

IEEE Guide to the Use of the ATLAS* Specification

*Abbreviated Test Language for All Systems

100 \$
TAGE 20MV ERR
ANGE 100HZ TO
0 SKT1-2 \$
AC SIGNAL, VOLTAGE
1000HZ,
2 LO SKT2-4 \$
MEASUREMENT
5 LL 195 \$
0 \$
0000 CALCULATE
10 APP AC SIGNAL, VOLTAGE 20MV ERR LMT $\pm 0.5\%$,
FREQ 'STIM-F' RANGE 100HZ TO 1000HZ ERR LMT $\pm 5\%$,
CNX HI SKT1-1 LO SKT1-2 \$
MEASURE, (VOLTAGE), AC SIGNAL, VOLTAGE MAX 10000MV,
FREQ MAX 1000HZ,
CNX HI SKT2-2 LO SKT2-4 \$
LOOP RETURN \$
CALCULATE, 'GAIN' = 'MEASUREMENT'/20 \$
COMPARE, 'GAIN', UL 205 LL 195 \$
50 GO TO, STEP 004030 IF HI,
STEP 000100 IF LO \$
60 CALCULATE, 'STIM-F' = 100 \$
70 COMPARE, 'STIM-F', GE 1000 \$
80 GO TO, STEP 004200 IF HI,
STEP 000100 IF LO \$
90 SETUP, AC SIGNAL, VOLTAGE 20MV ERR LMT $\pm 0.5\%$,
FREQ 'STIM-F' RANGE 100HZ TO 1000HZ ERR LMT $\pm 5\%$,
CNX HI SKT1-1 LO SKT1-2 \$
00 READ, (VOLTAGE), AC S
FREQ MAX 1000H
CNX HI SKT2-2 LI
GO TO, STEP 004030 \$
OPERATIONS COMPLETE
004030 CALCULATE, 'GAIN',
004040 COMPARE, 'GAIN', UL
004050 GO TO, STEP 200000 IF HI,
STEP 200100 IF LO \$
004060 CALCULATE, 'STIM-F' = 'STIM-F' + 100 \$
004070 COMPARE, 'STIM-F', GE 1100 \$
004080 GO TO, STEP 004200 IF GO \$
004090 SETUP, AC SIGNAL, VOLTAGE 20MV ERR LMT $\pm 0.5\%$,
FREQ 'STIM-F' RANGE 100HZ TO 1000HZ ERR LMT $\pm 5\%$,
CNX HI SKT1-1 LO SKT1-2 \$
004100 READ, (VOLTAGE), AC SIGNAL, VOLTAGE MAX 10000MV,
FREQ MAX 1000HZ,
CNX HI SKT2-2 LO SKT2-4 \$

Initialization

Including tests
and
event loops

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

November 20, 1989

IEEE Std 771-1989
(Revision of ANSI/IEEE
Std 771-1984)

IEEE Guide to the Use of the ATLAS* Specification

Sponsor

**IEEE SCC20 (ATLAS) Committee
of the
IEEE Standards Board**

Approved August 17, 1989
IEEE Standards Board

*Abbreviated Test Language for All Systems

© 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.*

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 that 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
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
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.



Documents

Document Sharing Hub

Foreword

(This Foreword is not a part of IEEE Std 771-1989, IEEE Guide to the Use of The ATLAS Specification.)

ANSI/IEEE Std 416-1984, ATLAS Test Language, IEEE Std 716-1989, IEEE Standard C/ATLAS Test Language, and ARINC Specification 626-1-1989, Standard ATLAS Subset for Modular Test, define ATLAS, a standardized test language for expressing test specifications and procedures. This guide is supplementary to ANSI/IEEE Std 416-1984, IEEE Std 716-1989, and ARINC Specification 626-1-1989, and is part of the documentation set for the language.

The need for a guide to the use of ATLAS arose from the experience of users in the preparation of test procedures in the language. This showed that differences of interpretation existed as to technique and method, if not as to definition. The purpose of this guide is to provide guidance in the matter of technique and usage; it is seen as particularly important in promoting the use of the language envisaged by its originators. This document is not intended as an instruction manual for ATLAS, nor as a substitute for some formal tuition, but by its nature it should find some applicability in the training environment.

This issue of ANSI/IEEE Std 771-1989 corresponds with the issue of ANSI/IEEE Std 416-1984, IEEE Std 716-1989 and ARINC Specification 626-1-1989. It is the intention of the IEEE ATLAS Committee that there shall be a continued correspondence between these standards and IEEE Std 771-1989.

The members of the ATLAS 771 sub-committee during the preparation of this guide were:

Anthony A. F. Vogelpoel, Cochairman
Linda O. Lund, Cochairman

George J. Diab
 Jim Forrest

Arnold M. Greenspan
 Ashley Hulme
 Neil A. McKinnon

William E. Reeves
 Wolfgang Ried

In addition, acknowledgement is made of the valuable individual contributions made by the following:

S. J. Cherry
 R. Duffour
 H. V. Ehrismann
 H. Grundner
 A. Hann
 M. G. Harrison

R. L. Heather
 K. E. Hermance
 R. J. Kelly
 H. A. Mason
 J. Massey
 R. Mattice

R. J. Patrick
 M. Prcic
 S. M. Schlosser
 J. L. Smeaton
 F. Szymanski
 T. H. Tabb

At the time it completed its review of this standard, the officers of the IEEE ATLAS Committee (SCC20) served as the balloting committee and consisted of the following members:

D. H. Abernathie
 M. Blair
 R. Brooks
 A. Bunsen
 L. D. Carpenter
 M. Carter
 C. R. Childress
 H. A. Churchill
 F. Conforti
 J. L. Cross
 G. Demars
 G. J. Diab
 B. Dinkluge
 H. U. Ehrismann
 H. D. Ey
 K. Goetz
 M. Gooding
 A. M. Greenspan
 A. Hann
 M. G. Harrison
 D. Hart

R. Hasenstab
 R. G. Hayes
 R. L. Heather
 R. T. Heatherington
 K. E. Hermance
 J. M. Barre
 H. M. Kaunzinger
 K. Kugler
 S. Kval
 J. Lewis
 L. Lund
 M. S. Martin
 K. McClure
 N. McKinnon
 V. Monie
 D. Morris
 J. W. Mowery
 T. Murphy
 H. Obermeier
 R. Oishi
 C. Ozenfant

J. Passwater
 R. J. Patrick
 F. D. Petry
 J. Pouilly
 W. Reeves
 J. Reh
 A. Sanson
 H. Scheibenzuber
 W. M. Scheinin
 S. M. Schlosser
 M. Seavey
 P. T. Smith
 T. H. Tabb
 H. Thomale
 A. Vogelpoel
 R. Vuarnoz
 C. J. Walton
 R. J. Weger
 P. L. Weimer
 K. D. Wilkinson
 W. Young



Documents

Document Sharing Hub

When the IEEE Standards Board approved this standard on August 17, 1989, it had the following membership:

Dennis Bodson, *Chairman*

Marco W. Migliaro, *Vice Chairman*

Andrew G. Salem, *Secretary*

Arthur A. Blaisdell
Fletcher J. Buckley
Allen L. Clapp
James M. Daly
Stephen R. Dillon
Donald C. Fleckenstein
Eugene P. Fogarty
Jay Forster*
Thomas L. Hannan

Kenneth D. Hendrix
Theodore W. Hissey, Jr.
John W. Horch
David W. Hutchins
Frank D. Kirschner
Frank C. Kitzantides
Joseph L. Koepfinger*
Edward Lohse

John E. May, Jr.
Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*
Richard E. Mosher
Stig Nilsson
L. John Rankine
Gary S. Robinson
Donald W. Zipse

*Member Emeritus

Contents

SECTION	PAGE
1. Introduction	1-1
1.1 For Whom This Volume Is Intended.....	1-1
1.2 Purpose of the Guide	1-1
1.3 Organization of the Guide.....	1-1
1.4 Related Documents.....	1-1
1.5 Standards References.....	1-1
1.6 Definition of Terms.....	1-2
2. The Need for a Test-Language Standard.....	2-1
2.1 The Importance of Unambiguous Communication.....	2-1
2.2 Types of Languages	2-1
2.2.1 Person-to-Person Communication Language	2-1
2.2.2 Person-to-Machine Communication Language.....	2-2
2.3 The Test Language.....	2-3
2.4 ATLAS Development.....	2-3
2.4.1 General.....	2-3
2.4.2 Constraints.....	2-7
2.5 Continued Regulated Growth of ATLAS	2-8
2.6 ATLAS as an Aid to the Management of Testing.....	2-8
2.7 Benefits of a Standard.....	2-8
3. ANSI/IEEE Std 416-1984, IEEE Std 716-1989 and ARINC Specification 626-1-1989	3-1
3.1 Definition of ATLAS.....	3-1
3.2 ANSI/IEEE Std 416-1984, IEEE Std 716-1989 and ARINC Specification 626-1-1989	3-1
3.2.1 General Definition.....	3-1
3.2.2 General Characteristics of ATLAS.....	3-1
3.2.3 Levels of ATLAS.....	3-1
3.2.4 ATLAS Test Procedure Structures.....	3-2
3.3 Formal Syntax Section.....	3-8
3.4 NON-ATLAS Usage	3-8
4. The Elements of ATLAS.....	4-1
4.1 Introduction.....	4-1
4.2 The ATLAS Character Set.....	4-1
4.3 Keywords.....	4-2
4.4 ATLAS Number Conventions.....	4-2
4.4.1 General.....	4-2
4.4.2 Decimal Numbers and Decimal Integers.....	4-2
4.4.3 Digital Numbers.....	4-3
4.4.4 Statement Numbers	4-5
4.4.5 Complex Numbers.....	4-6
4.5 User-Formed Strings	4-6
4.5.1 General.....	4-6
4.5.2 Constants	4-7
4.5.3 Variables.....	4-8
4.5.4 Types.....	4-10

SECTION	PAGE
4.5.5 Defined Names	4-11
4.5.6 Connections	4-11
4.5.7 Messages	4-12
4.5.8 Character Strings	4-12
4.6 Classification of the Language Elements	4-12
4.6.1 General	4-12
4.6.2 Verbs	4-12
4.6.3 Nouns and Modifiers	4-15
5. Writing a Test Procedure in ATLAS	5-1
5.1 Introduction	5-1
5.2 Analysis of the Test Problem	5-1
5.3 Segmenting the Test Procedure	5-1
5.4 Flow Charting	5-1
5.5 Logical Branching	5-2
5.6 Test Resources	5-2
5.7 Arrangement of Test Procedure	5-2
5.7.1 Preamble	5-2
5.7.2 Delimiting Statements	5-6
5.7.3 Label Control	5-6
5.8 GLOBAL Procedures	5-8
5.9 Instructions to and from the Operator	5-9
5.10 Deleted	
5.11 Deleted	
5.12 Applying Power to the UUT	5-11
5.13 Resource Allocation	5-13
5.14 Identifying Signals with Resources	5-13
5.15 Applying Test Signals	5-14
5.15.1 Applying Digital Signals	5-15
5.16 Sensing UUT Responses	5-16
5.17 Statement Execution Rate	5-18
5.17.1 General	5-18
5.17.2 PREPARE/EXECUTE Structure	5-18
5.18 Postponing Statement Execution	5-19
5.18.1 General	5-19
5.18.2 DELAY Statement	5-19
5.18.3 WAIT FOR, Event Statement	5-19
5.18.4 WHILE, THEN Structure	5-20
5.18.5 Synchronization and Strobing	5-20
5.19 Branching	5-21
5.19.1 General	5-21
5.19.2 Entry Points	5-22
5.20 Iteration in Testing	5-22
5.20.1 General	5-22
5.20.2 Loop Control	5-23
5.20.3 The Use of Loops	5-23
5.20.4 Procedures for Loops	5-28
5.20.5 Implied Iterations	5-31

SECTION	PAGE
5.20.6 Summary.....	5-31
5.21 Complex-Action Verbs.....	5-31
5.22 Multistatement Sequences.....	5-31
5.22.1 Adjustments.....	5-31
5.22.2 EVENT COUNTING.....	5-32
5.22.3 Escape Fields.....	5-32
5.23 Simultaneous Testing.....	5-32
5.23.1 Tasks.....	5-32
5.23.2 Communication Between Tasks and Mainstream Testing.....	5-33
5.24 Test Problems Not Explicitly Covered by ATLAS Vocabulary.....	5-33
5.24.1 General.....	5-33
5.24.2 EXTEND.....	5-33
5.24.3 DEFINE, 'Drawing'.....	5-35
5.25 Use of Comments.....	5-35
5.26 Shutting Down a UUT.....	5-35
5.26.1 General.....	5-35
5.26.2 At the Conclusion of Testing.....	5-36
5.26.3 On-Line Repair or Adjustment.....	5-36
5.26.4 Malfunction.....	5-36
5.26.5 Test Equipment Malfunction.....	5-36
5.27 Recommended Standards of Good ATLAS.....	5-36
5.27.1 User Preparation.....	5-37
5.27.2 Writing the Test Document.....	5-37
5.27.3 Verification and Validation of the ATLAS Tests.....	5-39
6. Documentation and Structuring of Test Requirements.....	6-1
6.1 Documentation of Test Requirements.....	6-1
6.2 Unit Under Test.....	6-1
6.3 Design Requirement.....	6-1
6.4 Production Test Procedure.....	6-3
6.4.1 General.....	6-3
6.4.2 Production Test Procedure Unit Type 4051-6.....	6-3
6.5 ATLAS Test Procedure.....	6-5
6.5.1 General.....	6-5
6.5.2 Sample ATLAS-Structured Programming Version (Preferred Method).....	6-5
6.5.3 Commentary on Structured Programming Version.....	6-5
6.5.4 Sample ATLAS GO TO Version (Less Preferred Method).....	6-12
6.5.5 Commentary on GO TO Version.....	6-12
6.5.6 Summary.....	6-17
7. ATLAS Elements.....	7-1
7.1 Introduction.....	7-1
7.2 ATLAS Verbs.....	7-1
7.3 ATLAS Nouns.....	7-2
7.4 ATLAS Noun Modifiers.....	7-4
7.5 ATLAS Units.....	7-13
7.6 Miscellaneous/Math Elements.....	7-21
8. Glossary of ATLAS Keywords.....	8-1

FIGURES

PAGE

Fig 1	ATLAS Test Specification Family Tree.....	3-3
Fig 2	ATLAS Program Structure Syntax Diagram.....	3-3
Fig 3	ATLAS Program Structure	3-4
Fig 4	Program Preamble Structure Syntax Diagram.....	3-6
Fig 5	Basic Statement Structure	3-7
Fig 6 (a)	State Diagram for a Source or Load Virtual Resource Without Event Monitor Using Single-Action and Multi-Action Verbs	4-16
Fig 6 (b)	State Diagram for a Virtual Analog Source or Load Resource Without Event Monitor ..	4-17
Fig 6 (c)	State Diagram for a Source or Load Virtual Resource Without Event Monitor Using Single-Action and Multi-Action Verbs	4-18
Fig 7 (a)	State Diagram for a Source or Load Virtual Resource With Event Monitor Using Single-Action and Multi-Action Verbs.....	4-19
Fig 7 (b)	State Diagram for a Virtual Analog Source or Load Resource With an Event Monitor ..	4-20
Fig 7 (c)	State Diagram for a Source or Load Virtual Resource With an Event Monitor Using Single-Action and Multi-Action Verbs.....	4-21
Fig 8 (a)	State Diagram for a Sensor Virtual Resource Without Event Monitor for Making a Measurement Using Single-Action and Multi-Action Verbs.....	4-22
Fig 8 (b)	State Diagram for a Sensor Virtual Analog Resource Without Event Monitor	4-23
Fig 8 (c)	State Diagram for a Sensor Virtual Resource Without Event Monitor for Making a Measurement Using Single-Action and Multi-Action Verbs.....	4-24
Fig 9 (a)	State Diagram for a Sensor Virtual Resource With Event Monitor for Making a Measurement Using Single-Action and Multi-Action Verbs.....	4-25
Fig 9 (b)	State Diagram for a Sensor Virtual Analog Resource Without Event Monitor	4-26
Fig 9 (c)	State Diagram for a Sensor Virtual Resource With Event Monitor for Making a Measurement Using Single-Action and Multi-Action Verbs.....	4-27
Fig 10	ATLAS VERBS vs. ATE Model.....	4-29
Fig 11	Examples of Flow Chart Level	5-3
Fig 12 (a)	Example of Program Flow Diagram IF, THEN/ELSE Structure; Forces Separation of Main Tests.....	5-4
Fig 12 (b)	Example of Program Flow Diagram GO TO STRUCTURE; Diagnostic Separated	5-5
Fig 13	Examples of Delimiting Statements	5-6
Fig 14	Hypothetical Example of Complex Structure With Poor Readability.....	5-25
Fig 15	Test Loop 1 Control Statements Are Among Test Statements.....	5-26
Fig 16	Test Loop 2 Control Statements Are After Test Statements.....	5-27
Fig 17	Test Loop 3 Control Statements Are Before Test Statements	5-28
Fig 18	Task State Diagram.....	5-29
Fig 19	Schematic Diagram of Fuel Control System	6-2
Fig 20	Schematic Diagram of Electronic Vehicle Fuel Control, Type 4051-6.....	6-2
Fig 21	Test Flow Diagram of Sample ATLAS Procedure, Structured Programming Version...	6-6
Fig 22	Printout of Sample ATLAS Procedure of Fig 21.....	6-8
Fig 23	Test Flowchart of Sample ATLAS Procedure, GO TO Version	6-13
Fig 24	Printout of Sample ATLAS Procedure of Fig 23.....	6-14

TABLES

Table 1	High and Low Level Languages	2-2
Table 2	ATLAS Character Set.....	4-1
Table 3	Examples of Typical Decimal Numbers.....	4-3
Table 4	ATLAS Constants.....	4-8

TABLES	PAGE
Table 5 Task State Table	4-28
Table 6 Resource Allocation Table.....	5-17
APPENDICES	
Appendix A. ATLAS in Analog Testing	A-1
A1 Introduction.....	A-1
A2 Analog Systems.....	A-1
A2.1 Steady State Signals.....	A-1
A2.2 Modulated Signals	A-1
A3 Language Elements	A-1
A3.1 ATLAS Nouns Used in Analog Testing.....	A-1
A3.2 ATLAS Noun Modifiers	A-5
A4 Examples.....	A-12
A4.1 A General Alternating Current Power Supply.....	A-12
A4.2 A General DC Logic Level Power Supply	A-13
A4.3 A Ramp Deflection Signal.....	A-13
A4.4 A Noise Signal Which Can Be Used in Amplifier Testings	A-13
A4.5 A Square Wave Signal.....	A-13
A4.6 A Step Signal That May Be Used in Servo Testing	A-14
A4.7 Using Waveform a Simple Sine-Wave May Be Synthesized.....	A-14
Appendix B. ATLAS in Digital Testing	B-1
B1 Introduction.....	B-1
B2 Digital Systems	B-1
B2.1 Combinational Logic.....	B-1
B2.2 Sequential Logic.....	B-1
B2.3 Digital Test Techniques.....	B-2
B2.4 Data Field Manipulation	B-2
B3 Language Elements	B-4
B3.1 ATLAS Nouns Used in Digital Testing	B-4
B3.2 ATLAS Noun Modifiers Used in Digital Test	B-9
B3.3 Flexible Level ATLAS for Digital Testing.....	B-22
B4 Examples.....	B-27
B4.1 Logic Nouns.....	B-27
B4.2 Digital Test.....	B-28
B4.3 Dynamic Digital Test.....	B-29
Appendix BB. IEEE Std 716-1989 and ARINC Specification 626-1-1989 DIGITAL Examples ...	BB-1
BB1 ENABLE DIGITAL CONFIGURATION.....	BB-1
BB2 DEFINE DIGITAL SOURCE/SENSOR	BB-1
BB2.1 Parallel Data Configuration.....	BB-1
BB2.2 Serial Data Configuration.....	BB-1
BB2.3 Serial - External Specification.....	BB-2
BB2.4 Serial - Sync Bit Transition.....	BB-2
BB2.5 Illegal State Indicator.....	BB-3
BB3 STIMULATE.....	BB-3
BB3.1 STIMULATE - STATIC	BB-3
BB3.2 STIMULATE - TIMED	BB-3

APPENDICES

PAGE

	BB3.3 STIMULATE - REPEAT	BB-3
	BB3.4 STIMULATE - ZERO, ONE, HIZ	BB-4
BB4	SENSE	BB-4
	BB4.1 SENSE - STATIC	BB-4
	BB4.2 SENSE - TIMED	BB-4
	BB4.3 SENSE - HIZ	BB-4
BB5	PROVE	BB-4
	BB5.1 PROVE - STATIC	BB-4
	BB5.2 PROVE - TIMED	BB-5
BB6	DO TIMED DIGITAL	BB-5
	BB6.1 DO TIMED DIGITAL - ITERATE AND PROCEED	BB-5
BB7	DO SIMULTANEOUS	BB-5
BB8	Multiphase Clocks	BB-6
BB9	External Measure Strobe	BB-7
BB10	External Clock	BB-7
BB11	Concurrent Testing	BB-8
BB12	Stop on Mismatch	BB-8
BB13	Start on Match	BB-9
Appendix C.	ATLAS in Air Navigation System Testing	C-1
C1	Introduction	C-1
	C1.1 New Concepts	C-1
C2	DME	C-1
	C2.1 System Description	C-1
	C2.2 Language Elements	C-2
	C2.3 Examples	C-6
C3	ILS	C-7
	C3.1 System Description	C-7
	C3.2 Language Elements	C-8
	C3.3 Examples	C-11
C4	TACAN	C-12
	C4.1 System Description	C-12
	C4.2 Language Elements	C-13
	C4.3 Examples	C-19
C5	VOR	C-20
	C5.1 System Description	C-20
	C5.2 Language Elements	C-21
	C5.3 Examples	C-24
C6	ADF	C-24
	C6.1 System Description	C-24
	C6.2 Language Elements	C-25
	C6.3 Examples	C-27
C7	ATC	C-27
	C7.1 System Description	C-27
	C7.2 Language Elements	C-28
	C7.3 Examples	C-31
C8	IFF	C-32
	C8.1 System Description	C-32

APPENDICES

PAGE

	C8.2	Language Elements.....	C-32
	C8.3	Examples.....	C-36
C9		Air Data System Testing.....	C-36
	C9.1	System Description.....	C-36
	C9.2	Language Elements.....	C-37
C10		Example of a Complete Test.....	C-39
Appendix D. ATLAS in Gas Turbine Engine Testing.....			D-1
D1		Introduction.....	D-1
	D1.1	Scope.....	D-1
	D1.2	Use.....	D-1
D2		Test Discipline.....	D-1
	D2.1	Effect.....	D-1
	D2.2	Concepts.....	D-1
	D2.3	Instrumentation.....	D-2
D3		Language Elements.....	D-3
	D3.1	Description.....	D-3
	D3.2	Propulsion Pin Descriptors.....	D-3
	D3.3	Gas Turbine Testing Nouns.....	D-4
D4		Examples.....	D-8
	D4.1	Use of the TURBINE ENGINE DATA Noun.....	D-8
	D4.2	Use of the FLUID SIGNAL Noun.....	D-8
	D4.3	Use of the AMBIENT CONDITIONS Noun.....	D-9
	D4.4	Use of the VIBRATION Noun.....	D-9
Appendix E. ATLAS in Inertial Navigation System Testing.....			E-1
E1		Introduction.....	E-1
E2		Basic Inertial Navigation Principles.....	E-1
E3		Test Procedures.....	E-3
	E3.1	Sensor Tests.....	E-4
	E3.2	Electronic Tests.....	E-4
	E3.3	System Tests.....	E-4
	E3.4	Diagnostic Tests.....	E-5
E4		Stimuli and Measurement Requirements for Testing Navigation Systems.....	E-5
	E4.1	Stimuli Requirements.....	E-5
	E4.2	Measurement Requirements.....	E-7
Appendix F. Timing and Synchronization in ATLAS.....			F-1
F1		Introduction.....	F-1
F2		Identifying Elements.....	F-1
	F2.1	Signal Based Events.....	F-2
	F2.2	Event Based Events.....	F-4
	F2.3	Event Intervals.....	F-7
	F2.4	Event Indicator.....	F-8
	F2.5	Timers.....	F-8
	F2.6	Time Base Events.....	F-9
	F2.7	Event Monitoring.....	F-9
F3		Controlling the Test Procedure.....	F-12

APPENDICES

PAGE

F3.1	Delaying Test Execution	F-12
F3.2	Limiting Test Execution Time	F-14
F3.3	Synchronizing Parallel Test Sequences	F-15
F4	Controlling UUT Signals.....	F-19
F4.1	Controlling Single UUT Signal Actions	F-19
F4.2	Controlling Multiple UUT Signal Actions	F-26
F4.3	Time Interval Measurement.....	F-29
F5	Example.....	F-29
Appendix G.	ATLAS in Electro Optical Testing	G-1
G1	Introduction.....	G-1
G2	Language Elements	G-1
G2.1	LIGHT.....	G-1
G2.2	Noun Modifiers for the Noun LIGHT	G-1
G2.3	INFRARED	G-3
G2.4	Noun Modifiers for the Noun INFRARED	G-3
G2.5	LASER.....	G-5
G2.6	Noun Modifiers for the Noun LASER	G-6
G2.7	LASER TARGET RANGE	G-7
G2.8	Noun Modifiers for the Noun LASER TARGET RANGE	G-7
G3	Examples.....	G-8
G3.1	Examples for LASER.....	G-8
G3.2	Example for LASER TARGET RANGE	G-8
Appendix H.	ATLAS in Data Bus Testing	H-1
H1	Overview.....	H-1
H1.1	Example 1.....	H-1
H1.2	Example 2.....	H-2
H1.3	Example 3.....	H-2
H1.4	Example 4.....	H-3
Appendix I.	ATLAS in Video Testing	I-1
I1	Introduction.....	I-1
I2	Language Elements	I-1
I2.1	VIDEO SIGNAL	I-1
I3	Examples for VIDEO SIGNAL	I-8
I3.1	Group Modifier ACTIVE-AREA.....	I-9
I3.2	Group Modifier CONTENT	I-9
I3.3	Group Modifier FIELD-SYNC	I-9
I3.4	Group Modifier LINE-SYNC.....	I-9
I3.5	Group Modifier REF-LEVELS	I-9
I3.6	Group Modifier SCAN-FORMAT	I-9
I3.7	Examples of the Use of VIDEO-SIGNAL for Specific Screen Displays	I-10

APPENDIX FIGURES

Fig A1	Pulse Definition.....	A-4
Fig A2	Ramp Deflection Signal Example	A-13
Fig A3	Square Wave Signal Example	A-14

APPENDIX FIGURES

PAGE

Fig A4 (a)	Voltage Ramped.....	A-16
Fig A4 (b)	Voltage Stepped.....	A-16
Fig B1	Digital Test Illustration.....	B-2
Fig B2	Illustration of RZ, NRZ, and BIP Pulse Types for B'10110010'.....	B-16
Fig C1	Pulse Definition.....	C-5
Fig E (1)	Simple Inertial Navigator.....	E-2
Fig E (2)	Stable Platform Schematic Diagram.....	E-3
Fig F1	Signal Based Event.....	F-3
Fig F2	Event Based Event.....	F-4
Fig F3	EVERY n OCCURRENCES OF.....	F-4
Fig F4	AFTER EVERY M OCCURRENCES OF.....	F-5
Fig F5	EVERY n OCCURRENCES OF <event> AFTER m OCCURRENCES OF.....	F-5
Fig F6	Gating Mechanism.....	F-6
Fig F7	Delayed <event> Capability.....	F-6
Fig F8	<event> Based on Start or Stop of <event> Interval.....	F-6
Fig F9	Event Interval (Fixed Time).....	F-7
Fig F10	Event Interval Between Two <events>.....	F-7
Fig F11	Logical AND of Two Event Intervals.....	F-8
Fig F12	Event Monitoring.....	F-9
Fig F13	Task State Diagram.....	F-17
Fig F14	Sync Subfield.....	F-21
Fig F15	Timing of Test Situation.....	F-23
Fig I1	Diagrammatic Representation of Screen Display for 'AIMING-MARK'.....	I-12
Fig I2	Diagrammatic Representation of Screen Display for 'CROSS-HATCH'.....	I-13

IEEE Guide to the Use of the ATLAS* Specification

1. Introduction

1.1 For Whom This Volume Is Intended. This guide has been written as an applications manual for engineers and authors who will be writing test procedures in ATLAS. The beginning chapters are intended to provide managers with a brief introduction to the application of ATLAS. This guide is not, in any way, intended to be a substitute for an instruction manual in the ATLAS language, but, rather, a practical extension to such a manual.

1.2 Purpose of the Guide. The purpose of this guide is to provide principles of good practice in the use of ATLAS for the writing of test procedures. This advice has been drawn from practical applications in many organizations and a number of companies who are contractually committed to prepare test procedures in ATLAS.

Since ATLAS must be capable of communicating complex test requirements, the language has a large vocabulary and complex syntax. Therefore, it is not always readily understandable. The guide concentrates on the proper use of the ATLAS vocabulary, syntax, and structure for the generation of unambiguous test procedures. The guide assists in the preparation of good test procedures, as well as in the writing of computer programs which will efficiently process ATLAS test procedures for automatic implementation.

1.3 Organization of the Guide. ATLAS language elements are used by many disciplines across the testing spectrum. This guide has, therefore, been organized into a main text and several appendices.

The main text contains information and language elements which apply to all test disciplines. Each appendix contains information that applies to a particular test discipline. This information includes language elements and structures used in the particular test discipline with some examples and descriptions of usage. Thus, an engineer preparing a test procedure for air navigation equipment, for example, needs only to consult the main text and the air navigation appendix for guidance.

1.4 Related Documents. The material in this guide is applicable to IEEE Std 716-1989¹, ANSI/IEEE Std 416-1984², and ARINC Specification 626-1-1989³. Supplements to these standards will be issued periodically and the users guide updated accordingly.

1.5 Standards References. This standard shall be used in conjunction with the following publications.

¹ IEEE publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, Piscataway, NJ 08855-1331 USA.

² ANSI/IEEE publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, Piscataway, NJ 08855-1331 USA, or from the Sales Department, American National Standards Institute, 1430 Broadway, New York, NY 10018 USA.

³ ARINC publications are available from Aeronautical Radio, Incorporated, 2551 Riva Road, Annapolis, MD 21401, USA.

[1] ISO 646-1983, ISO 7-Bit Coded Character Set for Information Interchange.⁴

[2] ISO 1000-1981, SI Units and Recommendations for the Use of Their Multiples and of Certain Other Units.

1.6 Definition of Terms. These terms have the following meanings within this guide.

AEEC. Airlines Electronic Engineering Committee, the ARINC committee that originated ATLAS.

argand plane. A graphical representation of a vector used in complex notation.

ARINC. Aeronautical Radio Incorporated, an organization concerned with providing services to airlines including sponsorship of voluntary standardization among airlines and airframe and avionics manufacturers.

ASCII. American Standard Code for Information Interchange.

ATE. Automatic Test Equipment.

ATLAS compiler. A program that converts high-order ATLAS statements into executable machine code.

ATLAS vocabulary. The range of words and symbols used in standard ATLAS.

Backus-Naur form (BNF). A particular metalanguage developed by Backus and Naur.

Backus-Naur notation. A general term relating to metalanguages that use the concepts developed by Backus and Naur.

BCD. Binary-coded decimal.

Boolean. A labeled variable which can assume only two states, true or false.

delimiter. A character that provides punctuation in an ATLAS statement.

diagnostic test. A test applied to a UUT with the purpose of isolating a fault to a lower level of assembly.

field. A defined subdivision of a statement.

flag. The first character of an ATLAS statement used to mark that statement as having a special purpose or capability.

functional test. A sequence of tests applied to a UUT to establish whether it is functioning correctly.

global. Relating to the whole of an ATLAS test procedure.

IEEE. Institute of Electrical and Electronics Engineers, Inc.

ISO. International Organization for Standardization.

⁴ In the U.S., ISO Standards are available from the Sales Department of the American National Standards Institute, 1430 Broadway, New York, NY 10018, USA.

local. Relating to a bounded part of an ATLAS test procedure.

macro. A defined procedure or sequence of operations or characters which is inserted in the procedure each time its name is invoked.

metalanguage. A form of notation used to rigorously define the syntax and sometimes the semantics of another language.

MTE. Manual Test Equipment.

nonpreferred ATLAS. An ATLAS structure which has been rendered redundant by a later structure, but which is retained in the standard for upward compatibility.

semantics. The connotative meaning of words within an ATLAS statement.

SI units. Système International d'Unités; definitions of fundamental physical units and dimensions agreed to at the Eleventh General Conference of Weights and Measures, 1960 (Conference generale des poids et mesures).

syntax. The grammatical rules pertaining to the structure of an ATLAS statement.

test. An action or group of actions performed on a particular UUT to evaluate a parameter or characteristic.

test procedure. A document that describes the tests, test methods, and test sequences to be performed on a UUT to verify conformance with its test specification with or without fault diagnosis and without reference to specific test equipment.

test program. A test program implements the tests, test methods, and test sequences to be performed on a UUT to verify conformance with its test specification with or without fault diagnosis and designed for execution on a specific test system.

test specification. A document that defines the tests to be performed on a UUT to verify conformance with its performance specification and without reference to any specific test equipment or test method.

UUT. Unit Under Test; the entity to be tested. It may range from a simple component to a complete system.

variable. A quantity the value of which is assigned at program run time.

virtual resource. A notational test resource the performance characteristics of which conform to a summary of related test requirements.

2. The Need for a Test-Language Standard

2.1 Importance of Unambiguous Communication. Communication is a process of transferring data from one source to another. In human terms it involves the transfer of ideas, commands, and requests for data (questions) from one human being to another. It takes many forms by virtue of a wide variety of languages and an even wider variety of dialects within any one language. It is an observed and easily proven fact that human communications, regardless of the language used, are imperfect. Perhaps the simplest proof of this is the old parlor game of passing a message down a chain of several individuals; the message received at the far end almost always deviates in some degree from the original message and frequently bears no resemblance to that original message.

Human communication is best accomplished in a face-to-face environment where each word and phrase can fluctuate in its meaning and implication based on an almost unlimited variety of other factors. Environment, mannerisms, and intonation all play a key part in human communications. The one phrase that best typifies human communications is that it is a "do what I mean" type of communication rather than a "do what I say" version. In the world of industry, almost all problems that occur in the day-to-day operation of a company can ultimately be traced back to a lack of communication. Nowhere is this more clearly seen than in the world of automatic test equipment. This world revolves around a series of data transfers starting with a design specification for the unit under test, progressing through a variety of test specifications for that unit, and ending in a test program to ensure the unit's operability and assist it in its return to operation in the event that it has failed. This entire process can take place over a significant period of time and can embrace several different organizations. Given this environment, it should come as no surprise that automatic testing requires an explicit communications medium. It should be even less surprising that much emphasis has been placed on improving communications within this environment. Over the last decade these attempts have been centered on test-language standardization and has led to a focus on the use of the standard language designated ATLAS. This section is intended to show the reasoning behind the utilization of this language and the steps that were taken to bring it to its present point of development.

2.2 Types of Languages.

2.2.1 Person-to-Person Communication Language. In 2.1 the imperfections surrounding human communications have been discussed. Of prime importance is the realization that the "do what I mean" approach to communication is only viable in a direct contact situation between two individuals with sufficient common background to permit an interpretation of data elements outside the actual communications language used.

Languages used for human communications are known as "natural" languages. By this we mean that the language functions within a set of rules and is defined by its usage rather than its pure grammatical structure. Examples of this are seen in all elements of day-to-day communication. The use of slang, idioms, and fashionable expressions is a part of our everyday lives, and interpretation of these elements, which bear no resemblance to the pure language grammar, is also part of normal human communications. This loose structuring of the communications language becomes the chief reason for ambiguity when the communications vehicle is the written rather than the spoken word. The external elements of intonation, fashion, expression, and environment are now removed, and the recipient of the communication is left to make his or her own interpretation of the precise meaning of the communicate. In an attempt to compensate for this, sections of society with a need for precise communications have evolved their own particular languages which serve their needs, but which may be almost useless when viewed by members of the society outside of those particular segments.

It is well established that automatic testing also has this need. Communications are not restricted to the human level, but the ATE engineer and manager must also operate with the person-to-machine interface. This environment forms the backdrop against which ATLAS has been developed.

2.2.2 Person-to-Machine Communication Language. Despite their growing complexity, machines represent the very epitome of simple-mindedness. The "do what I mean" approach has no place in the machine world; a machine will universally obey the "do what I say" ground rule, even to the point of destruction of itself or its operator. All elements of communication with a machine, whether they originate from a human source or from another machine, are singly interpretable, and consequently, totally without ambiguity. This factor, of course, greatly simplifies machine-to-machine communication, but serves as a complicating factor in person-to-machine communication, since the person is not used to functioning in this environment. In an attempt to smooth the person's transition into this world of literal translation, languages have been constructed which closely resemble human communications languages, but which in fact represent meaningful statements to the machine. The prime difference now is that while the statements are meaningful, they convey only a single meaning to the machine.

All person-to-machine communication languages fall into one of two categories. These categories are frequently given different names, but here they will be called high-order language (HOL) and low-order language (LOL). A high-order language is sometimes referred to as a problem-oriented language or, in the automatic testing world, a test-oriented language. It is characterized by the fact that it restricts itself to simple statements of a problem and avoids specifying methods of solution to that problem. For example, the statement shown in Table 1 as a high-order language states that it is desired to solve the equation $a = b^3$. No mention is made of any method of solution, and consequently, the method is left entirely up to the machine. Since the statement is also non-ambiguous and since a machine is a repetitive environment, the method of solution will always be the same, regardless of its adequacy, for a particular situation.

A low-order language takes the reverse approach and ignores the problem itself, choosing instead to focus on the elements that make up a solution. By doing this, it allows the user to construct a wide variety of solutions to a particular problem, but at the same time it offers no guarantee that any of those solutions will be correct. It also restricts the solution of the problem to a particular machine.

Regardless of the correctness of the solution, the machine will faithfully implement the instructions given and generate an answer. In short, the flexibility to implement more than one solution to a problem has been traded off against the guarantee that the problem itself will be solved by the machine. Table 1 also shows a low-order representation of the $a = b^3$ problem. Note that here no mention is made of the problem and that the language is entirely oriented toward the solution on a particular machine.

Table 1
High and Low Level Languages

Problem: Calculate $a = b^3$	
Solution:	
High Level	Low Level
$A = B * B * B$ ($a = b^3$)	(1) $LL * B$ (load accumulator with B) (2) $MM * B$ (multiply accumulator by B) (3) $MM * B$ (multiply accumulator by B) (4) $SL * A$ (store accumulator in A)
A high-level language addresses machine inputs and outputs.	
A low-level language addresses the machine or machine element, or both.	

In the world of automatic testing, standardization has focused on high-order languages. When examined in detail, it becomes obvious that this focus is indeed the correct one. The desire is to communicate a sequence of events which must occur, and it is left to the recipient of the communications, whether it be a person or a machine, to ensure that those events are faithfully reproduced. Any attempt to communicate specific solutions will fail for a multitude of reasons. The solution itself may be incorrect or incomplete, or both, the solution may require access to equipment or expertise that does not exist or is not available to the recipient, and in general low-level solutions are considerably more complex and correspondingly more voluminous than their high-order alternatives. The loss of flexibility suffered by moving to a high-order language is more than justified by the increase in communications ability and the freedom from depending on specific tools to do the job.

2.3 The Test Language. The major stumbling block in test-information communication is the need to express testing functions concisely and unambiguously. This need cannot be met by the use of a "natural" language with its many problems and ambiguities. This is not to say that the specialist terms required for testing do not exist in the natural language, or even that they are ill-defined. The meanings of everyday words are too wide. Too many imperfect synonyms exist to guarantee that total information transfer will be effected.

The obvious solution was to suggest use of a natural language that already had wide usage and to subset and redefine this language for the specific purpose of test-data information communications. This was the driving force behind the birth of ATLAS.

At the time ATLAS was conceived, many artificial languages existed for machine communications, and many special languages existed for particular automatic test equipment. All these languages lacked the general-purpose approach applicable to the testing domain. It was felt that only by addressing the general-purpose issue could the needed international acceptance of the language be achieved. An artificial language was required based on a subset of a natural language. This subset was oriented toward the testing problem.

By virtue of its international acceptance, the natural language chosen was English. The obvious area of subset focus was that of avionics testing, since it was in that area that the bulk of the testing problems occurred. Hence, ATLAS was conceived as an avionics test language based on a subset of the English language.

Thus, ATLAS was conceived as both a person-to-person communication language and a person-to-machine communication language. As a person-to-person communication language it provides precise constructs, grammar, and terminology to communicate test requirement information unambiguously and concisely. It permits the exact description of a test procedure in terms and expressions sufficient to enable understanding by humans. ATLAS is also a high-order language for machine communications. It utilizes constructs that are carefully selected to be independent of particular solutions or particular equipment. This allows faithful implementation of an ATLAS test procedure in a variety of ways and on a variety of equipments, both manual and automatic. The realization of these goals is the key to the understanding of the ATLAS language, and the establishment of an ATLAS standard is the key to the continued use of the ATLAS language.

2.4 ATLAS Development.

2.4.1 General. At the request of the airlines, Aeronautical Radio Inc. (ARINC) began in 1965 to formulate a standard test language based on English. Sponsored by ARINC, this activity was regulated by the Airlines Electronic Engineering Committee (AEEC) whose principal function is to formulate standards for electrical equipment and systems for the airlines.

In 1976, ARINC transferred the control of the development of the ATLAS language to the Institute of Electrical and Electronics Engineers (IEEE). It was recognized that ATLAS could be used for testing beyond that applied to the airline industry. IEEE was to become the body to assume the further enhancement and standardization of the ATLAS language.

The historical development of the ATLAS language can be briefly summarized and tabulated in correlation to the Development Revision Levels which identified as hyphenated numbers to the ARINC Specification 416 document. This level numbering continued for the preparation of revisions to ANSI/IEEE Std 416-1984.

Each published version of ANSI/IEEE Std 416-1984 is identified by a hyphenated year number. The corresponding numbered development Revision Levels are included in parentheses in the table following the yearly identifier for the published versions of ANSI/IEEE Std 416-1984.

The historical development of ATLAS can be briefly tabulated as follows:

416 Issue	Year	Development
	1965-1966	A study of the need for a standard test language was initiated as the airlines were considering using automatic test equipment for maintenance testing.
	January 1967	The first AEEC subcommittee meeting was held to formulate a test language.
	August 1967-September 1968	Seven preliminary drafts of ATLAS Specification 416 were generated.
ARINC 416-1	October 1968	The final draft was approved by AEEC for issue as ARINC Specification 416-1.
2-5	1969-1972	Miscellaneous changes and refinements were introduced in the language which, until this time, only covered analog testing requirements.
6	Oct 1972	A digital testing capability was introduced.
7	July 1973	Some minor changes were introduced.
8	July 1973	A digital macro construct was introduced to augment the handling of digital testing.
9	May 1974	Miscellaneous refinements were incorporated.
10	May 1975	The most significant changes were the introduction of decision table and tasking capabilities. Decision table defines and initiates branching a test requirement which is dependent upon a complex relation between a number of variables. Task allows for the use of separate test sequences to be performed simultaneously.

- | | | |
|---------------|---------------|---|
| 11 | January 1976 | A structured programming capability was introduced which allowed a test requirement to be divided into self-contained blocks of test routines. |
| 12 | April 1976 | Miscellaneous refinements relating to complex numbers and wave form handling amongst others were incorporated. |
| 13 | August 1976 | Further miscellaneous changes were introduced. |
| 13A | August 1976 | The ARINC Specification 416-13 was reformatted with no changes to improve the interpretation and the management of the document. |
| IEEE 416-1976 | November 1976 | The reformatted version, ARINC Specification 416-13A was adopted by the IEEE as their launching version of the ATLAS language standard. |
| 1978 (14) | November 1978 | A capability for the defining of test resources as a requirement for testing was incorporated. |
| 15 | October 1979 | Miscellaneous refinements were introduced. |
| 1981 (16) | July 1981 | Further miscellaneous refinements were incorporated. |
| 17 | October 1980 | New syntax and semantics were introduced to provide for:

Defining signal from version; inclusion of ATLAS and NON-ATLAS program modules; defining protocol procedures and data exchanges; execution of this allowed data bus testing to be handled. New vocabulary introduced related to defining electro-magnetic fields; radar signals; ambient-conditions; fluid signals; turbine engine and vibration testing. |
| 18 | April 1981 | A number of ATLAS statements were introduced to identify events during testing to improve the handling of time-related testing. Further statements were introduced to augment to handling of tasking to: suspend and continue tasks; request and release tasks. Vocabulary to input and output devices.

A statement was added to separate the preamble and procedural sections of ATLAS test requirements. New constructs for handling inputs and outputs of test information were introduced. |
| 19 | October 1981 | The if-then-else structure was enhanced to include else-if. User defined functions were introduced. New digital constructs were introduced providing more flexibility for handling modern digital testing |

		techniques. An enable escape mechanism was introduced to facilitate error handling.
20	May 1982	Further miscellaneous refinements were incorporated.
21	November 1982	Introduction of state diagrams to correlate single-action and multi-action verbs. Some further miscellaneous refinements were incorporated.
22	May 1983	Vocabulary was introduced for Scorsby-like motion; visible light; infra-red radiation. Further requirements to the bus testing constructs and new vocabulary for video signal testing were introduced.
23	November 1983	Minor miscellaneous refinements were incorporated.
24	May 1984	Quantities and dimensions conforming to ISO 1000-1981 [2] ⁵ , SI Units and Recommendations for the Use of Their Multiples and of Certain Other Units (SI) were introduced. These were for use on nonaerospace and nondefense projects, although they are equally useful in the aerospace and defense industries. New vocabulary related to target signals was introduced.
25	November 1984	Some refinements to the old and new digital testing constructs were incorporated.
1984 (21)	December 1984	Published version aligned to Development Revision Level 21.
26	May 1985	Structured data types were introduced to provide more rigorous handling of basic data types; constants, variables and parameters. Refinements to the vocabulary relating to waveforms were incorporated.
27	November 1985	Refinement to sensor type statements were incorporated to allow for sampled measurements. An extend mechanism was introduced to facilitate the inclusion of new vocabulary into ATLAS test requirements. Some nonpreferred ATLAS statements were deleted. Some infrequently used and superseded were made nonpreferred.
28	May 1986	Refinements for bit synchronization were added to the new digital testing constructs. The first refinements to the definitions of the ATLAS vocabulary review were incorporated.

⁵ The numbers in brackets correspond to those of the references listed in 1.5.

31	November 1987	Further refinements to the definitions of the ATLAS vocabulary resulting from the vocabulary review.
31-33	May 1988-November 1988	Further refinements to the syntax and semantics of ATLAS arising from the creation of the subsets IEEE Std 716-1989 and ARINC Specification 626-1-1989.
		Further refinements to the definitions of the ATLAS vocabulary arising from the vocabulary review.
IEEE Std 716		
1982 (17)	March 1982	Contained ATLAS constructs for analog testing and digital testing using the old digital constructs.
1985 (18)	September 1985	Refinements to the syntax and semantics were incorporated.
1989 (30)	May 1989	The subset was revised to include: SI Units; structured data types; identification of events; new digital constructs; data bus testing constructs; requirement for test resources.

2.4.2 Constraints. In the selection of subset elements of English for an inclusion in ATLAS, constraints had to be considered. The first required that the general syntax be restrictive, and to this end the imperative or command tense was selected. This imperative format is:

DO, SOMETHING, WITH SOMETHING ELSE, SOMEWHERE \$

While the format itself is at least English-like, the punctuation differs considerably. The continual insertion of commas between statement segments or fields reflects a desire to make the statement more readable by machine. The selection of the currency symbol rather than the period (full stop) as a terminator is equally practical; the single-minded machine would have no way of readily differentiating between the ubiquitous decimal point used in engineering and the statement terminator, or period, used in normal everyday English. The currency symbol was selected as a non-engineering element that could be a reasonable terminator for a statement.

The early versions of ATLAS were directed toward the testing of analog electronic equipment, although some content was included for logic signals. There were 34 verbs, 30 nouns, and approximately 200 other words in the vocabulary of ATLAS 416-1. The number of verbs, nouns, and other words has increased considerably; additionally many of these may be prefixed, suffixed, and sometimes doubly suffixed to produce distinct compound words.

Additional verbs cover macro-function or a multi-action for digital test purpose, multiple decisions, and "structured" programming techniques. Additional nouns provide further definition of digital signals and air navigation signals, as well as non-electronic terms, which are reflected in the change of meaning of the second A in the acronym ATLAS from "AVIONIC" to "ALL". In the fullest mathematical sense, logic as well as arithmetic operations can be specified.

Another constraint surrounding ATLAS development was that, in line with the need for a clear definition by a computer, as opposed to a complex interpretation by a human being, strict yet simple language rules were required. The technique adopted to lay down these rules is one that is well accepted in the computer world, and yet almost unknown outside of this sphere. It involves the use of a formal definition language or, as it is frequently termed, a metalanguage.

The particular metalanguage used in the definition of ATLAS is known as the Dehmer and Frank form. The ability to understand this form and to use it is a totally unnecessary skill for almost all ATLAS users, but it is helpful for the computer specialist faced with the problem of designing an ATLAS translator for a machine. The first formal definition, or metalinguistic definition, of the ATLAS language occurred simultaneously with the thirteenth language issue, known as ARINC Specification 416-13A. Since that time this definition has been refined and expanded and is now given in the Formal Syntax section of the current ANSI/IEEE ATLAS Standards.

2.5 Continued Regulated Growth of ATLAS. Rapidly evolving technologies can cause obsolescence in a language before the language has even established a mature foothold. Nowhere is this more true than in the rapidly expanding and ever evolving field of electronics. The continued invention of new devices and new techniques requires a well reasoned but simple approach to developing language elements that will allow the transmission of new technology information. To meet this challenge, ATLAS is controlled and evolved through the auspices of an international committee of language experts. These experts serve this committee on a voluntary basis and work together to produce new language elements as needed. Users of the language, regardless of nationality, affiliation, or interest, may generate ATLAS problem sheets for consideration by this committee. These same users are encouraged to attend committee meetings to assist with the problem solution.

Meetings of the ATLAS Committee at which these problems and their proposed solutions are considered are organized at least once a year by the IEEE.

2.6 ATLAS as an Aid to the Management of Testing. ATLAS is not restricted to information presentation. The language is structured to enforce rules not only related to syntax and semantics, but also to overall specification presentation. ATLAS will control the format of the test information, the sequence of the test information, the implementation of testing decisions, and the identification of test elements. Furthermore, it will permit a test to be related to a specific test environment.

By controlling these elements and by eliminating the redundancy in information communication, a feature of natural languages, ATLAS can improve efficiency and add clarity in communications, whether person-to-person or person-to-machine. The benefits of clarity are well-known and can lead to improvement in performance and reduction in costs.

2.7 Benefits of a Standard. It is difficult to quantify the benefits of adopting a standard, in particular ATLAS, because few suppliers have either the time or the resources to conduct an experiment to measure the total effect. More often than not, the early benefits accrue directly to the purchasers and users, and the supplier initially sees only disadvantages. Formal training, on-the-job learning, and the acquisition of more detailed information increase the cost and time scale of the task. In the longer term, especially when the testing has been planned from the outset, the use of a standard test language in a formalized documentation system will show savings over a conventional free-range text, as well as providing a more complete test document. The introduction of ATLAS does not just impinge on the authors of test documentation, but also on the designers and in-house test departments. In many companies tradition, prejudice, and

demarcation have hindered the communication process, wherein each department has tried to retain its autonomy. ATLAS, because it is impersonal, has helped to remove some barriers which had existed previously.

The ultimate benefit of a standard can only be seen when it becomes universally adopted; ATLAS is close to this universal adoption now. As more and more projects use ATLAS on an international scale, the language itself will be refined to meet a larger proportion of total applications. With this refinement and with this extended usage, the nonrecurring costs associated with training and learning curves will be minimized, and the basic cost reductions associated with simplified communications and improved data transfer will become clear. In this sense, ATLAS is no different from any other standard.

3. ANSI/IEEE Std 416-1984, IEEE Std 716-1989 and ARINC Specification 626-1-1989

3.1 Definition of ATLAS. ATLAS, a standard abbreviated English language, is used in the preparation and documentation of test procedures or test programs which can be implemented either manually or with automatic or semiautomatic test equipment. The ATLAS language is defined in ANSI/IEEE Std 416-1984, IEEE Std 716-1989 and ARINC Specification 626-1-1989. These include a general or functional definition, as well as a formal definition, written in a metanotation.

3.2 ANSI/IEEE Std 416-1984, IEEE Std 716-1989, and ARINC Specification 626-1-1989. ANSI/IEEE Std 416-1984 can be regarded as the reference document from which the other documents are derived. It functions as a "working paper" for the IEEE SCC20 ATLAS Committee. It contains all elements in the ATLAS language including those introduced in the language between published versions. The current version of the user's guide is applicable to ANSI/IEEE Std 416-1984.

IEEE Std 716-1989, is a subset of ANSI/IEEE Std 416-1984, and contains those elements of the language that were deemed necessary by the IEEE Std 716-1989 Subcommittee of SCC20. This standard is published on a regular basis and the current user's guide is applicable to IEEE Std 716-1989.

ARINC Specification 626-1-1989, Standard ATLAS Subset for Modular Test, is a subset of ANSI/IEEE Std 416-1984. It contains those elements of ATLAS deemed necessary by the community of manufacturers and airlines for the testing of avionics in commercial aircraft. This version of the user's guide is applicable to ARINC Specification 626-1-1989.

3.2.1 General Definition. The general definition is provided by means of functional descriptions and rules applicable to ATLAS constructs. Cross-references to the formal syntax are provided. The text is augmented in appropriate places by diagrammatic representations of valid syntax. These diagrams are called syntax diagrams and provide guidance in constructing valid ATLAS statements.

3.2.2 General Characteristics of ATLAS. This section presents the user with information regarding the general characteristics of ATLAS, the levels of ATLAS, and the basic structure of an ATLAS test procedure. The following are the general characteristics of ATLAS.

(1) Unit under test orientation. The language is dedicated to defining the test requirements of the unit under test (UUT) without reference to or dependence upon the test equipment that may be used. The latter may be automatic, manual, or of a hybrid design.

(2) Unambiguous communication. The selective use of English language terms, which are compatible with the description of test requirements, and a formal structure for their use, constitute an environment which will ensure an unambiguous description of the requirements of a test procedure for the UUT designers, developers, users, and maintenance personnel.

(3) Test equipment independence. The specification of test requirements in terms of the UUT by ATLAS facilitates the transportability of test specifications from implementation on one set of test equipment to another, provided that all the test requirements can be satisfied by the test equipments.

3.2.3 Levels of ATLAS. Four levels of ATLAS are defined: standard ATLAS, subset ATLAS, subset ATLAS with Extensions, and adapted ATLAS.

(1) **Standard ATLAS.** The complete language, including vocabulary, syntax, and rules, as described in IEEE Std 716-1989, is defined as standard ATLAS. ARINC Specification 626-1-1989 has also been developed by ARINC for use by the commercial airlines and their suppliers. It is quite similar to IEEE Std 716-1989, but does also contain some unique language elements.

(2) **Subset ATLAS.** ATLAS has evolved as a broad based test language from the collective requirements of a wide spectrum of users. The language was used primarily in the areas of avionic and electronic testing (both analog and digital), but has now been extended to include several areas of nonavionic and nonelectronic testing.

Individual ATLAS users seldom require or implement the entire language because of the broad spectrum of coverage, language redundancy, and economic constraints of implementation.

Subset ATLAS is, therefore, a language of which every constituent construct is included within standard ATLAS (that is, no extensions) except that, for commercial or technical reasons, it does not include all the vocabulary or statements and syntactic options of standard ATLAS.

(3) **Subset ATLAS with Extensions.** A version of ATLAS which is based upon an ATLAS subset, as defined in (2) above, and which includes NON-ATLAS extensions appended to this subset. Specific rules for creating ATLAS subsets and appended extensions are provided in ANSI/IEEE Std 416-1984, section 1.3.

(4) **Adapted ATLAS.** ATLAS is a precise language used to communicate test procedures among various users. When it would be advantageous to modify the ATLAS language to aid implementation, ATLAS lends itself to adaptation. This modification is known as adapted ATLAS. Adapted ATLAS is the term used to refer to the ATLAS procedure after it has been modified to suit a local application.

Adapted ATLAS, then, identifies a family of languages which conform in some manner to ANSI/IEEE Std 416-1984, but which have modified vocabularies and possible syntax distortions. These languages are not intended for use with a standard ATLAS compiler.

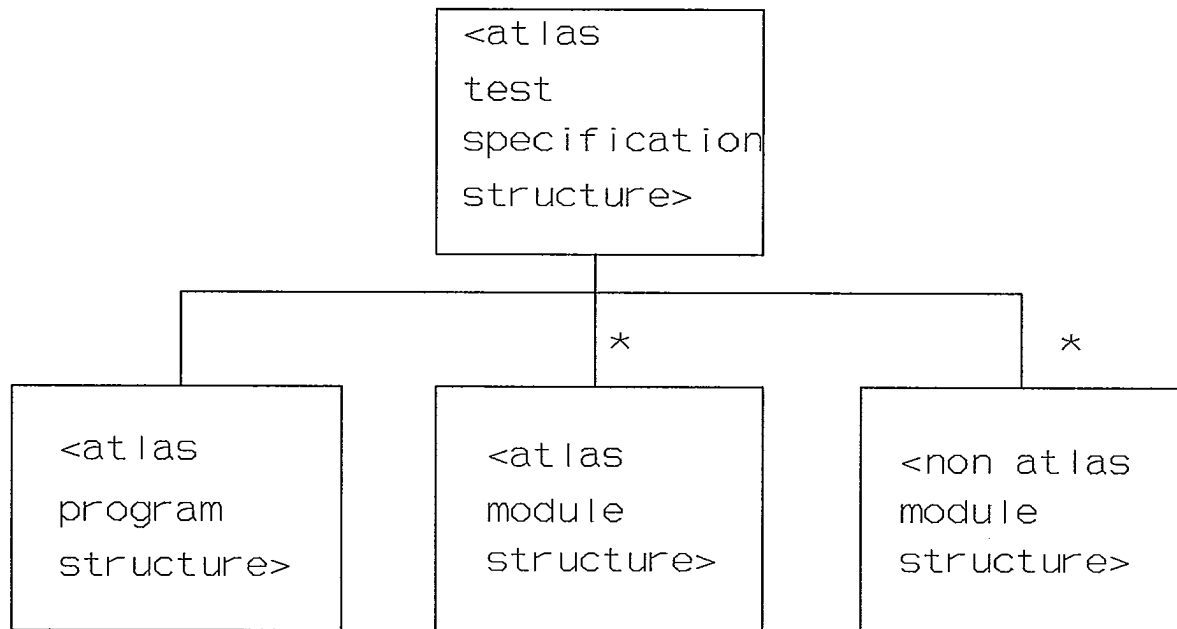
Because adapted ATLAS is not controlled by ANSI, IEEE, or ARINC, its use is discouraged.

3.2.4 ATLAS Test Procedure Structures.

3.2.4.1 General. An ATLAS Test Specification is structured into distinct groups which are defined herein. Figs 1 through 3 illustrate the structure of an ATLAS specification. These are an ATLAS Program Structure and optionally any number of ATLAS Module Structures and non-atlas module structures.

ATLAS and non-ATLAS modules exist as entities separate from the ATLAS Program Structure and are linked to that structure by INCLUDE statements in the main preamble structure. ATLAS and non-ATLAS modules provide a facility for preparing a large ATLAS Test Specification by using a number of smaller, manageable portions with a controlled interface. See Fig 1.

A syntax diagram (see Fig 2) is a diagrammatic representation of a level in the hierarchical structure of ATLAS. The title of the diagram is boxed by solid lines and is at the left-hand side of the diagram. Constituent elements and the manner in which they may be linked together are described in a notation similar to flow charting; these are written on the right-hand side of the diagram. The constituent elements may be either ATLAS reserved words (in uppercase) or other reserved characters such as \$ or /, or elements that require definition by a syntax diagram



Optional: There may be any number greater than or equal to 0

Fig 1
ATLAS Test Specification Family Tree

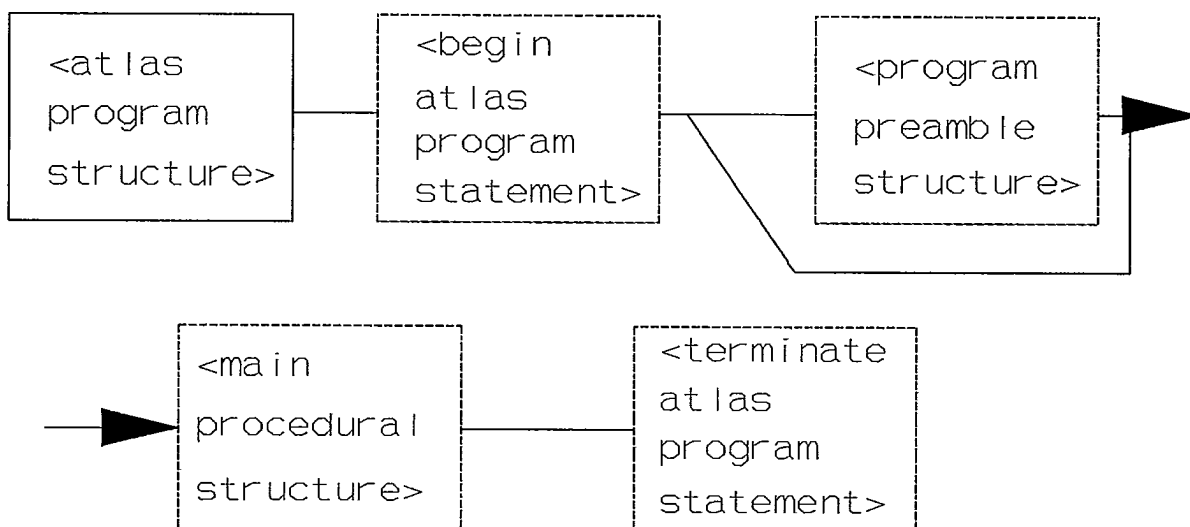


Fig 2
ATLAS Program Structure Syntax Diagram

	000100	BEGIN, ATLAS PROGRAM	\$
	05	FOR UUT,.....,MODS, ...	\$
	000200	REQUIRE,.....	\$
	.		
	000350	INCLUDE,.....	\$
	000400	DECLARE, ...	\$
	.		
	000450	FILL, ...	\$
	000500	DEFINE, ...,SOURCE, ...	\$
	.		
	000600	DEFINE, ...,PROCEDURE, ...	\$
	.		
		END, ...	\$
C		END OF PREAMBLE	\$
	100000	COMMENCE, MAIN PROCEDURE	\$
E	100001	APPLY, ...	\$
	.		
		MEASURE, ...	\$
	.		
		COMPARE, ...	\$
	.	IF THEN, ...	\$
	.	ELSE, ...	\$
E	200000	BEGIN, . . . BLOCK, . . . 'TIM' . . .	\$
	10	DECLARE, ...	\$
	20	DECLARE, ...	\$
	.		
	50	BEGIN, BLOCK, 'WAVE '	\$
	.		
		END, BLOCK, 'WAVE '	\$
		COMPARE, ...	\$
		END, BLOCK, 'TIM' . . .	\$
	.		
		FINISH, ...	\$
	.		
		TERMINATE, ATLAS PROGRAM	\$

P
R
E
A
M
B
L
EP
R
O
C
E
D
U
R
E

NOTE: FILL and COMMENCE are not in IEEE Std 716-1989 and ARINC Specification 626-1-1989.

Fig 3
ATLAS Program Structure

at the next lower level of the hierarchy. These latter elements are shown as dashed boxes containing the name of the element written in lower-case italic letters. A number is also included in the box to identify the paragraph in the Standard where that element is defined.

Flow lines which link or by-pass the constituent elements must be read from left to right, except where an arrowhead on a line indicates that additional passes may be made through the part of the diagram looped by this line. Where options exist, these are shown by parallel paths.

The main preamble structure precedes the main procedural structure and contains all the statements that predefine functions or declare types of variables for the ATLAS Test Procedure. Preamble statements do not cause any tests to be executed, but the information described in the preamble may be repeatedly referenced in the procedural section.

An ATLAS Test Procedure may contain a single main preamble, and each DEFINE, 'procedure', DEFINE, 'task', or DEFINE BLOCK may contain a single local preamble. In each case, the preamble serves to identify the name and attributes of a label which is subsequently referenced within the appropriate part of the procedure.

3.2.4.2 Program Preamble Structure.

(1) Function. The Program Preamble Structure, which appears immediately following the BEGIN, ATLAS PROGRAM statement, establishes labels that are global to the entire ATLAS Test Procedure Structure unless an identical label is established, at a lower level structure, within a local preamble structure. This preamble may contain FOR UUT, REQUIRE, INCLUDE, DECLARE, DEFINE, and FILL statements.

(2) Syntax diagram. A syntax diagram of a Program Preamble Structure is shown in Fig 4. The FOR UUT statement acts as an identifier in which relevant information on the UUT is given, such as part number, modification state, etc. It also establishes labels so that parts of the procedural section may be executed conditionally upon the part number and modification state of the UUT. (FOR UUT is not used in ARINC Specification 626-1-1989 or IEEE Std 716-1989.)

The second part of the Program Preamble Structure, the REQUIRE statement, is an optional implementation structure. An implementation structure is a series of statements which couple the UUT oriented and test equipment independent sections of the ATLAS Test Procedure (preamble and procedural structures) with the environment in which the test procedure must operate.

If the module facility is used, INCLUDE statements must be written to link the module documentation to the main procedure. Note that the preamble linking does not cause execution of these modules; the execution of a module is dependent on an appropriate PERFORM, 'procedure' or ENABLE, 'task' statement within the main program procedural section.

The DECLARE statement establishes labels and identifies specific storage spaces required, and the type of data to be assigned to each space.

The DEFINE statement establishes and names a section of the ATLAS specification that will be executed when the label is cited in a subsequent procedural statement. This section may be a series of statements, a single statement, or part of a statement.

By these means, sources, loads, and sensors which are applied or measured repeatedly can be defined and named once, and then, in the procedural section, they can be called by the use of

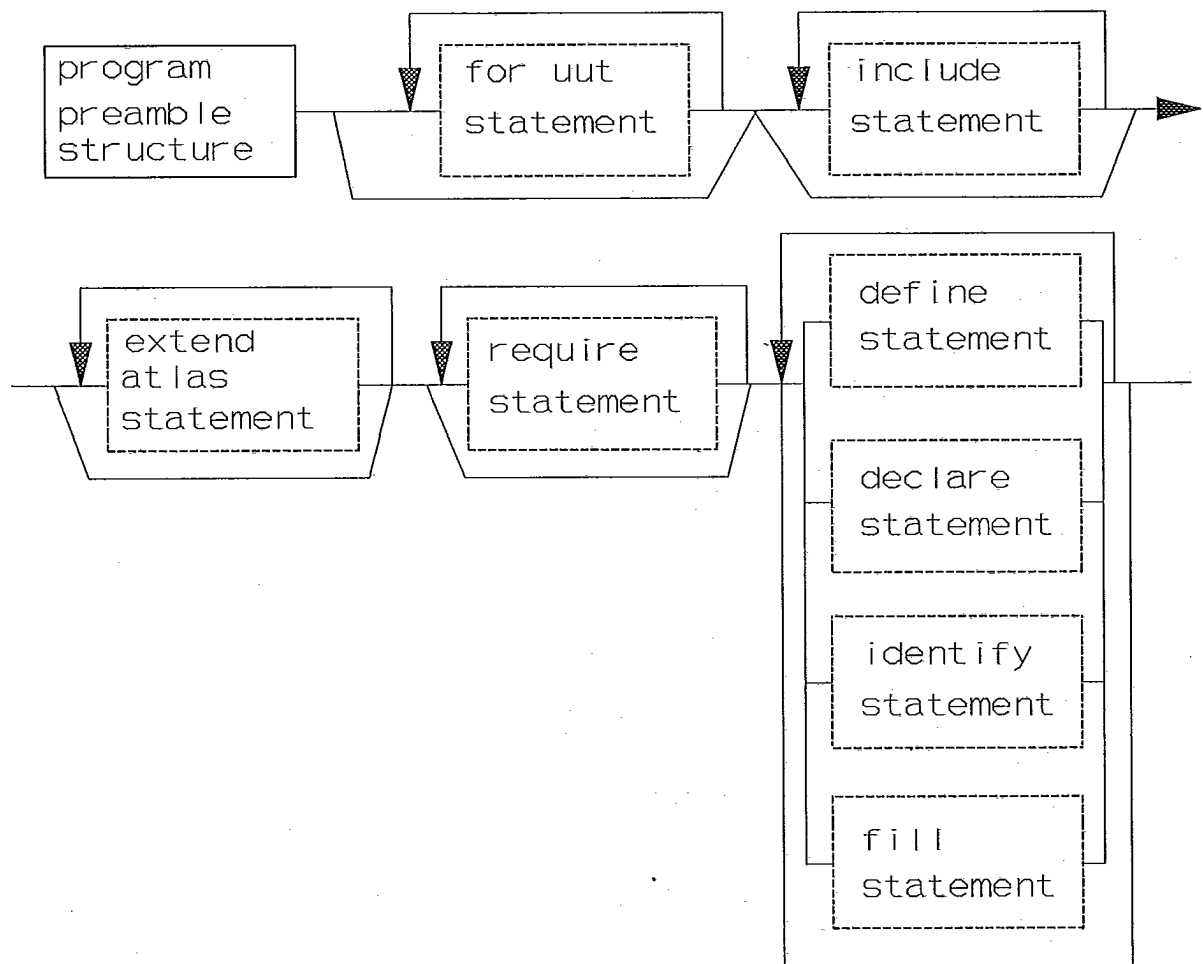


Fig 4
Program Preamble Structure Syntax Diagram

these names whenever required. Repeated messages can be handled in the same way:

Preamble

```
000020 DEFINE, 'DC-1', SOURCE, DC SIGNAL, VOLTAGE 5V, ... $
```

Procedure

```
100421 APPLY, 'DC-1', ..... $
```

is equivalent to:

```
100421 APPLY, DC SIGNAL, VOLTAGE 5V .....$
```

Complete tests, which are repeated several times, may be defined in a DEFINE, 'procedure' statement. In this case, the test is written as several complete separate statements following the

DEFINE, 'procedure' statement and is executed by the appropriate PERFORM statement in the procedural section.

The FILL statement is used to preload certain storage spaces with data that will be used during the execution of the procedure. (FILL is not used in IEEE Std 716-1989 or ARINC Specification 626-1-1989.)

Although there is no specific ATLAS symbol to denote the end of the main and a local preamble structure, it is good practice to indicate the end of a preamble structure by suitably worded commentary.

3.2.4.3 Main Procedural Structure. The Main Procedural Structure consists of a series of statements describing a portion of the required test. Each statement implies an instruction to proceed to the following statement immediately after completion of the present one, unless directed otherwise in a branching statement.

3.2.4.4 Basic Statement Structure. The elements of a general ATLAS statement structure are shown in Fig 5. The detailed format for each verb type is specified in the appropriate section of this document. There is no limit to the length of an ATLAS statement. It is not necessary to complete the statement in one line.

(1) General ATLAS statement structure. The writer can begin a new line after any complete language element 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 three fields (flag, statement number, and verb) must be written on the first row as required by the format.

(2) Flag field. The flag field designates different types of ATLAS statements.

(3) Statement number field. The statement number (statno) field is used to uniquely designate ATLAS statements.

(4) Verb field. The verb field identifies the test or data processing action for each statement. The ATLAS verbs are reserved words and must appear as the first word of a complete statement.

(5) 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 one or more spaces is designated for this purpose. Within some major fields, blank spaces are used as subfield separators.

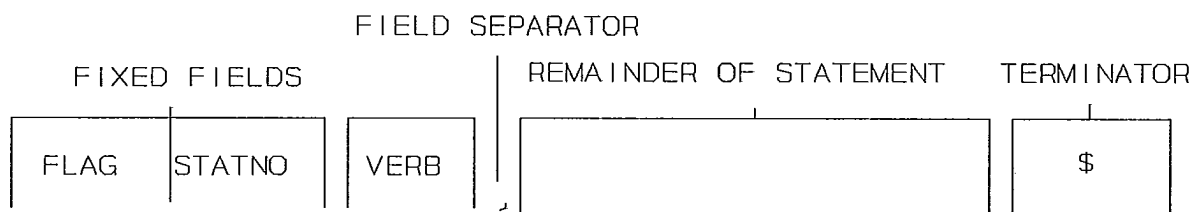


Fig 5
Basic Statement Structure

(6) Remainder of statement. This part of each particular type of ATLAS statement varies considerably. The detailed format following each verb type is specified in the appropriate section of the ATLAS standards.

(7) Statement terminator (\$). The last character of any ATLAS statement must be a currency symbol.

3.3 Formal Syntax Section. The Formal Syntax Section of ANSI/IEEE Std 416-1984 and IEEE Std 716-1989 was devised to give a very rigorous definition of the permitted structure of an ATLAS Test Procedure. It does this by employing an extension of the Backus-Naur notation to write syntax definitions in the form of "equations."

The Backus-Naur notation was invented by computer scientists in the 1950's to enable them to define exhaustively the permitted language structures of computer programming languages. It is known as a metalanguage because it is effectively a very limited language used to define more complex languages very rigorously.

The Formal Syntax Section defines what character sequences are permitted. Its treatment is more rigorous than that of the General Definitions Section, but it is not designed to give any indication as to the meaning of those character sequences. Although it is possible to make some assumptions from the English-language meanings about the meaning of the metalinguistic names, it is dangerous to do so.

ANSI/IEEE Std 416-1984, IEEE Std 716-1989, and ARINC Specification 626-1-1989 define both the structure of the language and its meaning (technically known as the syntax and the semantics, respectively). However, it has been found convenient to be a little imprecise in some cases in the syntax diagrams in order to make the diagrams more readable, supplementing the illustrations by rules to describe valid paths through the diagrams for particular circumstances. The Formal Syntax Section was added to present the permitted syntax in a formal machine processable form. It is presented separately so as not to impair the readily assimilated form of the General Definitions Section.

Most authors of ATLAS test specifications will find it unnecessary to refer to the Formal Syntax Section, but if a large number of specifications are to be written by a team, it is normally advantageous for one or two members of the team to understand the Formal Syntax Section to resolve any problems.

The material in the Formal Syntax Section takes precedence over that in the rest of the standard, since that section is a more rigorous treatment of the syntactic structure of ATLAS. ANSI/IEEE Std 416-1984, IEEE Std 716-1989, and ARINC Specification 626-1-1989 generate syntax diagrams from the formal syntax to greatly reduce the possibility of errors. This does not preclude the possibility of errors in the Formal Syntax Section. Should anyone encounter any inconsistency, he or she should highlight the problem in the form of a change proposal to the IEEE ATLAS Committee.

3.4 NON-ATLAS Usage. If, when writing an ATLAS test specification, the ATLAS language as defined in ANSI/IEEE Std 416-1984, or IEEE Std 716-1989 does not adequately cover a particular test discipline or problem, the NON-ATLAS text may be provided as a NON-ATLAS Module linked to the ATLAS specification by the INCLUDE statement in the main preamble and executed by an appropriate PERFORM in the procedural section.

It is recommended that the NON-ATLAS text be written in a statement format similar to ATLAS, and that the author submits to the IEEE ATLAS Committee the problem that he or she has encountered along with a possible solution. In this manner, ATLAS will continue to be upgraded.

4. The Elements of ATLAS

4.1 Introduction. Like any other language, ATLAS has a set of symbols or characters that may be used singly or collectively in a prescribed order to convey meaning. This section describes the characters and their uses and the "words" which may be formed by linking characters.

4.2 The ATLAS Character Set. The following table represents the most current printing ATLAS character set:

Table 2
ATLAS Character Set

Characters	Usage
Uppercase letters: A through Z	Keywords
Lowercase letters: a through z	Character string data
Numerals: 0 through 9	Integers/constants
Symbols:	Delimiters/punctuation
,	Comma
	Blank space
*	Multiplication and exponentiation sign, asterisk
.	Period, decimal point
=	Equal sign
+	Plus sign
-	Minus sign, hyphen
/	Division or lowercase sign, slash
(	Left parenthesis
)	Right parenthesis
'	Apostrophe or single quote
!	Exclamation point
"	Double quotes
#	Number sign
%	Percent sign
:	Colon
;	Semicolon
<	Less than
>	Greater than
@	At sign
[	Left bracket
]	Right bracket
\	Backslash
^	Circumflex
_	Underscore
{	Left brace
}	Right brace
	Vertical line
~	Tilde
&	Ampersand
?	Question mark
\$	Currency character
	Statement terminator

NOTES: (1) In handwritten ATLAS, O (alphabetic character) and 0 (zero) are separate characters. A slash should be used to distinguish the zero character.

(2) Where no lowercase alphabet is available in a subset, and an alpha character in a manufacturer's pin designation is required to be used literally as lowercase, a slash is written immediately preceding each affected letter. For example,

socket 1-bb
would be written as
SKT1-/B/B

(3) Blank spaces are used to improve readability and to separate words within some language elements. In addition, one blank space is required between words within any language elements containing two or more words. Blank spaces adjacent to all special characters that are not part of a language element (such as the , +, -, \$), and between alphabetic and numeric subfields, are optional.

(4) Although no particular character coding is specified, it is recommended that the USA (1) version of the ISO standard be employed. In this case, statement termination should use the US currency symbol (\$) which is encoded as octal value 044 in 7-bit code; otherwise, problems may occur in decoding coded material.

4.3 Keywords (Section 8). A keyword is any single character or sequence of characters recognized as being an element of ATLAS by inclusion in the glossary of ATLAS words.

Every verb, noun, modifier, dimension, and uppercase word in the ATLAS syntax diagrams is a keyword.

4.4 ATLAS Number Conventions

4.4.1 General. Test procedures may require the inclusion of several different number types for different purposes. ATLAS provides the following number types:

(1) Decimal numbers. Further subdivided into integers; fixed-point and floating-point forms

(2) Digital numbers. These are bit strings which may be integrated as holding various forms of binary and binary coded decimal

(3) Statement numbers

(4) Complex numbers

(a) Complex-polar. Complex numbers in the angle and magnitude forms

(b) Complex-rectangular. Complex numbers in the real and imaginary component forms

These number types are discussed in more detail in 4.4.2 through 4.4.5.

4.4.2 Decimal Numbers and Decimal Integers. Decimal numbers are the most familiar form of numbers in general use. Basic decimal numbers are written in ATLAS using the normal mathematical form, digits 0 through 9. They may or may not be preceded by a sign character (+ or -). The absence of a sign character implies a positive sense (+). A period (full stop) is used to indicate the position of the decimal point.

In other circumstances, such as the index of a list, it is essential to use an integer. As in normal mathematics, an integer is written without a decimal point. Good style requires that if a decimal point is written, a fractional part should also appear (i.e., 4.33).

Decimal numbers may be written in floating-point notation where the value is defined by a multiplier (the argument) and a power (the exponent). Thus 3E10 (or +3.0E10) expresses in ATLAS the value 3×10^{10} .

An example of the many floating-point representations of the value 1234.5 include:

1234.5E0 123.45E1
12345E-1 1.2345E3
0.12345E+4

Table 3
Examples of Typical Decimal Numbers

	Number	Sign	Type
(a)*	16393	Signed or	Integer
(b)	0	unsigned	Integer
(c)	-16393	Signed	Integer
(d)	1639.3	Signed or	Decimal Numbers
(e)	0.16393	unsigned	
(f)	-16.393	Signed	Decimal Numbers
(g)	+16393	Signed	Decimal Numbers
(h)	-0.16393	Signed	Decimal Numbers
(j)	+16393.	Signed	Integer

* Example (a) is equivalent to +16393. Example (j) is included as an example of legitimate, but poor ATLAS.

Table 3 above shows some typical decimal numbers. All these are equally acceptable in ATLAS. However, it is common practice to restrict usage so that all floating-point numbers have a common format. This practice is known as normalization. The most common normalization rules are as follows:

- | | | |
|-----|-------------------------------------|---|
| (1) | $1 < A < 1000$
E a multiple of 3 | This is effectively the basis of the SI system of units with standard prefixes (giga-, mega-, kilo-, milli-, micro-, and so on). (See ISO 1000-1981 [2]). |
| (2) | $1 < A < 10$ | Commonly used in the scientific community. |
| (3) | $0.1 < A < 1$ | Commonly used by the computer community because it is normal to work in fractional arithmetic. |

NOTE: A in items (1) through (3) represents the argument and E the exponent.

The rules for number construction are detailed in ANSI/IEEE Std 416-1984, IEEE Std 716-1989, and ARINC Specification 626-1-1989. Deviations, even if they are common national practices such as use of a comma for the decimal point or to separate integral and fractional parts into groups of three, are forbidden. Similarly, there must be no spaces anywhere within a number or even between the sign and the digits of the number.

4.4.3 Digital Numbers.

4.4.3.1 General. Digital numbers are used in association with two-state circuits and thus based on numbers with a radix 2. Such numbers are called binary numbers. The digits of a binary number take the values 0 and 1.

Long strings of 1's and 0's tend to be incomprehensible, inconvenient, and error prone. The use of radices that are powers of 2 reduces the size of the numbers while retaining the ability to distinguish single bit values. Radices of 8 (octal system) and 16 (hexadecimal system) are used. In the hexadecimal system, the representations for the decimal values 10 through 16 are the letters A to F.

Since the digits which occur in binary also occur (but with different significance) in decimal, octal, and hexadecimal systems, it is necessary to distinguish the various number types. This is done by enclosing the digital numbers in single quotes and preceding them by a prefix letter to indicate the type of digital number. The letters used are:

B for Binary
O for Octal
X for Hexadecimal

For example,

B'11110000110101101'
O'360655'
X'1E1AD'

all of which represent the same binary pattern with the decimal value 123309 if the bits are given integral binary weighting. In ANSI/IEEE Std 416-1984 and ARINC Specification 626-1-1989, octal and hexadecimal representations may include a digit between the key letter and the first quote to indicate a smaller number of bits and the most significant character position.

Bits in digital numbers can be given a variety of weightings. Some weightings are commonly encountered, in particular binary weighting, with the radix point at any point in the binary string. Another weighting is binary coded decimal (BCD). Here the binary bits are grouped into quadruplets with each group representing the value of a decade. Conversions between binary and decimal numbers are handled by mathematical functions. Standard functions are provided for binary and BCD conversion. In ANSI/IEEE Std 416-1984 and ARINC Specification 626-1-1989, there is a facility to define other functions for different weightings.

There is one abnormal number type which does not comply with the normal radix definition given above. This is the binary-coded-decimal (BCD) form. Here the binary bits are grouped into fours, with each group representing the value of a decade.

As digital numbers are used primarily for testing digital UUTs, it is necessary to take account of the peculiarity of digital circuits when defining digital numbers.

Even when a digital circuit takes a value, it is not always possible to predict which it will take, either because of inadequate information about the history of the device or, because of possible unpredictable pulses in the circuit. In this case, it is necessary that the particular bit be indicated as being a "don't know" or "don't care" bit. This is indicated by an X.

From the preceding, it will be seen that the legitimate digits for each number type are as follows:

Binary	0, 1, X
Octal	0, 1, 2, 3, 4, 5, 6, 7, X
Decimal	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X
Hexadecimal	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, X

Many UUT's require that the digital words or fields received by them be self-checking by inclusion of a parity check bit. When the stimulus VALUE is a dependent variable it is not possible at the time of writing the procedure to allocate a specific '1' or '0' as the parity check bit. Depending on whether odd or even parity checking is required the function PO or PE may be used to generate the parity 'field' in a digital word stimulus.

4.4.3.2 Digital Number Descriptors. Because a binary digit has only two possible information states, 0 and 1, it is necessary to define a convention that permits a logic bit to define whether a number is positive or negative. If it is negative, it is also necessary to define the convention by which the numeric value is represented. The number type and the negative convention are defined as parameters of the conversion function by digital number descriptors. The ATLAS digital number descriptors are:

BNR	Binary Number. Pattern with no sign bit and no negative form.
BSM	Binary Sign and Magnitude. The most significant bit is a 0 for positive and a 1 for negative. The remaining bits are effectively a BNR and are the same for both positive and negative.
B1C	Binary Ones Complement. The most significant bit is a sign bit with the same convention as BSM. The remaining bits are all inverted for negative numbers so that a 1 in a positive number is a 0 in a negative number, and vice versa. It has one main disadvantage: there are two different forms for 0 (that is, 00000000 and its complement 11111111).
B2C	Binary Twos Complement. The negative form is formed by adding a 1 in the least significant position to the B1C negative form. This results in only one form for 0 (00000000). It is the most common form for use in digital computers since it most conveniently fits in with simple arithmetic devices.
BCD	Binary Coded Decimal. The unsigned form has one group of four bits for each decade. The signed form has one extra bit which is the most significant and is set to 0 for positive and 1 for negative.
n ISO	A character from the n-bit ISO code (n is either 6, 7, or 8) described in ISO 646-1983 [1].
n ASCII	A character from the n-bit ASCII code (ASCII is the ISO code with US national deviations. ASCII: American Standard Code for Information Interchange).

4.4.4 Statement Numbers. Statement numbers are number forms in their own right, namely, effectively unsigned decimal integers of restricted length. These are used solely for identifying statements in an ATLAS procedure. The number is normally six digits long. The leading four digits are called the test number, and the final two digits the step number. The test number may be replaced by four blank characters if the test number of a statement is the same as that of the previous statement; otherwise, when written, it must have four digits and include leading 0s as fillers when required. Although it is not mandatory to use statement numbers it is recommended practice to do so. When the statement number is omitted it must be replaced by six blank characters.

4.4.5 Complex Numbers. There are some quantities that are conveniently expressed in complex numbers. Complex numbers are represented as points in the Argand plane. Two forms of numbers are available for expressing these values, complex-polar and complex-rectangular. They both are comprised of two signed decimal numbers linked either by the operator/ or the operator J. The use of one form or the other must be consistent throughout a statement except as authorized by the rules of the verb CALCULATE.

(1) Complex-polar. In this notation, a point in the Argand plane is represented by its distance from the origin and the angle between the vector joining the point to the origin and the positive real axis.

A point at a distance R from the origin at an angle Q is expressed as R/Q.

(2) Complex-rectangular. In this notation, a point in the Argand plane is represented by the distance along the real and imaginary axes, where they are crossed by lines through the point orthogonal to the axes.

A point at a distance X along the real axis and Y along the imaginary axis is represented by $X + JY$, if Y is positive, and $X - JY$, if Y is negative.

For example, a point which is +2.0 along both the real and the imaginary axes is represented as

2.8284/45 DEG in complex-polar

and

2.0+J2.0 in complex-rectangular

If the imaginary component were negative, it would be represented by 2.8284/-45 DEG and 2.0- J2.0, respectively.

4.5 User-Formed Strings.

4.5.1 General. ATLAS language rules provide a facility for users to create character strings, which are not listed as keywords, for purposes that are determined by the context in which they are to be used. Constants, variables and types are considered in this section.

Any number of characters from a relevant subset of the ATLAS set can be linked to make a user-formed string. Subsets are chosen to avoid delimiting problems in the context in which the string is used. Obviously, a user must not introduce ambiguity by forming identical strings with different meanings. A difference in one character within the first sixteen characters of the string guarantees uniqueness (unless that difference is a blank space which is considered to be insignificant in variables and defined names). Within the connection field, a blank space is very significant since it is the separator between connection designators.

Names created by the user, apart from connection designations, are called labels and must be enclosed within single quotes.

Examples of user-formed names are:

'THREE PHASE POWER 115V'	}	Identical up to the 17th character.
'THREE PHASE POWER 120V'		
'THREE PHASE POWER 200V'		Unique at the 16th character
'AC POWER'	}	Considered identical
'ACPOWER'		
'AC-POWER'		Hyphen ensures uniqueness
SKT-1-5		Connection designation. No spaces are permitted.
'INPUT-PIN'		Connection label. This is a CONN variable, not a connection designation.

Uniqueness should be established as early as possible in the string, for example,

'115V THREE PHASE POWER'
'120V THREE PHASE POWER'

differ at the second character instead of the seventeenth character as in the earlier example.

One reason for allowing user-formed strings is to improve the readability of the test documentation. It is recommended that meaningful labels, neither too abbreviated nor too lengthy, be used. ATLAS module structures may be referenced by more than one program. Therefore, care is essential in the choice of labels and their categorization as **EXTERNAL** in module preambles to avoid duplication of label names having global application.

4.5.2 Constants. A constant is the name chosen by the test writer to describe an implementation-time location which holds numerical, literal, connection or enumeration data for use in signal orientated, computational, logical, or input/output operations. The value of the constant does not change during the course of the test.

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. Table 4 shows the types that can be used for constants.

Constants must be declared in either the main preamble or a local preamble. Constants which are declared in different preambles can use identical labels. The span during which a constant is valid is determined by the type of preamble. Constants declared in the main preamble are said to be global and access to them can be made from any subsequent statement except when the same label has been declared in a local preamble. In this case, the local constant takes precedence in those procedural statements directly or indirectly associated with the local preamble containing the identical label. Use of the constants and variables in this way is not preferred.

The following shows some examples of declarations of constants:

DECLARE, CONSTANT, 'PI' IS 3.141 \$
DECLARE, CONSTANT, 'MESSAGE' IS 'THIS IS A MESSAGE' \$

Table 4
ATLAS Constants

Type	Contents
INTEGER	A base 10 numeric value which has no fractional part
DECIMAL	A base 10 fixed point or floating point number which may have a fractional part
LONG-DECIMAL (used in IEEE Std 716-1989 only)	A base 10 fixed point or floating point number which is beyond the range or precision for DECIMAL
BIT	A digital number specifying one bit
STRING OF BIT	A digital number specifying more than one bit
CHAR	A character string specifying one character
STRING OF CHAR	A character string specifying more than one character
ENUMERATION	Enumeration elements are constants ordered by the sequence in which they are written into the declaration
CONNECTIONS	A <connection>, COMMON, EARTH, or ATMOS preceded by the word CONNECTION

4.5.3 Variables. Similar to constants, variables are names chosen by the test writer to describe implementation time locations holding numerical, literal or logical data for use in signal orientated, computational, logical or I/O operations.

The types associated with variables can be divided into three main classes: Base-type, Predeclared type and Structured type.

4.5.3.1 Base Type. The base type is the simplest type a variable can assume. The following types are classified as base type:

INTEGER	A base 10 numeric value which has no fractional part
DECIMAL	A base 10 fixed point or floating point number which may have a fractional part
LONG-DECIMAL (used in IEEE Std 716-1989 only)	A base 10 fixed point or floating point number which is beyond the range or precision for DECIMAL (NOT USED IN ARINC Specification 626-1-1989, BUT IS INCLUDED IN IEEE Std 716-1989)
BIT	A digital number specifying one bit
CHAR	A character string specifying one character
ENUMERATION	Enumeration elements are constants ordered by the sequence in which they are written into the declaration
CONNECTION	A <connection>, or a number of <connection>s preceded by the word CONNECTION

Examples of declarations of base type variables are shown below:

DECLARE, VARIABLE, 'NUMBER' IS DECIMAL \$

DECLARE, VARIABLE, 'PARITY' IS BIT \$

DECLARE, VARIABLE, 'PIN1' IS CONNECTION(J1) \$

DECLARE, VARIABLE, 'DIGIT' IS ENUMERATION
('1','2','3','4','5','6','7','8','9','0') \$

4.5.3.2 Predeclared Type. There are three predeclared types: BOOLEAN, DIG-CLASS, and CHAR-CLASS. These exist as though each ATLAS program contained an enumeration type declaration (see 4.5.4). For BOOLEAN these enumeration elements are:

DECLARE, GLOBAL, TYPE, 'BOOLEAN' IS ENUMERATION
('FALSE','TRUE') \$

For DIG-CLASS these enumeration elements are:

'BNR','B1C','B2C','BSM','BCD' and 'SBCD'

and for CHAR-CLASS they are:

'ASCII7' and 'ISO7' (NOT IN ARINC Specification 626-1-1989)

Examples of declarations for variables of the predeclared type are shown below:

DECLARE, VARIABLE, 'COMPARISON' IS BOOLEAN \$

DECLARE, VARIABLE, 'A' IS DIG-CLASS \$

There are two ways of assigning one of the enumeration element values to the variable:

- (1) By means of the INITIAL field one can assign a value in the preamble.
- (2) By means of a CALCULATE statement anywhere in the test program.

An example of the first method would be:

DECLARE, VARIABLE, 'COMPARISON' IS BOOLEAN INITIAL=FALSE \$

An example of the second method would be:

CALCULATE, 'A' = BCD \$

4.5.3.3 Structured Type. There exist four structured types:

STRING OF BIT/CHAR, ARRAY, RECORD OF, and FILE OF.

STRING OF BIT is used for variables that represent digital numbers. It specifies the number of bits the digital number may occupy. An example of a declaration using this type is:

DECLARE, VARIABLE, 'ADDRESS' IS STRING (8) OF BIT \$

A later CALCULATE statement can assign a value to 'ADDRESS':

CALCULATE, 'ADDRESS' = X'4F'

ARRAY is used to declare a number of memory locations. A possible declaration using this type could be:

```
DECLARE, VARIABLE, 'DATA' IS ARRAY (1 THRU 10)
  OF STRING (8) OF BIT $
```

This would declare ten memory locations of eight bits each. Individual array elements can be assigned values in CALCULATE statements as follows:

```
CALCULATE, 'DATA'(1) = B'10011001',
'DATA'(2) = B'10010010'.
      etc.
```

It is possible to declare multidimensional arrays as follows:

```
DECLARE, VARIABLE, 'MARRAY' IS ARRAY (1 THRU 10, 1 THRU 5)
  OF ARRAY (1 THRU 5) OF STRING (8) OF BIT $
```

(This example is nonpreferred)

This would declare a two-dimensional 10 x 5 array of memory locations each one eight bits long.

RECORD OF is used to declare a collection of data items that need not be of the same type or structure. Individual fields within a record are given field names. An example of a record declaration could be:

```
DECLARE, VARIABLE, 'DATE' IS RECORD OF
  ['DAY', 'MONTH', 'YEAR' IS INTEGER] $
```

Individual fields of the record can be assigned values as follows:

```
CALCULATE, 'DATE'. 'MONTH' = 8,
'DATE'. 'DAY' = 12,
'DATE'. 'YEAR' = 1989 $
```

This statement would specify the date August 12, 1989.

FILE OF is used to declare a collection of data elements where each item has a position in the FILE beginning with zero and advancing by one with each subsequent data element. There are three classes of file:

- (1) A known type file consists of a sequence of data items all of which have the same type.
- (2) An untyped file consists of bytes of data which could represent anything. Input and output for this type of file requires a detailed knowledge of the data format, which may be very machine dependent. (This is not used in ARINC Specification 626-1-1989).
- (3) A text file consists of a stream of characters which equate to declared data types, after possible automatic I/O conversion.

4.5.4 Types. In addition to variables and constants it is possible to declare types. Type declarations are used to avoid repetition of identical descriptions of data structures in VARIABLE declarations.

For instance, suppose two digital variables 'ADDRESS' and 'DATA' are declared as follows:

```
000100    DECLARE, VARIABLE, 'ADDRESS' IS ARRAY(1 THRU 10)
           OF STRING(8) OF BIT $
000200    DECLARE, VARIABLE, 'DATA' IS ARRAY(1 THRU 10)
           OF STRING(8) OF BIT $
```

It would be better to declare a type 'DIGNUM' as follows:

```
001000    DECLARE, TYPE, 'DIGNUM' IS ARRAY (1 THRU 10)
           OF STRING(8) OF BIT $
```

and then to declare 'ADDRESS' and 'DATA' as being of the type 'DIGNUM' :

```
001100    DECLARE, VARIABLE, 'ADDRESS','DATA' IS 'DIGNUM' $
```

4.5.5 Defined Names. Where test information is used repeatedly throughout the test procedure, ATLAS provides a method of writing this information once in the preamble. The information is allocated a name by means of a DEFINE statement, the name being recognized from then on as a label. Calling that label during the remainder of the test procedure accesses all the test information allocated to it.

Characteristics associated with a label of a DEFINE statement must be applicable to the syntax and semantics of any statement which subsequently references that label.

Although any arrangement of alphanumeric characters can be used as a label, it has been found useful to choose labels that are meaningful to the human reader. Of course this is of no consequence to the machine, but for the reader it helps make the interpretation of the procedure easier. For example,

'GAIN-CHECK', 'POWER-ON', 'DC-1'

where the ATLAS test procedure is so extensive that it is generated by more than one user, it is considered prudent to exercise overall label control. This ensures that when only uppercase letters are available in a subset, all sections of the final procedure use the same labels for the same variables. Again this may be unimportant to a machine, but it does alleviate confusion for the human interpreter. It is not a difficult task to ensure uniqueness of labels, even if it is only achieved by the use of a prefix or suffix. When organizing overall label control it is advisable to cover all labels in both DECLARE and DEFINE statements.

4.5.6 Connections. Connections on the UUT and on special hardware defined in DEFINE drawings obviously have descriptions assigned by the manufacturer. In ATLAS statements reference will be made to the connections in the CNX and terminal fields. Each description of a connection will be an alphanumeric string without a space between any of the characters. The apostrophe, currency symbol, comma, left and right parentheses, and equal sign are forbidden characters. When only uppercase letters are available in a subset, lowercase letters are represented by preceding each uppercase letter by a slash /. The slash character must not be used for any other purpose in the connection field. Separators between shell description and individual pins should be hyphens; for example,

J3-/A is jack 3, pin a
PLA3-12 is plug A3, pin 12

ATLAS keywords, in particular pin descriptors, are not permitted as connection descriptions.

4.5.7 Messages. The text of messages must be written using any symbol from the authorized character set, except the apostrophe and currency symbols. The layout of the written characters must be the one that is required to be displayed or printed. Allowance must be made for any characters that may be inserted in the common text or DEFINE messages when these definitions are written.

4.5.8 Character Strings. It may seem unusual to include character strings at this point. However, they have most of the attributes of ATLAS numbers. The ATLAS syntax defines where they may be used. Character strings are written in a form similar to digital numbers; they are enclosed in quotes and preceded by the character C, for example,

123456 CALCULATE, 'TEXT'= C 'THIS STRING MUST ONLY USE PERMITTED CHARACTERS'S

Any ATLAS character except the currency symbol and a quote may occur in a character string. (THIS EXAMPLE CANNOT BE USED IN ARINC Specification 626-1-1989.)

4.6 Classification of the Language Elements.

4.6.1 General. The parts of syntax of ATLAS are very similar to those of any natural language and comprise verbs, nouns, modifiers (adjectives), connectives (conjunctions), and delimiters (punctuation).

An ATLAS statement is very similar to a natural language sentence. The verb is the action word, especially when that verb is used in the imperative (command) tense. The majority of ATLAS statements are in this tense and thus require the verb-predicate format of the natural language. For uniformity of statement construction this order is maintained for all statements, even when there is no command or, in some cases, only a pseudo verb.

4.6.2 Verbs. In ANSI/IEEE Std 416-1984, IEEE Std 716-1989, and ARINC Specification 626-1-1989, the verbs are presented in groups according to their functions. This grouping is repeated here to present to the first-time reader the scope and power of the language.

NOTE: Items marked * are not in IEEE Std 716-1989
Items marked † are not in ARINC Specification 626-1-1989

(1) Structure delimiting (4.0)

BEGIN, ATLAS PROGRAM
BEGIN, ATLAS MODULE
BEGIN, BLOCK*
COMMENCE, MAIN PROCEDURE*
END, BLOCK*
LEAVE, BLOCK*
TERMINATE, ATLAS PROGRAM
TERMINATE, ATLAS MODULE

(2) Preamble (6.0)

DECLARE

THE ATLAS SPECIFICATION

DEFINE,'converter'†*
DEFINE,'digital configuration'
DEFINE,'digital timing' (IEEE Std 716-1989 only)
DEFINE,'digital sensor'
DEFINE,'digital source'
DEFINE,'drawing' *
DEFINE,'exchange'
DEFINE,'function' *
DEFINE,'procedure'
DEFINE,'protocol'*
DEFINE 'signal'
DEFINE, 'task' †*
END, 'converter' †*
END,'digital configuration'
END,'function' *
END,'procedure'
END,'protocol'*
END,'task'†*
ESTABLISH 'protocol'(IEEE Std 716-1989 only)
EXTEND, ATLAS connections for
EXTEND, ATLAS modifiers for
EXTEND, ATLAS nouns
FOR UUT†*
IDENTIFY, EVENT (event)
IDENTIFY, EVENT (signal)
IDENTIFY, EVENT INDICATOR
IDENTIFY, EVENT INTERVAL
IDENTIFY, TIME BASED EVENT*
IDENTIFY, TIMER
INCLUDE
LEAVE,'function' *
LEAVE,'procedure'
LEAVE,'task'†*
REQUIRE
SPECIFY,'delay min'

(3) Data processing (8.0)

CALCULATE
COMPARE
CREATE FILE (ARINC Specification 626-1-1989 only)
DELETE FILE (ARINC Specification 626-1-1989 only)

(4) Input/output (9.0)

DISABLE FILE ACCESS
ENABLE FILE ACCESS
FILL†*
INPUT
OUTPUT

(5) Program Control (10.0)

CONTINUE,'task'† *
DISABLE, ESCAPE

DISABLE, MANUAL INTERVENTION*
DISABLE, 'task'†*
ELSE
ELSE, IF
ELSE, IF-THEN*
ENABLE, DIGITAL CONFIGURATION +
ENABLE, ESCAPE*
ENABLE, MANUAL INTERVENTION*
ENABLE, 'task'†*
END, FOR
END, IF
END, WHILE
FINISH
FOR, THEN
GO TO
IF, THEN
LEAVE, FOR
LEAVE, IF
LEAVE, WHILE
PERFORM
RELEASE†*
REQUEST†*
RESUME, FOR *
RESUME, WHILE*
SUSPEND 'task' †*
WAIT FOR, MANUAL INTERVENTION*
WHILE, THEN

(6) Signal-oriented (11.0)

ADJUST†*
APPLY
ARM
CHANGE
CONNECT
DISABLE, 'event'
DISCONNECT
ENABLE, 'event'
FETCH
INITIATE
MEASURE
MONITOR
PROVE
READ
REMOVE
RESET
SENSE
SETUP
STIMULATE
TO MAXIMIZE†*
TO MINIMIZE†*
TO REACH†*
VERIFY

(7) Timing (12.0)

DELAY (ARINC Specification 626-1-1989 only)
 DISABLE, TIME-LIMIT†*
 DO, SIMULTANEOUS
 DO, TIMED DIGITAL
 ENABLE, TIME-LIMIT†*
 END, DO
 READ, TIMER (ARINC Specification 626-1-1989 only)
 RESET, TIMER
 WAIT FOR
 WAIT FOR, EVENT
 WAIT FOR, 'time'
 WAIT FOR, TIMER

(8) Macro statements restricted to digital testing (13.0)

DO † *

(9) Macro statements related to data-bus testing (13.3) (In IEEE Std 716-1989)

DO, EXCHANGE
 UPDATE, EXCHANGE†

Other criteria could have been used to divide the set of verbs into different groups or subgroups from those listed in (1) through (9), and it is useful to consider some different groupings.

One of the ground rules for the development of ATLAS is "a procedure shall be written in a step-by-step manner." Due to this, some verbs are described in terms of other simpler verbs. This creates the terms multiple-action signal-oriented verbs versus single-action signal-oriented verbs. Further subgrouping produces the terms source (and load) type and sensor type. The relationship between single-action and multiple-action verbs is illustrated in the state diagrams Figs 6 through 9. These diagrams show the states which can be adopted by a resource and the legal transitions which can be caused by single-action and multiple-action verbs. Any transitions which are not shown are illegal. Figure 10 shows the ATLAS verbs in respect to an ATE model.

A source type verb is one that is appropriate to use with source or load functioning of the equipment, for example, APPLY or ADJUST. That is, APPLY may be used to force a signal to the UUT or to place a load on the UUT, but it may not be used to perform a measurement function.

A sensor type verb may be used for measurement functions, for example, MEASURE or VERIFY.

Anomalies are apparent in some groupings. Although TO MAXIMIZE, TO MINIMIZE, TO REACH, START WHEN, and STOP WHEN are written in sensor-form statements, START WHEN and STOP WHEN are not true sensor-type verbs in that the value stored in the 'MEASUREMENT' label is not overwritten by the value measured in START WHEN or STOP WHEN statements. (These are not used in ARINC Specification 626-1-1989 or IEEE Std 716-1989.)

Similarly, DO and MEASURE, (SAMPLE), WAVEFORM, for which specific locations for the storage of measured data are mandatory, do not use or overwrite the 'MEASUREMENT' label.

4.6.3 Nouns and Modifiers. In ANSI/IEEE Std 416-1984, IEEE Std 716-1989 and ARINC Specification 626-1-1989, each noun has a list of associated noun modifiers in order that semantically inappropriate or nonsensical combinations cannot be formed. Modifiers, together with dimensional quantity or modifier descriptors, form statement characteristics to qualify the nouns. The complete list of nouns and noun modifiers is listed in Section 7 of this guide.

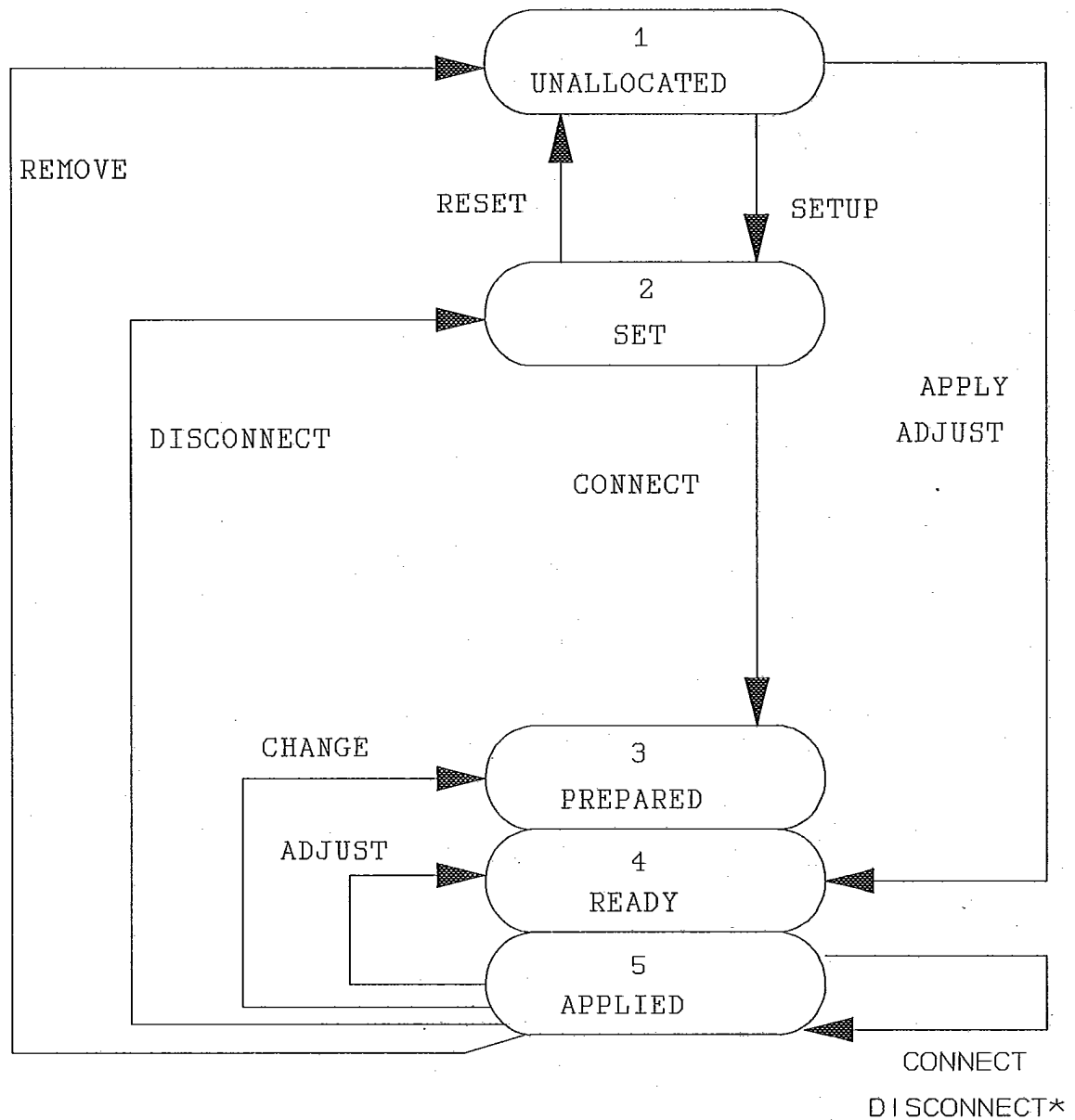


Fig 6
(a) State Diagram for a Source or Load Virtual Resource Without Event Monitor Using
Single-Action and Multi-Action Verbs
 (Applicable to ANSI/IEEE Std 416-1984)

* Multiple connections of a source or load resource to separate UUT interface points are permitted; thus DISCONNECT actions will leave the resource in the APPLIED state unless all UUT interface connections of the resource are disconnected, at which time it is changed to the SET state.