

IEEE Standard

C/ATLAS

Test Language



Published by
The Institute of Electrical and
Electronics Engineers, Inc.

SH12518

IEEE Standard C/ATLAS Test Language



Published by
The Institute of Electrical and Electronics Engineers, Inc

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE which have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old, and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
345 East 47th Street
New York, NY 10017
USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

IEEE Std 716-1989
(Revision of ANSI/IEEE
Std 716-1985)

C/ATLAS* Test Language

Sponsor
IEEE Standards Coordinating Committee 20,
ATLAS
Abbreviated Test Language for All Systems

* Common ATLAS, a Subset of the ATLAS Test Language

Second Printing
June 1991

ISBN 1-55937-006-8
Library of Congress Catalog Number 89-080474

© Copyright 1989 by

The Institute of Electrical and Electronics Engineers, Inc
345 East 47th Street, New York, NY 10017 USA

*No part of this publication may be reproduced in any form,
in an electronic retrieval system or otherwise,
without the prior written permission of the publisher.*

April 28, 1989

SH12518

Foreword

(This foreword is not part of IEEE Standard 716-1989, IEEE Standard C/ATLAS Test Language).

IEEE Standard 716 (C/ATLAS) is a companion language to IEEE Standard 416 (ATLAS). This revision of C/ATLAS was balloted by the ATLAS voting constituency in August 1987 and July 1988 and approved by the IEEE Standards Board as IEEE STD 716 in 1989.

C/ATLAS has been conceived as a widely acceptable ATLAS subset with a single and unique construct to express each language facility selected to support structured programming. Major emphasis was placed on long-term stability between C/ATLAS updates.

This publication represents the culmination of numerous reviews, analyses and assessments by the Major User representatives and representatives from various corporations through participation in the IEEE SCC 20 ATLAS Committee and the 716 C/ATLAS Subcommittee. The Major Users consist of representatives of the United States Department of Defense (US DOD), the Federal Republic of Germany Ministry of Defense (FRG MOD), the French Ministry of Defense (FR MOD) and BNI, the United Kingdom Ministry of Defense (UK MOD), the Swedish Ministry of Defense (SW MOD), and the airlines represented by Aeronautical Radio, Inc (ARINC).

Special thanks go to (a) the Major Users for sponsoring the development of the proposed changes, reviewing and integrating the proposals; (b) the United States Department of Defense for providing the graphics, integrating the proposals, providing quality assurance, editing and typesetting of the final book; (c) the Federal Republic of Germany Ministry of Defense for providing the formal syntax; (d) and the United States Department of Defense for providing the implementation and field testing of the IEEE Standard 716 prior to publication.

When this revised standard was developed during the period 1984 to 1989, the 716 Subcommittee consisted of the following members:

716 SUBCOMMITTEE EXECUTIVE BOARD

ARINC	Mr. Lawrence Carpenter (84-88)
FR MOD	Ms. Catherine Ozenfant (84-)
FRG MOD	Mr. Eckhart Schwab (84-88), Mr. Hans-Dieter Hessling (88-)
SW MOD	Mr. Alf Gustafsson (88-)
UK MOD	Mr. Colin White (84,85), Mr. Michael Browning(86), Mr. Michael Cosslett (87-)
US DOD	Mr. Joseph Batz (84-), Mr. Paul Gross (Alternate 84-)

716 SUBCOMMITTEE CHAIRMAN/VICE CHAIRMAN/SECRETARY

<u>YEAR</u>	<u>CHAIRMAN</u>	<u>VICE CHAIRMAN</u>	<u>SECRETARY</u>
1984	Helmuth Kaunzinger (US DOD)	Colin White (UK MOD)	Kenneth Wilkinson (GAI)
1985	Colin White (UK MOD)	John Lewis (US DOD)	Kenneth Wilkinson (GAI)

1986	John Lewis (US DOD)	Eckhard Obel (FRG MOD)	Kenneth Wilkinson (ATEAM Corp)
1987	Eckhard Obel (FRG MOD)	Charles Lillie (USDOD)	Kenneth Wilkinson (ATEAM Corp)
1988	Charles Lillie (US DOD)	Catherine Ozenfant (FR MOD)	Kenneth Wilkinson (ATEAM Corp)
1989	Catherine Ozenfant (FR MOD)	Helmuth Kaunzinger (US DOD)	Kenneth Wilkinson (ATEAM Corp)

QUALITY ASSURANCE WORKING GROUP CHAIRMEN / VICE CHAIRMEN

Charles Walton (84)
John Lewis / Ernst-Wolfgang Dieterich (85)
James Cottrell (86-)

TECHNICAL EDITOR

Gregory deMare, GAI

716 SUBCOMMITTEE AND MAJOR USERS TECHNICAL WORKING GROUP MEMBERS (1984-1989)

Donald H. Abernathie, GDE	Fil Feit, SofTech
Richard Bailey, MEMBRAIN	Ken Fox, Sperry
Alan Baldwin, UK MOD	Ulrich Fritzler, GPP
Lester I. Birman, Grumman	George Gohagan, US DOD
Tim Bearse, US DOD	Michael J. Gooding, Emerson
Patrice Brard, SFENA	Paul Gross, US DOD
Margaret Brody, US DOD	Gene P. Hall, Boeing
Ronald G. Brooks, SofTech	Arthur Hann, Consultant
Joseph Buckland, US DOD	Raymond L. Hasenstab, Northrop
Antonius Bunsen, GPP	Raymond Heather, Admiral ITECH
James Burns, Sperry	John Heckel, Laser Graphics
William E. Byrnes, Emerson	Werner Henn, FRG MOD
Mick Carter, Minster Technology Consultancy	Kenneth E. Hermance, Bendix
Robert Cays, US DOD	Lennart Hoglund, FFV Test Systems AB
May Cheong, US DOD	Gordon L. Hyatt, Harris
Howard Churchill, US DOD	Guy Ingram, US DOD
James E. Cottrell, US DOD	Ulla Jacobsson, FFV Test Systems AB
Gregory deMare, GAI	Timothy V. Johnson, Honeywell
Ernst-Wolfgang Dieterich, GPP	Gunther Kainz, US DOD
Bernd Dinklage, GPP	Helmuth M. Kaunzinger, US DOD
Keith Edwards, GAI	Ulf Kempe, LM Ericsson
Bernd F. Eichenauer, GPP	Joseph Kerjean, FR MOD
Marcie E. Emas, AMG	Roger Koble, US DOD
Wolf-Dieter Erdmann, KAE	George Konomos, US DOD
Horst Ey, FRG MOD	John Kroplin, US DOD
James Fay, RCA	James L. Kuehn, Lockheed

Donald Kump, Grumman
Steven Kushnir, GPP
Mike Kuzemchak, Harris
Sonja Kval, FFV Test Systems AB
M. Lacroix, Aerospatiale
Lars-Olov Larsson, FFV Test Systems AB
Dominique Laurencin
Gerald Lejeune, Belgian AF
Thomas Leuthold, GPP
John Lewis, US DOD
Charles W. Lillie, US DOD
Robert A. Lindley, RCA
Ted Lunsford, GAI
Charles Macrina, Sperry
Michael S. Martin, US DOD
Harry A. Mason, US DOD
Glenn W. Meyers, US DOD
Pat McCown, Bendix
Susan Micheel, ATEAM
Robert Moore, Laser Graphics
Thomas Murphy, US DOD
Goran Nordstrom, FFV Test Systems AB
Eckhard Obel, FRG MOD
Hans Obermeier, GPP
Roy Oishi, ManTech
Catherine Ozenfant, FR MOD
Joel R. Owens, US DOD
James Passwater
Dick Patrick, GEC Avionics LTD

Frederick Petry, US DOD
Duane Polson, Boeing
Claude Pommere, SFENA
Frank A. Pullo, Grumman
Brian W. Raffaelli, British Aerospace plc
Johannes Reh, GPP
Frederick Richter, TDC
Albert Sanson, MATRA
Paul Saucier, RCA
Bruno Schlecht
Helmuth Scheibenzuber, MBB
Mike Seavy, Honeywell
John Shumate US DOD
John W. Slaterry, GDE
Philip Smith, US DOD
Larry Sollman, US DOD
Harvey A. Soloway, Northrop
Joseph Stanco, SSAI
John F. Tehan, Emerson
Robert V. Tieche, US DOD
Paul Thomas, US DOD
Scott Tucker, US DOD
L. W. Wagner, GD
Charles Walton, US DOD
Dana R. Wheeler, US DOD
Diane Wilkinson, ATEAM
Kenneth D. Wilkinson, ATEAM
James B. Wyburn, US DOD
Will W. Young, Bendix

The following persons were on the balloting committee that approved this document for submission to the IEEE Standards Board:

D. H. Abernathie	M. G. Harrison	J. A. Passwater
J. C. Batz	D. C. Hart	R. J. Patrick
P. Brard	R. L. Hasenstab	F. D. Petry
R. G. Brooks	S. J. Hay	K. E. Potts
J. Buckland	R. L. Heather	J. Pouilly
A. Bunsen	R. T. Heatherington	J. R. Reeder
W. E. Byrnes	K. E. Hermance	J. Reh
L. D. Carpenter	G. L. Hyatt	F. H. Richter
M. T. Carter	U. Jacobsson	A. Sanson
W. J. Cartmell	H. M. Kaunzinger	H. Scheibenzuber
R. E. Cays	K. M. Kugler	W. M. Scheinin
B. J. Christiansen	M. Kuzemchak	B. Schlecht
M. F. Cosslett	S. V. Kval	S. M. Schlosser
J. E. Cottrell	M. LaCroix	M. L. Seavey
L. J. Cross	L.-O. Larsson	J. B. Sheffler
G. M. Demare	G. Lejeune	J. L. Shumate
B. Dinklage	J. E. Lewis	P. T. Smith
B. F. Eichenauer	C. W. Lillie	A. Soik
E. M. Emas	A. Lowenstein	L. Sollman
W.-D. Erdmann	M. S. Martin	H. A. Soloway
H. D. Ey	H. Mason	J. J. Stanco
J. E. Fay	J. Meeusen	H. Thomale
K. J. Fox	V. Monie	P. Voletsky
U. Fritzler	D. S. Morris	L. W. Wagner
G. R. Gohagan	J. W. Mowery	C. J. Walton
M. J. Gooding	T. W. Murphy	J. R. Weger
T. W. Griffin	G. B. Nordstrom	L. Wheelwright
P. J. Gross	H. Obermeier	K. D. Wilkinson
A. Hann	R. T. Oishi	W. W. Young
	C. Ozenfant	

When the IEEE Standards Board approved this standard on March 8, 1989, it had the following membership:

Dennis Bodson, *Chairman*

Marco W. Migliaro, *Vice Chairman*

Andrew G. Salem, *Secretary*

Arthur A. Blaisdell	Kenneth D. Hendrix	John E. May, Jr.
Fletcher J. Buckley	Theodore W. Hissey, Jr.	Lawrence V. McCall
Allen L. Clapp	John W. Horch	L. Bruce McClung
James M. Daly	David W. Hutchins	Donald T. Michael*
Stephen R. Dillon	Frank D. Kirschner	Richard E. Mosher
Donald C. Fleckenstein	Frank C. Kitzantides	Stig Nilsson
Eugene P. Fogarty	Joseph L. Koepfinger*	L. John Rankine
Jay Forster*	Edward Lohse	Gary S. Robinson
Thomas L. Hannan		Donald W. Zipse

*Member Emeritus

TABLE OF CONTENTS

TITLE	PAGE
1.0 Purpose	1-1
1.1 Language Processors	1-1
1.2 Document Control	1-1
2.0 Reference Material	2-1
2.1 Applicable Documents	2-1
2.2 Document Precedence	2-1
2.3 Document Organization and Conventions	2-1
3.0 Complete C/ATLAS Test Program	3-1
3.1 <complete atlas test program structure>	3-1
3.2 Basic Statement Elements	3-3
4.0 Structure Delimiter Statements	4-1
4.1 BEGIN/TERMINATE Statements	4-1
4.1.1 BEGIN, ATLAS PROGRAM Statement	4-1
4.1.2 TERMINATE, ATLAS PROGRAM Statement	4-1
4.1.3 BEGIN, ATLAS MODULE Statement	4-2
4.1.4 TERMINATE, ATLAS MODULE Statement	4-3
5.0 Reserved For Future Use	5-1
6.0 Preamble Statements	6-1
6.1 Main Preamble Structure	6-1
6.2 <local preamble structure>	6-4
6.3 DECLARE Statement	6-4
6.4 DEFINE Statements Definition	6-18
6.5 Deleted	6-19
6.6 DEFINE <signal> Statement	6-20
6.7 PROCEDURE Definition	6-24
6.7.1 <define procedure structure>	6-24
6.7.2 DEFINE PROCEDURE Statement	6-25
6.7.3 PROCEDURE Body	6-26
6.7.4 LEAVE <procedure> Statement	6-27
6.7.5 END PROCEDURE Statement	6-28
6.8 REQUIRE Statement	6-28
6.9 INCLUDE Statement	6-33
6.10 IDENTIFY Statement	6-34
6.11 IDENTIFY TIMER Statement	6-36
6.12 IDENTIFY SIGNAL BASED EVENT Statement	6-36
6.13 IDENTIFY EVENT BASED EVENT Statement	6-39
6.14 IDENTIFY EVENT INTERVAL Statement	6-41
6.15 IDENTIFY EVENT INDICATOR Statement	6-43
6.16 IDENTIFY TIME BASED EVENT Statement	6-44
6.17 DIGITAL CONFIGURATION Definition	6-45

6.17.1 <define digital configuration structure>	6-45
6.17.2 DEFINE DIGITAL CONFIGURATION Statement	6-46
6.17.3 DEFINE DIGITAL SOURCE Statement	6-46
6.17.4 DEFINE DIGITAL SENSOR Statement	6-57
6.17.5 Deleted	6-62
6.17.6 END DIGITAL CONFIGURATION Statement	6-62
6.18 EXTEND Statement	6-63
6.19 ESTABLISH PROTOCOL Statement	6-68
6.20 DEFINE EXCHANGE Statement	6-71
6.21 DEFINE DIGITAL TIMING Statement	6-75
7.0 Procedural Statements	7-1
7.1 <main procedural structure>	7-1
7.2 <main procedural statements>	7-1
8.0 Procedural Statements, Data Processing	8-1
8.1 CALCULATE Statement	8-2
8.2 COMPARE Statement	8-17
9.0 Procedural Statements, Input/Output	9-1
9.1 Deleted	9-1
9.2 INPUT Statement	9-1
9.3 OUTPUT Statement	9-7
9.4 ENABLE FILE ACCESS Statement	9-12
9.5 DISABLE FILE ACCESS Statement	9-14
10.0 Procedural Statements, Control	10-1
10.1 IF THEN ELSE Capability	10-4
10.1.1 IF THEN ELSE Structure	10-4
10.1.2 IF THEN Statement	10-6
10.1.3 <procedural segment> (IF THEN ELSE)	10-7
10.1.4 LEAVE, IF Statement	10-8
10.1.5 ELSE Statement	10-9
10.1.6 END, IF Statement	10-10
10.2 WHILE THEN Capability	10-11
10.2.1 WHILE THEN Structure	10-11
10.2.2 WHILE THEN Statement	10-12
10.2.3 <procedural segment> (WHILE THEN)	10-13
10.2.4 LEAVE, WHILE Statement	10-13
10.2.5 END, WHILE Statement	10-14
10.3 FOR THEN Capability	10-15
10.3.1 FOR THEN Structure	10-15
10.3.2 FOR THEN Statement	10-17
10.3.3 <procedural segment> (FOR THEN)	10-19
10.3.4 LEAVE, FOR Statement	10-19
10.3.5 END, FOR Statement	10-20
10.4 GO TO Statement	10-21
10.5 PERFORM Statement	10-22
10.6 FINISH Statement	10-23
10.7 ENABLE DIGITAL CONFIGURATION Statement	10-23

11.0 Signal Oriented Statements	11-1
11.1 <procedural statements, signal>	11-1
11.2 Single-Action Verb	11-10
11.2.1 Single Action Verb: General Description	11-10
11.2.2 SETUP Statement	11-15
11.2.3 CONNECT Statement	11-17
11.2.4 DISCONNECT Statement	11-19
11.2.5 ARM Statement	11-21
11.2.6 FETCH Statement	11-23
11.2.7 CHANGE Statement	11-24
11.2.8 ENABLE EVENT Statement	11-26
11.2.9 DISABLE EVENT Statement	11-27
11.2.10 RESET Statement	11-28
11.3 Multiple-action Verb	11-29
11.3.1 Multiple-action Verb - General Description	11-29
11.3.2 APPLY Statement	11-31
11.3.3 REMOVE Statement	11-32
11.3.4 MEASURE Statement	11-34
11.3.5 MONITOR Statement	11-35
11.3.6 VERIFY Statement	11-36
11.3.7 READ Statement	11-37
11.3.8 INITIATE Statement	11-38
11.4 Digital Verb	11-39
11.4.1 Digital Verb- General Description	11-39
11.4.2 STIMULATE Statement	11-40
11.4.3 SENSE Statement	11-43
11.4.4 PROVE Statement	11-44
12.0 Procedural Statements Timing	12-1
12.1 Timing Statements (General)	12-1
12.2 Deleted	12-2
12.3 Deleted	12-2
12.4 Deleted	12-2
12.5 READ TIMER Statement	12-2
12.6 WAIT FOR Statement	12-2
12.7 RESET TIMER Statement	12-4
12.8 DO/END DO CAPABILITY	12-4
12.8.1 <do simultaneous structure>	12-4
12.8.2 DO SIMULTANEOUS Statement	12-5
12.8.3 <do simultaneous body>	12-6
12.8.4 END DO Statement	12-7
12.8.5 <do timed digital structure>	12-8
12.8.6 DO TIMED DIGITAL Statement	12-8
12.8.7 <do timed digital body>	12-11
13.0 Procedural Statements, Macro	13-1
13.1 <procedural statement macro>	13-2
13.2 Deleted	13-2

13.3 DO EXCHANGE Statement	13-2
13.4 UPDATE EXCHANGE Statement	13-10
14.0 Field and Subfield Definition	14-1
14.1 <statement characteristics>	14-1
14.2 <real characteristic subfield>	14-2
14.3 <digital characteristic subfield>	14-4
14.4 <sync subfield>	14-5
14.5 <real errlim> (Real Error Limit)	14-8
14.6 <dim lim>(Dimension Limit) Deleted	14-9
14.7 <pc lim>(Percent Limit) Deleted	14-9
14.8 <measured characteristic>	14-10
14.9 <evaluation field>	14-12
14.10 <eval statement characteristics> (Evaluation Statement Characteristics)	14-17
14.11 Deleted	14-18
14.12 <max time>	14-18
14.13 <time quantity>	14-18
14.14 <condition>	14-19
14.15 Deleted	14-19
14.16 Deleted	14-19
14.17 Deleted	14-19
14.18 <conn> (Connection Field)	14-20
14.19 <conn set> (Connection Set)	14-20
14.20 Deleted	14-28
14.21 Deleted	14-28
14.22 Deleted	14-28
14.23 <signal value>	14-28
14.24 Deleted	14-29
14.25 Deleted	14-29
14.26 Deleted	14-29
14.27 Deleted	14-29
14.28 Deleted	14-29
14.29 Deleted	14-29
14.30 Deleted	14-29
14.31 Deleted	14-29
14.32 Deleted	14-29
14.33 Deleted	14-29
14.34 <real quantity>	14-29
14.35 <index>	14-29
14.36 Deleted	14-30
14.37 Deleted	14-30
14.38 Deleted	14-30
14.39 Deleted	14-30
14.40 Deleted	14-30
14.41 Deleted	14-30
14.42 Deleted	14-30
14.43 Deleted	14-30

14.44 <noun field>	14-30
14.45 <dimensioned number>	14-31
14.46 <require control>	14-31
14.47 <require capability>	14-32
14.48 <require limit>	14-33
14.49 <require cnx>	14-34
14.50 <digital quantity>	14-35
14.51 Deleted	14-35
14.52 <data store>	14-35
14.53 <array range>	14-36
14.54 <parameter>	14-38
14.55 <gate field>	14-39
14.56 <stim rate>	14-40
14.57 <sense rate>	14-40
14.58 <sense delay>	14-41
14.59 <stim event>	14-41
14.60 <sense event>	14-42
14.61 <when field>	14-43
14.62 <digital source characteristic>	14-44
14.63 <digital sensor characteristic>	14-44
14.64 <on field>	14-45
14.65 <exchange expression>	14-46
14.66 <exchange frame>	14-47
14.67 <bus parameter>	14-52
14.68 <protocol parameter>	14-53
14.69 <role field>	14-55
14.70 <command field>	14-56
14.71 <data field>	14-57
14.72 <status field>	14-58
15.0 Language Component Specification	15-1
15.1 Authorized Characters	15-1
15.2 Number Representations	15-1
15.2.1 Deleted	15-2
15.2.2 <decimal-number>	15-2
15.2.3 <unsigned decimal number>	15-2
15.2.4 <unsigned decimal integer>	15-3
15.2.5 <digital number>	15-3
15.2.6 Deleted	15-3
15.2.7 <long decimal number>	15-3
15.3 Flag and Statement Numbers	15-3
15.4 Blank Space Requirements	15-6
15.5 <label>	15-6
15.6 Retention of Labelled Information	15-8
15.7 Range Subfield	15-11
15.8 Dimensions	15-11
15.9 Character Code Definitions	15-16
15.10 Special Character Sets	15-20
15.11 Miscellaneous Language Elements	15-21

16.0 Nouns and Their Modifiers	16-1
16.1 AC SIGNAL (Alternating Current Signal)	16-3
16.2 ADF (Automatic Direction Finder)	16-4
16.3 AM SIGNAL (Amplitude Modulation Signal)	16-5
16.4 AMBIENT CONDITIONS	16-6
16.5 ATC (Air Traffic Control)	16-7
16.6 COMMON	16-10
16.7 DC SIGNAL (Direct Current Signal)	16-12
16.8 DISPLACEMENT	16-12
16.9 DME (Distance Measuring Equipment)	16-13
16.10 DOPPLER	16-14
16.11 EARTH	16-15
16.12 EM FIELD (Electro Magnetic Field)	16-15
16.13 EVENTS	16-16
16.14 FLUID SIGNAL	16-17
16.15 FM SIGNAL (Frequency Modulation Signal)	16-19
16.16 HEAT	16-20
16.17 IFF (Identification, Friend or Foe)	16-20
16.18 ILS (Instrument Landing System)	16-22
16.19 IMPEDANCE	16-25
16.20 LIGHT	16-25
16.21 LOGIC CONTROL	16-26
16.22 LOGIC DATA	16-30
16.23 LOGIC LOAD	16-37
16.24 LOGIC REFERENCE	16-38
16.25 MANOMETRIC	16-39
16.26 PAM (Pulse Amplitude Modulation)	16-40
16.27 PM SIGNAL (Phase Modulated Signal)	16-41
16.28 PULSED AC (Pulsed Alternating Current Signal)	16-41
16.29 PULSED AC TRAIN	16-42
16.30 PULSED DC (Pulse Direct Current Signal)	16-45
16.31 PULSED DC TRAIN	16-47
16.32 PULSED DOPPLER	16-49
16.33 RADAR SIGNAL	16-49
16.34 RAMP SIGNAL	16-50
16.35 RANDOM NOISE	16-51
16.36 RESOLVER	16-51
16.37 ROTATION	16-52
16.38 SHORT	16-58
16.39 SQUARE WAVE	16-58
16.40 STEP SIGNAL	16-59
16.41 SUP CAR SIGNAL (Suppressed Carrier Signal)	16-59
16.42 SYNCHRO	16-60
16.43 TACAN (Tactical Air Navigation)	16-61
16.44 TIME INTERVAL	16-67
16.45 TRIANGULAR WAVE SIGNAL	16-68
16.46 TURBINE ENGINE DATA	16-68
16.47 VIBRATION	16-70
16.48 VOR (VHF Omnidirectional Radio Range)	16-71
16.49 WAVEFORM	16-73

17.0 NOUN-MODIFIER DEFINITIONS	17-1
17.1 Mnemonics for Pulse-Type Signals	17-1
17.2 Modifier Prefixes and Suffixes	17-2
17.3 Digital Noun-Modifiers	17-7
17.4 Noun Modifiers	17-8
18.0 Standard C/ATLAS Syntax	18-1
18.1 Introduction	18-1
18.1.1 Purpose	18-1
18.1.2 Organization	18-1
18.1.3 References to the Formal Syntax Definition	18-2
18.2 Syntax Notation	18-3
18.2.1 Syntax Language Elements	18-3
18.2.2 Structure of a Meta-Program	18-4
18.2.3 Rule Bodies	18-5
18.3 Program Structure Syntax	18-7
18.3.1 Range and Order of the Definition	18-7
18.3.2 Name Conventions	18-8
18.3.3 Syntax Definition	18-9
18.4 Statements and Field Syntax	18-13
18.4.1 Range and Order of the Definition	18-13
18.4.2 Name Conventions	18-13
18.4.3 Notes	18-13
18.4.4 Syntax Definition	18-15
18.5 Lexical Symbols Syntax	18-46
18.5.1 Range and Order of the Definition	18-46
18.5.2 Name Conventions	18-46
18.5.3 Classes of Lexical Symbols	18-48
18.5.4 Syntax Definition	18-49
18.6 Cross-Reference Listings of Syntax Variables and Symbols	18-67
18.6.1 Cross-Reference List of Rule Names	18-67
18.6.2 Cross-Reference List of Literals	18-82

1.0 Purpose

IEEE Std 716, Common ATLAS (C/ATLAS) 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.

This document defines an operational standard which is based upon the language defined by the reference document for the ATLAS language, IEEE Std. 416, in that, for those cases where a capability was required that went beyond the existing reference document and working papers or it was not possible to conveniently subset the 416 material, extensions were developed for IEEE Standard 716.

Differences do exist between IEEE Stds 716 and 416 primarily due to different development and publication cycles. Every reasonable effort will continue to be made to maintain compatibility between these standards.

1.1 Language Processors

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.

1.2 Document Control

All aspects of definition, creation, extensibility, management, maintenance, quality assurance and proposal handling shall be the responsibility of the Std 716 Subcommittee to the satisfaction of the Steering Committee of the IEEE Standards Coordination Committee 20 (SCC-20).

2.0 Reference Material

2.1 Applicable Documents

The following documents are applicable to this standard:

- [1] ANSI X3/TR-1-1983, American National Standard dictionary for Information Processing.¹
- [2] ANSI X3.41-1974, Code Extension Techniques for Use with the 7-Bit Coded Character Set of American National Standard Code for Information Interchanges (ASCII).¹
- [3] ANSI/IEEE Std 416-1984, IEEE Standard ATLAS Test Language (including material from Revision Levels up to 30).²
- [4] ANSI/IEEE Std 771-1984, IEEE Guide to the Use of ATLAS.²
- [5] ARINC 573, Aircraft Integrated Data Systems-Mark 2 (AIDS) for the NRZ Code.³
- [6] IEEE Std 100-1988, Standard Dictionary of Electrical and Electronics Terms.⁴
- [7] ISO 646-1983, International Standard Seven-Bit Character Set.⁵
- [8] ISO 1000-1981, SI Units and Recommendations for the Use of their Multiples and of Certain Other Units.⁵
- [9] MIL-STD-291B, Standard Tactical Navigation (TACAN) Signal, 13 Dec 67.⁶
- [10] 106-71, Telemetry Standards (revised January 1971, Figure 4, p 42).⁷

2.2 Document Precedence

In case of any conflict in the definition of C/ATLAS within IEEE Std 716, the definition follows the precedence in the order listed:

- a. Formal Syntax
- b. Rules and Descriptions
- c. Syntax Diagrams
- d. Notes and Examples

FOOTNOTES

- ¹ ANSI publications are available from the Sales Department, American National Standards Institute, 1430 Broadway, New York, NY 10018.
- ² ANSI/IEEE publications are available from the Sales Department, American National Standards Institute, 1430 Broadway, New York, NY 10018, or from the Institute of Electrical and Electronic Engineers, Service Center, Piscataway, NJ 08854-1331.
- ³ ARINC publications are available from ARINC Research Corporation, Document Section, 2551 Riva Rd., Annapolis, MD 21401.
- ⁴ IEEE publications are available from the Institute of Electrical and Electronic Engineers, Service Center, Piscataway, NJ 08854-1331.
- ⁵ ISO publications are available in the US from the Sales Department, American National Standards Institute, 1430 Broadway, New York, NY 10018.
- ⁶ Naval Publications and Forms Center, 5801 Tabor Avenue, Philadelphia, PA, 91920-5094.
- ⁷ Available from the Telemetry Working Group, Inter-Range Commanders Council, White Sands Missile Range, NM 88002.

2.3 Document Organization and Conventions

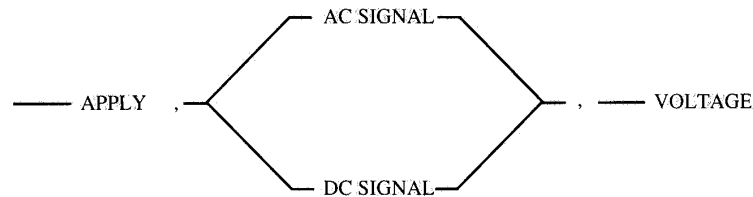
2.3.1 Extensibility

The use of the extend capability for adding nouns, noun modifiers, and pin descriptors must 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 Section 18. Names in subsections entitled "Formal Syntax" refer to corresponding names in Section 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 must 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 which satisfy the variable <label>, then the following are valid sequences:

```
'HI-VAL'
'HI-VAL',    'LO-VAL'
'HI-VAL',    'LO-VAL',    'AVRG-VAL'
```

There are three types of C/ATLAS elements which are included in C/ATLAS syntax diagrams:

- (1) literals — uppercase and special characters which are included in ATLAS sequences exactly as they appear.
- (2) components — lowercase letters enclosed by the characters "<" and ">" which are replaced by valid symbols and/or numbers as described in paragraph 15.
- (3) subdiagrams — broken bordered boxes containing a component-like name and the paragraph number where the subdiagram is specified. The box is replaced by sequences valid according to that specified paragraph.

Subdiagrams which are 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 name and the paragraph number in which the definition appears.

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 section where each structure or statement is located in this standard.

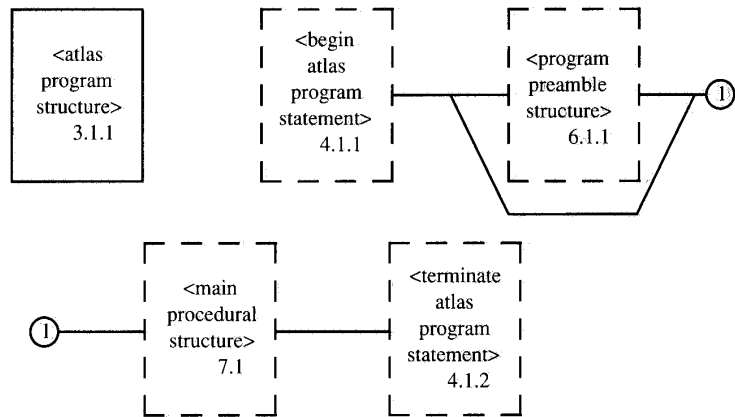


Figure 3-2. C/ATLAS Program Structure

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 section where each structure or statement is located in this standard.

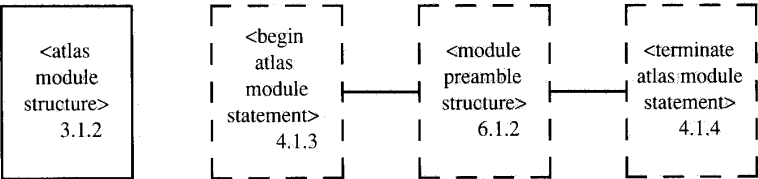


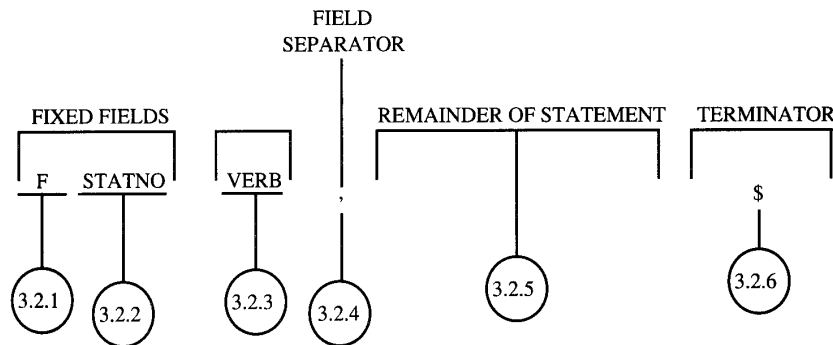
Figure 3-3. C/ATLAS Module Structure

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 section 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 paragraph 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) must be written on the first line of the statement.



Note: Numbers in circles refer to paragraphs 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 must 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 section 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

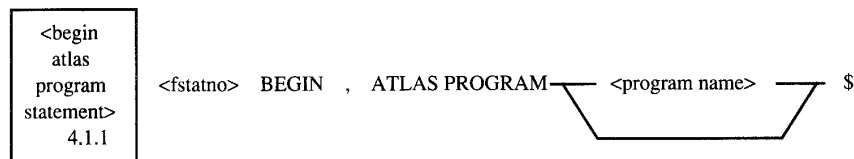
4.1.1.1 Function

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

4.1.1.2 Formal Syntax

Reference begin-atlas-program-stmt

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

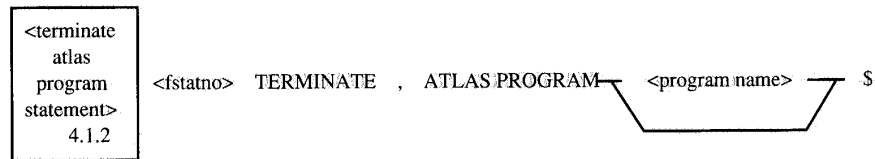
4.1.2.1 Function

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

4.1.2.2 Formal Syntax

Reference terminate-atlas-program-stmt

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> must 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

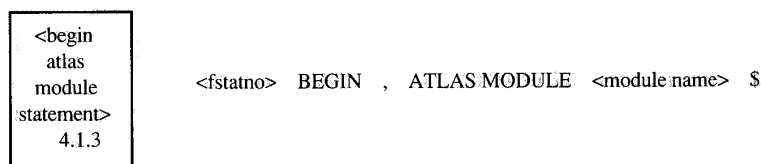
4.1.3.1 Function

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

4.1.3.2 Formal Syntax

Reference: begin-atlas-module-stmt

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> must be unique within one <complete atlas test program structure>.

4.1.3.5 Notes and Examples

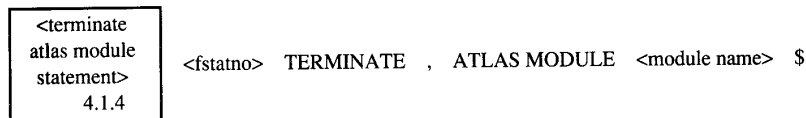
```
120000 BEGIN, ATLAS MODULE 'CALIBRATION PROCS' $
```

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-stmt

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

5.0 This paragraph 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 Section 15.6.1.

6.1 Main Preamble Structure

6.1.1 <program preamble structure>

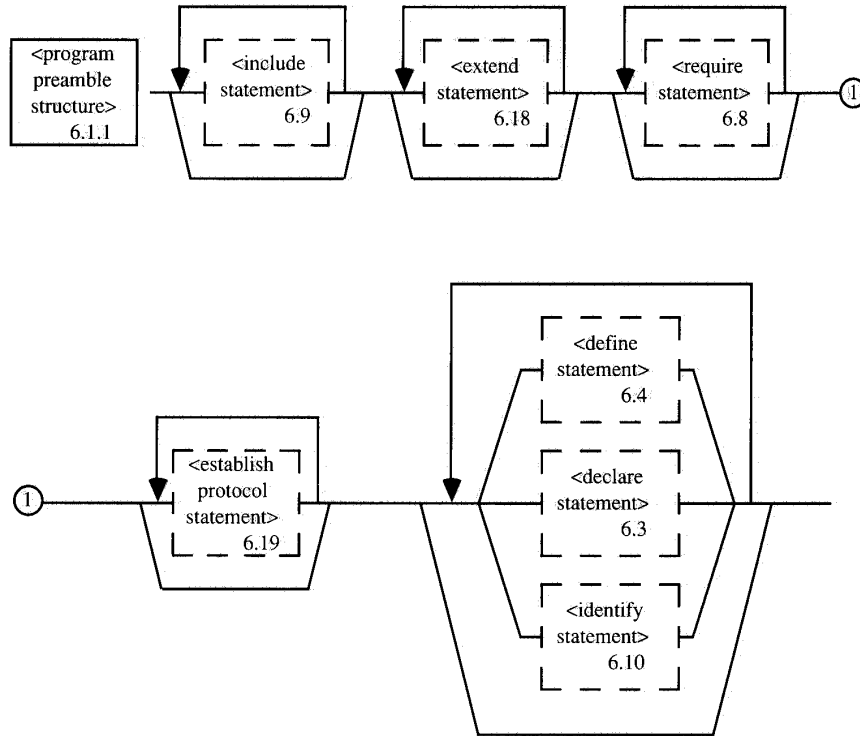
6.1.1.1 Function

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-struct

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>` must 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>`.

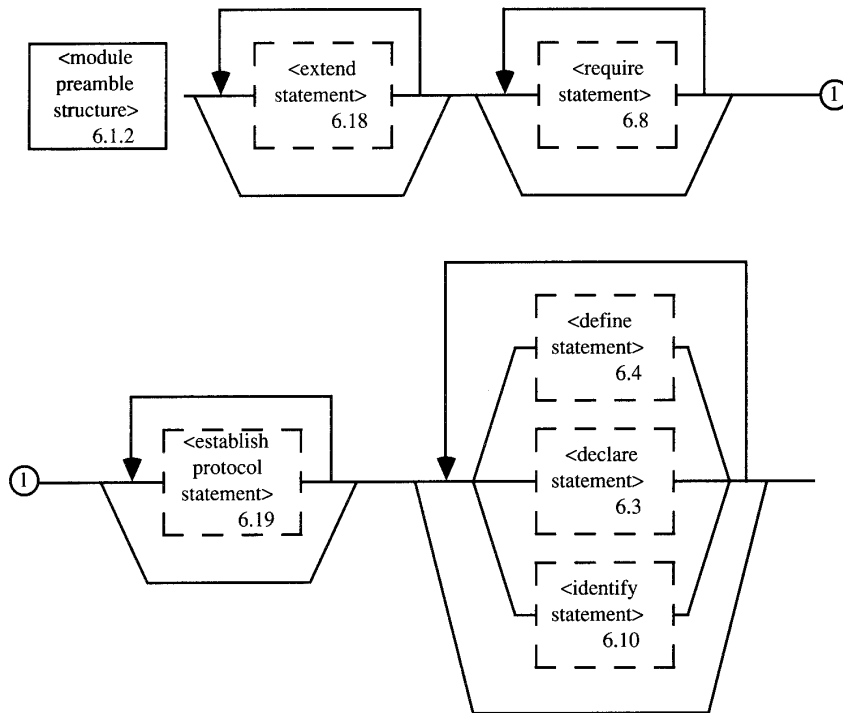
6.1.2 `<module preamble structure>`

6.1.2.1 Function

The `<module preamble structure>` which 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-struct

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 must 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>

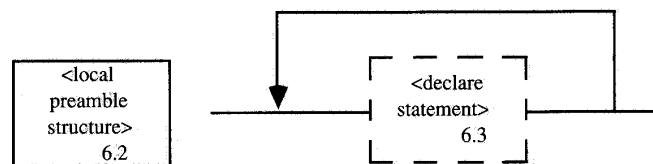
6.2.1 Function

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

6.2.2 Formal Syntax

Reference local-preamble-struct

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> must 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

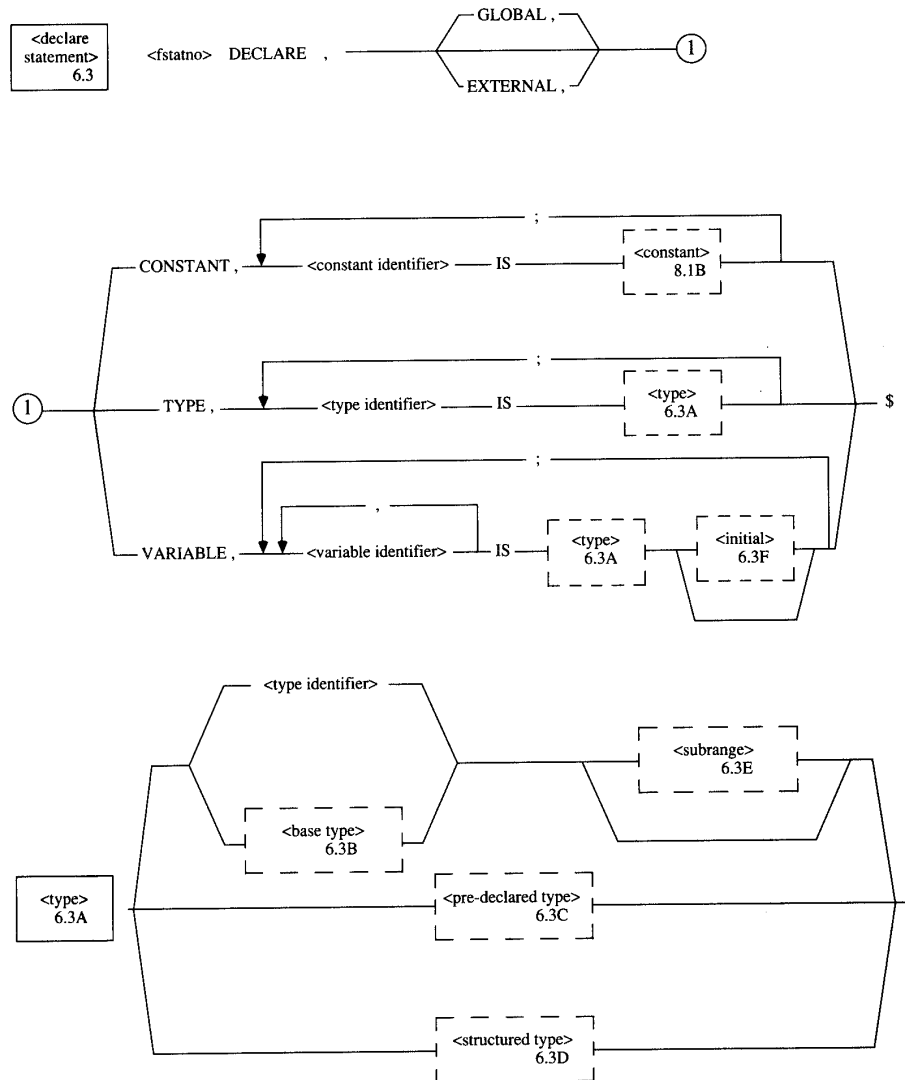
6.3.1 Function

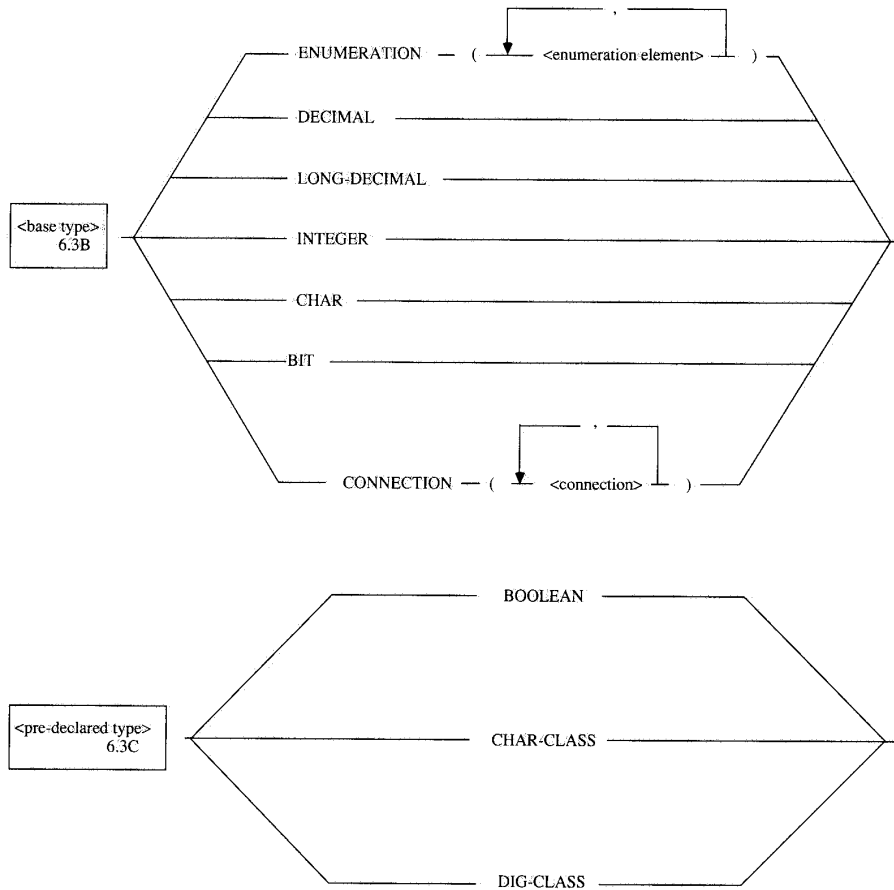
To establish, label, and identify data types, variables, and constants.

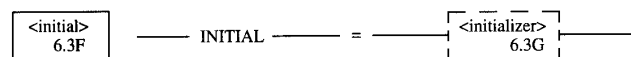
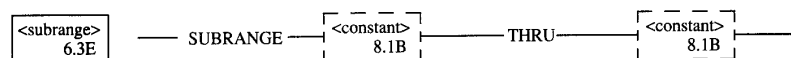
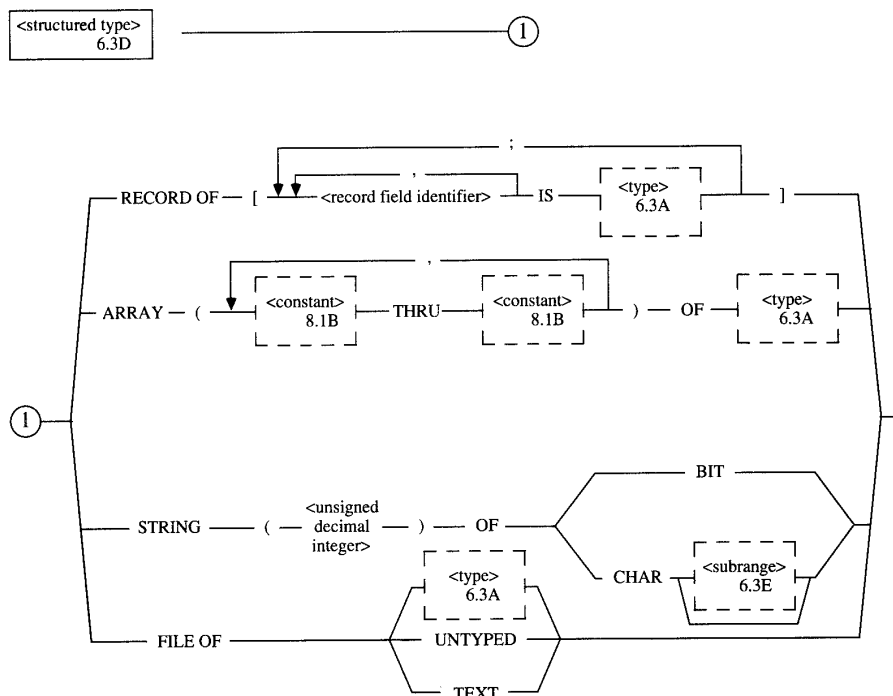
6.3.2 Formal Syntax

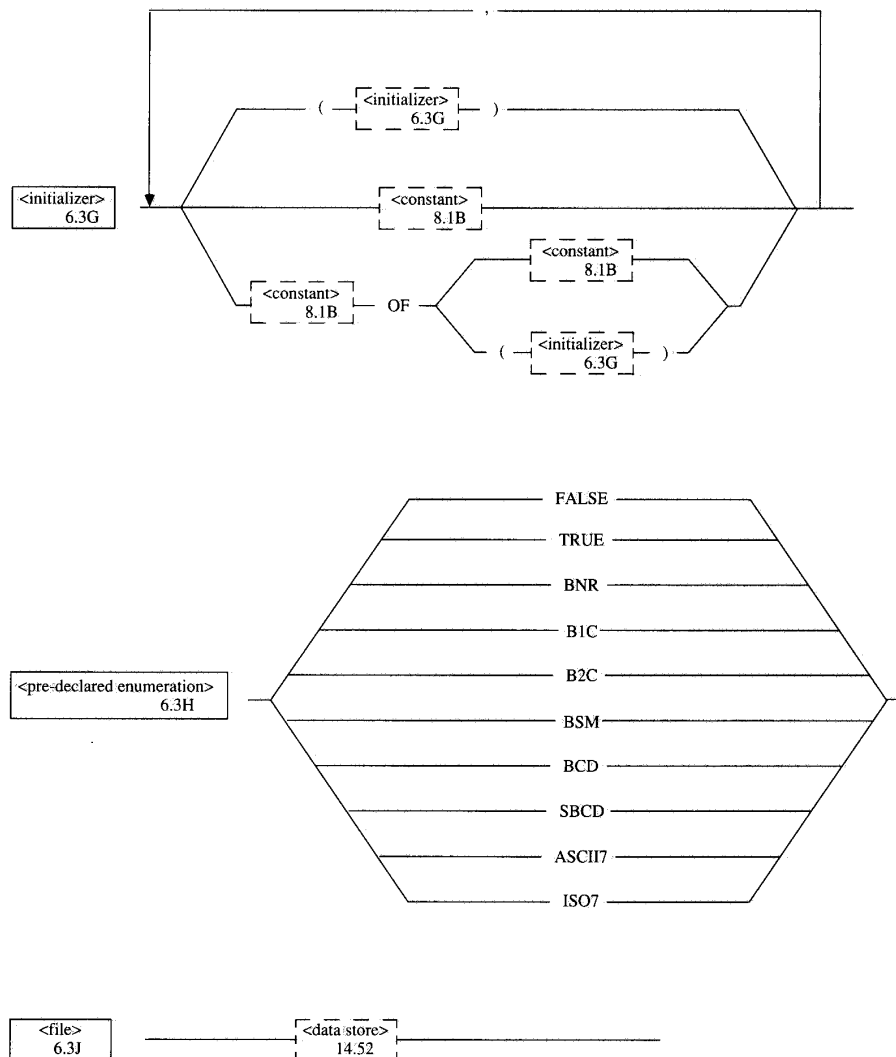
Reference declare-stmt

6.3.3 Syntax Diagram









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 must 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 must 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 must be of the same <type> and:
 - a. if they are ARRAYS, the corresponding dimensions must have the same number of components with the same index number
 - b. if they are STRINGS, they must 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>

1. 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 paragraph 8.1.4.5.

6.3.4.3 Rules for TYPE form of <declare statement>

1. 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 section 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 <parameter> subfield, the description of the <type> can be stated explicitly without reference to a <type identifier> label.
3. Two <expression>s are type compatible if they are:
 - a. of the same <base type>, or
 - b. records with identical structure and corresponding type compatible components, or
 - c. arrays with the same number of dimensions and type compatible components, or
 - d. strings with type compatible components, or
 - e. files that are:
 - i. of the same type, or
 - ii. both UNTYPED, or
 - iii. both TEXT.
4. An <expression> is assignment compatible with a <data store> if:
 - a. The <expression> and <data store> are type compatible and
 - i. if they are ARRAYS, the corresponding dimensions have the same number of components, or

- ii. if they are STRINGS, the current dynamic length of the <expression> is less than or equal to the declared length of the <data store>, or
- b. the <expression> is of type INTEGER and the <data store> is of type DECIMAL or LONG-DECIMAL, or
- c. the <expression> is of type DECIMAL and the <data store> is of type LONG-DECIMAL.

6.3.4.6 Rules for <base type>

1. A variable of <base type> must 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 must 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 14.18 <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 must 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 Section 15.2G.

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 which 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 section 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 section 15.9.1)

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

6.3.4.7.1 Notes

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 BITS.

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 must 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 must be of the same <type>. The <type> must 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 equal numbers of elements. The only operations permitted between entire arrays are assignment, test for equality and test for inequality.
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, test for equality and test for inequality. Records must be assignment compatible and have equal numbers of elements.
6. The constant specifying the length of a string must be greater than zero and of <base type> INTEGER.
7. The <base type> of a STRING must be either BIT or CHAR or a subrange of CHAR.
8. A STRING has an associated dynamic length attribute which holds the current length of the STRING.
 - a. Initialization, input, assignment, and parameter passing operations 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 parameter.
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 parameter.

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 which 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 must be to and from a variable of identical type.
 - b. An untyped file consists of bytes of data which 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.
 - c. A text file consists of a stream of characters which 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 which may be assigned to a variable to lie within a sub-division of the range of values which may be assigned to its <base type>.
2. The type of <constant> must conform to the <base type>.
3. The value of the <constant>s must lie within the bounds of the <type> being ranged. The value of the first constant must be less than or equal to the value of the second constant.
4. The <type> being ranged must be ENUMERATION, DECIMAL, LONG-DECIMAL, INTEGER or CHAR and must 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 parameter passing to subrange variables must 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> which corresponds to a simple variable.
- b. A <constant> number of <constant>s where the first <constant> must evaluate to a positive integer which 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 c above. This is in the form of "<constant> OF (list of values)" where <constant> must be a positive integer which 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 <constant> value supplied by the <initializer> must be type compatible with the corresponding variable with the exception that DECIMAL and LONG-DECIMAL variables may be initialized with <decimal number>.

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

6.3.4.11 Rules for <file>

- 1. The <data store> must be of type FILE.

6.3.5 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', 'PINSET2' IS ARRAY (1 THRU 4) OF
    CONNECTION (J1, J3, J5, J7, J9, J11, J13, J15,
                J17) $
```

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. The following 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

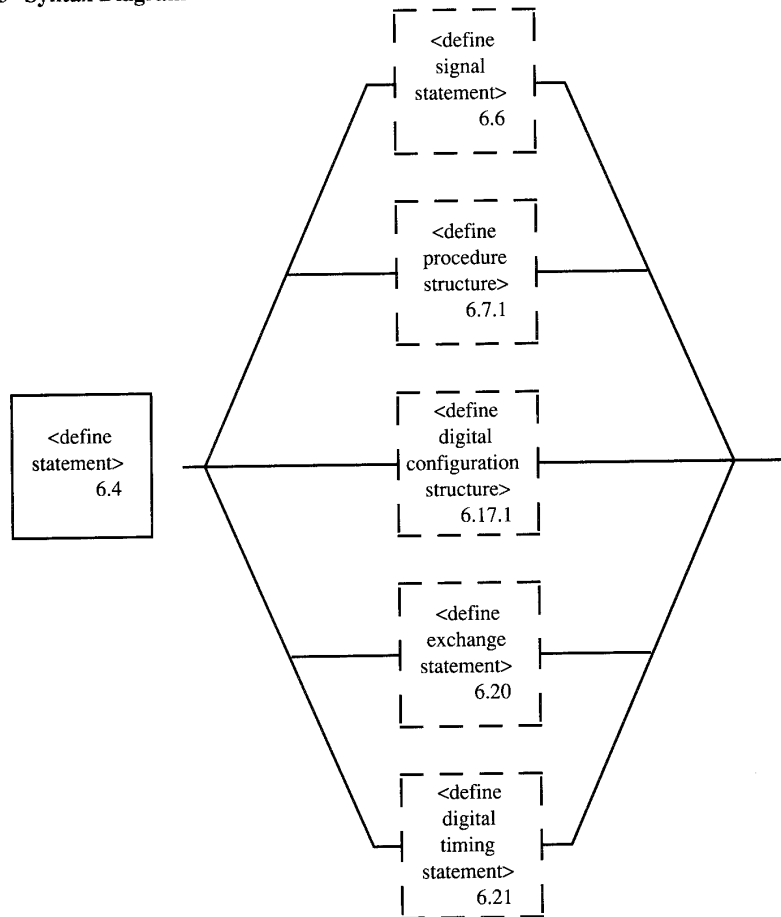
To establish and label a portion of the <complete atlas test program structure> which is not executed until the label is invoked by direct or indirect reference in a subsequent <main procedural statement>.

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-stmt

6.4.3 Syntax Diagram**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 Deleted

6.6 DEFINE <signal> Statement

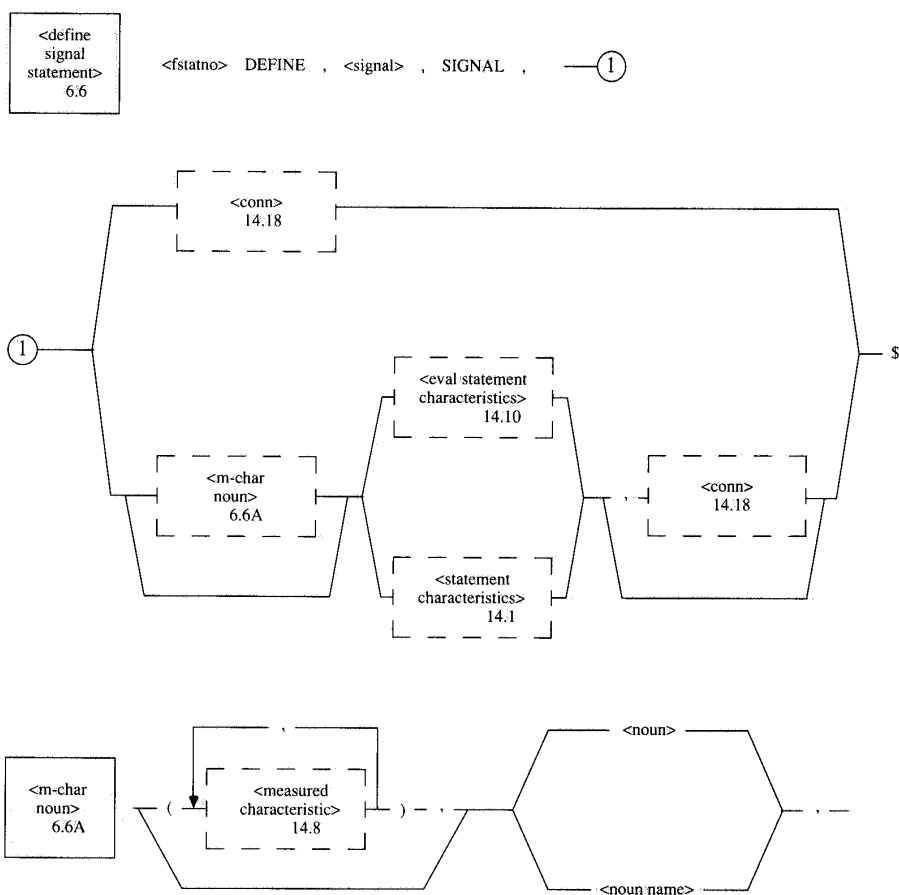
6.6.1 Function

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 cited in the subsequent <define statement> or <main procedural statement>. In a <define signal statement> the number of spaces between the parentheses is not important.

6.6.2 Formal Syntax

Reference define-signal-stmt

6.6.3 Syntax Diagram



NOTE:

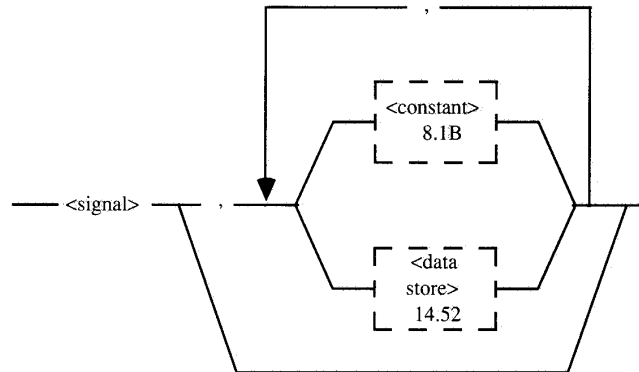
1. A <decimal number>, <digital number>, and <connection> may be designated as variable by replacing the item by a pair of parentheses. <label>s defined in a <define signal statement> may be used either in subsequent DEFINE statements or in procedural statements. The value for any parentheses pairs in the definition must 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 must 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).

6.6.4 Rules

1. This type of statement is employed to establish and identify one or more fields which 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 which 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 which will be assigned to that characteristic in subsequent statements.
4. In the statement which 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 must be filled explicitly. They may not be filled by "daisy-chaining" through another pair of parentheses in the calling DEFINE statement.
6. It is not allowed to specify a dimension as variable.
7. The DEFINE <signal> statement provides a standard format which 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 Section 11 or in the corresponding Formal Syntax Definition.

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



6.6.5 Notes and Examples

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

```
000011 DEFINE, 'RZ STIM', SIGNAL, AC SIGNAL,
      VOLTAGE ( )
      RANGE 100V TO 200V, 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 0V TO 10V
      ERRLMT + - 10 PC,
      CNX HI J2-3 LO J2-4 $
111144 APPLY, DC SIGNAL, AC-COMP MAX 5MV, 'GRID', 5 $
111155 APPLY, AC SIGNAL, DC-OFFSET 20V, 'GRID', 1.10 $
```

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 $
```