
```

3. For a SENSOR to define the following ranges: 3.0 + infinity V for binary 1, 2.0 to -infinity V for binary 0, and 2.0 to 3.0 V not defined:

```
VOLTAGE-ONE MIN 3.0 V, VOLTAGE-ZERO MAX 2.0 V
```

4. For a SOURCE:

```
TYPE SERIAL-MSB-FIRST,
PULSE-CLASS NRZ, VALUE B'101101',
VOLTAGE-ONE 4.5 V ERRLMT +-0.5 V,
VOLTAGE-ZERO 0.5 V ERRLMT +-0.5 V
```

The associated voltage ranges and a possible pulse train are shown in figure 17-5:

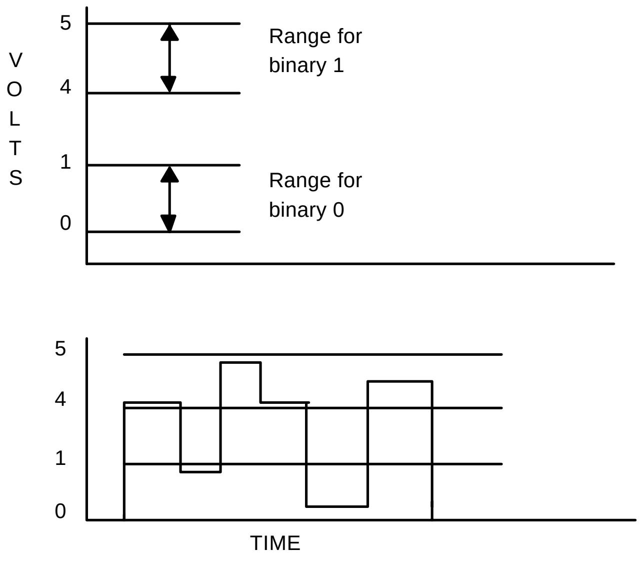


Figure 17-5. Defined voltage ranges

5. For a SENSOR

```
TYPE SERIAL-LSB-FIRST,
PULSE-CLASS NRZ,
VOLTAGE-ONE RANGE 4.0 V TO 5.0 V,
VOLTAGE-ZERO RANGE 0 V TO 1.0 V
```

These ranges defined for the binary 0 and 1 are the same as shown in Rule 4.

6. The following applies to all non-bi-phase LOGIC DATA SENSOR waveforms driven from a voltage source:

a) The VOLTAGE-ONE, VOLTAGE-ZERO modifiers, when specified in terms of a MIN, MAX, or RANGE, imply a test measurement and evaluation. That is, the test system will react to the UUT output voltage amplitude and the stated tolerance or threshold limits that define the binary 1 and 0 ranges and classify the logic signal as binary 1 or 0.

b) If the UUT voltage falls outside both the binary 1 and 0 ranges, the binary interpretation is determined by the test system implementation, but generally is one of the following:

   1) The binary digit is interpreted as a third state, undefined.

   2) The binary digit is interpreted as either a binary 1 or 0, depending on the test system implementation.

7. The binary 1 state need not necessarily be represented by the higher potential. Thus,

   a) VOLTAGE-ONE 5 V, VOLTAGE-ZERO 0 V

   b) VOLTAGE-ONE 0 V, VOLTAGE-ZERO 5 V

   c) VOLTAGE-ONE 5 V, VOLTAGE-ZERO -1 V

8. The process of determining the 1 or 0 status of each bit in a UUT response, given the VOLTAGE-ONE, VOLTAGE-ZERO values, is a digital problem. This process does not involve taking an amplitude "measurement" the measurement being taken is the determination of the 1, 0 bit pattern. The actual "measurement" of these analog pulse amplitudes is an analog problem; accordingly, the conventional modifiers for PULSED DC are used.

9. The use of the suffix DIFF and the modifiers TRUE and COMPL normally result in the statement carrying redundant information. It is recommended that, since the intelligence is carried by the voltage difference, the suffix DIFF is always used with either a MAX or MIN qualifier.

17.4.242 VOLTAGE-RAMPED

17.4.242.1 Definition

Interpolation rule for the generation of a WAVEFORM from a set of amplitude samples. This rule indicates that the voltage levels indicated are to be obtained at successive time intervals. During each time interval a linear voltage ramp is to be generated from the preceding to the succeeding value so that the resulting waveform consists of a series of connected linear ramps.

17.4.242.2 Rules

The modifier field contains only the mnemonic.

17.4.243 VOLTAGE-STEPPED

17.4.243.1 Definition

Interpolation rule for the generation of a WAVEFORM from a set of amplitude samples. This rule indicates that the voltage levels indicated by the list elements are each to be held for a single time interval so that the resulting waveform consists of a series of steps.

17.4.243.2 Rules

The modifier field contains only the mnemonic.

17.4.244 VOLUME-FLOW

17.4.244.1 Definition

A volumetric fluid flow rate, expressed in units of volume per unit time.

17.4.245 WAVE-LENGTH

17.4.245.1 Definition

For a periodic wave, the distance between points of corresponding phase of two consecutive cycles.

17.4.246 WIND-SPEED

17.4.246.1 Definition

The velocity of the wind at the UUT.

17.4.246.2 Rules

17.4.246.3 Notes and examples

It is normally expressed in units of KT (knots).

17.4.247 WORD-LENGTH

17.4.247.1 Definition

The number of bits that are contained in a digital word.

17.4.247.2 Rules

1. Optional for all LOGIC DATA statements. Cannot be used as a SENSOR measured characteristic.

2. Form: WORD-LENGTH b { BITS, DIGITS, CHAR }

17.4.248 WORD-RATE

17.4.248.1 Definition

Number of digital words transmitted per unit time.

17.4.249 ZERO-INDEX

17.4.249.1 Definition

Standard angular reference position from which the various angular shaft positions of a device are determined for a standard direction of rotation.

17.4.250 ZERO-POWER

17.4.250.1 Definition

This modifier is used as a measured characteristic to establish the zero power reference. This is a run-time parameter dependent on temperature and path selection.

17.5 Function and function characteristics

17.5.1 AM

17.5.1.1 Definition

This complex function mnemonic indicates that the COMPLEX SIGNAL modulation type is amplitude modulation.

17.5.2 AM-SENSITIVITY

17.5.2.1 Definition

The percent of amplitude modulation to be generated per volt of modulation signal input.

17.5.3 DIFFERENCE

17.5.3.1 Definition

This <function characteristic mnemonic> is used with the <function mnemonic> MIX in the <specify complex function statement>. It indicates that a difference type mixing is to occur for the generation of the lower sideband(s).

17.5.4 FM

17.5.4.1 Definition

This complex function mnemonic indicates that the COMPLEX SIGNAL modulation type is frequency modulation.

17.5.5 FM-SENSITIVITY

17.5.5.1 Definition

The fm-deviation in hertz per volt of INPUT modulating signal.

17.5.6 LSSB (lower single side band)

17.5.6.1 Definition

This complex function mnemonic indicates that the COMPLEX SIGNAL modulation type is lower single side band.

17.5.7 MIX

17.5.7.1 Definition

This complex function mnemonic performs a heterodyne conversion on two signals producing an intermediate frequency signal. The intermediate frequency is the sum or difference of one frequency and the integral multiple of the frequency of the other signal.

17.5.8 MOD-FUNCTION

17.5.8.1 Definition

Describes the mathematical operation of the modulation as it is applied to the base signal. The only allowable values are LOG (logarithmic) and LIN (linear).

17.5.9 MOD-INDEX

17.5.9.1 Definition

For amplitude modulation, the ratio of the minimum to the maximum voltage of the signal envelope, V min / V max. For phase and frequency modulation, the ratio of the peak carrier frequency deviations of the modulated wave to the frequency of the modulated function delta Fc / Fm, where delta Fc is peak carrier frequency deviation and Fm is the modulation frequency. The MOD-INDEX is equal to the phase deviation, expressed in radians.

17.5.10 PAM

17.5.10.1 Definition

This complex function mnemonic indicates that the COMPLEX SIGNAL modulation type is pulse amplitude modulation in which the modulating wave is caused to amplitude modulate a pulse carrier.

17.5.11 PHASE-LOCK

17.5.11.1 Definition

This complex function indicates that the COMPLEX SIGNAL is to operate at a constant phase angle relative to a reference.

17.5.12 PHASE-SENSITIVITY

17.5.12.1 Definition

The percent of phase deviation to be generated per volt of modulation signal input.

17.5.13 PM

17.5.13.1 Definition

This complex function mnemonic indicates that the COMPLEX SIGNAL modulation type is phase modulation.

17.5.14 PULSE

17.5.14.1 Definition

This complex function mnemonic indicates that the COMPLEX SIGNAL modulation type is pulse modulation.

17.5.15 PULSE-SENSITIVITY

17.5.15.1 Definition

The attribute of a complex function mnemonic that describes the on/off thresholds of a modulating pulse signal.

17.5.16 SUM

17.5.16.1 Definition

This complex function mnemonic indicates that the function applied to a component of a COMPLEX SIGNAL is summing.

17.5.17 SUMMATION

17.5.17.1 Definition

This <function characteristic mnemonic> is used with the <function mnemonic> MIX in the <specify complex function statement>. It indicates that a summing type mixing is to occur for upper sidebands.

17.5.18 SUPP-CAR

17.5.18.1 Definition

This complex function mnemonic indicates that the COMPLEX SIGNAL modulation type is suppressed carrier.

17.5.19 SUPPRESS-LEVEL

17.5.19.1 Definition

The attribute of a complex function that describes the attenuation level in decibels for which the specified modulated signal component is reduced from the reference of 100% modulation.

17.5.20 SWEEP

17.5.20.1 Definition

This complex function mnemonic indicates that the function applied to a component signal is a sweep of one of the parameters of that signal (i.e., the parameter of one signal will vary instantaneously and in proportion with the parameter of another signal). A SWEEP-CONTROL signal is provided to this function and the sweep action follows proportionally the magnitude of a designated characteristic of this SWEEP-CONTROL signal. The <proportionality subfield> shall be used with this function.

17.5.21 USSB

17.5.21.1 Definition

This complex function mnemonic indicates that the modulation type is upper single side band modulation.

17.5.22 VECTOR

17.5.22.1 Definition

This complex function mnemonic indicates that the modulation type is vector modulation, which is simultaneous amplitude and phase (also time and frequency) modulation. There are three modulation signals for this function: I, D, Q: I-Inphase, D-Digital, and Q-Quadrature.

17.5.23 VECTOR-TYPE

17.5.23.1. Definition

The modulation mode used to create the vector modulation COMPLEX SIGNAL.

The following <modifier descriptor>s are valid for VECTOR-TYPE:

17.5.24 PSK (phase shift keying)

17.5.24.1 Definition

Digital modulation using vector techniques where the transferred information is contained in specific vector phase angles. Typical types are BPSK (binary phase shift keying), QPSK (quadrature phase shift keying), 8PSK (8 phase shift keying), and OQPSK (offset quadrature phase shift keying).

17.5.25 QAM (quadrature amplitude modulation)

17.5.25.1 Definition

Digital modulation using vector techniques where the transferred information is contained in specific vector phase angles and amplitudes. Typical types are OQAM (offset quadrature amplitude modulation), 16QAM, 64QAM and 256QAM.

17.5.26 MSK (minimum shift keying) e.g., a special case of OQAM.

17.5.27 FSK (frequency shift keying)

17.5.27.1 Definition

Frequency modulation, where the modulating signal shifts the output frequency between predetermined values and the output wave, has no phase discontinuity. Using digital vector techniques, the transferred information is contained in two specific frequency values BFSK (Binary FSK). Additional types are CPFSK (continuous phase frequency shift keying) and MFSK (multiple frequency shift keying, e.g., more than two frequencies).

17.5.28 256QAM (256 quadrature amplitude modification)

17.5.28.1 Definition

Digital modulation using vector techniques for encoding 8 bits. Modulation information is contained in specific combinations of vector phase angles and amplitudes.

17.5.29 OQPSK (offset quadriphase shift key)

17.5.29.1 Definition

Digital modulation using vector techniques for encoding 2 bits. Modulation is contained in specific vector phase angles.

17.5.30 CPFSK (coherent phase frequency shift key)

17.5.30.1 Definition

Digital modulation technique used for multiple coherent frequency generation. Modulation is contained in specific signal frequencies.

17.6 Keyword definitions for the <specify signal conditioning statement>

17.6.1 ATTEN

17.6.1.1 Definition

This mnemonic used in the <filter structure> indicates the depth of attenuation in dB and is required for a NOTCH-FILTER.

17.6.2 BAND-PASS-FILTER

17.6.2.1 Definition

Used with RANGE where the lower value indicates the low frequency cutoff and the upper value indicates the high frequency cutoff.

17.6.3 DIVIDE

17.6.3.1 Definition

The integer factor used to create a sub-multiple of the signal frequency.

17.6.4 DE-EMPHASIS

17.6.4.1 Definition

Indicates the time constant, in seconds, typically evaluated at the 3dB point of a first order filter to de-emphasize the magnitude of the signal. The extent of de-emphasis is specified using the gain field.

17.6.5 GAIN

17.6.5.1 Definition

The specification of signal amplification.

17.6.6 HI-PASS-FILTER

17.6.6.1 Definition

The frequency representing the low end *cutoff.

17.6.7 ISOLATION

17.6.7.1 Definition

The degree of separation of the signal path from the rest of the system. This mnemonic is used in the <specify signal conditioning statement> and is used to indicate that the type of conditioning is isolation. Its specification is in dBm.

17.6.8 LO-PASS-FILTER

17.6.8.1 Definition

The frequency representing the high end *cutoff.

17.6.9 NOTCH-FILTER

17.6.9.1. Definition

This mnemonic specifies a frequency or band of frequencies that are attenuated.

17.6.10 PRE-EMPHASIS

17.6.10.1 Definition

Indicates the time constant, in seconds, typically evaluated at the 3dB point of a first order filter to emphasize the magnitude of the signal. The extent of emphasis is specified using the gain field.

17.6.11 ROLLOFF, ROLLOFF-LOWER, ROLLOFF-UPPER

17.6.11.1 Definition

These conditioning parameters indicate the rate at which attenuation is applied by a filter. It is specified in dB/octave or W/HZ. ROLLOFF indicates that the rate of change of both the upper and lower frequency cutoffs are the same. ROLLOFF-LOWER indicates the rate of cutoff at the low frequency end while ROLLOFF-UPPER indicates the rate of cutoff at the high frequency end as they apply to the pass bands of LO-PASS-, HI-PASS-, and BAND-PASS-FILTERs and to the rejection band of a NOTCH-FILTER.

17.6.11.2 Rules

17.6.11.3 Notes and examples

The frequency cutoff points insertion loss is normally 3dB.

17.7 Keyword definitions for the <mark descriptor subfield>

17.7.1. Definition

The mark descriptors have the following meanings, where n denotes a positive integer:

POWER-MARK 0	Represents the maximum power peak within the frequency range of a sweep.
POWER-MARK n	n>0, represents the nth decreasing power peak above or below the frequency of POWER-MARK 0 within the frequency range of sweep (increasing n indicates decreasing power peaks).
POWER-MARK-UPPER n	n>0, represents the nth decreasing power peak at a frequency above that of POWER-MARK 0 within the frequency range of sweep (increasing n indicates decreasing power peaks).

POWER-MARK-LOWER n	n>0, represents the nth decreasing power peak at a frequency below that of POWER-MARK 0 within the frequency range of sweep (increasing n indicates decreasing power peaks).
FREQ-MARK 0	Represents the frequency position at the peak power (POWER-MARK 0) position within the frequency range of sweep.
FREQ-MARK n	n>0, represents the frequency at the nth decreasing power-peak above or below the frequency of FREQ-MARK 0 within the frequency range of sweep (increasing n indicates decreasing power peaks).
FREQ-MARK-UPPER n	n>0, represents the frequency of the nth indicated signal peak above POWER-MARK 0, irrespective of the power magnitude above signal threshold (increasing n indicates increasing frequency).
FREQ-MARK-LOWER n	n>0, represents the frequency of the nth indicated signal peak below POWER-MARK 0, irrespective of the power magnitude above signal threshold (increasing n indicates decreasing frequency).
FREQ-MARK-HARM n	n>0, represents the frequency of the signal power at the nth harmonic of the signal frequency above FREQ-MARK 0. This power value may not coincide with a power peak; it is dependent on the nature of the signal.

17.8 Keyword definitions for the <sweep configuration subfield>

17.8.1 FORWARD

17.8.1.1 Definition

Indicates the action of a sweep with or without a trigger. Sweep forward in sweep time to the last value then reset, then either wait for a trigger to begin again, or begin again immediately.

17.8.2 FORWARD-HOLD

17.8.2.1 Definition

Indicates the action of a sweep with a trigger. Sweep forward in sweep time to the last value. Hold that value until the receipt of a trigger, then reset and start the scenario again.

17.8.3 FORWARD-HOLD-RESET-HOLD

17.8.3.1 Definition

Indicates the action of a sweep with trigger. Sweep forward through the sweep time, then hold at the last value waiting for the next trigger, then reset to the first value and wait for the next trigger. That last trigger will then start the complete scenario again.

17.8.4 FORWARD-HOLD-REVERSE-HOLD

17.8.4.1 Definition

Indicates the action of a sweep with trigger. Sweep forward through the sweep time, then hold at the last value waiting for the next trigger, then sweep in the reverse direction to the first value and wait for the next trigger. That last trigger will then start the complete scenario again.

17.8.5 FORWARD-REVERSE

17.8.5.1 Definition

Indicates the action of a sweep with or without a trigger. Sweep forward to the last value in one half sweep time, then sweep in the reverse direction in one half sweep time to the first value. Then either start again or wait for a trigger.

17.8.6 REVERSE

17.8.6.1. Definition

Indicates the action of a sweep with or without a trigger. Jump to the last value of sweep then sweep down to the first value in sweep time. Wait for a trigger to begin again or start again immediately.

17.9 Keyword definitions for <step sweep descriptor>

17.9.1 FSK

17.9.1.1 Definition

Indicates a form of frequency sweeping in which the signal is shifted in frequency within a set of specified values. Each frequency shift occurs at the first zero crossing of the signal after the trigger. SWEEP-TIME is not allowed with this sequence.

17.9.2 SEQUENCE

17.9.2.1 Definition

Indicates a sweep action forward, one specified step at a time, from the first defined step of the sweep parameter to the last defined step, within the SWEEP-TIME specified. Following the last step, the signal is then reset to the first step value. If no trigger is specified, the sweep action starts again after the reset. If SWEEP-TRIG is specified, the sweep action restarts at each trigger occurrence.

17.9.3 SEQUENCE-HOLD

17.9.3.1 Definition

Indicates a sweep action, with a trigger, forward through all defined steps of the sweep parameter to the last value, within the SWEEP-TIME specified. The last step value is held by the signal until the receipt of the next trigger, at which time the signal is reset to the first step value and this sweep and hold sequence is repeated.

17.9.4 SEQUENCE-HOLD-RESET-HOLD

17.9.4.1. Definition

Indicates a sweep action, with a trigger, forward through all defined steps of the sweep parameter to the last value, within the SWEEP-TIME specified. The last step sweep value is held until the receipt of the next trigger. At this next trigger the signal is reset to the first step value and then held at the reset value. The next trigger, subsequent to the reset, will restart this complete sequence.

17.9.5 SEQUENCE-STEP

17.9.5.1 Definition

Indicates a sweep action, with a trigger, forward through each defined step of the sweep parameter at each occurrence of the trigger. When all specified steps have been covered, the sequence restarts again from the first defined step of the sequence. SWEEP-TIME is not permitted with this sequence.

18.0 Standard C/ATLAS syntax

18.1 Introduction

18.1.1 Purpose

This clause provides the formal syntax definition (FSD) of the C/ATLAS Language as defined in this standard.

The language provided in this chapter is to be used for the writing of test programs for units under test (UUTs) so that these programs can operate on various makes and models of automatic test equipment. The definition provided in this chapter is intended to be used primarily for the development of language processors for C/ATLAS.

A formal syntax facilitates the development of language processors by having the following properties:

a) It consists solely of text with no figures, tables, or definition by example.

b) There are precise rules for developing a test program that conforms syntactically to the formal syntax definition.

c) The notations used in specifying the syntax rules have the conciseness and clarity of modern algebraic notations.

A formal syntax does not provide information for translating a test program into specific actions (this information is commonly referred to as semantics). Thus, this clause should be utilized in conjunction with the other clauses of this document that do provide semantic information.

18.1.2 Organization

The clause is self-contained in that it contains material explaining the formal syntax rules used, the actual formal syntax, and a section providing alphabetized cross-reference lists for the formal syntax definition.

The explanation of the formal syntax rules is provided in 18.2. It is presented in an instructional, rather than formal, manner since it is intended to be used to aid in the understanding of the formal syntax definition that follows, rather than in developing theoretical attributes of the language.

The formal syntax definition comprises three parts: program structure, statements and fields, and lexical symbols. The program structure part provides the overall structure and the substructures of a C/ATLAS Program. Every C/ATLAS statement appears as a right-hand side in at least one rule of the program structure but is not defined further. The statements and fields part includes the definitions of the C/ATLAS statements (see clauses 4, 6, and 8-13) and the definitions of fields and subfields (see clause 14). The lexical symbols part contains textual entities within a C/ATLAS program such as labels or character strings.

Cross-reference lists are provided in 18.6. The cross-reference lists provide a quick and easy reference to the different lexical symbols, rule names, and literals that occur within the production rules. These cross-reference lists make it easy to find the definition and the usages of a name or a literal. Furthermore, they provide a very convenient overview of the language and demonstrate the completeness of the description. The cross-reference subclause is divided into two parts: a list of rule names, and a list of literals that occur in the production rules.

In the cross-reference list of rule names, three fields appear under the headings: RULE-NAME, DEFINITION, and USAGE.

RULE NAME	gives the rule name
DEFINITION	gives the line number in the Formal Syntax, where the corresponding rule name is defined
USAGE	gives the line numbers of each use of a rule name in rule bodies

In the cross-reference list of literals, the SYMBOL and the USAGE fields, respectively, contain the literal and the line numbers in which they are referenced.

18.1.3 References to the formal syntax definition

To facilitate the parallel use of this specification and the formal syntax definition (FSD), the following conventions are introduced:

a) The names of syntactical variables used in the SPEC are also used in the FSD.

b) The clause numbers of the SPEC where a syntactical variable is defined are also used in the FSD.

c) The clause numbers of the SPEC where a syntactical variable is defined are also used in the FSD.

d) In the FSD, the SPEC clause numbers are written as a comment in the right margin of the left- hand-side of a rule. In the program structure part of the FSD, the right-hand-sides of the rule are also marked with clause numbers.

   Example:

```
    end-procedure-statement:              ! 6.6.5

    fstatno "END" fd procedure statement-terminator ;
```

e) The order of the clause numbers of the rules in the program structure part is strictly ascending.

f) In the statements and fields part, the clause numbers are ascending, with few exceptions. If a syntactical sub-construct, such as a field, is used exclusively in a certain statement, then its definition is put immediately after the statement definition. In the place where the sub-construct definition should occur according to its clause number, a reference as to where to find the definition is given.

g) In the lexical symbols part, clause numbers are in ascending order within each of the seven subsets of lexical symbols.

h) Syntactic variables of the FSD that do not have a counterpart within the SPEC are given a new clause number created by appending "- 1", "-2", etc., to the clause number of the rule in which they appear on the right-hand side.

   Example:

require-statement: ! 6.7

...

require-analog: ! 6.7-1

...

require-timer: ! 6.7-2

...

If such a new syntax-variable appears on the right-hand side of different rules, then the greatest common portion of the clause number of the relevant rules is used to construct the new clause number.

Example:

single-action-body: ! 11.2-1

appears on the right-hand-side of rules 11.2.2 through 11.2.4 and 11.2.7.

18.2 Syntax notation

The following notation used to describe the syntax of synthetic languages was derived from the Backus-Naur Format (BNF) and was originally proposed by DeRemer and Frank in 1974. It was extended and slightly modified by the ATLAS Committee to ease the formal definition and to simplify the analysis of the syntax of C/ATLAS.

18.2.1 Syntax language elements

The text that represents the formal syntax definition of the C/ATLAS language consists of a sequence of so-called "terminal symbols" or "tokens". A token is one of the following:

a) An Identifier consisting of a sequence of lower case letters, digits, and hyphens beginning with a letter and followed by zero or more hyphens, letters, or digits.

Examples:

define-statement

procedure body

fd

digital-number

b) A Literal is a self-defining character string. A literal is enclosed in double quotes. A double quote itself is represented by two double quotes.

Examples:

"VOLTAGE"

"ATLAS PROGRAM"

c) An Operator is a character indicating a relationship between one or more identifiers or literals. Operators are used to manifest the structure of a meta-program (see 18.2.2) and to formulate syntax expressions that are described in 18.4. The following keywords and special characters are used as operators:

Operator	**Meaning**
SYNTAX-RULES	beginning of description
FINIS	end of description
:	definition
*	repetition 0 or more times
%	repetition 1 or more times
|	alternation
#	separation
+	optional chaining
{ }	grouping braces
[ ]	optionality brackets
;	end of a rule
!	comment-indicator

In addition to terminal symbols, the formal syntax provides for blanks and comments. Blanks are used to separate tokens and may be omitted if the tokens can be identified without separation by blanks. The character ! denotes the start of a comment that is information for the human reader. The comment consists of all characters after ! until the end of the line.

18.2.2 Structure of a meta-program

The formal description of the syntax of a language is called a "meta-program". A meta-program begins with the keyword, SYNTAX-RULES, and ends with the keyword, FINIS.

Between these two keywords the syntax of C/ATLAS is described by a sequence of so-called "production rules".

A production rule consists of:

a) An identifier (rule name) used to reference the rule body in other production rules

b) A colon to separate the name of the rule from the rule body

c) A rule body that specifies the syntax or part of the syntax of a language

d) A semicolon to terminate the production rule

A meta-program is said to be complete if:

– Each rule name identifies one and only one rule body

– One and only one name exists that is used in no rule body (start symbol)

18.2.3 Rule bodies

A rule body is represented by a syntax expression that is a string of identifiers and/or literals separated by operators. The following items are supported by the notation in this clause:

18.2.3.1 Chaining

A particular sequence of tokens is represented by listing the tokens in the required order.

Example.

    a : identifier "," identifier ;

18.2.3.2 Alternatives

Alternatives in a syntax expression are separated by means of a vertical bar.

Example.

    a : b | c ;

    a is defined as either b or c.

18.2.3.3 Options

If one alternative string of tokens is merely an extension of another alternative, the remainder of the longer string can be enclosed in square brackets to indicate that it is optional.

Example. Using square brackets the production rule:

    a : b c d | b ;

may be represented in the form;

    a : b [c d] ;

Here, it is understood that if b does not appear, then the b alternative is taken to be a null alternative. Thus, if we have

    a : [c d] ;

the alternative choice is

    a : null ;

18.2.3.4 Repetition

When a rule body contains a symbol that shall occur one or more times, the symbol has to be followed by a percent sign (%) indicating that the symbol may be repeated.

Example. The production rule:

    a : b [a] ;

may also be represented as

```
a : b % ;
```

If the symbol may occur zero or more times, then an asterisk (*) is used instead of a percent sign.

Example.

```
a : [ b [ a ] ] ;
```

is equivalent to

```
a : b * ;
```

Here it is understood that if b is a set containing more than one element, then it is not necessary to choose the same element of b each time.

18.2.3.5 Separation

The operator # indicates that repeated occurrences of the left operand are separated by the right operand.

Example. The rule:

```
a : b | b "," a ;
```

produces

```
  b

 b,b

b,b,b

  .

  .

  .
```

The same factual situation can be represented in the form:

```
a : b # "," ;
```

18.2.3.6 Optional chaining

The operator + is used to indicate that the operand in front of and the operand behind the operator can occur either alone or together. If they occur together then they have to occur in the sequence defined by the syntax expression.

Example.

```
a : b [ c ] | c ;
```

is equivalent to

```
    a : b  +  c ;
```

18.2.3.7 Use of braces

The following rules govern the use of braces ({ }) in syntax expressions:

– Syntax expressions enclosed in braces or square brackets act like single tokens.

– Depending on the precedence of the operators in a syntax expression, braces may be suppressed. Operators occurring in syntax expressions have the following relative precedence:

```
Highest               * %
      |
      |               chaining
      |
      |               |
      |
      |               #
      |
Lowest                +
```

– Syntax expressions with operators of equal precedence are processed from left to right.

Example.

```
a  :  b  #  c ;  b :  d  #  e ;

 a  : {  d  #  e } #  c ;

  a  :  d  #  e  #  c ;
```

18.3 Program structure syntax

The program structure syntax is presented in the following groupings in the formal syntax definition:

*	COMPLETE C/ATLAS TEST PROGRAM	see 18.6.1
*	STRUCTURE DELIMITER STATEMENTS	see 18.6 2
*	PREAMBLE STATEMENTS	see 18.6.3
*	PROCEDURAL STRUCTURE	see 18.6.4
*	PROCEDURAL STATEMENTS DATA PROCESSING	see 18.6.5
*	PROCEDURAL STATEMENTS INPUT/OUTPUT	see 18.6.6
*	PROCEDURAL STATEMENTS CONTROL	see 18.6.7
*	PROCEDURAL STATEMENTS SIGNAL	see 18.6.8
*	PROCEDURAL STATEMENTS MACRO	see 18.6.9
*	PROCEDURAL STATEMENTS DATABUS	see 18.6.10
*	FIELDS AND SUBFIELDS	see 18.6.11

18.3.1 Range and order of the definition

This portion of C/ATLAS formal syntax defines all program elements involving more than one statement. These include both the complete-atlas-program-structure itself, its structures, and substructures. Structures and substructures define a fixed sequence of statements or substructures. Statement sets are used to express the alternative use of statements out of a set, at a fixed position within a substructure.

Every statement of C/ATLAS appears as a terminal symbol within the program structure syntax. The start symbol is the complete-atlas-program-structure.

The order of these rules is in accordance with that in the IEEE Std 716 C/ATLAS.

18.3.2 Name conventions

The following suffixes are used for syntax variables:

"-structure"	for structures and substructures
"-body"	for embedded substructures
"-statements"	for statement sets
"-statement"	for statements

The suffix "-statements" may be followed by another suffix in order to use the same notation as in C/ATLAS (e.g., procedural-statements-signal).

18.4 Statements and field syntax

18.4.1 Range and order of the definition

The statements and field syntax portion of the C/ATLAS formal syntax defines all C/ATLAS statements, fields, and subfields. The order of the rules is in accordance with this specification, with the following exceptions:

a) Definitions of fields and subfields that are used in one and only one statement are located immediately after this statement definition.

b) Some fields are defined as lexical symbols.

To facilitate cross-reference with the IEEE Std 716 C/ATLAS, references are given to the location of the definition of any missing paragraph number. Such references appear where the paragraph number would appear.

18.4.2 Name conventions

It was sometimes necessary to form a few concise rules out of one construct that appears as a single entity within the C/ATLAS specification. In such cases, where new syntax-variables had to be introduced, their names usually were constructed using the same prefixes as the basic construct.

Example.

```
sync-subfield:                    ! 14.4
    ...sync-body...
sync-body:                        ! 14.4A
  ...sync-initial ...
sync-initial:                     ! 14.4B
```

18.4.3 Notes

The formal syntax definition (FSD) allows cases that are syntactically correct but semantically invalid.

The formal syntax definition does not include the following:

a) Correctness of a particular label-identifier in the given context

b) Illegal data type handling

c) Define function usage

d) Program flow

e) Correct usage of
   1) noun vs. modifier
   2) modifier vs. dimension
   3) signal noun vs. pin descriptors
   4) verb vs. noun
   5) minimum necessary modifier set per noun
   6) default modifier conditions
   7) measured characteristic vs. noun
   8) measured characteristic vs. range subfield

f) Signal state checking
   1) order
   2) flow
   3) conflict
   4) path
   5) values

g) Formal vs. actual parameter verification
   1) number
   2) type
   3) order / position

18.5 Lexical symbols syntax

18.5.1 Range and order of the definition

The lexical symbols syntax defines that C/ATLAS character strings are valid for the C/ATLAS element that is represented by each lexical symbol of the C/ATLAS language.

Example.

```
unsigned-integer-number:
       digit * ;
```

Meaning: The C/ATLAS element represented by the lexical symbols unsigned-integer-number may contain any sequence of the digits 0 through 9.

The lexical symbols of C/ATLAS are presented in the following groupings:

* BASIC SYMBOLS AND CHARACTER SETS	see 18.6.12
* IDENTIFIERS AND LABELS	see 18.6.13
* SEPARATORS AND STATEMENT NUMBERS	see 18.6.14
* OPERATORS	see 18.6.15
* DESCRIPTORS	see 18.6.16
* NOUNS	see 18.6.17
* NOUN MODIFIERS	see 18.6.18
* NOUN MODIFIER PREFIXES AND SUFFIXES	see 18.6.19
* DIMENSIONS	see 18.6.20
* ABBREVIATIONS	see 18.6.21

18.5.2 Name conventions

The meta-variable denoting a lexical symbol may be composed of a main part and a suffix. The suffixes are used to establish a grouping of lexical symbols for the convenience of the reader. This grouping is irrelevant with respect to the formal syntax definition.

The following suffixes are used:

- set	for character sets
- identifier	
- number	
- operator	
- descriptor	
- noun	
- dim	for dimension
- pfx	for noun modifier prefix
- sfx	for noun modifier suffix

The lexical modifier-descriptor is defined via several syntax variables. The names of these syntax variables are constructed of the prefix "mod" and the name of the noun modifier where this descriptor applies.

18.5.2.1 Operators

A relational operator is classified as either a comparison-operator or an equal-notequal-operator.

18.5.2.2 Nouns

Nouns are divided into groups for analog usage and for digital (logic) usage.

18.5.2.3 Noun modifiers (mod)

The noun modifiers are listed in alphabetical order.

18.5.2.4 Dimensions (dim)

The list of dimensions contains literals (e.g., "V") in the order of table 15-1 and symbols representing separately defined subsets of dimensions. Separately defined are simple subsets such as freq-dim, digital-dim, time-dim, and sets that are constructed using the subsets and the full set of dimensions, namely rate-dim and product-dim.

18.5.2.5 Abbreviations

Abbreviations are used to represent typical pairs of alternatives. This shortens the notation of some rules within the statements syntax.

18.5.3 Classes of lexical symbols

There are several classes of lexical symbols, each of which are treated separately. These symbols and a description of their treatment follows:

18.5.3.1 Blanks

One or more blanks have to be used as separators between any two lexical symbols, except for the following cases:

Blanks may be omitted adjacent to:

- fd
- arith-string-operator
- parentheses
- label
- "/"
- statement-terminator

Blanks may be omitted between:

- if the first symbol is a prefix
- if the second symbol is a suffix

18.5.3.2 Sets

Sets are used to facilitate the definition of the identifiers and literals. All character sets are based upon the common char-set, each of which is contained in its successor. The label-char-set is formed by adding different characters to the common-char-set.

18.5.3.3 Identifiers

For the identifier of connections and of variables, two lexical symbols (connection-identifier) and label) are used because of their different structure and usage in C/ATLAS.

18.5.3.4 Literals

In C/ATLAS, string constants may be defined using the lexical symbols char-string-literal, message- string-set, and message-text-literal. Although the symbols unsigned-integer-number and unsigned- decimal-number are literals too, they are suffixed by "-number", akin to the notation in the earlier sections.

18.6 Syntax definition

```
SYNTAX-RULES


! Comments on the right are the section numbers
! for the definition in this C/ATLAS Standard.


!  18.6.1  COMPLETE C/ATLAS TEST PROGRAM


complete-atlas-test-program-structure:                  ! 3.1
   atlas-program-structure
   { { atlas-module-structure } * |
   { non-atlas-module-structure } * } ;

atlas-program-structure:                                ! 3.1.1
   begin-atlas-program-statement
   [ program-preamble-structure ]
   main-procedural-structure
   terminate-atlas-program-statement ;

atlas-module-structure:                                 ! 3.1.2
   begin-atlas-module-statement
   module-preamble-structure
   terminate-atlas-module-statement ;

non-atlas-module-structure:                             ! 3.1.3
   non-atlas-module-body ;


!  18.6.2  STRUCTURE DELIMITER STATEMENTS


begin-atlas-program-statement:                          ! 4.1.1
   fstatno "BEGIN" fd "ATLAS PROGRAM"
   [ program-name ]
   statement-terminator ;

terminate-atlas-program-statement:                      ! 4.1.2
   fstatno "TERMINATE" fd "ATLAS PROGRAM"
   [ program-name ]
   statement-terminator ;

begin-atlas-module-statement:                           ! 4.1.3
   fstatno "BEGIN" fd "ATLAS MODULE"
   module-name
   statement-terminator ;
```

```

terminate-atlas-module-statement:                       ! 4.1.4
   fstatno "TERMINATE" fd "ATLAS MODULE"
   module-name
   statement-terminator ;


!  18.6.3  PREAMBLE STATEMENTS


program-preamble-structure:                             ! 6.1.1
   include-statement *
   extend-statement *
   require-statement *
   establish-protocol-statement *
   { define-statements |
   declare-statement |
   identify-statements } * ;

module-preamble-structure:                              ! 6.1.2
   extend-statement *
   require-statement *
   establish-protocol-statement *
   { define-statements |
   declare-statement |
   identify-statements } * ;

local-preamble-structure:                               ! 6.2
   declare-statement % ;

declare-statement:                                      ! 6.3
   fstatno "DECLARE" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ]
   { var-declare | const-declare | type-declare }
   statement-terminator ;

const-declare:                                          ! 6.3-1
   "CONSTANT" fd
   { { constant-identifier "IS" constant } # ";" } ;

type-declare:                                           ! 6.3-2
   "TYPE" fd
   { { type-identifier "IS" type } # ";" } ;

var-declare:                                            ! 6.3-3
   "VARIABLE" fd
   { { { variable-identifier # fd }
   "IS" type [ initial ] } # ";" } ;

type:                                                   ! 6.3A
   { { type-identifier | base-type } [ subrange ] |
   pre-declared-type |
   structured-type } ;
```

```

base-type:                                                 ! 6.3B
   { "ENUMERATION"
   "(" { enumeration-element # fd } ")" |
   "CONNECTION" "(" { connection # fd } ")" |
   "LONG-DECIMAL" |
   "DECIMAL" |
   "INTEGER" |
   "CHAR" |
   "BIT" } ;

pre-declared-type:                                         ! 6.3C
   { "BOOLEAN" | "CHAR-CLASS" | "DIG-CLASS" } ;

structured-type:                                           ! 6.3D
   { array-structure |
   string-structure |
   record-structure |
   file-of-structure } ;

record-structure:                                          ! 6.3D-1
   "RECORD OF"
   "[" { { { record-field-identifier # fd }
   "IS" type } # ";" } "]" ;

array-structure:                                           ! 6.3D-2
   "ARRAY" "(" { { constant "THRU" constant } # fd } ")"
   "OF" type ;

string-structure:                                          ! 6.3D-3
   "STRING" "(" unsigned-integer-number ")" "OF"
   { "CHAR" [ subrange ] | "BIT" } ;

file-of-structure:                                         ! 6.3D-4
   "FILE OF" { type | "UNTYPED" | "TEXT" } ;

subrange:                                                  ! 6.3E
   "SUBRANGE" constant "THRU" constant ;

initial:                                                   ! 6.3F
   "INITIAL =" initializer ;

initializer:                                               ! 6.3G
   { { constant
   [ "OF" { constant | "(" initializer ")" } ] |
   "(" initializer ")" } # fd } ;

define-statements:                                         ! 6.4
   { define-signal-statement |
   define-procedure-structure |
   define-digital-configuration-structure |
   define-exchange-statement |
   define-digital-timing-statement |
   define-complex-signal-structure |
   define-exchange-configuration-statement } ;

define-signal-statement:                                  ! 6.5
   fstatno "DEFINE" fd
   signal fd "SIGNAL" fd
   signal-definition statement-terminator ;

signal-definition:                                        ! 6.5-1
   { [ m-char-noun ]
   { eval-statement-characteristics |
   { statement-characteristics # fd } }
   [ fd conn ] } | conn ;

m-char-noun:                                              ! 6.5A
   [ "(" { measured-characteristic # fd } ")" fd ]
   { noun | noun-name } fd ;

define-procedure-structure:                               ! 6.6.1
   define-procedure-statement
   procedure-body
   end-procedure-statement ;

define-procedure-statement:                               ! 6.6.2
   fstatno "DEFINE" fd procedure fd
   [ "GLOBAL" fd | "EXTERNAL" fd ]
   [ "TIMED DIGITAL" fd ] "PROCEDURE"
   [ "(" { parameter # fd } ")" ]
   [ "RESULT" "(" { parameter # fd } ")" ]
   statement-terminator ;

procedure-body:                                           ! 6.6.3
   [ local-preamble-structure ]
   [ main-procedural-structure ] ;

leave-procedure-statement:                                ! 6.6.4
   fstatno "LEAVE" fd procedure
   statement-terminator ;

end-procedure-statement:                                  ! 6.6.5
   fstatno "END" fd procedure
   statement-terminator ;

require-statement:                                        ! 6.7
   fstatno "REQUIRE" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ]
   { { requirement fd } % }
   { require-analog | require-timer |
   require-event } fd
   { noun | noun-name }
   [ require-control ]
   [ require-capability ]
   [ require-limit ]
   [ require-cnx ]
   statement-terminator ;

require-analog:                                                   ! 6.7-1
   { "SENSOR"
   "(" { measured-characteristic-mnemonic # fd } ")" |
   "SOURCE" | "LOAD" } ;

require-timer:                                                    ! 6.7-2
   "TIMER" | "DIGITAL TIMER" ;

require-event:                                                    ! 6.7-3
   "EVENT MONITOR"
   "(" measured-characteristic-mnemonic ")" ;

include-statement:                                                ! 6.8
   fstatno "INCLUDE" fd
   { "ATLAS MODULE" | "NON-ATLAS MODULE" }
   { module-name # fd }
   statement-terminator ;

identify-statements:                                              ! 6.9
   { identify-timer-statement |
   identify-signal-based-event-statement |
   identify-event-based-event-statement |
   identify-event-interval-statement |
   identify-event-indicator-statement |
   identify-time-based-event-statement } ;

identify-timer-statement:                                         ! 6.10
   fstatno "IDENTIFY" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ]
   { "TIMER" timer | "DIGITAL TIMER" digital-timer }
   "TIME INTERVAL" [ "USING" requirement ] fd
   { statement-characteristics # fd }
   statement-terminator ;

identify-signal-based-event-statement:                            ! 6.11
   fstatno "IDENTIFY" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ] "EVENT" event "AS"
   { analog-signal | digital-signal }
   statement-terminator ;

analog-signal:                                                    ! 6.11A
   "(" measured-characteristic-mnemonic ")" fd
   noun-field fd "EQ" real-quantity
   [ real-errlim ] fd [ increasing-decreasing fd ]
   { statement-characteristics # fd } fd conn ;

digital-signal:                                                   ! 6.11B
   "(" "VALUE" ")" "REF" expression
   [ { fd "MASK-ONE" | fd "MASK-ZERO" } expression ]
   { fd "MINIMUM-SENSE-RATE" expression dim |
   fd "SENSE-EVENT" event } fd on-field ;

identify-event-based-event-statement:                    ! 6.12
   fstatno "IDENTIFY" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ] "EVENT" event "AS"
   { occurrences-of | event gate-field |
   time-quantity  "AFTER" event |
   "MANUAL-INTERVENTION" } statement-terminator ;

occurrences-of:                                          ! 6.12A
   [ "EVERY" ] unsigned-integer-number
   "OCCURRENCES OF" event
   [ "AFTER" unsigned-integer-number
   "OCCURRENCES OF" event ] ;

identify-event-interval-statement:                       ! 6.13
   fstatno "IDENTIFY" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ] "EVENT INTERVAL"
   event-interval "AS"
   { identify-event-and-or | identify-event-from-to }
   statement-terminator ;

identify-event-and-or:                                   ! 6.13-1
   [ "NOT" ] event-interval
   { { "AND" | "OR" | "XOR" }
   [ "NOT" ] event-interval } % ;

identify-event-from-to:                                  ! 6.13-2
   "FROM" event
   [ "FOR" time-quantity | "TO" event ] ;

identify-event-indicator-statement:                      ! 6.14
   fstatno  "IDENTIFY" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ] "EVENT INDICATOR"
   event-indicator "AS SET BY" event
   statement-terminator ;

identify-time-based-event-statement:                     ! 6.15
   fstatno  "IDENTIFY" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ] "EVENT" event "AS"
   { "DIGITAL TIMER" digital-timer | "TIMER" timer }
   "EQ" time-quantity statement-terminator ;

define-digital-configuration-structure:                  ! 6.16.1
   define-digital-configuration-statement
   configuration-body
   end-digital-configuration-statement ;

configuration-body:                                      ! 6.16.1-1
   { define-digital-source-statement |
   define-digital-sensor-statement } % ;

define-digital-configuration-statement:                  ! 6.16.2
```

```
   fstatno "DEFINE" fd configuration fd
   [ "GLOBAL" fd | "EXTERNAL" fd ]
   "DIGITAL CONFIGURATION"
   statement-terminator ;

define-digital-source-statement:                       ! 6.16.3
   fstatno "DEFINE" fd digital-source fd
   "DIGITAL" "SOURCE"
   { fd "SAME AS" digital-source |
   digital-source-spec }
   statement-terminator ;

digital-source-spec:                                   ! 6.16.3A
   [ fd "SPEC"
   { external-digital-specification # fd } ]
   [ fd noun-field ]
   [ fd { statement-characteristics # fd } ]
   [ fd source-logic-levels ]
   [ fd "ILLEGAL-STATE-INDICATOR" boolean-variable ]
   [ digital-format ] fd
   "CNX" conn-set ;

source-logic-levels:                                   ! 6.16.3B
   { { { { "LEVEL-LOGIC-ZERO" | "LEVEL-LOGIC-ONE" }
   digital-source-characteristics
   [ logic-alternation ] } |
   { { "TRANS-LOGIC-ONE" | "TRANS-LOGIC-ZERO" }
   { { { "1" | "0" } { "1" | "0" } } # "OR" } } |
   { { "LEVEL-LOGIC-HIZ" | "LEVEL-LOGIC-QUIES" }
   digital-source-characteristics } |
   { "TRANS-SYNC"
   { { { "1" | "QUIES" | "0" }
   { "1" | "0" } } # "OR" } } |
   "TRANS-SYNC-BIT" } # fd } ;

digital-format:                                        ! 6.16.3C
   { [ { fd "SERIAL-LSB-FIRST" |  fd "SERIAL-MSB-FIRST" }
   fd "WORD-LENGTH" unsigned-integer-number "BITS" ]
   [ fd pulse-class ]
   [ fd digital-synchronization ] } |
   { [ fd "SKEW-TIME MAX" time-quantity ] } ;

logic-alternation:                                     ! 6.16.3D
   "EXCEPT AT EVERY" unsigned-integer-number
   "CONSECUTIVE OCCURRENCES" ;

pulse-class:                                           ! 6.16.3E
   "PULSE-CLASS"
   { "SPEC" external-pulse-class-specification |
   "HDB" unsigned-integer-number |
   "RZ" | "NRZ" | "BIP" | "AMI" } ;

digital-synchronization:                               ! 6.16.3F
```

```
   { { "BIT-RATE" | "BIT-PERIOD" } real-quantity
   [ "RANGE" unsigned-integer-number dim
   "TO" unsigned-integer-number dim ] } |
   { "SYNC-BIT-TRANSITION TO EVENT"
   event max-time } ;

define-digital-sensor-statement:                          ! 6.16.4
   fstatno "DEFINE" fd digital-sensor fd
   "DIGITAL" "SENSOR"
   { fd "SAME AS" digital-sensor |
   digital-sensor-spec }
   statement-terminator ;

digital-sensor-spec:                                      ! 6.16.4A
   [ fd "SPEC"
   { external-digital-specification # fd } ]
   [ fd "(" measured-characteristic-mnemonic ")" ]
   [ fd noun-field ]
   [ fd { statement-characteristics # fd } ]
   [ fd sensor-logic-levels ]
   [ fd "ILLEGAL-STATE-INDICATOR" boolean-variable ]
   [ digital-format ] fd
   "CNX" conn-set ;

sensor-logic-levels:                                      ! 6.16.4B
   { { { { "LEVEL-LOGIC-ZERO" | "LEVEL-LOGIC-ONE" }
   digital-sensor-characteristics
   [ logic-alternation ] } |
   { { "LEVEL-LOGIC-HIZ" | "LEVEL-LOGIC-QUIES" }
   digital-sensor-characteristics } |
   { { "TRANS-LOGIC-ONE" | "TRANS-LOGIC-ZERO" }
   { { { "1" | "0" } { "1" | "0" } } # "OR" } } |
   { "TRANS-SYNC"
   { { { "1" | "QUIES" | "0" }
   { "1" | "0" } } # "OR" } } |
   { "TRANS-SYNC-BIT" } } # fd } ;

end-digital-configuration-statement:                      ! 6.16.5
   fstatno "END" fd configuration
   statement-terminator ;

extend-statement:                                         ! 6.17
   fstatno "EXTEND" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ] "ATLAS"
   { extend-noun | extend-modifier |
   extend-connection | extend-databus }
   statement-terminator ;

extend-noun:                                              ! 6.17A
   "NOUN" "'" noun-name "'" fd "FOR VERBS"
   { verb-info # "/" }
   fd "SPEC" external-specification ;

```

```
extend-modifier:                                              ! 6.17B
   "MODIFIERS FOR" { noun | noun-name } fd
   { { { { modifier-definition fd
   "USAGE" usage-info } # fd } fd
   "SPEC" external-specification } # fd } ;

extend-connection:                                            ! 6.17C
   "CONNECTIONS FOR" { noun | noun-name } fd
   extend-cnx fd "SPEC" external-specification ;

modifier-definition:                                          ! 6.17D
   { "'" modifier-name "'" mod-type |
   modifier-name | modifier-mnemonic } ;

usage-info:                                                   ! 6.17E
   { "STIMULUS-RESPONSE-MEASUREMENT" |
   "RESPONSE-MEASUREMENT" |
   "STIMULUS-RESPONSE" |
   "STIMULUS" |
   "RESPONSE" } ;

extend-cnx:                                                   ! 6.17F
   "CNX" { "'" pin-descriptor-name "'" |
   pin-descriptor-name |
   pin-descriptor } %  ;

verb-info:                                                    ! 6.17G
   signal-oriented-verb ;

signal-oriented-verb:                                         ! 6.17G-1
   { "INITIATE" | "SETUP"   | "CONNECT" | "DISCONNECT" |
   "IDENTIFY" | "FETCH"   | "ARM"     | "CHANGE"     |
   "APPLY"    | "REMOVE"  | "MEASURE" | "DEFINE"     |
   "REQUIRE"  | "MONITOR" | "VERIFY"  | "READ"       |
   "RESET"    | "EXTEND" } ;

mod-type:                                                     ! 6.17H
   "MOD-TYPE"
   { { "VALUE" | "ARRAY-RANGE" }
   { "DECIMAL" | "INTEGER" |
   "LONG-DECIMAL" | "DIGITAL" } mod-dim |
   "DECIMAL-RANGE" mod-dim | mod-desc-info } ;

mod-dim:                                                      ! 6.17I
   "MOD-DIM"
   { { { "'" dim-name "'" } % "DIM-SPEC"
   external-specification | dim-name | dim } % | "NONE" } ;

mod-desc-info:                                                ! 6.17J
   { "MODIFIER-DESCRIPTOR"
   { mod-desc-data # fd } |
   "MNEMONIC-ONLY" } ;

```

```
mod-desc-data:                                          ! 6.17K
   { { { "'" modifier-descriptor-name "'" } % }
   "MOD-DESC-SPEC" external-specification |
   modifier-descriptor-name |
   modifier-descriptor } ;

extend-databus:                                         ! 6.17L
   { extend-bus-parameter |
   extend-protocol-parameter |
   extend-bus-mode |
   extend-test-equip-role |
   extend-exchange-pin } ;

extend-bus-parameter:                                   ! 6.17L-1
   "BUS-PARAMETER" { { "'" bus-parameter-name "'" fd
   extend-parameter-definition } # fd } ;

extend-protocol-parameter:                              ! 6.17L-2
   "PROTOCOL-PARAMETER"
   { { "'" protocol-parameter-name "'" fd
   extend-parameter-definition } # fd } ;

extend-bus-mode:                                        ! 6.17L-3
   "BUS-MODE" { { "'" bus-mode-name "'" } # fd } ;

extend-test-equip-role:                                 ! 6.17L-4
   "TEST-EQUIP-ROLE"
   { { "'" test-equip-role-name "'" } # fd } ;

extend-exchange-pin:                                    ! 6.17L-5
   "EXCHANGE-PIN"
   { { "'" pin-descriptor-name "'" } # fd } ;

extend-parameter-definition:                            ! 6.17M
   "MOD-TYPE" { "VALUE" { "DECIMAL" |
   "LONG-DECIMAL" | "INTEGER" | "DIGITAL" } mod-dim |
   mod-desc-info } ;

establish-protocol-statement:                           ! 6.18
   fstatno "ESTABLISH" fd
   [ "GLOBAL" fd | "EXTERNAL" fd ]
   "BUS PROTOCOL" protocol fd
   "SPEC" { external-bus-specification # fd } fd
   databus-definition fd
   { "STANDARD" | "FAULT-TEST" }
   [ "PRIMARY" | "REDUNDANT" ] "BUS"
   [ fd "ALTERNATE-BUS-TRANSMIT" protocol
   [ "DELAY" time-quantity ] ] fd
   "CNX" { pin-descriptor % }
   statement-terminator ;

databus-definition:                                     ! 6.18A
   test-equip-bus-role
   [ test-equip-bus-monitor ]
   { exchange-model # fd }
   [ fd establish-bus-parameter ]
   [ fd establish-protocol-parameter ] ;

test-equip-bus-role:                                        ! 6.18B
   "TEST-EQUIP-ROLE"
   { { "MASTER" | "MONITOR" | "SLAVE" |
   test-equip-role-name } fd } % ;

test-equip-bus-monitor:                                     ! 6.18C
   "TEST-EQUIP-MONITOR"
   { { "COMMAND" | "DATA" | "STATUS" } fd } % ;

exchange-model:                                             ! 6.18D
   "BUS-MODE"
   { "CON-RT" | "RT-CON" | "RT-RT" | bus-mode-name |
   "CON-MODE" | "TALKER-LISTENER" | "ALL-LISTENER" }
   [ fd "TALKER"
   { { "TEST-EQUIP" | device-identifier | "UUT" }
   # fd } [ "(" type ")" ] ]
   [ fd "LISTENER"
   { { "TEST-EQUIP" | device-identifier | "UUT" }
   # fd } [ "(" type ")" ] ]
   [ fd "COMMAND" "(" type ")" ]
   [ fd "DATA" "(" type ")" ]
   [ fd "STATUS" "(" type ")" ] ;

establish-bus-parameter:                                    ! 6.18E
   "BUS-PARAMETER"
   { { [ fetch-update-spec ]
   bus-parameter-info } # fd } ;

bus-parameter-info:                                         ! 6.18F
   { "WORD-LENGTH" | "WORD-GAP" | "MESSAGE-GAP" |
   "RESPONSE-TIME" | "ZERO-AMPLITUDE" | "ONE-AMPLITUDE" |
   "ZERO-CROSSING" | bus-parameter-name }
   [ limit-field | constant ] ;

establish-protocol-parameter:                               ! 6.18G
   "PROTOCOL-PARAMETER"
   { { fetch-update-spec
   protocol-parameter-info } # fd } ;

protocol-parameter-info:                                    ! 6.18H
   { protocol-parameter-name [ "(" type ")" ] |
   { "COMMAND-WORD" | "DATA-WORD" | "STATUS-WORD" }
   { "SYNC" | "T-R" | "PULSE-CODE" | "PARITY" |
   "LENGTH" } | "WORD-COUNT" } ;

fetch-update-spec:                                          ! 6.18I
   { "UPDATABLE-FETCHABLE" | "FETCHABLE" | "UPDATABLE" }
   { "PROTOCOL-EXCHANGE" | "PROTOCOL" | "EXCHANGE" } ;
```

```

define-exchange-statement:                               ! 6.19
   fstatno "DEFINE" fd exchange fd
   [ "GLOBAL" fd | "EXTERNAL" fd ]
   "EXCHANGE" fd "PROTOCOL" protocol fd
   "BUS-MODE" { "CON-RT" | "RT-CON" | "RT-RT" |
   bus-mode-name | "CON-MODE" | "TALKER-LISTENER" |
   "ALL-LISTENER" }
   [ fd "TALKER"
   { "UUT" | "TEST-EQUIP" | device-identifier }
   [ "(" { constant | data-store } ")" ] ]
   [ fd "LISTENER" { {
   { "UUT" | "TEST-EQUIP" | device-identifier }
   [ "(" { constant | data-store } ")" ] } # fd } ]
   [ fd command-field ]
   [ fd data-field ]
   [ fd status-field ]
   [ fd bus-parameter ]
   [ fd set-protocol-parameter ]
   statement-terminator ;

define-digital-timing-statement:                         ! 6.20
   fstatno "DEFINE" fd digital-timing fd
   "DIGITAL TIMING"
   { fd stim-event [ fd sense-event ] |
   fd sense-event }
   statement-terminator ;

define-complex-signal-structure:                         ! 6.21.1
   define-complex-signal-statement
   [ specify-complex-signal-characteristics-statement ]
   [ [ specify-complex-function-statement ]
   specify-component-signal-statement %
   [ specify-signal-conditioning-statement ] ]
   end-complex-signal-statement ;

define-complex-signal-statement:                         ! 6.21.2
   fstatno "DEFINE" fd complex-signal fd
   [ "GLOBAL" fd | "EXTERNAL" fd ]
   "COMPLEX SIGNAL" statement-terminator ;

specify-complex-signal-characteristics-statement:        ! 6.21.3
   fstatno "SPECIFY" fd "COMPLEX SIGNAL"
   { complex-signal | noun | noun-name } fd
   { statement-characteristics # fd }
   statement-terminator ;

specify-complex-function-statement:                      ! 6.21.4
   fstatno "SPECIFY" fd "COMPLEX FUNCTION" fd
   { "SPEC" external-complex-function-specification |
   complex-function-mnemonic }
   [ fd { { function-characteristic-mnemonic
   { modifier-descriptor |
   real-characteristic-subfield } } % } |
   proportionality-subfield ]
   statement-terminator ;

complex-function-mnemonic:                                   ! 6.21.4-2
   { "AM" | "FM" | "LSSB" | "MIX" | "PHASE-LOCK" |
   "PM" | "PAM" | "PULSE" | "SUM" | "SUPP-CAR" |
   "SWEEP" | "USSB" | "VECTOR" } ;

function-characteristic-mnemonic:                            ! 6.21.4-3
   { "MOD-INDEX" | "MOD-FUNCTION" mod-mod-mode |
   "AM-SENSITIVITY" | "FREQ-DEV" | "FM-SENSITIVITY" |
   "SUPPRESS-LEVEL" | "SUMMATION" | "DIFFERENCE" |
   "PHASE-ANGLE" | "PHASE-JIT" | "PHASE-DEV" |
   "PULSE-SENSITIVITY" | "PHASE-SENSITIVITY" |
   "VECTOR-TYPE" mod-vector-descriptor } ;

specify-component-signal-statement:                           ! 6.21.5
   fstatno "SPECIFY" fd "AS"
   { "REFERENCE-SIGNAL" | "MOD-SIGNAL" |
   "MOD-SIGNAL-D" | "MOD-SIGNAL-I" |
   "MOD-SIGNAL-Q" | "SWEEP-CONTROL" |
   "REFERENCE-LEVEL" | "OSCILLATOR" |
   "COMPONENT" | "CARRIER" |
   "SPEC" external-signal-name-specification } fd
   noun-field [ fd { statement-characteristics # fd } ]
   [ fd gate-field ]
   [ fd conn ] statement-terminator ;

specify-signal-conditioning-statement:                        ! 6.21.6
   fstatno "SPECIFY" fd "SIGNAL CONDITIONING" fd
   { { { "SPEC"
   external-signal-conditioning-specification |
   signal-conditioning-mnemonic }
   [ real-characteristic-subfield |
   modifier-descriptor ] | filter-structure } # fd }
   statement-terminator ;

signal-conditioning-mnemonic:                                 ! 6.21.6-1
   { "ATTEN" | "DE-EMPHASIS" | "DIVIDE" | "GAIN" |
   "ISOLATION" | "PHASE-SHIFT" | "PRE-EMPHASIS" } ;

filter-structure:                                             ! 6.21.6A
   { "LO-PASS-FILTER" real-characteristic-subfield
   [ fd "ROLLOFF-UPPER" real-characteristic-subfield ] |
   "HI-PASS-FILTER" real-characteristic-subfield
   [ fd "ROLLOFF-LOWER" real-characteristic-subfield ] |
   { "BAND-PASS-FILTER" | "NOTCH-FILTER" }
   real-characteristic-subfield rolloff-info }
   [ fd "ATTEN" real-characteristic-subfield ] ;

rolloff-info:                                                 ! 6.21.6B
   { [ fd "ROLLOFF-UPPER" real-characteristic-subfield ]
   [ fd "ROLLOFF-LOWER" real-characteristic-subfield ] |
   [ fd "ROLLOFF" real-characteristic-subfield ] } ;

end-complex-signal-statement:                            ! 6.21.7
   fstatno "END" fd complex-signal
   statement-terminator ;

define-exchange-configuration-statement:                 ! 6.22
   fstatno "DEFINE" fd exchange-configuration
   fd "EXCHANGE-CONFIGURATION"
   { protocol # fd } statement-terminator ;


!  18.6.4  PROCEDURAL STRUCTURE


main-procedural-structure:                               ! 7.1
   main-procedural-statements % ;

main-procedural-statements:                              ! 7.2
   { procedural-statements-data-processing |
   procedural-statements-input-output |
   procedural-statements-control |
   procedural-statements-signal |
   procedural-statements-timing |
   procedural-statements-databus } ;


!  18.6.5  PROCEDURAL STATEMENTS - DATA PROCESSING


procedural-statements-data-processing:                   ! 8.0
   { calculate-statement |
   compare-statement } ;

calculate-statement:                                     ! 8.1
   fstatno "CALCULATE" fd
   { { data-store "=" expression } # fd }
   statement-terminator ;

expression:                                              ! 8.1A
   { { { { [ unary-operator ]
   { expression-info
   # arith-string-operator } }
   # boolean-operator } }
   # comparison-operator } ;

expression-info:                                         ! 8.1B
   { pre-defined-function
   ["(" { expression # fd } ")"] |
   "(" expression ")" | data-store | constant |
   condition | boolean-evaluation } ;

```

```
constant:                                                  ! 8.1C
   { decimal-number |
   digital-number |
   character-string |
   enumeration-element |
   constant-identifier |
   "CONNECTION" connection |
   pre-declared-enumeration } ;

boolean-evaluation:                                        ! 8.1D
   data-store [ "NOM" real-quantity ]
   uplowlimit-real ;

uplowlimit-real:                                           ! 8.1D-1
  "UL" real-quantity "LL" real-quantity ;

pre-declared-enumeration:                                  ! 8.1E
   { "FALSE" | "TRUE" |
   "BNR" | "B1C" | "B2C" | "BSM" | "BCD" | "SBCD" |
   "ASCII7" | "ISO7" } ;

compare-statement:                                         ! 8.2
   fstatno "COMPARE" fd
   data-store fd evaluation-field
   statement-terminator ;


!  18.6.6  PROCEDURAL STATEMENTS - INPUT / OUTPUT


procedural-statements-input-output:                        ! 9.0
   { input-statement |
   output-statement |
   enable-file-access-statement |
   disable-file-access-statement } ;

input-statement:                                           ! 9.1
   fstatno "INPUT" fd
   { "GO-NOGO" |
   known-type-input | text-input | untyped-input }
   statement-terminator ;

known-type-input:                                          ! 9.1.5
   "FROM" file [ expression ] fd
   "INTO" { data-store # fd } ;

text-input:                                                ! 9.1.6
   [ "FROM" file fd ] "INTO"
   { { data-store
   [ "HEXADECIMAL" | "BINARY" | "OCTAL" ] } # fd } ;

untyped-input:                                             ! 9.1.7
   "FROM" file [ expression ] fd
   "INTO" { data-store # fd } ;

output-statement:                                       ! 9.2
   fstatno "OUTPUT" fd
   { known-type-output | text-output | untyped-output }
   statement-terminator ;

known-type-output:                                      ! 9.2.4
   "TO" file [ expression ] fd { expression # fd } ;

text-output:                                            ! 9.2.5
   [ "TO" file fd ]
   { { expression [ text-format ] } # fd } ;

text-format:                                            ! 9.2.5A
   ":" [ "E-FORMAT" | "OCTAL" |
   "HEXADECIMAL" | "BINARY" ]
   expression [ ":" expression ] ;

untyped-output:                                         ! 9.2.6
   "TO" file [ expression ] fd { expression # fd } ;

enable-file-access-statement:                           ! 9.3
   fstatno "ENABLE" fd
   { { "OUTPUT TO" | "I-O" }
   { "OLD" | "NEW" } |
   "INPUT FROM" }
   expression fd "VIA" file
   statement-terminator ;

disable-file-access-statement:                          ! 9.4
   fstatno "DISABLE" fd file
   statement-terminator ;


!  18.6.7  PROCEDURAL STATEMENTS - CONTROL


procedural-statements-control:                          ! 10.0
   { leave-statements |
   if-then-else-structure |
   while-then-structure |
   for-then-structure |
   go-to-statement |
   perform-statement |
   finish-statement |
   enable-digital-configuration-statement |
   disable-digital-configuration-statement |
   escape-structure } ;

leave-statements:                                       ! 10.0A
   { leave-procedure-statement |
   leave-if-statement |
   leave-while-statement |
   leave-for-statement } ;

if-then-else-structure:                                   ! 10.1.1
   if-then-statement
   procedural-segment-if-then-else
   [ else-statement
   procedural-segment-if-then-else ]
   end-if-statement ;

if-then-statement:                                        ! 10.1.2
   fstatno "IF" fd expression fd "THEN"
   statement-terminator ;

procedural-segment-if-then-else:                          ! 10.1.3
   main-procedural-structure ;

leave-if-statement:                                       ! 10.1.4
   fstatno "LEAVE" fd "IF"
   [ "STEP" statement-number ]
   statement-terminator ;

else-statement:                                           ! 10.1.5
   fstatno "ELSE"
   statement-terminator ;

end-if-statement:                                         ! 10.1.6
   fstatno "END" fd "IF"
   [ fd "'" message-text "'" ]
   statement-terminator ;

while-then-structure:                                     ! 10.2.1
   while-then-statement
   procedural-segment-while-then
   end-while-statement ;

while-then-statement:                                     ! 10.2.2
   fstatno "WHILE" fd expression
   fd "THEN"
   statement-terminator ;

procedural-segment-while-then:                            ! 10.2.3
   main-procedural-structure ;

leave-while-statement:                                    ! 10.2.4
   fstatno "LEAVE" fd "WHILE"
   [ "STEP" statement-number ]
   statement-terminator ;

end-while-statement:                                      ! 10.2.5
   fstatno "END" fd "WHILE"
   [ fd "'" message-text "'" ]
   statement-terminator ;
```

```

for-then-structure:                                        ! 10.3.1
   for-then-statement
   procedural-segment-for-then
   end-for-statement ;

for-then-statement:                                        ! 10.3.2
   fstatno "FOR" fd data-store "="
   for-then-range fd "THEN"
   statement-terminator ;

for-then-range:                                            ! 10.3.2-1
   { expression "THRU" expression
   [ "BY" expression ] |
   { expression # fd } } ;

procedural-segment-for-then:                               ! 10.3.3
   main-procedural-structure ;

leave-for-statement:                                       ! 10.3.4
   fstatno "LEAVE" fd "FOR"
   [ "STEP" statement-number ]
   statement-terminator ;

end-for-statement:                                         ! 10.3.5
   fstatno "END" fd "FOR"
   [ fd "'" message-text "'" ]
   statement-terminator ;

go-to-statement:                                           ! 10.4
   fstatno "GO TO"
   fd "STEP" statement-number
   statement-terminator ;

perform-statement:                                         ! 10.5
   fstatno "PERFORM" fd
   [ "TIMED DIGITAL" ] procedure
   [ "(" { expression # fd } ")" ]
   [ "RESULT" "(" { data-store # fd } ")" ]
   statement-terminator ;

finish-statement:                                          ! 10.6
   fstatno "FINISH"
   statement-terminator ;

enable-digital-configuration-statement:                    ! 10.7
   fstatno "ENABLE" fd
   "DIGITAL CONFIGURATION" configuration
   statement-terminator ;

disable-digital-configuration-statement:                   ! 10.8
   fstatno "DISABLE" fd
   "DIGITAL CONFIGURATION" [ configuration ]
   statement-terminator ;

escape-structure:                                          ! 10.9.0
   { enable-escape-to-procedure-statement |
   disable-escape-to-procedure-statement } ;

enable-escape-to-procedure-statement:                      ! 10.9.1
   fstatno "ENABLE" fd "ESCAPE TO PROCEDURE"
   procedure "IF"
   { { expression "ESCAPE NUMBER" unsigned-integer-number
   [ "WITH PRIORITY" unsigned-integer-number ] } # fd }
   statement-terminator ;

disable-escape-to-procedure-statement:                     ! 10.9.2
   fstatno "DISABLE" fd "ESCAPE TO PROCEDURE"
   { "ESCAPE NUMBER" { unsigned-integer-number # fd } |
   "ALL" } statement-terminator ;


!  18.6.8  PROCEDURAL STATEMENTS - SIGNAL


procedural-statements-signal:                              ! 11.1
   { single-action-statements |
   multiple-action-statements |
   digital-statements } ;

single-action-statements:                                  ! 11.2
   { setup-statement |
   connect-statement |
   disconnect-statement |
   arm-statement |
   fetch-statement |
   change-statement |
   enable-event-statement |
   disable-event-statement |
   enable-complex-signal-statement |
   disable-complex-signal-statement |
   reset-statement } ;

setup-statement:                                           ! 11.2.2
   fstatno "SETUP"
   [ single-action-body ] fd
   noun-field fd
   { statement-characteristics # fd }
   gate-conn-or-event-field
   statement-terminator ;

single-action-body:                                        ! 11.2.2-1
   fd "(" { { [ "COMPONENT" complex-signal ]
   measured-characteristic-mnemonic
   [ real-errlim ] } # fd } ")" ;

```

```
gate-conn-or-event-field:                                 ! 11.2.2-2
   { [ fd gate-field ] fd conn |
   fd event-field } ;

event-field:                                              ! 11.2.2-3
   { "FROM" event "TO" event |
   "OF" event-interval } fd max-time ;

connect-statement:                                        ! 11.2.3
   fstatno "CONNECT"
   [ single-action-body ] fd
   noun-field
   [ fd { statement-characteristics # fd } ]
   gate-conn-or-event-field
   statement-terminator ;

disconnect-statement:                                     ! 11.2.4
   fstatno "DISCONNECT"
   [ single-action-body ] fd
   noun-field
   [ fd { statement-characteristics # fd } ]
   gate-conn-or-event-field
   statement-terminator ;

arm-statement:                                            ! 11.2.5
   fstatno "ARM"
   single-action-body fd
   noun-field fd
   { statement-characteristics # fd }
   gate-conn-or-event-field
   statement-terminator ;

fetch-statement:                                          ! 11.2.6
   fstatno "FETCH" fd
   "(" { measured-characteristic # fd } ")" fd
   noun-field fd { statement-characteristics # fd }
   gate-conn-or-event-field
   statement-terminator ;

change-statement:                                         ! 11.2.7
   fstatno "CHANGE"
   [ single-action-body ] fd
   noun-field fd
   { statement-characteristics # fd }
   gate-conn-or-event-field
   statement-terminator ;

enable-event-statement:                                   ! 11.2.8
   fstatno "ENABLE" fd
   "EVENT" { event # fd }
   statement-terminator ;

disable-event-statement:                                  ! 11.2.9
```

```
   fstatno "DISABLE" fd
   "EVENT" { { event # fd } | "ALL" }
   statement-terminator ;

enable-complex-signal-statement:                          ! 11.2.10
   fstatno "ENABLE" fd "COMPLEX SIGNAL"
   complex-signal statement-terminator ;

disable-complex-signal-statement:                         ! 11.2.11
   fstatno "DISABLE" fd "COMPLEX SIGNAL"
   complex-signal statement-terminator ;

reset-statement:                                          ! 11.2.12
   fstatno "RESET"
   [ single-action-body ] fd
   noun-field fd
   { statement-characteristics # fd }
   gate-conn-or-event-field statement-terminator ;

multiple-action-statements:                               ! 11.3
   { apply-statement |
   remove-statement |
   measure-statement |
   monitor-statement |
   verify-statement |
   read-statement |
   initiate-statement } ;

apply-statement:                                          ! 11.3.2
   fstatno "APPLY" fd noun-field
   [ fd { statement-characteristics # fd } ]
   [ fd gate-field ] fd conn
   statement-terminator ;

remove-statement:                                         ! 11.3.3
   fstatno "REMOVE" fd
   { remove-characteristics | "ALL" }
   statement-terminator ;

remove-characteristics:                                   ! 11.3.3A
   { { remove-component # fd } |
   { { noun-field fd conn } # fd } | { conn # fd } } ;

remove-component:                                         ! 11.3.3B
   [ "(" { { [ "COMPONENT" complex-signal ]
   measured-characteristic-mnemonic } # fd } ")" fd ]
   noun-field
   [ fd { statement-characteristics # fd } ] fd conn ;

measure-statement:                                        ! 11.3.4
   fstatno "MEASURE" fd
   "(" { measured-characteristic # fd } ")" fd
   noun-field fd
   { statement-characteristics # fd }
   gate-conn-or-event-field
   statement-terminator ;

monitor-statement:                                          ! 11.3.5
   fstatno "MONITOR" fd
   "(" measured-characteristic ")" fd noun-field fd
   { statement-characteristics # fd }
   gate-conn-or-event-field
   statement-terminator ;

verify-statement:                                           ! 11.3.6
   fstatno "VERIFY" fd
   "(" measured-characteristic ")" fd noun-field fd
   eval-statement-characteristics
   gate-conn-or-event-field
   statement-terminator ;

read-statement:                                             ! 11.3.7
   fstatno "READ" fd
   "(" { measured-characteristic # fd } ")" fd
   noun-field fd
   { statement-characteristics # fd }
   gate-conn-or-event-field
   statement-terminator ;

initiate-statement:                                         ! 11.3.8
   fstatno "INITIATE" fd
   "(" { { [ "COMPONENT" complex-signal ]
   measured-characteristic-mnemonic
   [ real-errlim ] } # fd } ")" fd
   noun-field fd { statement-characteristics # fd }
   gate-conn-or-event-field
   statement-terminator ;

digital-statements:                                         ! 11.4
   { stimulate-statement |
   sense-statement |
   prove-statement } ;

stimulate-statement:                                        ! 11.4.2
   fstatno "STIMULATE" fd
   [ when-field fd ]
   { { stimulate-multiple-values | expression }
   [ fd "HIZ" { expression |
   stimulate-multiple-values } ] fd
   on-field | { { { "ZERO" | "ONE" | "HIZ" } fd
   on-field } # fd } }
   statement-terminator ;

stimulate-multiple-values:                                  ! 11.4.2A
   array-range [ fd "REPEAT"
   { array-range | unsigned-integer-number |
   data-store } ] ;

sense-statement:                                              ! 11.4.3
   fstatno "SENSE" fd [ when-field fd ]
   [ "SENSE-WINDOW" time-quantity fd ]
   "(" { "VALUE" | "HIZ" }
   "INTO" { data-store | array-range } ")" fd
   on-field statement-terminator ;

prove-statement:                                              ! 11.4.4
   fstatno "PROVE" fd [ when-field fd ]
   [ "SENSE-WINDOW" time-quantity fd ]
   { { { { "ZERO" | "ONE" | "HIZ" } fd
   on-field } # fd } |
   prove-hiz-or-value } statement-terminator ;

prove-hiz-or-value:                                           ! 11.4.4A
   "(" { "VALUE" | "HIZ" }
   [ "INTO" { data-store | array-range } ] ")" "REF"
   { expression | array-range }
   [ fd { "MASK-ONE" | "MASK-ZERO" }
   { expression | array-range } ]
   [ fd "SAVE-COMP" { data-store | array-range } ]
   [ fd "ERROR" { data-store | array-range } ]
   [ fd "ERROR-INDEX" { data-store | array-range } ]
   [ fd "FAULT-COUNT" data-store ] fd
   on-field ;


!  18.6.9  PROCEDURAL STATEMENTS - TIMING


procedural-statements-timing:                                 ! 12.0
   { read-timer-statement |
   wait-for-statement |
   reset-timer-statement |
   do-simultaneous-structure |
   do-timed-digital-structure } ;

read-timer-statement:                                         ! 12.2
   fstatno "READ"  fd
   "(" "TIME" "INTO" data-store ")" fd
   "TIMER" timer statement-terminator ;

wait-for-statement:                                           ! 12.3
   fstatno "WAIT FOR" fd
   { time-quantity
   [ fd "BEFORE STEP" statement-number ] |
   { "EVENT" event max-time |
   "TIMER" timer time-quantity }
   [ fd "THEN RESET" timer ] |
   "DIGITAL TIMER" digital-timer time-quantity
   [ fd "THEN RESET" { digital-timer # fd } ] }
   statement-terminator ;

reset-timer-statement:                                       ! 12.4
   fstatno "RESET" fd
   { "TIMER" { { timer # fd } | "ALL" } |
   "DIGITAL TIMER"
   { { digital-timer # fd } | "ALL" } }
   statement-terminator ;

do-simultaneous-structure:                                   ! 12.5.1
   do-simultaneous-statement
   do-simultaneous-body
   end-do-statement ;

do-simultaneous-statement:                                   ! 12.5.2
   fstatno "DO" fd "SIMULTANEOUS"
   [ fd when-field ]
   [ fd "WITHIN" time-quantity ]
   statement-terminator ;

do-simultaneous-body:                                        ! 12.5.3
   { procedural-statements-signal |
   read-timer-statement |
   reset-timer-statement |
   do-timed-digital-structure } % ;

end-do-statement:                                            ! 12.5.4
   fstatno "END" fd "DO"
   statement-terminator ;

do-timed-digital-structure:                                  ! 12.5.5
   do-timed-digital-statement
   do-timed-digital-body
   end-do-statement ;

do-timed-digital-statement:                                  ! 12.5.6
   fstatno "DO" fd "TIMED DIGITAL"
   [ fd when-field ]
   { { fd { stim-rate | stim-event }
   [ { fd sense-delay | fd sense-event } ] } |
   { fd { sense-event | sense-rate } } |
   { fd "DIGITAL TIMER" { digital-timer # fd } } }
   [ fd "ITERATE"
   { unsigned-integer-number | data-store } "TIMES" ]
   [ fd "FAULT-INDICATOR" variable-identifier ]
   [ fd "PROCEED" ]
   [ fd max-time ]
   statement-terminator ;

do-timed-digital-body:                                       ! 12.5.7
   { procedural-statements-data-processing |
   perform-statement |
   digital-statements |
   wait-for-statement |
   reset-timer-statement |
   do-timed-digital-structure |
   do-timed-control-structures |
   do-simultaneous-substructure } % ;

do-timed-control-structures:                           ! 12.5.7A
   { for-then-statement
   do-timed-digital-body
   end-for-statement |
   while-then-statement
   do-timed-digital-body
   end-while-statement |
   do-timed-if-then-structure |
   go-to-statement |
   leave-if-statement |
   leave-for-statement |
   leave-while-statement } ;

do-simultaneous-substructure:                          ! 12.5.7B
   do-simultaneous-statement
   [ { sense-statement |
   prove-statement } % |
   stimulate-statement % ]
   end-do-statement ;

do-timed-if-then-structure:                            ! 12.5.7C
   if-then-statement do-timed-digital-body
   [ else-statement do-timed-digital-body ]
   end-if-statement ;


!  18.6.10  PROCEDURAL STATEMENTS - DATABUS


procedural-statements-databus:                         ! 13.1
   { do-exchange-statement |
   update-exchange-configuration-statement |
   fetch-exchange-configuration-statement |
   enable-exchange-configuration-statement |
   connect-exchange-configuration-statement |
   disconnect-exchange-configuration-statement |
   disable-exchange-configuration-statement } ;

do-exchange-statement:                                 ! 13.2
   fstatno "DO" fd "EXCHANGE"
   "USING" exchange-configuration
   [ fd when-field ]
   [ fd role-field ] fd
   { { "START" | exchange-expression } fd
   { "PROCEED" | "WAIT" } |
   "HOLD" fd exchange-expression } [ fd max-time ]
   statement-terminator ;
```

```
update-exchange-configuration-statement:                  ! 13.3
   fstatno "UPDATE" fd
   { update-protocol | update-exchange }
   statement-terminator ;

update-protocol:                                          ! 13.3A
   protocol fd "PROTOCOL"
   "USING" exchange-configuration
   [ fd when-field ]
   { fd "STANDARD" | [ fd bus-parameter ]
   [ fd set-protocol-parameter ] } ;

update-exchange:                                          ! 13.3B
   exchange fd "EXCHANGE"
   "USING" exchange-configuration
   [ fd when-field ]
   [ fd "TALKER"
   { device-identifier | "TEST-EQUIP" | "UUT" }
   [ constant | data-store ] ]
   [ fd "LISTENER"
   { { { device-identifier | "TEST-EQUIP" | "UUT" }
   [ constant | data-store ] } # fd  } ]
   [ fd command-field ]
   [ fd data-field ]
   [ fd status-field ]
   [ fd bus-parameter ]
   [ fd set-protocol-parameter ] ;

fetch-exchange-configuration-statement:                   ! 13.4
   fstatno "FETCH" fd
   { fetch-protocol | fetch-exchange }
   statement-terminator ;

fetch-protocol:                                           ! 13.4A
   protocol fd "PROTOCOL" "USING"
   exchange-configuration
   [ fd when-field ]
   [ fd bus-parameter ]
   [ fd fetch-protocol-parameter ] ;

fetch-exchange:                                           ! 13.4B
   exchange fd "EXCHANGE" "USING"
   exchange-configuration
   [ fd when-field ]
   fetched-exchange-data
   [ fd max-time ] ;

fetched-exchange-data:                                    ! 13.4C
   [ fd "TALKER"
   { device-identifier | "TEST-EQUIP" | "UUT" }
   [ constant | data-store ] ]
   [ fd "LISTENER"
   { { { device-identifier | "TEST-EQUIP" | "UUT" }
   [ constant | data-store ] } # fd } ]
   [ fd "COMMAND" databus-fetch-data ]
   [ fd "DATA" databus-fetch-data ]
   [ fd "STATUS" databus-fetch-data ]
   [ fd bus-parameter ]
   [ fd fetch-protocol-parameter ] ;

enable-exchange-configuration-statement:                 ! 13.5
   fstatno "ENABLE" fd "EXCHANGE-CONFIGURATION"
   exchange-configuration fd
   { { "PROTOCOL" protocol
   protocol-initialization } # fd }
   statement-terminator ;

protocol-initialization:                                 ! 13.5A
   [ fd bus-parameter ]
   [ fd set-protocol-parameter ]
   fd "CNX" conn-set ;

connect-exchange-configuration-statement:                ! 13.6
   fstatno "CONNECT" fd "EXCHANGE-CONFIGURATION"
   exchange-configuration fd
   { { "PROTOCOL" protocol fd "CNX" conn-set } # fd }
   statement-terminator ;

disconnect-exchange-configuration-statement:             ! 13.7
   fstatno "DISCONNECT" fd "EXCHANGE-CONFIGURATION"
   exchange-configuration fd
   { { "PROTOCOL" protocol fd
   "CNX" conn-set } # fd }
   statement-terminator ;

disable-exchange-configuration-statement:                ! 13.8
   fstatno "DISABLE" fd "EXCHANGE-CONFIGURATION"
   exchange-configuration
   statement-terminator ;


!  18.6.11  FIELDS AND SUBFIELDS


statement-characteristics:                               ! 14.1
   { [ "COMPONENT" complex-signal ]
   modifier-mnemonic
   [ real-characteristic-subfield |
   modifier-descriptor ] |
   "STROBE-TO-EVENT" event [ max-time ] |
   digital-characteristic-subfield | sync-subfield |
   mark-descriptor-subfield |
   sweep-configuration-subfield } ;

real-characteristic-subfield:                            ! 14.2
```

```
   { real-control-characteristic |
   real-capability-characteristic |
   real-limit-characteristic } ;

real-control-characteristic:                                  ! 14.2A
   signal-value [ real-errlim ] ;

real-capability-characteristic:                               ! 14.2B
   { { max-min | "NOM" } signal-value |
   "RANGE" signal-value "TO" signal-value } ;

real-limit-characteristic:                                    ! 14.2C
   { "LIMIT-TO" { max-min signal-value |
   "RANGE" signal-value "TO" signal-value } } ;

digital-characteristic-subfield:                              ! 14.3
   { { "TRANS-ZERO" | "TRANS-SYNC" | "TRANS-ONE" }
   { { bit-q-0-1 % } # "OR" } |
   "TRANS-PERIOD" bit-q-0-1 bit-0-1 |
   digital-char-volt-curr |
   "TYPE" { "PARALLEL" | "SERIAL-LSB-FIRST" |
   "SERIAL-MSB-FIRST" } |
   "PULSE-CLASS" modifier-descriptor |
   "VALUE" expression } ;

digital-char-volt-curr:                                       ! 14.3-1
   { "VOLTAGE-ONE" | "VOLTAGE-ZERO" |
   "VOLTAGE-QUIES" | "CURRENT-ONE" |
   "CURRENT-ZERO" | "CURRENT-QUIES" }
   real-characteristic-subfield ;

bit-q-0-1:                                                    ! 14.3-2
   { "QUIES" | "0" | "1" } ;

bit-0-1:                                                      ! 14.3-3
   { "0" | "1" } ;

sync-subfield:                                                ! 14.4
   "SYNC" [ sync-body ]
   "TO EVENT" event [ max-time ] ;

sync-body:                                                    ! 14.4A
   { modifier-mnemonic [ evaluation-field ]
   pos-neg-slope |
   "INITIAL" sync-initial |
   "BIT-TRANSITION" |
   "WORD-TRANSITION" |
   "FREQ" } ;

sync-initial:                                                 ! 14.4B
   { { "VOLTAGE 0V" | "CURRENT 0A" }
   [ real-errlim ]  pos-neg-slope |
   "VOLTAGE-ONE" | "VOLTAGE-ZERO" |
   "VOLTAGE-QUIES"| "CURRENT-ONE" |
   "CURRENT-ZERO" | "CURRENT-QUIES" |
   "VOLTAGE-P-POS" | "VOLTAGE-P-NEG" |
   "CURRENT-P-POS" | "CURRENT-P-NEG" } ;

real-errlim:                                              ! 14.5
   "ERRLMT" dim-lim ;

dim-lim:                                                  ! 14.5A
   { "+-" unsigned-decimal-number { dim | "PC" } |
   { "+" unsigned-decimal-number { dim | "PC" }
   "-" unsigned-decimal-number { dim | "PC" } } |
   { "-" unsigned-decimal-number { dim | "PC" }
   "+" unsigned-decimal-number { dim | "PC" } } } ;

measured-characteristic:                                  ! 14.6
   [ "COMPONENT" complex-signal ]
   measured-characteristic-mnemonic [ real-errlim ]
   [ "INTO" data-store ] ;

evaluation-field:                                         ! 14.7
   { [ "NOM" real-or-digital-quantity ]
   uplowlimit-real-dig |
   comparison-operator real-or-digital-quantity } ;

uplowlimit-real-dig:                                      ! 14.7-1
   { "UL" real-or-digital-quantity
   "LL" real-or-digital-quantity } ;

real-or-digital-quantity:                                 ! 14.7A
   real-quantity | digital-quantity ;

eval-statement-characteristics:                           ! 14.8
   evaluation-field
   [ fd { statement-characteristics # fd } ] ;

max-time:                                                 ! 14.9
   "MAX-TIME" time-quantity ;

time-quantity:                                            ! 14.10
   expression time-dimension ;

condition:                                                ! 14.11
   { "MAX-TIME" |
   "LO" | "HI" |
   "NOGO" | "GO" } ;

conn:                                                     ! 14.12
   "CNX" conn-set
   [ "REF" { reference-phase | conn-set } ] ;

conn-set:                                                 ! 14.13
   { [ pin-descriptor ] conn-descriptor } % ;
```

```

conn-descriptor:                                          ! 14.13-1
   { connection | data-store } % ;

signal-value:                                             ! 14.14
   { expression
   { max-min decimal-number dim | range-to } |
   array-range dim | decimal-number dim }
   [ "NORMALIZE" array-range "SPEC"
   external-compensation-specification ] ;

real-quantity:                                            ! 14.15
   expression dim ;

index:                                                    ! 14.16
   expression ;

noun-field:                                               ! 14.17
   { "COMPLEX SIGNAL" complex-signal |
   { noun  | noun-name } [ "USING" requirement ] } ;

dimensioned-number:                                       ! 14.18
   decimal-number dim ;

require-control:                                          ! 14.19
   fd "CONTROL"
   { fd modifier-mnemonic control-fields } % ;

control-fields:                                           ! 14.19A
   { { { range-to { "BY" dimensioned-number |
   "CONTINUOUS" } } | { dimensioned-number # fd } }
   [ real-errlim ] |
   modifier-descriptor } ;

range-to:                                                 ! 14.19B
   "RANGE" dimensioned-number
   "TO"    dimensioned-number ;

require-capability:                                       ! 14.20
   fd "CAPABILITY"
   { fd modifier-mnemonic capability-fields } % ;

capability-fields:                                        ! 14.20A
   { { range-to [ "BY" dimensioned-number ] |
   max-min dimensioned-number } [ real-errlim ] |
   modifier-descriptor } ;

require-limit:                                            ! 14.21
   fd "LIMIT" fd
   { { modifier-mnemonic limit-field } # fd } ;

limit-field:                                              ! 14.21A
   { range-to [ "BY" dimensioned-number ] |
   max-min dimensioned-number } ;

require-cnx:                                                    ! 14.22
   fd "CNX"
   { pin-descriptor } %
   [ "REF" { pin-descriptor } % ] ;

digital-quantity:                                               ! 14.23
   expression ;

data-store:                                                     ! 14.24
   variable-identifier
   [ { "(" { index # fd } ")" |
   "." record-field-identifier } % ] ;

array-range:                                                    ! 14.25
   data-store
   "(" { { index "THRU" index [ "BY" index ] |
   { index # "THEN" } } # fd }  ")" ;

parameter:                                                      ! 14.26
   { variable-identifier # fd } "IS" type ;

gate-field:                                                     ! 14.27
   "GATED"
   { "FROM" event "TO" event |
   "BY" event-interval } ;

stim-rate:                                                      ! 14.28
   "STIM-RATE"
   { unsigned-decimal-number | data-store } dim ;

sense-rate:                                                     ! 14.29
   "SENSE-RATE"
   { unsigned-decimal-number | data-store } dim ;

sense-delay:                                                    ! 14.30
   "SENSE-DELAY" time-quantity ;

stim-event:                                                     ! 14.31
   "STIM-EVENT"
   { { event | event-interval } # fd }
   [ max-time ] ;

sense-event:                                                    ! 14.32
   "SENSE-EVENT" { event # fd } [ max-time ] ;

when-field:                                                     ! 14.33
   "WHEN"
   { { timer | digital-timer }
   time-quantity [ real-errlim ] |
   event max-time } ;

```

```
digital-source-characteristics:                         ! 14.34
   { { modifier-mnemonic
   real-characteristic-subfield } # fd } ;

digital-sensor-characteristics:                         ! 14.35
   [ evaluation-field fd ]
   { { modifier-mnemonic
   real-characteristic-subfield } # fd } |
   evaluation-field ;

on-field:                                               ! 14.36
   "ON" { digital-source | digital-sensor }
   [ [ "EXCEPT" ] "CNX" conn-set ] ;

exchange-expression:                                    ! 14.37
   { { { "(" { exchange-frame # { "OR" | "AND" } }
   ")" } # "THEN" } |
   exchange-frame } ;

exchange-frame:                                         ! 14.38
   "STARTING" when-field
   [ fd "WITH PRIORITY" unsigned-integer-number ]
   [ fd "EVERY" { "EVENT" event | time-quantity } ]
   [ fd "UNTIL"
   { [ "TIMER" timer "EQ" ] time-quantity |
   "EVENT" event |
   "EXCHANGE" exchange } ]
   { fd "EXCHANGE" databus-transaction } % ;

databus-transaction:                                    ! 14.38A
   exchange [ fd role-field ]
   [ fd test-equip-transaction-role ]
   [ fd "DELAY" time-quantity ] ;

test-equip-transaction-role:                            ! 14.38B
   [ "TEST-EQUIP-SIMULATE" test-equip-sim ]
   [ fd "TEST-EQUIP-MONITOR"
   { fetch-protocol-parameter |
   bus-parameter | fetch-databus-data } ] ;

test-equip-sim:                                         ! 14.38C
   [ "TALKER" simulate-device ]
   [ "LISTENER" { simulate-device # fd } ] ;

simulate-device:                                        ! 14.38D
   { device-identifier | "UUT" }
   [ "(" data-store ")" | "(" constant ")" ] ;

fetch-databus-data:                                     ! 14.38E
   [ fd "COMMAND" databus-fetch-data ]
   [ fd "DATA" databus-fetch-data ]
   [ fd "STATUS" databus-fetch-data ] ;

```

```
bus-parameter:                                       ! 14.39
   "BUS-PARAMETER"
   { { { "WORD-LENGTH" | "WORD-GAP" |
   "MESSAGE-GAP" | "RESPONSE-TIME" |
   "ZERO-AMPLITUDE" | "ONE-AMPLITUDE" |
   "ZERO-CROSSING" | bus-parameter-name }
   bus-parameter-data } # fd } ;

bus-parameter-data:                                  ! 14.39A
   [ { constant | data-store } dim |
   modifier-descriptor-name ] ;

set-protocol-parameter:                              ! 14.40
   "PROTOCOL-PARAMETER"
   { { [ "INSERT-INVALID" ]
   { set-word-info |
   "WORD-COUNT" [ sign unsigned-integer-number ] |
   protocol-parameter-name [ data-store ] } } # fd } ;

set-word-info:                                       ! 14.40A
   { "COMMAND-WORD" | "DATA-WORD"  | "STATUS-WORD" }
   [ [ unsigned-integer-number ]
   { "BIT" unsigned-integer-number "PULSE-CODE" |
   "SYNC" | "T-R" | "PARITY" | "LENGTH" } ] ;

role-field:                                          ! 14.41
   "TEST-EQUIP-ROLE"
   { test-equip-role-name |
   "MONITOR" | "MASTER" | "SLAVE" } ;

command-field:                                       ! 14.42
   "COMMAND"
   { [ "ENTIRE" ] array-range |
   data-store | constant } ;

data-field:                                          ! 14.43
   "DATA" { [ "ENTIRE" ]
   { "FROM" file [ expression ] |
   array-range } |
   data-store | constant } ;

status-field:                                        ! 14.44
   "STATUS"
   { [ "ENTIRE" ] array-range |
   data-store | constant } ;

mark-descriptor-subfield:                            ! 14.45
   "AT" { power-mark-descriptor
   "TO" power-mark-descriptor|
   frequency-mark-descriptor "TO"
   frequency-mark-descriptor } ;

power-mark-descriptor:                               ! 14.45A
```

```
   { "POWER-MARK" | "POWER-MARK-UPPER" |
   "POWER-MARK-LOWER" } unsigned-integer-number ;

frequency-mark-descriptor:                                ! 14.45B
   { "FREQUENCY-MARK" | "FREQUENCY-MARK-UPPER" |
   "FREQUENCY-MARK-LOWER" |
   "FREQUENCY-MARK-HARM" } unsigned-integer-number ;

proportionality-subfield:                                 ! 14.46
   numerator-modifier-mnemonic "PROPORTIONAL TO"
   denominator-modifier-mnemonic fd
   numerator-modifier-mnemonic
   "RANGE" signal-value "TO" signal-value fd
   denominator-modifier-mnemonic "RANGE"
   signal-value "TO" signal-value ;

numerator-modifier-mnemonic:                              ! 14.46A
   modifier-mnemonic ;

denominator-modifier-mnemonic:                            ! 14.46B
   modifier-mnemonic ;

sweep-configuration-subfield:                             ! 14.47
   { { "SWEEP-CONFIGURATION"
   smooth-step-info sweep-info } # fd } ;

smooth-step-info:                                         ! 14.47A
   { "SMOOTH" fd
   smooth-sweep-parameter fd
   smooth-sweep-descriptor |
   "STEP" fd
   { step-sweep-parameter-1 |
   step-sweep-parameter-2 } fd
   step-sweep-descriptor } ;

sweep-info:                                               ! 14.47B
   [ { { fd "SWEEP-TYPE"
   { "SPEC" external-sweep-specification |
   "RANDOM" | "LIN" | "LOG" | "SIN" } |
   "SWEEP-TIME" real-characteristic-subfield |
   "SWEEP-TRIG" sync-subfield |
   "SWEEP-ACTIVE" gate-field } % } ] ;

smooth-sweep-parameter:                                   ! 14.47C
   { "FREQ" | "VOLTAGE" | "PHASE-ANGLE" |
   "POWER" } "RANGE" signal-value "TO" signal-value ;

smooth-sweep-descriptor:                                  ! 14.47D
   { "FORWARD" | "FORWARD-HOLD" |
   "FORWARD-HOLD-RESET-HOLD" |
   "FORWARD-HOLD-REVERSE-HOLD" |
   "FORWARD-REVERSE" | "REVERSE" } ;

```

```
step-sweep-parameter-1:                                  ! 14.47E
   { "FREQ" | "VOLTAGE" |
   "PHASE-ANGLE" | "POWER" } array-range ;

step-sweep-parameter-2:                                  ! 14.47F
   { "FREQ" | "VOLTAGE" |
   "PHASE-ANGLE" | "POWER" }
   "RANGE" signal-value "TO" signal-value
   [ "BY" dimensioned-number ] ;

step-sweep-descriptor:                                   ! 14.47G
   { "FSK" | "SEQUENCE" |
   "SEQUENCE-HOLD" | "SEQUENCE-HOLD-RESET-HOLD" |
   "SEQUENCE-STEP" } ;

fetch-protocol-parameter:                                ! 14.48
   "PROTOCOL-PARAMETER"
   { { fetch-word-info |
   protocol-parameter-name
   { "FAULT-COUNT" data-store | data-store } } # fd } ;

fetch-word-info:                                         ! 14.48A
   { { "COMMAND-WORD" | "DATA-WORD" | "STATUS-WORD" }
   { "SYNC" | "T-R" | "PULSE-CODE" | "PARITY" |
   "LENGTH" } | "WORD-COUNT" }
   "FAULT-COUNT" data-store ;

databus-fetch-data:                                      ! 14.49
   { { file [ expression ] | array-range }
   [ fd "COUNT-INTO" data-store ] |
   data-store } ;

file:                                                    ! 14.50
   data-store ;


! 18.6.12  BASIC SYMBOLS AND CHARACTER SETS


non-printing-character:                                  ! 15.1.1
   { "\" {"NUL" | "SOH" | "STX" | "ETX" | "EOT" |
   "ENQ" | "ACK" | "BEL" | "BS"  | "HT"  |
   "LF"  | "VT"  | "FF"  | "CR"  | "SO"  |
   "SI"  | "DLE" | "DC1" | "DC2" | "DC3" |
   "DC4" | "NAK" | "SYN" | "ETB" | "CAN" |
   "EM"  | "SUB" | "ESC" | "FS"  | "GS"  |
   "RS"  | "US"  | "DEL" } "\" } ;

common-char-set:                                         ! 15.1.1
   { symbol-extension-set | lower-case-letter |
   letter | digit | "*" | "." | "=" | "+" | "-" } ;

!  Symbols not contained in the common-char-set are:
```

```
!  ",", "$", "/", "(", ")", " ", "'",
!  and the non-printable control-characters.

symbol-extension-set:                                  ! 15.1.1
   { "|" | """ | "#" | "%" | "&" | ":" |
   ";" | "<" | ">" | "?" | "@" | "^" |
   "[" | "]" | "\" | "*" | "." | "!" |
   "{" | "}" | "~" | "`" | "'" | "_" } ;

lower-case-letter:                                     ! 15.1.1
   { "a" | "b" | "c" | "d" | "e" | "f" |
   "g" | "h" | "i" | "j" | "k" | "l" |
   "m" | "n" | "o" | "p" | "q" | "r" |
   "s" | "t" | "u" | "v" | "w" | "x" |
   "y" | "z" } ;

decimal-number:                                        ! 15.2.1
   [ sign ] unsigned-decimal-number ;

sign:
   { "+" | "-" } ;

unsigned-decimal-number:                               ! 15.2.2
   { digit % [ "." [ digit % ] ] |  "." digit % }
   [ "E" [ sign ] digit % ] ;

unsigned-integer-number:                               ! 15.2.3
   digit % ;

digital-number:                                        ! 15.2.4
   { "B" "'" { "0" | "1" } * "'" |
   "O" "'" { "0" | "1" | "2" | "3" |
   "4" | "5" | "6" | "7" } * "'" |
   "X" "'" { digit |
   "A" | "B" | "C" | "D" | "E" | "F" } * "'" } ;

blank:                                                 ! 15.4
   " " ;

parentheses:
   { "(" | ")" } ;

character-string:                                      ! 15.10.1
   "C" "'" char-string-set * "'" ;

char-string-set:
   { non-printing-character | common-char-set |
   blank | "/" | "," |  parentheses } ;

message-text:                                          ! 15.10.2
   char-string-set % ;

label-char-set:
   { common-char-set | "/" | "," } ;

name-char-set:
   { common-char-set | blank | parentheses } ;

letter:                                                   ! 15.10.3
   { "A" | "B" | "C" | "D" | "E" | "F" |
   "G" | "H" | "I" | "J" | "K" | "L" |
   "M" | "N" | "O" | "P" | "Q" | "R" |
   "S" | "T" | "U" | "V" | "W" | "X" |
   "Y" | "Z" } ;

digit:                                                    ! 15.10.4
   { "0" | "1" | "2" | "3" | "4" | "5" |
   "6" | "7" | "8" | "9" } ;


!  18.6.13  IDENTIFIERS AND LABELS


connection:                                               ! 14.13
   { { letter | lower-case-letter | "/" letter |
   digit | "+" | "-" | "." } % |
   "COMMON" | "EARTH" | "ATMOS" } ;

label:                                                    ! 15.5
   "'" { blank * label-char-set } % blank * "'" ;

requirement:                                              ! 15.6.8
   label ;

complex-signal:                                           ! 15.6.10
   label ;

module-name:                                              ! 15.10.6
   label ;

program-name:                                             ! 15.10.7
   label ;

extend-label:                                             ! 15.10.8
   { blank * label-char-set } % blank * ;

constant-identifier:                                      ! 15.11.2
   label ;

type-identifier:                                          ! 15.11.3
   label ;

variable-identifier:                                      ! 15.11.4
   label ;

enumeration-element:                                      ! 15.11.5
```

```
   label ;

record-field-identifier:                                  ! 15.11.6
   label ;

procedure:                                                ! 15.11.8
   label ;

signal:                                                   ! 15.11.9
   label ;

event:                                                    ! 15.11.10
   label ;

event-indicator:                                          ! 15.11.11
   label ;

event-interval:                                           ! 15.11.12
   label ;

timer:                                                    ! 15.11.13
   label ;

protocol:                                                 ! 15.11.14
   label ;

exchange:                                                 ! 15.11.15
   label ;

configuration:                                            ! 15.11.16
   label ;

digital-source:                                           ! 15.11.17
   label ;

digital-sensor:                                           ! 15.11.18
   label ;

dim-name:                                                 ! 15.11.19
   extend-label ;

noun-name:                                                ! 15.11.22
   extend-label ;

pin-descriptor-name:                                      ! 15.11.24
   extend-label ;

modifier-name:                                            ! 15.11.25
   extend-label ;

digital-timing:                                           ! 15.11.27
   label ;

```

```
device-identifier:                                      ! 15.11.28.1
   label ;

test-equip-role-name:                                   ! 15.11.28.2
   extend-label ;

bus-parameter-name:                                     ! 15.11.28.3
   extend-label ;

protocol-parameter-name:                                ! 15.11.28.4
   extend-label ;

bus-mode-name:                                          ! 15.11.28.7
   extend-label ;

external-bus-specification:                             ! 15.11.29
   label ;

modifier-descriptor-name:                               ! 15.11.30
   extend-label ;

external-compensation-specification:                    ! 15.11.32
   label ;

external-complex-function-specification:                ! 15.11.33
   label ;

external-signal-conditioning-specification:             ! 15.11.34
   label ;

external-signal-name-specification:                     ! 15.11.35
   label ;

external-sweep-specification:                           ! 15.11.37
   label ;

external-specification:
   label ;

external-digital-specification:
   label ;

external-pulse-class-specification:
   label ;

digital-timer:                                          ! 15.11.50
   label ;

exchange-configuration:                                 ! 15.11.51
   label ;

boolean-variable:                                       ! 15.11.52
   variable-identifier ;
```

```
! 18.6.14 SEPARATORS AND STATEMENT-NUMBERS


fd:                                                        ! 3.2.4
   "," ;

statement-terminator:                                      ! 3.2.6
   "$" ;

fstatno:                                                   ! 15.3.1
   { "E" | blank } statno ;

test-number:                                               ! 15.3.1
   digit digit digit digit ;

step-number:                                               ! 15.3.1
   digit digit ;

comment-statement:                                         ! 15.3.2
   { "B" | "C" } [ message-text ]
   statement-terminator ;

statno:                                                    ! 15.3.1
   { test-number step-number |
   blank blank blank blank step-number |
   blank blank blank blank blank blank } ;

statement-number:                                          ! 15.3.3
   test-number step-number ;


! 18.6.15 OPERATORS


comparison-operator:
   { "EQ" | "NE" | "GT" | "LT" | "GE" | "LE" } ;

equal-notequal-operator:
          { "EQ" | "NE" } ;

unary-operator:
   { "+" | "-" | "NOT" } ;

pre-defined-function:
   { "EOF" | "ODD" | "PE" | "PO" | "ORD" | "LEN" |
   "LOCN" | "COUNT" | "SIZE" | "SUCC" | "PRED" |
   "DATE" | "CHAR" | "BITS" | "COPY" | "ROTATE" |
   "SHIFT" | "A-SHIFT" | "DIG" | "DEC" | "INT" |
   "ROUND" | "ABS" | "SQRT" |
   "SIN" | "COS" | "TAN" | "ATAN" |
   "LN" | "EXP" | "LOG" | "ALOG" |
   "DELETE" | "INSERT" | "TRIM" } ;

arith-string-operator:
   { "**" | "*" | "/" | "DIV" | "MOD" |
   "+" | "-" | "&" } ;

boolean-operator:
   { "AND" | "OR" | "XOR" } ;


!  18.6.16  DESCRIPTORS


modifier-descriptor:                                                  ! 15.10.5
   { mod-atc-mode |
   mod-atmos |
   mod-channel |
   mod-fluid-type |
   mod-fuel-supply |
   mod-ident-sig |
   mod-iff-mode |
   mod-pulse-class |
   mod-pulses-incl-excl |
   mod-ref-pulses |
   mod-tacan-mode |
   mod-logic-data-type |
   mod-mod-mode |
   mod-trans-bits |
   mod-vector-descriptor } ;

mod-atc-mode:
   { "A" | "B" | "C" | "D" | "A-C" } ;

mod-atmos:
   { "ARDC" | "ICAN" |
   "ICAO" | "WADC" } ;

mod-channel:
   unsigned-integer-number [ "-X" | "-Y" ] ;

mod-fluid-type:
   { "AIR" | "FUEL" | "GAS" | "N2" | "OIL" | "WATER" |
   "DTD-" unsigned-integer-number |
   "MIL-" unsigned-integer-number } ;

mod-fuel-supply:
   { "ON" | "OFF" } ;

mod-ident-sig:
   letter [ letter [ letter [ letter ] ] ] ;

mod-iff-mode:
   {"1" | "2" | "3-A" | "3-C" | "4" | "C" } ;
```

```

mod-pulse-class:
   { "RZ" | "NRZ" | "BIP" | "MIP" |
   "AMI" | "HDB" unsigned-integer-number } ;

mod-pulses-incl-excl:
   { unsigned-integer-number # "-" } | "ALL" ;

mod-ref-pulses:
   { "MAIN" | "AUX" }
   [ unsigned-integer-number # "-" ] ;

mod-tacan-mode:
   { "RO" | "GA" | "AA" | "AB" } "-" { "I" | "N" } ;

mod-trans-bits:
   { "0" | "1" | "Q" } %
   [ "OR" { "0" | "1" | "Q" } % ] ;

mod-logic-data-type:
   { "PARALLEL" |
   "SERIAL-LSB-FIRST" | "SERIAL-MSB-FIRST" } ;

mod-mod-mode:
   { "LIN" | "LOG" } ;

mod-vector-descriptor:
   { "BPSK" | "QPSK" | "OQPSK" | "8PSK" |
   "OQAM" | "16QAM" | "64QAM" | "256QAM" |
   "CPFSK" | "BFSK" | "MFSK" } ;

reference-phase:                                       ! 15.11.1
   { "PHASE-A" | "PHASE-AB" | "PHASE-AC" |
   "PHASE-B" | "PHASE-BC" | "PHASE-CB" |
   "PHASE-C" | "PHASE-CA" | "PHASE-BA" } ;

pin-descriptor:                                        ! 15.11.23
   { pin-descriptor-name |
   "HI" |  "A" |
   "LO" |  "B" |
   "VIA" | "C" | "N" |
   "TRUE" |  "TO" | "SCREEN" |
   "COMPL" | "FROM" | "GUARD" |
   "R1" | "S1" | "X" |
   "R2" | "S2" | "Y" |
   "R3" | "S3" | "Z" |
   "R4" | "S4" } ;


!  18.6.17  NOUNS


noun:                                                  ! 16
```

```
{ noun-name |
"AC SIGNAL" |
"ADF" |
"AM SIGNAL" |
"AMBIENT CONDITIONS" |
"ATC" |
"COMMON" |
"COMPLEX SIGNAL" |
"DC SIGNAL" |
"DISPLACEMENT" |
"DME" |
"DOPPLER" |
"EARTH" |
"EM FIELD" |
"EVENTS" |
"FLUID SIGNAL" |
"FM SIGNAL" |
"HEAT" |
"IFF" |
"ILS" |
"IMPEDANCE" |
"LIGHT" |
"LOGIC CONTROL" |
"LOGIC DATA" |
"LOGIC LOAD" |
"LOGIC REFERENCE" |
"MANOMETRIC" |
"PAM" |
"PM SIGNAL" |
"PULSED AC" |
"PULSED AC TRAIN" |
"PULSED DC" |
"PULSED DC TRAIN" |
"PULSED DOPPLER" |
"RADAR SIGNAL" |
"RAMP SIGNAL" |
"RANDOM NOISE" |
"RESOLVER" |
"ROTATION" |
"SHORT" |
"SQUARE WAVE" |
"STEP SIGNAL" |
"SUP CAR SIGNAL" |
"SYNCHRO" |
"TACAN" |
"TIME INTERVAL" |
"TRIANGULAR WAVE SIGNAL" |
"TURBINE ENGINE DATA" |
"VIBRATION" |
"VOR" |
"WAVEFORM" } ;
```

```
! 18.6.18  NOUN-MODIFIERS


measured-characteristic-mnemonic:
   modifier-mnemonic ;

modifier-mnemonic:                                         ! 17.4
   { modifier-name |
   "AC-COMP" |
   "AC-COMP-FREQ" |
   "AGE-RATE" |
   "ALT" |
   "ALT-RATE" |
   "AM-COMP" |
   "AM-SHIFT" |
   "AMPL-MOD" [ "-C" | "-F" ] |
   "ANGLE" [ angle-sfx ] |
   "ANGLE-ACCEL" [ angle-sfx ] |
   "ANGLE-RATE" [ angle-sfx ] |
   "ANT-SPEED-DEV" |
   "ATMOS" |
   "ATTEN" |
   "BANDWIDTH" |
   "BAROMETRIC-PRESS" |
   "BIT-RATE" |
   "BURST" |
   "BURST-DROOP" |
   "BURST-REP-RATE" |
   "CAP" |
   "CAR-AMPL" |
   "CAR-FREQ" |
   "CAR-HARMONICS" |
   "CAR-PHASE" |
   "CAR-RESID" |
   "CHANNEL" mod-channel |
   "COMPL" |
   "CONDUCTANCE" |
   "COUNT" |
   "CREST-FACTOR" |
   "CURRENT" [ ampl-sfx | phase-sfx | pulse-sfx |
               log-noun-mod-sfx ]
   "CURRENT-LMT" |
   "CW-LEVEL" |
   "DBL-INT" |
   "DC-OFFSET" |
   "DDM" |
   "DEBRIS-COUNT" |
   "DEBRIS-SIZE" |
   "DEWPOINT" |
   "DISS-FACTOR" |
   "DISTANCE" [ dist-sfx | angle-sfx ] |
   "DISTORTION" |
   "DOMINANT-MOD-SIG" |
   "DOPPLER-BANDWIDTH" |
   "DOPPLER-FREQ" |
   "DOPPLER-SHIFT" |
   "DROOP" [ pulse-sfx ] |
   "DUTY-CYCLE" |
   "EFF" |
   "EFFICACY" |
   "FALL-TIME" [ pulse-sfx ] |
   "FIELD-STRENGTH" |
   "FLUID-TYPE" |
   "FLUX-DENS" [ flux-dens-sfx ] |
   "FM-COMP" |
   "FORCE" |
   "FORCE-RATE" |
   "FREQ" [ log-noun-mod-sfx ] |
   "FREQ-DEV" |
   "FREQ-PAIRING" |
   "FREQ-WINDOW" |
   "FUEL-SUPPLY" |
   "GLIDE-SLOPE" |
   "HARMONICS" |
   "HARM" "-" xxx "-" { "PHASE" | "POWER" |
                        "VOLTAGE" } |
   "HI-MOD-FREQ" |
   "HUMIDITY" |
   "IAS" |
   "IDENT-SIG" [ "-EP" | "-FREQ" | "-MOD" ] |
   "ILLUM" |
   "IND" |
   "INT" { "-JITTER" | "-RATE" } |
   "LO-MOD-FREQ" |
   "LOCALIZER" |
   "LUMINANCE" |
   "LUM" { "-FLUX" | "-INT" | "-TEMP" } |
   "MAG-BEARING" [ "-RATE" ] |
   "MARKER-BEACON" |
   "MASS-FLOW" |
   "MEAN-MOD" |
   "MOD" { "-AMPL" | "-DIST" | "-FREQ" | "-OFFSET" |
           "-PHASE" }|
   "MODE" [ mod-atc-mode | mod-iff-mode |
            mod-tacan-mode ] |
   "NEG-SLOPE" |
   "NOISE" [ "-AV" | "-P" | "-PP" | "-TRMS" ] |
   "NOISE-AMPL-DENS" |
   "NOISE-PWR-DENS" |
   "NON-HARMONICS" |
   "NON-LIN" |
   "OPER-TEMP" |
   "OVERSHOOT" [ pulse-sfx ] |
   "P-AMPL" |
   "P3-DEV" |
   "P3-LEVEL" |
   "PAIR-DROOP" |
   "PAIR-SPACING" |
   "PEAK-DEGEN" |
   "PERIOD" |
   "PHASE-ANGLE" |
   "PHASE-DEV" |
   "PHASE-JIT" |
   "PHASE-SHIFT" |
   "PLA" |
   "PLA-RATE" |
   "POS-SLOPE" |
   "POWER" [ "-AV" | "-P" ] |
   "POWER-DIFF" |
   "PRESHOOT" [ pulse-sfx ] |
   [ press-temp-pfx ] "PRESS-A" |
   [ press-temp-pfx ] "PRESS-G" |
   [ press-temp-pfx ] "PRESS-RATE" |
   "PRESS-OSC-AMPL" |
   "PRESS-OSC-FREQ" |
   "PRF" |
   "PULSE-CLASS" |
   "PULSE-IDENT" |
   "PULSE-POSN" pulse-posn-sfx |
   "PULSE-SPECT" |
   "PULSE-SPECT-THRESHOLD" |
   "PULSE-WIDTH" [ pulse-sfx ] |
   "PULSES-EXCL" |
   "PULSES-INCL" |
   "PWR-LMT" |
   "Q" |
   "QUAD" |
   "RADIAL" |
   "RADIAL-RATE" |
   "RANGE-PULSE-DEV" |
   "RANGE-PULSE-ECHO" |
   "REACTANCE" |
   "REF-FREQ" |
   "REF-INERTIAL" |
   "REF-PHASE-FREQ" |
   "REF-POWER" |
   "REF-PULSES" |
   "REF-PULSES-EXCL" |
   "REF-PULSES-INCL" |
   "REF-PULSES-DEV" |
   "REF-UUT" |
   "REF-VOLT" |
   "REL-BEARING" |
   "REL-BEARING-RATE" |
   "RELATIVE-HUMIDITY" |
   "RELATIVE-WIND" |
   "REPLY-EFF" |
   "RES" |
   "RESP" |
   "RINGING" [ pulse-sfx ] |
   "RISE-TIME" [ pulse-sfx ] |
   "ROTOR SPEED" |
   "ROUNDING" [ pulse-sfx ] |
   "SAMPLE" |
   "SAMPLE-SPACING" |
   "SAMPLE-TIME" |
   "SAMPLE-WIDTH" |
   "SETTLE-TIME" |
   "SHAFT-SPEED" |
   "SKEW-TIME" |
   "SLANT-RANGE" |
   "SLANT-RANGE-ACCEL" |
   "SLANT-RANGE-RATE" |
   "SLEW-RATE" |
   "SLS-DEV" |
   "SLS-LEVEL" |
   "SPACING" pulse-spacing-sfx |
   "SPEC-GRAV" |
   "SPEC-TEMP" |
   "SQTR-DIST" [ "-" xxx ] |
   "SQTR-RATE" |
   "STIM" |
   "SUB-CAR-FREQ" |
   "SUB-CAR-MOD" |
   "SUSCEPTANCE" |
   "SWR" |
   "TARGET-RANGE" |
   "TARGET-RANGE-ACCEL" |
   "TARGET-RANGE-RATE" |
   "TAS" |
   [ press-temp-pfx ] "TEMP" |
   "TEMP-COEF-CAP" |
   "TEMP-COEF-CURRENT" |
   "TEMP-COEF-IND" |
   "TEMP-COEF-REACT" |
   "TEMP-COEF-RES" |
   "TEMP-COEF-VOLT" |
   "THREE-PHASE-DELTA" |
   "THREE-PHASE-WYE" |
   "THRUST" |
   "TIME" |
   "TIME-ASYM" |
   "TIME-JIT" |
   "TORQUE" |
   "TRANS-ONE" mod-trans-bits |
   "TRANS-ZERO" mod-trans-bits |
   "TRANS-PERIOD" mod-trans-bits |
   "TRIG" |
   "TRUE" |
   "TYPE" |
   "UNDERSHOOT" [ pulse-sfx ] |
   "VALUE" |
   "VAR-PHASE-FREQ" |
   "VAR-PHASE-MOD" |
   "VIBRATION-ACCEL" |
   "VIBRATION-AMPL" vibration-sfx |
   "VIBRATION-RATE" |
   "VOLTAGE" [ ampl-sfx | phase-sfx | pulse-sfx ] |
   "VOLTAGE" log-noun-mod-sfx |
   "VOLTAGE-RAMPED" |
   "VOLTAGE-STEPPED" |
   "VOLT-LMT" |
   "VOLUME-FLOW" |
   "WAVE-LENGTH" |
   "WIND-SPEED" |
   "WORD-LENGTH" |
   "WORD-RATE" |
   "ZERO-INDEX" } [ "-REF" ] ;


!  18.6.19  NOUN-MODIFIER PREFIXES AND SUFFIXES


ampl-sfx:                                                  ! 17.2.2
   { param-sfx [ "-IN-PHASE" | "-QUAD" ] |
   { "-IN-PHASE" | "-QUAD" } | "INST" } ;

angle-sfx:                                                 ! 17.2.5
   { "THETA" | "PHI" | "-X" | "-Y" | "-Z" | "-R" } ;

dist-sfx:                                                  ! 17.2.4
   { "-X" | "-Y" | "-Z" | "-R" } ;

flux-dens-sfx:                                             ! 17.2.8
   { "-IN-PHASE" | "-QUAD" } ;

log-noun-mod-sfx:                                          ! 17.2.9/10
   { "-ONE" | "-QUIES" | "-ZERO" } ;

param-sfx:                                                 ! 17.2.2
   { "-AV" | "-P" | "-PP" | "-P-NEG" | "-P-POS" |
   "-TRMS" } ;

phase-sfx:                                                 ! 17.2.3
   { "-PHASE-A" | "-PHASE-AB" | "-PHASE-AC" |
   "-PHASE-B" | "-PHASE-BC" | "-PHASE-CB" |
   "-PHASE-C" | "-PHASE-CA" | "-PHASE-BA" } ;

press-temp-pfx:                                            ! 17.2.1
   { "STATIC-" | "TOTAL-" } ;

pulse-sfx:                                                 ! 17.2.6
   "-P" xxx ;

pulse-posn-sfx:                                            ! 17.4
```

```
   { { "-A" | "-B" | "-C" | "-D" }
   { "1" | "2" | "3" | "4" | "5" } |
   "-X" | "-SPI" | "-" unsigned-integer-number } ;

pulse-spacing-sfx:                                         ! 17.2.6
   pulse-sfx "-" posneg pulse-sfx "-" posneg ;

vibration-sfx:                                             ! 17.2.2.3
   { "-P" | "-PP" | "-TRMS" } ;


!  18.6.20  DIMENSIONS


!  Dimensions are listed in the order of Table 15-1
!  of the ATLAS Standard except for time-dimension
!  which is defined separately in this section.


dim:
   { dim-name |
   "DEG" | "MIL" |                                        !  Angle,Plane
   "MRAD" | "RAD" |
   "URAD" | "REV" |

   "SR" | "MSR" |                                          !  Angle, Solid

   "CYCLES" | "PULSES" |                                   !  Burst Length

   "FD" | "UFD" |                                          !  Capacitance
   "NFD" | "PFD" |

   "C" | "KC" |                                            !  Charge
   "UC" | "NC" |

   "S" |                                                   !Conductance,
                                                            Susceptance

   "A" | "KA" |                                            !  Current
   "MA" | "UA" | "NA" |

   "BITS" | "CHAR" | "DIGITS" |                            !  Digital

   "BITS/SEC" | "KBITS/SEC" |                              !  Digital-Rate
   "MBITS/SEC" | "WORDS/SEC" |
   "KWORDS/SEC" | "MWORDS/SEC" |

   "M" | "KM" |                                            !  Distance,
                                                              wavelength
   "MM" | "UM" | "NM" |
   "IN" | "FT" | "S-MILE" |
   "N-MILE" |

"J" | "KJ" | "MJ" |                                      !  Energy,
                                                            Work, Heat
"EV" | "KEV" | "MEV" |

"TIMES" |                                                !  Events

"WB" | "MWB" |                                           !  Flux, Magnetic

"T" | "MT" |                                             !  Flux, Density
"UT" | "GAM" |

"N" | "KN" |                                             !  Force
"MN" | "UN" |

"HZ" | "KHZ" |                                           !  Frequency
"MHZ" | "GHZ" |
"PPS" | "KPPS" |

"LX" |                                                   !  Illuminance

"H" | "MH" |                                             !  Inductance
"UH" | "NH" | "PH" |

"NT" |                                                   !  Luminance

"LM" |                                                   !  Luminous Flux

"CD" |                                                   !  Luminous Intensity

"KG" | "G" |                                             !  Mass
"MG" | "UG" |

"DEG" |                                                  !  Phase Shift

"W" | "KW" | "MW" | "UW" |                               !  Power
"DBM" | "DBW" | "DBK" |

"PA" | "KPA" |                                           !  Pressure
"MPA" | "UPA" |
"MB" | "MMHG" | "INHG" |

"PULSES" |                                               !  Pulse

"DB" | [ "PC" ] |                                        !  Ratio

"OHM" | "KOHM" |                                         !  Resistance
"MOHM" |

"DEGK" | "DEGF" |                                        !  Temperature
"DEGC" |

"N-M" |                                                  !  Torque
```

```

   "FT/SEC" | "M/SEC" |                                     !  Velocity, Linear
   "KT" | "MACH" |

   "RAD/SEC" |                                              !  Velocity, Angular
   "DEG/SEC" | "RPS" | "RPM" |
   "RPH" |

   "V" | "KV" |                                             !  Voltage
   "MV" | "UV" |

   "L" | "ML" |                                             !  Volume

rate-dimension |
product-dimension |
time-dimension } ;

rate-dimension:                                             ! 15.8.1
   dim "/" time-dimension ;

product-dimension:                                          ! 15.8.2
   dim "-" dim ;

time-dimension:                                             ! 15.8.7
   { "HR" | "MIN" | "SEC" |
   "MSEC" | "USEC" | "NSEC" | "PSEC" } ;


!  18.6.21 ABBREVIATIONS


max-min:
   { "MAX" | "MIN" } ;

posneg:
   { "POS" | "NEG" } ;

pos-neg-slope:
   { "POS-SLOPE" | "NEG-SLOPE" } ;

increasing-decreasing:
   { "INCREASING" | "DECREASING" } ;

xxx:
   digit [ digit [ digit ] ] ;

FINIS
```

18.7 Cross-reference listing of syntax variables and symbols

18.7.1 Cross-reference list of rule names

RULENAME	DEFINITION	USAGE
ampl-sfx	2507	2313,2491
analog-signal	250	247
angle-sfx	2511	2290,2291,2292,2324
apply-statement	1083	1075
arith-string-operator	2117	727
arm-statement	1026	980
array-range	1600	1159,1160,1167,1179,1180, 1182, 1183,1184,1185,1539,1540, 1723, 1729,1734,1799,1825
array-structure	126	116
atlas-module-structure	22	13
atlas-program-structure	16	12
base-type	102	98
begin-atlas-module-statement	44	23
begin-atlas-program-statement	34	17
bit-0-1	1460	1444
bit-q-0-1	1457	1443,1444
blank	1886	1897,1906,1929,1944,2074, 2088, 2089
boolean-evaluation	746	735
boolean-operator	2121	728
boolean-variable	2060	331,386
bus-mode-name	2021	495,541,584
bus-parameter	1691	595,1330,1346,1358,1378, 1389, 1676

bus-parameter-data	1699	1697
bus-parameter-info	558	556
bus-parameter-name	2015	486,561,1696
calculate-statement	719	716
capability-fields	1574	1572
change-statement	1041	982
char-string-set	1895	1893,1900
character-string	1892	740
command-field	1721	592,1343
comment-statement	2082	
common-char-set	1845	1896,1903,1906
compare-statement	758	717
comparison-operator	2098	729,1502
complete-atlas-test- program-structure	11	
complex-function-mnemonic	635	628
complex-signal	1934	615,621,688,998,1061,1065,1099, 1136,1416,1495,1550
condition	1521	735
configuration	1985	313,404,943,948
configuration-body	308	305
conn	1526	166,254,658,1003,1086,1096,1102
conn-descriptor	1533	1531
conn-set	1530	333,388,1391,1396,1403,1527, 1528,1650
connect-exchange- configuration-statement	1393	1307

connect-statement	1010	978
connection	1923	105,743,1534
const-declare	84	81
constant	737	86,127,138,144,145,562,588,591, 734,1339,1342,1371,1374,1684, 1700,1724,1730,1735
constant-identifier	1946	86,742
control-fields	1560	1558
data-field	1726	593,1344
data-store	1595	588,591,721,734,747,760,781,785, 790,903,934,1161,1167,1179,1183, 1184,1185,1186,1202,1257,1339, 1342,1371,1374,1497,1534,1601, 1615,1619,1684,1700,1708,1724, 1730,1735,1816,1822,1826,1827, 1830
databus-definition	523	515
databus-fetch-data	1824	1375,1376,1377,1687,1688,1689
databus-transaction	1667	1665
decimal-number	1866	738,1538,1539,1554
declare-statement	78	64,72,76
define-complex-signal- statement	614	607
define-complex-signal- structure	606	154
define-digital-configuration- statement structure	312	304
define-digital-configuration- structure	303	151
define-digital-sensor- statement	372	310

digital-source- characteristics	1638	337,342
digital-source-spec	325	322
digital-statements	1143	974,1266
digital-synchronization	365	352
digital-timer	2054	239,300,1212,1213,1220,1255,1634
digital-timing	2006	600
dim	2558	259,367,368,465,1488,1489,1490, 1491,1492,1538,1539,1544,1554, 1615,1619,1700,2659,2662
dim-lim	1487	1485
dim-name	1994	464,465,2559
dimensioned-number	1553	1561,1562,1567,1568,1575,1576, 1584,1585,1805
disable-complex-signal- statement	1063	986
disable-digital- configuration-statement	946	837
Disable-escape-to- procedure-statement	962	953
disable-event-statement	1054	984
disable-exchange- configuration-statement	1406	1309
disable-file-access-statement	820	771
disconnect-exchange- configuration-statement	1399	1308
disconnect-statement	1018	979
dist-sfx	2514	2324
do-exchange-statement	1311	1303

do-simultaneous-body	1234	1225
do-simultaneous-statement	1228	1224,1287
do-simultaneous-structure	1223	1197
do-simultaneous-substructure	1286	1271
so-timed-control-structures	1273	1270
do-timed-digital-body	1263	1246,1275,1278,1294,1295
do-timed-digital-statement	1249	1245
do-timed-digital-structure	1244	1198,1238,1269
do-timed-if-then-structure	1293	1280
else-statement	865	849,1295
enable-complex-signal- statement	1059	985
enable-digital-configuration- statement	941	836
enable-escape-to- procedure-statement	955	952
enable-event-statement	1049	983
enable-exchange- configuration-statement	1381	1306
enable-file-access-statement	812	770
end-complex-signal-statement	687	612
end-digital- configuration-statement	403	306
end-do-statement	1240	1226,1247,1291
end-for-statement	920	900,1276
end-if-statement	869	851,1296
end-procedure-statement	193	175
end-while-statement	892	877,1279

enumeration-element	1955	104,741
equal-notequal-operator	2101	
escape-structure	951	838
establish-bus-parameter	553	527
establish-protocol-parameter	564	528
establish-protocol-statement	510	62,70
eval-statement-characteristics	1511	164,1122
evaluation-field	1499	760,1468,1512,1643,1646
event	1967	246,260,264,265,266,271,273,288, 289,294,299,370,1007,1051,1056, 1209,1420,1465,1610,1626,1630, 1636,1660, 1663
event-field	1006	1004
event-indicator	1970	294
event-interval	1973	278,283,285,1008,1611,1626
exchange	1982	580,1334,1362,1664,1668
exchange-configuration	2057	692,1313,1328,1335,1356,1363, 1383, 1395,1401,1408
exchange-expression	1652	1316,1318
exchange-frame	1657	1653,1655
exchange-model	539	526
expression	724	257,258,259,721,733,734,780,789, 798,802,807,810,817,854,880,908, 909,910,933,958,1151,1152,1180, 1182,1449,1519,1537,1544,1547, 1593,1728,1825
expression-info	731	726
extend-bus-mode	494	481
extend-bus-parameter	485	479

extend-cnx	440	427
extend-connection	425	411
extend-databus	478	411
extend-exchange-pin	501	483
extend-label	1943	1995,1998,2001,2004,2013,2016, 2019,2022,2028
extend-modifier	419	410
extend-noun	414	410
extend-parameter-definition	505	487,492
extend-protocol-parameter	489	480
extend-statement	407	60,68
extend-test-equip-role	497	482
external-bus-specification	2024	514
external-compensation- specification	2030	1541
external-complex- function-specification	2033	627
external-digital- specification	2048	327,381
external-pulse-class- specification	2051	361
external-signal-conditioning- specification	2036	663
external-signal-name- specification	2039	655
external-specification	2045	417,423,427,465,474
external-sweep-specification	2042	1781

fd	2067	35,40,45,50,79,80,85,89,93,94, 104,105,123,127,146,158,159,165, 166,169,170,178,179,180,181,182, 190,194,198,199,200,202,212,223, 225,237,238,240,241,245,246,251, 252,253,254,258,259,260,263,264, 276,277,292,293,298,299,313,314, 319,321,326,327,328,329,330,331, 332,346,349,350,351,352,353,373, 375,380,381,382,383,384,385,386, 387,401,404,408,409,415,417,420, 421,422,423,426,427,469,486,487, 491,492,495,499,503,511,512,513, 514,515,518,519,526,527,528,533, 537,543,545,546,548,549,550,551, 556,567,580,581,582,586,589,591, 592,593,594,595,596,600,602,603, 615,616,620,621,622,626,629,649, 655,656,657,658,661,666,675,677, 680,683,684,685,688,692,693,694, 720,721,733,759,760,774,780,781, 784,786,789,790,793,798,801,802, 810,813,817,821,854,861,870,871, 880,881,888,893,894,903,904,910, 916,921,922,927,931,933,934,942, 947,956,959,963,964,991,992,993, 998,1000,1003,1004,1008,1012, 1014,1020,1022,1028,1029,1030, 1035,1036,1037,1043,1044,1045, 1050,1051,1055,1056,1060,1064, 1069,1070,1071,1084,1085,1086, 1090,1095,1096,1100,1102,1105, 1106,1107,1108,1113,1114,1115, 1120,1121,1127,1128,1129,1130, 1135,1138,1139,1149,1150,1152, 1153,1154,1155,1159,1164,1165, 1167,1171,1172,1173,1174,1181, 1183,1184,1185,1186,1201,1202, 1206,1208,1211,1213,1217,1218, 1220,1229,1230,1231,1241,1250, 1251,1252,1253,1254,1255,1256, 1258,1259,1260,1312,1314,1315, 1316,1318,1322,1327,1329,1330, 1331,1334,1336,1337,1340,1342, 1343,1344,1345,1346,1347,1350, 1355,1357,1358,1359,1362,1364, 1366,1369,1372,1374,1375,1376, 1377,1378,1379,1382,1383,1385, 1389,1390,1391,1394,1395,1396, 1400,1401,1402,1403,1407,1513, 1557,1558,1562,1571,1572,1580, 1581,1588,1597,1603,1606,1626, 1630,1640,1643,1645,1659,1660, 1661,1665,1668,1669,1670,1674,

		1680,1687,1688,1689,1697,1708, 1754,1756,1768,1771,1772,1774, 1776,1780,1816,1826
fetch-databus-data	1686	1676
fetch-exchange	1361	1351
fetch-exchange- configuration-statement	1349	1305
fetch-protocol	1354	1351
fetch-protocol-parameter	1812	1359,1379,1675
fetch-statement	1034	981
fetch-update-spec	575	555,566
fetch-word-info	1818	1814
fetched-exchange-data	1368	1365
file	1829	780,784,789,798,801,810,817,821, 1728,1825
file-of-structure	134	119
filter-structure	673	666
finish-statement	937	835
flux-dens-sfx	2517	2337
for-then-range	907	904
for-then-statement	902	898,1274
for-then-structure	897	832
frequency-mark-descriptor	1747	1740,1741
fstatno	2073	35,40,45,50,79,158,178,190, 194,198,223,237,245,263,276, 292,298,313,319,373,404,408, 511,580,600,615,620,626,649, 661,688,692,720,759,774,793, 813,821,854,861,866,870,880, 888,893,903,916,921,926,931, 938,942,947,956,963,990,1011, 1019,1027,1035,1042,1050, 1055,1060,1064,1068,1084,1090,

		1105,1113,1120,1127,1135,1149, 1164,1171,1201,1206,1217,1229, 1241,1250,1312,1322,1350,1382, 1394, 1400,1407
Function-characteristic- mnemonic	640	629
gate-conn-or-event-field	1002	994,1015,1023,1031,1038,1046, 1072,1109,1116,1123,1131,1140
gate-field	1608	265,657,1003,1086,1785
go-to-statement	925	833,1281
identify-event-and-or	282	279
Identify-event-based-event- statement	262	231
identify-event-from-to	287	279
Identify-event-indicator- statement	291	233
Identify-event-interval- statement	275	232
Identify-signal-based-event- statement	244	230
identify-statements	228	65,73
Identify-time-based-event- statement	297	234
identify-timer-statement	236	229
if-then-else-structure	846	830
if-then-statement	853	847,1294
include-statement	222	59
increasing-decreasing	2681	253
index	1546	1597,1602,1603
initial	140	95

initializer	143	141,145,146
initiate-statement	1134	1081
input-statement	773	768
known-type-input	779	776
known-type-output	797	794
label	1928	1932,1935,1938,1941,1947,1950, 1953,1956,1959,1962,1965,1968, 1971,1974,1977,1980,1983,1986, 1989,1992,2007,2010,2025,2031, 2034,2037,2040,2043,2046,2049, 2052,2055,2058
label-char-set	1902	1929,1944
leave-for-statement	915	844,1283
leave-if-statement	860	842,1282
leave-procedure-statement	189	841
leave-statements	840	829
leave-while-statement	887	843,1284
letter	1908	1847,1924,2164
limit-field	1583	562,1581
local-preamble-structure	75	186
log-noun-mod-sfx	2520	2314,2341,2492
logic-alternation	355	338,393
lower-case-letter	1859	1846,1924
m-char-noun	168	163
main-procedural-statements	703	701
main-procedural-structure	700	19,187,858,885,913
mark-descriptor-subfield	1737	1422
max-min	2672	1434,1438,1538,1576,1585

max-time	1515	370,1008,1209,1260,1318,1366, 1420, 1465,1627,1630,1636
measure-statement	1104	1077
measured-characteristic	1494	169,1036,1106,1114,1121,1128
measured-characteristic- mnemonic	2277	212,220,251,382,999,1100,1137, 1496
message-text	1899	871,894,922,2083
mod-atc-mode	2145	2129,2367
mod-atmos	2148	2130
mod-channel	2152	2131,2308
mod-desc-data	472	469
mod-desc-info	467	460,508
mod-dim	462	459,460,507
mod-fluid-type	2155	2132
mod-fuel-supply	2160	2133
mod-ident-sig	2163	2134
mod-iff-mode	2166	2135,2367
mod-logic-data-type	2187	2140
mod-mod-mode	2191	641,2141
mod-pulse-class	2169	2136
mod-pulses-incl-excl	2173	2137
mod-ref-pulses	2176	2138
mod-tacan-mode	2180	2139,2368
mod-trans-bits	2183	2142,2478,2479,2480
mod-type	455	430
mod-vector-descriptor	2194	646,2143

modifier-definition	429	421
modifier-descriptor	2128	476,630,666,1419,1448,1564,1577
modifier-descriptor-name	2027	473,475,1701
modifier-mnemonic	2280	431,1417,1468,1558,1572,1581, 1639, 1644,1761,1764,2278
modifier-name	2003	430,431,2281
module-name	1937	46,51,225
module-preamble-structure	67	24
monitor-statement	1112	1078
multiple-action-statements	1074	973
name-char-set	1905	
non-atlas-module-body	0	28
non-atlas-module-structure	27	14
non-printing-character	1836	1896
noun	2220	170,203,420,426,621,1551
noun-field	1549	252,328,383,656,992,1013,1021, 1029,1037,1044,1070,1084,1096, 1101,1107, 1114,1121,1129,1139
noun-name	1997	170,203,415,420,426,621,1551, 2221
numerator-modifier-mnemonic	1760	1753,1755
occurrences-of	269	265
on-field	1648	260,1154,1155,1168,1174,1187
output-statement	792	769
param-sfx	2523	2508
parameter	1605	181,182
parentheses	1889	1897,1906

perform-statement	930	834,1265
phase-sfx	2527	2313,2491
pin-descriptor	2204	443,520,1531,1589,1590
pin-descriptor-name	2000	441,442,503,2205
pos-neg-slope	2678	1469,1477
posneg	2675	2544
power-mark-descriptor	1743	1738,1739
pre-declared-enumeration	753	744
pre-declared-type	112	99
pre-defined-function	2107	732
press-temp-pfx	2532	2394,2395,2396,2464
procedural-segment-for-then	912	899
procedural-segment-if- then-else	857	848,850
procedural-segment-while-then	884	876
procedural-statements-control	828	706,
procedural-statements- data-processing	715	704,1264
procedural-statements-databus	1302	709
procedural-statements- input-output	767	705
procedural-statements-signal	971	707,1235
procedural-statements-timing	1193	708
procedure	1961	178,190,194,932,957
procedure-body	185	174
product-dimension	2661	2655
program-name	1940	36,41

program-preamble-structure	58	18
proportionality-subfield	1752	632
protocol	1979	513,518,582,694,1327,1355,1384, 1396,1402
protocol-initialization	1388	1385
protocol-parameter-info	569	567
protocol-parameter-name	2018	491,570,1708,1815
prove-hiz-or-value	1177	1175
prove-statement	1170	1146,1289
pulse-class	359	351
pulse-posn-sfx	2538	2402
pulse-sfx	2535	2313,2330,2334,2376,2393,2405, 2433,2434,2436,2484,2491,2544
pulse-spacing-sfx	2543	2450
range-to	1566	1538,1561,1575,1584
rate-dimension	2658	2654
read-statement	1126	1080
read-timer-statement	1200	1194,1236
real-capability-characteristic	1433	1427
real-characteristic-subfield	1425	631,665,674,675,676,677,679,680, 683,684,685,1418,1455,1640,1645, 1783
real-control-characteristic	1430	1426
real-errlim	1484	253,1000,1138,1431,1477,1496, 1563,1576,1635
real-limit-characteristic	1437	1428
real-or-digital-quantity	1508	1500,1502,1505,1506
real-quantity	1543	252,366,747,751,1509

record-field-identifier	1958	123,1598
record-structure	121	118
reference-phase	2199	1528
remove-characteristics	1094	1091
remove-component	1098	1095
remove-statement	1089	1076
require-analog	210	201
require-capability	1570	205
require-cnx	1587	207
require-control	1556	204
require-event	218	202
require-limit	1579	206
require-statement	197	61,69
require-timer	215	201
requirement	1931	200,240,1551
reset-statement	1067	987
reset-timer-statement	1216	1196,1237,1268
role-field	1716	1315,1668
rolloff-info	682	679
sense-delay	1621	1253
sense-event	1629	602,603,1253,1254
sense-rate	1617	1254
sense-statement	1163	1145,1288
sensor-logic-levels	390	385
set-protocol-parameter	1703	596,1331,1347,1390

set-word-info	1710	1706
setup-statement	989	977
sign	1869	1707,1867,1874
signal	1964	159
signal-conditioning-mnemonic	669	664
signal-definition	162	160
signal-oriented-verb	448	446
signal-value	1536	1431,1434,1435,1438,1439,1756, 1758,1789,1804
simulate-device	1682	1679,1680
single-action-body	997	991,1012,1020,1028,1043,1069
single-action-statements	976	972
smooth-step-info	1770	1768
smooth-sweep-descriptor	1791	1773
smooth-sweep-parameter	1787	1772
source-logic-levels	335	330
specify-complex- function-statement	625	609
specify-complex-signal- characteristics-statement	619	608
specify-component- signal-statement	648	610
specify-signal- conditioning-statement	660	611
statement-characteristics	1415	165,241,254,329,384,622,656,993, 1014,1022,1030,1037,1045,1071, 1085,1102,1108,1115,1130,1139, 1513
statement-number	2091	862,889,917,927,1208

statement-terminator	2070	37,42,47,52,82,160,183,191,195, 208,226,242,248,267,280,295,301, 316,323,377,405,412,521,597,604, 617,623,633,658,667,689,694,722, 761,777,795,818,822,855,863,867, 872,882,890,895,905,918,923,928, 935,939,944,949,960,965,995, 1016,1024,1032,1039,1047,1052, 1057,1061,1065,1072,1087,1092, 1110,1117,1124,1132,1141,1156, 1168,1175,1203,1214,1221,1232, 1242,1261,1319,1324,1352,1386, 1397,1404,1409,2084
statno	2086	2074
status-field	1732	594,1345
step-number	2079	2087,2088,2092
step-sweep-descriptor	1807	1777
step-sweep-parameter-1	1797	1775
step-sweep-parameter-2	1801	1776
stim-event	1624	602,1252
stim-rate	1613	1252
stimulate-multiple-values	1158	1151,1153
stimulate-statement	1148	1144,1290
string-structure	130	117
structured-type	115	100
subrange	137	98,132
sweep-configuration-subfield	1766	1423
sweep-info	1779	1768
symbol-extension-set	1853	1846
sync-body	1467	1464
sync-initial	1475	1470

sync-subfield	1463	1421,1784
terminate-atlas-module- statement	49	25
terminate-atlas-program- statement	39	20
test-equip-bus-monitor	535	525
test-equip-bus-role	530	524
test-equip-role-name	2012	499,533,1718
test-equip-sim	1678	1673
test-equip-transaction-role	1672	1669
test-number	2076	2087,2092
text-format	804	802
text-input	783	776
text-output	800	794
time-dimension	2664	1519,2656,2659
time-quantity	1518	266,289,301,353,519,1165,1172, 1207,1210,1212,1231,1516,1622, 1635,1660,1662,1670
timer	1976	239,300,1203,1210,1211,1218, 1634,1662
type	97	90,95,124,128,135,548,549,550, 551,570,1606
type-declare	88	81
type-identifier	1949	90,98
unary-operator	2104	725
unsigned-decimal-number	1872	1488,1489,1490,1491,1492,1615, 1619,1867
unsigned-integer-number	1876	131,270,272,350,356,362,367,368, 958,959,964,1160,1257,1659,1707, 1712,1713,1745,1750,2153,2157, 2158,2171,2174,2178,2541

untyped-input	788	776
untyped-output	809	794
update-exchange	1333	1323
update-exchange- configuration-statement	1321	1304
update-protocol	1326	1323
uplowlimit-real	750	748
uplowlimit-real-dig	1504	1501
usage-info	433	422
var-declare	92	81
variable-identifier	1952	94,1258,1596,1606,2061
verb-info	445	416
verify-statement	1119	1079
vibration-sfx	2546	2489
wait-for-statement	1205	1195,1267
when-field	1632	1150,1164,1171,1230,1251,1314, 1329,1336,1357,1364,1658
while-then-statement	879	875,1277
while-then-structure	874	831
xxx	2684	2348,2453,2536

18.7.2 Cross-reference list of literal symbols

RULENAME	USAGE
`""`	1887
`"!"`	1856
`"""`	1854

"AGE-RATE"	2284
"AIR"	2156
"ALL"	965,1056,1091,1218,1220, 2174
"ALL-LISTENER"	542,585
"ALOG"	2114
"ALT"	2285
"ALT-RATE"	2286
"ALTERNATE-BUS-TRANSMIT"	518
"AM"	636
"AM SIGNAL"	2224
"AM-COMP"	2287
"AM-SENSITIVITY"	642
"AM-SHIFT"	2288
"AMBIENT CONDITIONS"	2225
"AMI"	363,2171
"AMPL-MOD"	2289
"AND"	284,1653,2122
"ANGLE"	2290
"ANGLE-ACCEL"	2291
"ANGLE-RATE"	2292

"B2C"	755
"BAND-PASS-FILTER"	678
"BANDWIDTH"	2296
"BAROMETRIC-PRESS"	2297
"BCD"	755
"BEFORE STEP"	1208
"BEGIN"	35,45
"BEL"	1838
"BFSK"	2197
"BINARY"	786,806
"BIP"	363,2170
"BIT"	110,132,1713
"BIT-PERIOD"	366
"BIT-RATE"	366,2298
"BIT-TRANSITION"	1471
"BITS"	350,2110,2579
"BITS/SEC"	2581
"BNR"	755
"BOOLEAN"	113
"BPSK"	2195

"BS"	1838
"BSM"	755
"BURST"	2299
"BURST-DROOP"	2300
"BURST-REP-RATE"	2301
"BUS"	517
"BUS PROTOCOL"	513
"BUS-MODE"	495,540,583
"BUS-PARAMETER"	486,554,1692
"BY"	909,1561,1575,1584,1602, 1611,1805
"C"	1884,1893,1909,2083,2146, 2167, 2208,2571
"CALCULATE"	720
"CAN"	1841
"CAP"	2302
"CAPABILITY"	1571
"CAR-AMPL"	2303
"CAR-FREQ"	2304
"CAR-HARMONICS"	2305
"CAR-PHASE"	2306

"CAR-RESID"	2307
"CARRIER"	654
"CD"	2616
"CHANGE"	450,1042
"CHANNEL"	2308
"CHAR"	109,132,2110,2579
"CHAR-CLASS"	113
"CNX"	333,388,441,520,1391,1396, 1403, 1527,1588,1650
"COMMAND"	537,549,1375,1687,1722
"COMMAND-WORD"	571,1711,1819
"COMMON"	1926,2227
"COMPARE"	759
"COMPL"	2210,2309
"COMPLEX FUNCTION"	626
"COMPLEX SIGNAL"	617,620,1060,1064,1550, 2228
"COMPONENT"	654,998,1099,1136,1416, 1495
"CON-MODE"	542,584
"CON-RT"	541,583
"CONDUCTANCE"	2310

"EOF"	2108
"EOT"	1837
"EQ"	252,301,1662,2099,2102
"ERRLMT"	1485
"ERROR"	1184
"ERROR-INDEX"	1185
"ESC"	1842
"ESCAPE NUMBER"	958,964
"ESCAPE TO PROCEDURE"	956,963
"ESTABLISH"	511
"ETB"	1841
"ETX"	1837
"EV"	2591
"EVENT"	246,264,299,1051,1056, 1209,1660,1663
"EVENT INDICATOR"	293
"EVENT INTERVAL"	277
"EVENT MONITOR"	219
"EVENTS"	2235
"EVERY"	270,1660

"FILE OF"	135
"FINISH"	938
"FLUID SIGNAL"	2236
"FLUID-TYPE"	2336
"FLUX-DENS"	2337
"FM"	636
"FM SIGNAL"	2237
"FM-COMP"	2338
"FM-SENSITIVITY"	642
"FOR"	289,903,916,921
"FOR VERBS"	415
"FORCE"	2339
"FORCE-RATE"	2340
"FORWARD"	1792
"FORWARD-HOLD"	1792
"FORWARD-HOLD-RESET-HOLD"	1793
"FORWARD-HOLD-REVERSE-HOLD"	1794
"FORWARD-REVERSE"	1795
"FREQ"	1473,1788,1798,1802,2341
"FREQ-DEV"	642,2342

"FREQ-PAIRING"	2343
"FREQ-WINDOW"	2344
"FREQUENCY-MARK"	1748
"FREQUENCY-MARK-HARM"	1750
"FREQUENCY-MARK-LOWER"	1749
"FREQUENCY-MARK-UPPER"	1748
"FROM"	288,780,784,789,1007,1610, 1728, 2210
"FS"	1842
"FSK"	1808
"FT"	2587
"FT/SEC"	2642
"FUEL"	2156
"FUEL-SUPPLY"	2345
"G"	1910,2618
"GA"	2181
"GAIN"	670
"GAM"	2598
"GAS"	2156
"GATED"	1609

"HIZ"	1152,1154,1166,1173,1178
"HOLD"	1318
"HR"	2665
"HT"	1838
"HUMIDITY"	2351
"HZ"	2603
"I"	1910,2181
"I-O"	814
"IAS"	2352
"ICAN"	2149
"ICAO"	2150
"IDENT-SIG"	2353
"IDENTIFY"	237,245,263,276,292,298, 450
"IF"	854,861,870,957
"IFF"	2239
"ILLEGAL-STATE-INDICATOR"	331,386
"ILLUM"	2354
"ILS"	2240
"IMPEDANCE"	2241
"IN"	2587

"INCLUDE"	223
"INCREASING"	2682
"IND"	2355
"INHG"	2628
"INITIAL"	1470
"INITIAL ="	141
"INITIATE"	449,1135
"INPUT"	774
"INPUT FROM"	816
"INSERT"	2115
"INSERT-INVALID"	1705
"INST"	2509
"INT"	2111,2356
"INTEGER"	108,458,507
"INTO"	781,784,790,1167,1179, 1202,1497
"IS"	86,90,95,124,1606
"ISO7"	756
"ISOLATION"	671
"ITERATE"	1256
"J"	1910,2590

"ORD"	2108
"OSCILLATOR"	653
"OUTPUT"	793
"OUTPUT TO"	814
"OVERSHOOT"	2376
"P"	1911
"P-AMPL"	2377
"P3-DEV"	2378
"P3-LEVEL"	2379
"PA"	2626
"PAIR-DROOP"	2380
"PAIR-SPACING"	2381
"PAM"	637,2248
"PARALLEL"	1446,2188
"PARITY"	572,1714,1820
"PC"	1488,1489,1490,1491,1492, 2632
"PE"	2108
"PEAK-DEGEN"	2382
"PERFORM"	931
"PERIOD"	2383

"REF-PULSES-EXCL"	2421
"REF-PULSES-INCL"	2422
"REF-UUT"	2424
"REF-VOLT"	2425
"REFERENCE-LEVEL"	653
"REFERENCE-SIGNAL"	650
"REL-BEARING"	2426
"REL-BEARING-RATE"	2427
"RELATIVE-HUMIDITY"	2428
"RELATIVE-WIND"	2429
"REMOVE"	451,1090
"REPEAT"	1159
"REPLY-EFF"	2430
"REQUIRE"	198,452
"RES"	2431
"RESET"	453,1068,1217
"RESOLVER"	2258
"RESP"	2432
"RESPONSE"	438
"RESPONSE-MEASUREMENT"	435

"RESPONSE-TIME"	560,1694
"RESULT"	182,934
"REV"	2562
"REVERSE"	1795
"RINGING"	2433
"RISE-TIME"	2434
"RO"	2181
"ROLLOFF"	685
"ROLLOFF-LOWER"	677,684
"ROLLOFF-UPPER"	675,683
"ROTATE"	2110
"ROTATION"	2259
"ROTOR SPEED"	2435
"ROUND"	2112
"ROUNDING"	2436
"RPH"	2647
"RPM"	2646
"RPS"	2646
"RS"	1843
"RT-CON"	541,583

"STROBE-TO-EVENT"	1420
"STX"	1837
"SUB"	1842
"SUB-CAR-FREQ"	2456
"SUB-CAR-MOD"	2457
"SUBRANGE"	138
"SUCC"	2109
"SUM"	637
"SUMMATION"	643
"SUP CAR SIGNAL"	2263
"SUPP-CAR"	637
"SUPPRESS-LEVEL"	643
"SUSCEPTANCE"	2458
"SWEEP"	638
"SWEEP-ACTIVE"	1785
"SWEEP-CONFIGURATION"	1767
"SWEEP-CONTROL"	652
"SWEEP-TIME"	1783
"SWEEP-TRIG"	1784
"SWEEP-TYPE"	1780

"SWR"	2459
"SYN"	1841
"SYNC"	572,1464,1714,1820
"SYNC-BIT-TRANSITION TO EVENT"	369
"SYNCHRO"	2264
"T"	1912,2597
"T-R"	572,1714,1820
"TACAN"	2265
"TALKER"	543,586,1337,1369,1679
"TALKER-LISTENER"	542,584
"TAN"	2113
"TARGET-RANGE"	2460
"TARGET-RANGE-ACCEL"	2461
"TARGET-RANGE-RATE"	2462
"TAS"	2463
"TEMP"	2464
"TEMP-COEF-CAP"	2465
"TEMP-COEF-CURRENT"	2466
"TEMP-COEF-IND"	2467
"TEMP-COEF-REACT"	2468

"TYPE"	89,1446,2483
"U"	1912
"UA"	2577
"UC"	2572
"UFD"	2568
"UG"	2619
"UH"	2610
"UL"	751,1505
"UM"	2586
"UN"	2601
"UNDERSHOOT"	2484
"UNTIL"	1661
"UNTYPED"	135
"UPA"	2627
"UPDATABLE"	576
"UPDATABLE-FETCHABLE"	576
"UPDATE"	1322
"URAD"	2562
"US"	1843
"USAGE"	422

"n"	1862
"o"	1862
"p"	1862
"q"	1862
"r"	1862
"s"	1863
"t"	1863
"u"	1863
"v"	1863
"w"	1863
"x"	1863
"y"	1864
"z"	1864
"{"	1857
"|"	1854
"}"	1857
"~"	1857

19.0 Bibliography

The following documents provide reference material cited in, and applicable to, this standard:

[B1] ARDC Atmosphere, Survey 115, Cambridge Research Centre.

[B2] ICAN Atmosphere, International Committee on Air Navigation—1924.

[B3] ICAO Anex 10, sections 3.1 and 3.5, DME and ILS Frequency Pairing.

[B4] ICAO Atmosphere, Standard Atmosphere Document 7488.2-1964.

[B5] Telemetry Std 106-71 (revised Jan. 1971).

[B6] WADC Atmosphere, WADC Technical Report 54-215

[B.7] IEEE Std 416-1984 (W1993), IEEE Standard ATLAS Test Language

Standards Survey

The IEC would like to offer you the best quality standards possible. To make sure that we continue to meet your needs, your feedback is essential. Would you please take a minute to answer the questions overleaf and fax them to us at +41 22 919 03 00 or mail them to the address below. Thank you!

Customer Service Centre (CSC)

International Electrotechnical Commission
3, rue de Varembé
1211 Genève 20
Switzerland

or

Fax to: **IEC**/CSC at +41 22 919 03 00

Thank you for your contribution to the standards-making process.

Nicht frankieren
Ne pas affranchir

Non affrancare
No stamp required

A Prioritaire

RÉPONSE PAYÉE

SUISSE

Customer Service Centre (CSC)
International Electrotechnical Commission
3, rue de Varembé
1211 GENEVA 20
Switzerland

Q1 Please report on **ONE STANDARD** and **ONE STANDARD ONLY**. Enter the exact number of the standard: *(e.g. 60601-1-1)*

....

Q2 Please tell us in what capacity(ies) you bought the standard *(tick all that apply).* I am the/a:

purchasing agent	❏
librarian	❏
researcher	❏
design engineer	❏
safety engineer	❏
testing engineer	❏
marketing specialist	❏
other...	

Q3 I work for/in/as a: *(tick all that apply)*

manufacturing	❏
consultant	❏
government	❏
test/certification facility	❏
public utility	❏
education	❏
military	❏
other...	

Q4 This standard will be used for: *(tick all that apply)*

general reference	❏
product research	❏
product design/development	❏
specifications	❏
tenders	❏
quality assessment	❏
certification	❏
technical documentation	❏
thesis	❏
manufacturing	❏
other...	

Q5 This standard meets my needs: *(tick one)*

not at all	❏
nearly	❏
fairly well	❏
exactly	❏

Q6 If you ticked NOT AT ALL in Question 5 the reason is: *(tick all that apply)*

standard is out of date	❏
standard is incomplete	❏
standard is too academic	❏
standard is too superficial	❏
title is misleading	❏
I made the wrong choice	❏
other	

Q7 Please assess the standard in the following categories, using the numbers:
(1) unacceptable,
(2) below average,
(3) average,
(4) above average,
(5) exceptional,
(6) not applicable

timeliness...
quality of writing...................................
technical contents................................
logic of arrangement of contents ....
tables, charts, graphs, figures...............
other ....

Q8 I read/use the: *(tick one)*

French text only	❏
English text only	❏
both English and French texts	❏

Q9 Please share any comment on any aspect of the IEC that you would like us to know:

....
....
....
....
....
....
....
....
....
....
....
....

ISBN 2-8318-4940-3

9 782831 849409

ICS 35.240.50

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND