



IEEE Guide to the Use of ATLAS



Published by
The Institute of Electrical and Electronics Engineers, Inc



Distributed in cooperation with
Wiley-Interscience, a division of John Wiley & Sons, Inc

Approved September 13, 1984

IEEE Standards Board

James H. Beall, *Chairman*

John E. May, *Vice Chairman*

Sava I. Sherr, *Secretary*

J. J. Archambault
John T. Boettger
J. V. Bonucchi
Rene Castenschiold
Edward Chelotti
Edward J. Cohen
Len S. Corey
Donald C. Fleckenstein
Jay Forster

Daniel L. Goldberg
Donald N. Heirman
Irvin N. Howell
Jack Kinn
Joseph L. Koepfinger*
Irving Kolodny
George Konomos
R. F. Lawrence

John E. May
Donald T. Michael*
J. P. Riganati
Frank L. Rose
Robert W. Seelbach
Jay A. Stewart
Clifford O. Swanson
W. B. Wilkens
Charles J. Wylie

*Member emeritus

ISBN 471-82746-0

Library of Congress Catalog Number 84-043097

© Copyright 1984 by

The Institute of Electrical and Electronics Engineers, Inc

*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.*

December 21, 1984

IEEE Std 771-1984

**(Revision of ANSI/IEEE
Std 771-1980)**

IEEE Guide to the Use of ATLAS*

Sponsor

**IEEE SCC 20 (ATLAS) Committee of the
IEEE Standards Board**

***Abbreviated Test Language for All Systems**

Published by

**The Institute of Electrical and Electronics Engineers, Inc
345 East 47th Street, New York, NY 11117**



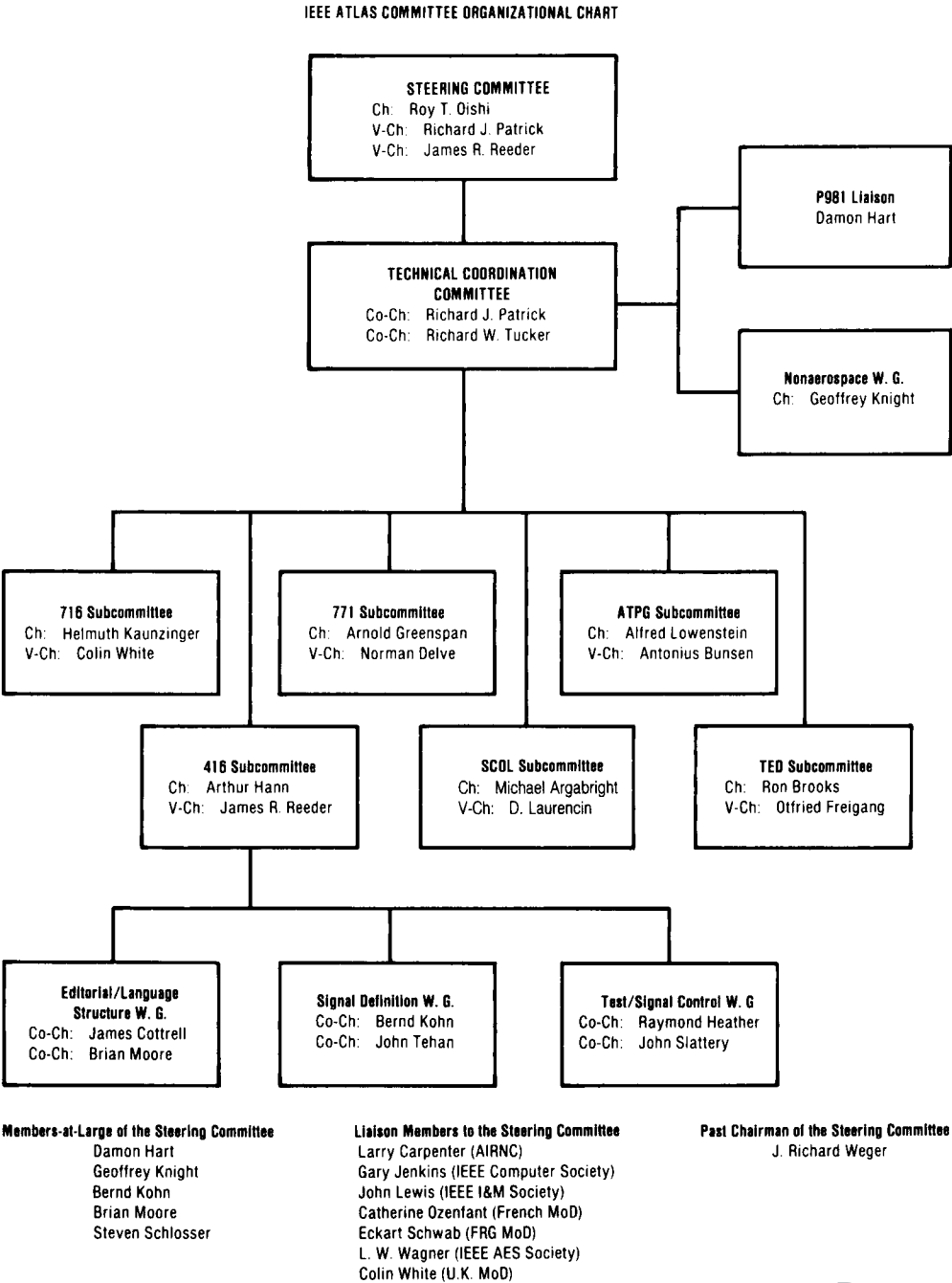
Documents

Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25, 2026 at 23:57:23 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 771-1984

At the time it completed its review of this standard the officers of the IEEE/ATLAS Steering Committee consisted of:



Foreword

(This Foreword is not a part of IEEE Std 771-1984, IEEE Guide to the Use of ATLAS.)

ANSI/IEEE Std 416-1984, ATLAS Test Language, defines ATLAS, a standardized test language for expressing test specifications and procedures. This guide is supplementary to ANSI/IEEE Std 416-1984 and is part of the documentation set for the language. An earlier ATLAS users guide had been issued at the time when the language specification was sponsored by Aeronautical Radio Inc. This Specification ARINC 418, which related to an earlier state of ATLAS, was not updated and is no longer extant.

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 771-1984 corresponds with the issue of ANSI/IEEE Std 416-1984 (at Supplement 21). It is the intention of the IEEE ATLAS Committee that there shall be a continued correspondence between ANSI/IEEE Std 416 and IEEE Std 771.

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

A. M. Greenspan, *Chairman*

N. R. Delve, *Vice Chairman*

S. J. Cherry

Robert Duffour

Hans V. Ehrismann

Hans Grundner

Kenneth E. Hermance

Harry A. Mason

Jim Massey

Richard Mattice

Michael D. Myles

Mike Prcic

Francois Szymanski

Thomas H. Tabb

Anthony A. F. Vogelpoel

Charles Zuroff

The assistance rendered by the UK Ministry of Defense and the US Department of Defense during preparation of this guide is gratefully acknowledged. In addition, acknowledgement is made of the valuable, individual contributions made by the following:

A. D. Bitter

R. G. Blodgett

T. S. Bush

P. M. Castle

W. H. Duke

W. R. Finch

R. A. Fletcher-Lord

D. P. Fuller

A. Hann

M. G. Harrison

R. L. Heather

R. J. Kelly

C. B. Knott

B. Kohn

M. Kuzemchak

G. Lawrence

R. J. Patrick

S. M. Schlosser

J. Slattery

J. L. Smeaton

M. R. Smith

W. E. Stamm

B. C. Stewart

R. E. Turkington

J. R. Weger

Contents

SECTION	PAGE
1. Introduction	17
1.1 For Whom This Volume Is Intended	17
1.2 Purpose of the Guide	17
1.3 Organization of the Guide	18
1.4 Related Documents	18
1.5 Standards References	18
1.6 Definition of Terms	18
1.7 Language Level	20
2. The Need for a Test-Language Standard	21
2.1 The Importance of Unambiguous Communication	21
2.2 Types of Languages	22
2.2.1 Person-to-Person Communication Language	22
2.2.2 Person-to-Machine Communication Language	22
2.3 The Test Language	24
2.4 ATLAS Development	24
2.4.1 General	24
2.4.2 Constraints	26
2.5 Continued Regulated Growth of ATLAS	27
2.6 ATLAS as an Aid to the Management of Testing	27
2.7 Benefits of a Standard	27
3. ANSI/IEEE Std 416-1984	29
3.1 Definition of ATLAS	29
3.2 ANSI/IEEE Std 416-1984	29
3.2.1 General Definition	29
3.2.2 General Characteristics of ATLAS	29
3.2.3 Levels of ATLAS	30
3.2.4 ATLAS Test Procedure Structures	30
3.3 ANSI/IEEE Std 416-1984, Formal Syntax Section	35
3.4 Non-ATLAS Usage	36
4. The Elements of ATLAS	37
4.1 Introduction	37
4.2 The ATLAS Character Set	37
4.3 Keywords	38
4.4 ATLAS Number Conventions	38
4.4.1 General	38
4.4.2 Decimal Numbers and Decimal Integers	39
4.4.3 Digital Numbers	40
4.4.4 Statement Numbers	42
4.4.5 Complex Numbers	42
4.5 User-Formed Strings	42
4.5.1 General	42
4.5.2 Variables	43

SECTION	PAGE
4.5.3 Defined Names	44
4.5.4 Connections	45
4.5.5 Messages	45
4.5.6 Character Strings	45
4.6 Classification of the Language Elements	45
4.6.1 General	45
4.6.2 Verbs	46
4.6.3 Nouns and Modifiers	50
5. Writing a Test Procedure in ATLAS	55
5.1 Introduction	55
5.2 Analysis of the Test Problem	55
5.3 Segmenting the Test Procedure	55
5.4 Flow Charting	56
5.5 Logical Branching	56
5.6 Test Resources	56
5.7 Arrangement of Test Procedure	60
5.7.1 Preamble	60
5.7.2 Delimiting Statements	60
5.7.3 Label Control	61
5.8 Global Procedures	62
5.9 Instructions to and from the Operator	64
5.10 (Part of 5.9)	67
5.11 (Part of 5.9)	67
5.12 Applying Power to the UUT	67
5.13 Resource Allocation	68
5.14 Identifying Signals with Resources	68
5.15 Applying Test Signals	69
5.15.1 Applying Digital Signals	71
5.16 Sensing UUT Responses	73
5.17 Statement Execution Rate	74
5.17.1 General	74
5.17.2 PREPARE/EXECUTE Structure	74
5.18 Postponing Statement Execution	75
5.18.1 General	75
5.18.2 DELAY Statement	75
5.18.3 WAIT FOR Time Statement	75
5.18.4 WAIT FOR, Time Statement	76
5.18.5 WHILE, THEN Structure	77
5.18.6 Synchronization and Strobing	77
5.19 Branching	79
5.19.1 General	79
5.19.2 DECISION-TABLES	79
5.20 Entry Points	81
5.21 Iteration in Testing	82

SECTION	PAGE
5.21.1 General	82
5.21.2 Loop Control	82
5.21.3 The Use of Loops	82
5.21.4 Procedures for Loops	88
5.21.5 Implied Iterations	90
5.21.6 Summary	90
5.22 Complex-Action Verbs	90
5.23 Multistatement Sequences	90
5.23.1 Adjustments	90
5.23.2 Interval Measurements	90
5.23.3 EVENT COUNTING	91
5.23.4 Escape Fields	91
5.24 Simultaneous Testing	92
5.24.1 Tasks	92
5.24.2 Communication Between Tasks and Mainstream Testing	92
5.25 Test Problems Not Covered by ATLAS Vocabulary	92
5.25.1 General	92
5.25.2 DEFINE, 'Drawing'	94
5.25.3 LEAVE/RESUME ATLAS	94
5.26 Use of Comments	94
5.27 Shutting Down a UUT	95
5.27.1 General	95
5.27.2 At the Conclusion of Testing	95
5.27.3 On-Line Repair or Adjustment	95
5.27.4 Malfunction	95
5.27.5 Test Equipment Malfunction	95
5.28 Recommended Standards of Good ATLAS	96
5.28.1 User Preparation	96
5.28.2 Writing the Testing Document	96
5.28.3 Verification and Validation of the ATLAS Tests	98
6. Examples	101
6.1 Introduction	101
6.1.1 General	101
6.1.2 Documentation of Test Requirements	101
6.2 Unit Under Test	102
6.3 Design Requirement	102
6.4 Production Test Procedure	104
6.4.1 General	104
6.4.2 Production Test Procedure — Unit Type 4051-6	104
6.5 ATLAS Test Procedure	105
6.5.1 General	105
6.5.2 Sample ATLAS — Structured Programming Version	105
6.5.3 Commentary on Structured Programming Version	105
6.5.4 Sample ATLAS — GO TO Version (Non-preferred)	114

SECTION	PAGE
6.5.5 Commentary on GO TO Version	114
6.5.6 Summary	118
7. ATLAS Problem/Proposal Preparation Guide	121
7.1 Scope	121
7.2 Description of Data in Problem/Proposal Form	121
7.2.1 Subject	121
7.2.2 Date	121
7.2.3 Submitter	121
7.2.4 Telephone	121
7.2.5 Organization	121
7.2.6 Address	121
7.2.7 Reference	121
7.2.8 Summary of Change Proposal	122
7.2.9 Problem Description	122
7.2.10 Change Proposal	122
7.3 Editorial Guidelines	122
7.3.1 General	122
7.3.2 Style	122
7.3.3 Subsections	122
7.3.4 Example of a Proposal	123
8. ATLAS Elements	125
8.1 Introduction	125
8.2 ATLAS Verbs	125
8.3 ATLAS Nouns	126
8.4 ATLAS Noun Modifiers	127
8.5 ATLAS Units	133
8.6 Miscellaneous Elements	136
9. Glossary of ATLAS Keywords	143

FIGURES

Fig 1 ATLAS Test Specification Family Tree	31
Fig 2 ATLAS Program Structure	32
Fig 3 ATLAS Program Structure Syntax Diagram	33
Fig 4 Program Preamble Structure Syntax Diagram	34
Fig 5 Basic Statement Structure	35
Fig 6 State Diagram for a Source or Load Virtual Resource with Event Monitor Using Single-Action and Multi-Action Verbs	50
Fig 7 State Diagram for a Source or Load Virtual Resource with Event Monitor Using Single-Action and Multi-Action Verbs	51
Fig 8 State Diagram for a Sensor Virtual Resource with Event Monitor for Making a Measurement Using Single-Action and Multi-Action Verbs	52

FIGURES

PAGE

Fig 9	State Diagram for a Sensor Virtual Resource with Event Monitor for Making a Measurement Using Single-Action and Multi-Action Verbs	53
Fig 10	ATLAS Verbs vs ATE Model	54
Fig 11	Examples of Flow Chart Level	57
Fig 12	Example of Program Flow Diagrams (a) IF, THEN/ELSE Structure; Forces Separation of Main Tests ..	58
	(b) GO TO Structure; Diagnostic Separated	59
Fig 13	Examples of Delimiting Statements	61
Fig 14	Hypothetical Example of Complex Structure with Poor Readability	84
Fig 15	Test Loop 1. Control Statements Are Among Test Statements	85
Fig 16	Test Loop 2. Control Statements Are After Test Statements	86
Fig 17	Test Loop 3. Control Statements Are Before Test Statements	87
Fig 18	Task State Diagram	93
Fig 19	Schematic Diagram of Fuel-Control System	102
Fig 20	Schematic Diagram of Electronic Vehicle Fuel Control, Type 4051-6	103
Fig 21	Test Flow Diagram of Sample ATLAS Procedure, Structured Programming Version	106
Fig 22	Printout of Sample ATLAS Procedure of Fig 21	108
Fig 23	Test Flow Chart of Sample ATLAS Procedure, GO TO Version	115
Fig 24	Printout of Sample ATLAS Procedure of Fig 23	116
Fig 25	ATLAS Problem/Proposal Form	124

TABLES

Table 1	High and Low Level Languages	23
Table 2	ATLAS Character Set	37
Table 3	Examples of Typical Decimal Numbers	39
Table 4	ATLAS Variables	44

APPENDIXES

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-2
A3.1	ATLAS Nouns Used in Analog Testing	A-2
A3.2	ATLAS Noun Modifiers	A-4
A4.	Examples	A-13
A4.1	A General Alternating Current Power Supply	A-13
A4.2	A General DC Logic Level Power Supply	A-13
A4.3	A Ramp Deflection Signal	A-13

SECTION	PAGE
A4.4 A Noise Signal Which Can Be Used in Amplifier Testing	A-14
A4.5 A Square Wave Signal	A-14
A4.6 A Step Signal Which May Be Used in Servo Testing	A-15
A4.7 Using Waveform a Simple Sine-Wave May Be Synthesized . . .	A-16
A4.8	A-16
A4.9	A-16
A4.10	A-16
A4.11	A-17
A4.12	A-17
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-2
B2.3 Data Field Manipulation	B-2
B3. Language Elements	B-3
B3.1 ATLAS Nouns Used in Digital Testing	B-3
B3.2 ATLAS Noun Modifiers Used in Digital Test	B-8
B4. Examples	B-19
B4.1 LOGIC NOUNS	B-19
B4.2 DIGITAL TEST	B-20
Appendix C. Air Navigation System Testing	C-1
C.1 Introduction	C-1
C.1.1 New Concepts	C-1
C2. DME	C-2
C2.1 System Description	C-2
C2.2 Language Elements	C-2
C2.2.1 Nouns	C-2
C2.2.2 Moun Modifiers	C-2
C2.3 Examples	C-10
C3. ILS	C-10
C3.1 System Description	C-10
C3.2 Language Elements	C-11
C3.2.1 Nouns	C-11
C3.2.2 Moun Modifiers	C-11
C3.3 Examples	C-15
C4. TACAN	C-16
C4.1 System Description	C-16
C4.2 Language Elements	C-17
C4.2.1 Nouns	C-17
C4.2.2 Noun Modifiers	C-17
C5. VOR	C-23
C5.1 System Description	C-23

SECTION	PAGE
C5.2 Language Elements	C-24
C5.2.1 Nouns	C-24
C5.2.2 Moun Modifiers	C-24
C5.3 Examples	C-27
C6. ADF	C-28
C6.1 System Description	C-28
C6.2 Language Elements	C-28
C6.2.1 Nouns	C-28
C6.2.2 Moun Modifiers	C-28
C6.3 Examples	C-30
C7. ATC	C-31
C7.1 System Description	C-31
C7.2 Language Elements	C-31
C7.2.1 Nouns	C-31
C7.2.2 Moun Modifiers	C-31
C7.3 Examples	C-35
C8. IFF	C-36
C8.1 System Description	C-36
C8.2 Language Elements	C-37
C8.2.1 Nouns	C-37
C8.2.2 Moun Modifiers	C-37
C8.3 Examples	C-41
C9. Air Data System Testing	C-42
C9.1 System Description	C-42
C9.2 Language Elements	C-42
C9.2.1 Nouns	C-42
C9.2.2 Moun Modifiers	C-43
C10. Example of a Complete Test	C-45
 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-4
D3.3 Valve Operation — COUPLE and UNCOUPLE	D-4
D3.4 Gas Turbine Engine Testing Nouns	D-4
D3.4.1 TURBINE ENGINE DATA	D-5
D3.4.2 FLUID SIGNAL	D-6
D3.4.3 AMBIENT CONDITIONS	D-7

SECTION	PAGE
D3.4.4 STEPPER MOTOR	D-7
D3.4.5 VIBRATION	D-8
D4. Examples	D-8
D4.1 Use of TURBINE ENGINE DATA Noun	D-8
D4.2 Use of FLUID SIGNAL Noun	D-9
D4.3 Use of AMBIENT CONDITIONS Noun	D-9
D4.4 Use of STEPPER MOTOR Noun	D-10
D4.5 Use of VIBRATION Noun	D-10
Appendix E. ATLAS in Inertial Navigation System Testing	E-1
E1. Introduction	E-1
E2. Basic Inertial Navigation Principles	E-2
E3. Test Procedures	E-2
E3.1 Sensor Tests	E-3
E3.1.1 Gyros	E-3
E3.1.2 Accelerometer	E-4
E3.1.3 Gimbal System	E-4
E3.2 Electronics Tests	E-4
E3.2.1 Sensors and Control Loops	E-4
E3.2.2 Digital Tests	E-5
E3.3 System Tests	E-5
E3.3.1 Power Supplies	E-5
E3.3.2 Control Loops	E-5
E3.3.3 Calibration of Gimbal Synchro Transmitters	E-5
E3.3.4 Drift Test	E-5
E3.4 Diagnostic Tests	E-6
E4. Stimuli and Measurement Requirements for Testing Inertial Navigation Systems	E-6
E4.1 Stimuli Requirements	E-6
E4.2 Measurement Requirements	E-7
APPENDIX FIGURES	
Fig A1 Pulse Definition	A-6
Fig A2 (a) Voltage Ramped (b) Voltage Stepped	A-15
Fig B1 Illustration of RZ, NRZ, and BIP Pulse Types of B'10110010'	B-13
Fig C1 Pulse Definition	C-9
Fig C2 ADF Antenna Pattern	C-28
Fig C3 Relative Bearing	C-29
Fig E1 Simple Inertial Navigator	E-3
Fig E2 Stable Platform Schematic Graph	E-4
APPENDIX TABLES	
Table B1 Digital Modifier Usage	B-22

IEEE Guide to the Use of ATLAS

1. Introduction

1.1 For Whom This Volume Is Intended. This guide has mainly been written as an applications manual for engineers and authors who will be writing test procedures in ATLAS. The beginning chapters are also intended to provide managers with a brief introduction to the application of ATLAS in their organizations. 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.

It has been assumed that this guide will prove most useful to those engineers who are preparing test requirements in ATLAS. The cross referencing, therefore, between this guide and ANSI/IEEE Std 416-1984 has been confined to the classical definitions presented in ANSI/IEEE Std 416-1984. Appropriate cross references are included in parentheses following section headings, and unless otherwise stated, they relate to ANSI/IEEE Std 416-1984.

1.2 Purpose of the Guide. The purpose of this guide is to advise on the 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. A number of companies are contractually committed to prepare test procedures in ATLAS, a system which must be unambiguously interpreted by both personnel and machines. The experiences of test engineers from these companies have been distilled to prepare this guide.

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. Herein lies the need for this guide.

The guide concentrates on the proper use of the ATLAS vocabulary, syntax, and structure for the generation of unambiguous test procedures. If the guide assists in the preparation of good test procedures, it should also assist 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 in many disciplines across the testing spectrum. This fact may make it difficult for an engineer working in a particular discipline to locate and use an element that he needs. Therefore, this guide has been organized into a main text and several appendixes.

The main text contains information and language elements that 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 ANSI/IEEE Std 416-1984. Supplements to ANSI/IEEE Std 416-1984 will be issued periodically and the users guide updated accordingly.

1.5 Standards References. The following standards references were used in the preparation of this guide.

ISO 646-1973, 7-Bit Coded Character Set for Information Processing Interchange.¹

ISO 1000-1973, SI Units and Recommendations for the Use of Their Multiples and of Certain Other Units.¹

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 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.

¹ In the US, ISO Standards are available from the Sales Department of American National Standards Institute, 1430 Broadway, New York, NY 10018.

BCD. Binary-coded decimal.

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

Decision Table. A matrix-providing program branching which may be a complex function of a number of variables.

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 Standards Organization.

List. An ordered single-dimensioned set of data.

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.

Non-preferred ATLAS. An ATLAS structure which is found to be superfluous or troublesome in real use, 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 (Conférence générale 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.

1.7 Language Level. This guide has been published in accordance with ATLAS Language, level 21 (ANSI/IEEE Std 416-1984). Subsequent revisions will be published as the language advances.

2. The Need for a Test-Language Standard

2.1 The 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 to assist 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 around 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 touched on. 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 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, which is further complicated by the fact that communications are not restricted to the human level. 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 to the person and may even be ambiguous in isolation, they contain 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

Table 1
High and Low Level Languages

<i>Problem:</i> Calculate $a = b^3$	
<i>Solution:</i>	
High Level	Low Level
$A = B^{**}3$ ($a = b^3$)	(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.	

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.

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 the 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. It becomes increasingly obvious that the major stumbling block in test-information communication is the need to express testing functions concisely and unambiguously and that 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 meaning of everyday words is simply too wide, and too many meanings or too many imperfect synonyms exist to guarantee that total information transfer will be effected. Add to this the requirement that the test information be readable by both humans and machines, and the communications problem is further complicated.

The obvious solution was to examine a natural language that already had wide usage and to subset and redefine the language for the specific purpose of test-data information communications. This was the driving factor 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. On examination, all these languages lacked the general-purpose approach applicable to the testing domain, and 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 that was based on a subset of a natural language, that subset being oriented toward the testing problem. The natural language chosen was English by virtue of its international acceptance, and 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.

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. This

activity was regulated by the Airlines Electronic Engineering Committee (AEEC), which could supply the technical personnel, and was sponsored by ARINC. The principal function of the AEEC is to formulate standards for electrical equipment and systems for the airlines.

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.
1	October 1968	The final draft was approved by AEEC for issue as 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	October 1972	A digital testing capability was introduced.
7/8	July 1973	Some minor changes were introduced by issue 7. The significant change in issue 8 was that a digital test macro capability was introduced.
9/10	May 1974- May 1975	Miscellaneous refinements were incorporated. The significant changes introduced by issue 10 were decision table and task capabilities. Task allowed for 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 routine.
12/13	April 1976- August 1976	Miscellaneous refinements, particularly relating to complex numbers and waveform, were incorporated.
13A	August 1976	ARINC Specification 416 was reformatted to include all ATLAS material contained in issues 1 through 13 with no changes. The revised format was introduced to improve the interpretation and the management of the specification.

ANSI/IEEE Std 416-1978	November 1976	The reformatted version, issue 13A, was adopted by the IEEE to become their standard for the ATLAS language.
ANSI/IEEE Std 416-1978	November 1978	Addition of REQUIRE and other changes.
ANSI/IEEE Std 416-1981	July 1981	Addition of GLOBAL, AND EXTERNAL attributes, 'modules', Data Bus Testing, and additional Digital Macro Functions. New descriptions for Air Navigation nouns and modifiers incorporated, with other miscellaneous refinements and classifications.

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 Englishlike, 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 less obvious, but 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 nonengineering 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 because many of these may be prefixed, suffixed, and sometimes doubly suffixed to produce distinct compound words.

The additional verbs cover mainly a macrofunction or a multiaction for digital test purpose, multiple decisions, and "structured" programming techniques. The additional nouns mainly provide further definitions of digital signals and air navigation signals, as well as nonelectronic 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 which surrounded 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 meta-language.

The particular metalanguage used in the definition of ATLAS is known as the Dehemer 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 ANSI/IEEE Std 416-1984. Thus ANSI/IEEE Std 416-1984 forms the present definition of the ATLAS language.

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 twice yearly 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. Along these lines 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 existed previously.

The ultimate benefit of a standard can only been 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

3.1 Definition of ATLAS. ATLAS is a standard abbreviated English language 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, which includes the material previously published in two separate volumes, namely, a general or functional definition and a formal definition written in a metanotation.

3.2 ANSI/IEEE Std 416-1984 (1.0)

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.

Section 3.2.2 presents the user with information regarding the general characteristics of ATLAS, the levels of ATLAS, and the basic structure of an ATLAS test procedure, as applicable to ANSI/IEEE Std 416-1984. The content of 3.2.2 is comprised of extracts from ANSI/IEEE Std 416-1984, expanded where necessary, together with explanatory paragraphs.

3.2.2 General Characteristics of ATLAS. The following are general characteristics of ATLAS.

(1) *Unit under test orientation.* The language is dedicated to defining the test requirements of the unit under test (UUT) with no reference to nor dependence upon the test equipment that may be used. The latter may be automatic, manual or of 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 (as defined in ANSI/IEEE Std 416-1984) constitute an environment that 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 those test speci-

fications 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 (1.2). 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 ANSI/IEEE Std 416-1984, is defined as standard ATLAS.

(2) *Subset ATLAS (1.2.2)*. 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 in 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 is defined as subset ATLAS with extensions. Specific rules for creating ATLAS subsets and appended extensions are provided in ANSI/IEEE Std 416-1984, paragraph 1.3.

(4) *Adapted ATLAS*. ATLAS is a precise language used to communicate test procedures among various users. ATLAS lends itself to the adaptation for a particular test environment where it would be advantageous to modify the ATLAS language to aid implementation. Any such modification is known as adapted ATLAS. Hence, 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 Std 416-1984, its use is not recommended.

3.2.4 ATLAS Test Procedure Structures (3.1)

3.2.4.1 General. An ATLAS Test Specification is structured into distinct groups which are defined herein. Figures 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 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 in a number of smaller, manageable portions with a controlled interface. See Fig 1.

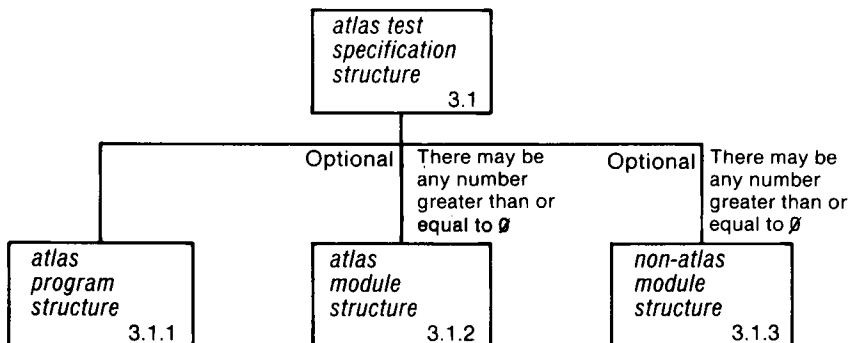


Fig 1
ATLAS Test Specification Family Tree

A syntax diagram (see Fig 3) 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 and 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 at the next lower level of the hierarchy. These latter elements are shown as dashed boxes containing the name of the element written in lowercase italic letters. A number is also included in the box to identify the paragraph in ANSI/IEEE Std 416-1984 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.

000100	BEGIN, ATLAS PROGRAM	\$	Preamble
05	FOR UUT, ...,MODS, ...	\$	
000200	REQUIRE, ...	\$	
.			
.			
000350	INCLUDE, ...	\$	
000400	DECLARE, ...	\$	
.			
.			
000450	FILL, ...	\$	
.			Procedure
000500	DEFINE, ...,SOURCE,.....	\$	
.			
.			
.			
000600	DEFINE, ...,PROCEDURE(.....) \$		
.	END,.....\$		
C	END OF PREAMBLE \$		
100000	COMMENCE, MAIN PROCEDURE, ...	\$	
E100001	APPLY, ...	\$	
.	MEASURE, ...	\$	LOCAL PREAMBLE
.	COMPARE, ...	\$	
.	IF THEN, ...	\$	
.	ELSE, ...	\$	
.			
E200000	BEGIN, ..BLOCK, ...'TIM' ..\$		
10	DECLARE, ...	\$	
20	DECLARE, ...	\$	
.			
50	BEGIN, BLOCK, 'WAVE' \$		
.			LOCAL PREAMBLE
.	END, BLOCK, 'WAVE' \$		
	COMPARE, ...	\$	
	END, BLOCK, 'TIM' \$		
.	FINISH, ...	\$	
.			
.	TERMINATE, ATLAS PROGRAM	\$	

Fig 2
ATLAS Program Structure

3.2.4.2 Program Preamble Structure (6.1.1)

(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.

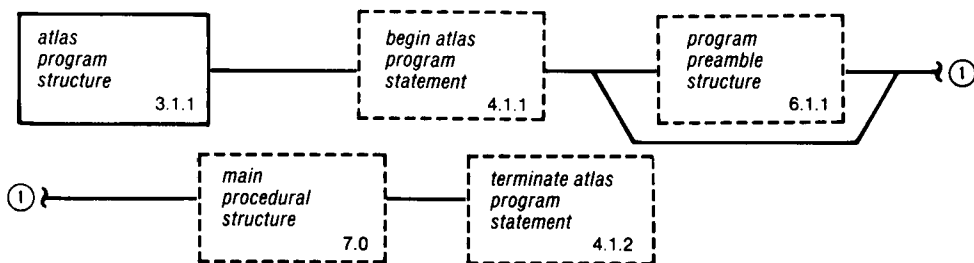


Fig 3
ATLAS Program Structure Syntax Diagram

(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.

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 with UUT oriented and test equipment independent sections of the ATLAS Test Procedure (preamble and procedural structures) to 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 (6.4) establishes labels (15.5) and identifies specific storage spaces required and establishes the type of data to be assigned to each space.

The DEFINE statement (6.5) 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 these names whenever required. Repeated messages can be handled in the same way.

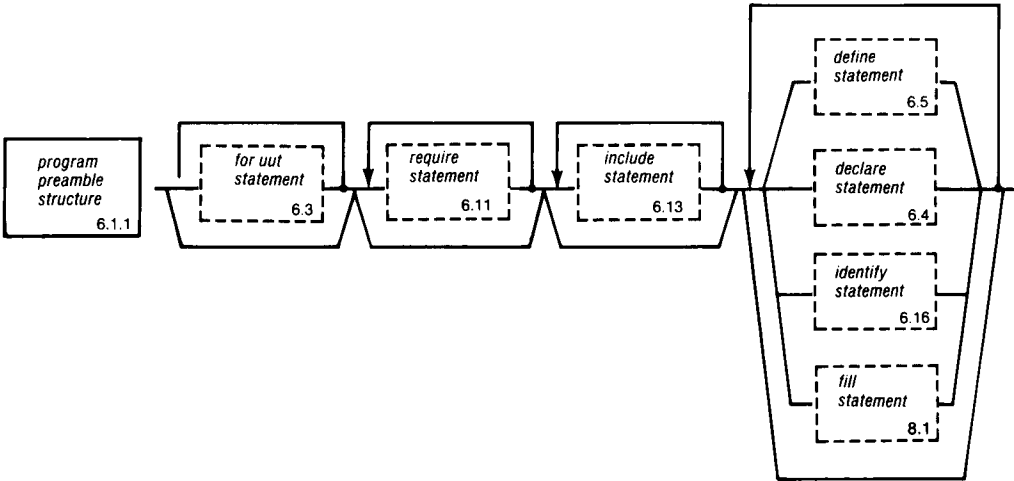


Fig 4
Program Preamble Structure Syntax Diagram

Preamble
000020 DEFINE, 'DC-1', SOURCE, DC SIGNAL, VOLTAGE 5V, ...\$
Procedure
100421 APPLY, 'DC-1', ...\$
is equivalent to:
100421 APPLY, DC SIGNAL, VOLTAGE 5V\$
and so on.

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.

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 (7.1). The Main Procedural Structure consists of a series of statements, which describes 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 (3.2). 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.

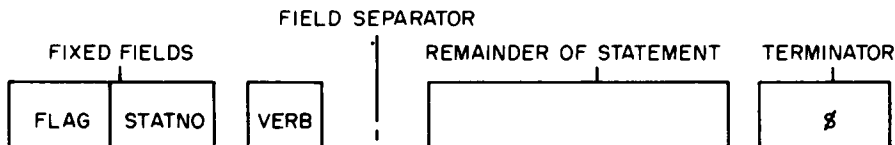


Fig 5
Basic Statement Structure

(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.

(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 ANSI/IEEE Std 416-1984.

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

3.3 ANSI/IEEE Std 416-1984, Formal Syntax Section. The Formal Syntax Section of ANSI/IEEE Std 416-1984 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 1950s 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 details of the particular extension of the Backus-Naur notation used in the Formal Syntax Section are explained. There are many computer-science text books that give a fuller treatment to the principles behind metalanguages in general and the Backus-Naur notation in particular.

The Formal Syntax Section of ANSI/IEEE Std 416-1984 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 defines 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 ANSI/IEEE Std 416-1984 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 eliminate these imprecisions, but 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 so that they can help to resolve any problems by reference to that section.

The definitions within ANSI/IEEE Std 416-1984 should be totally compatible. If there are any discrepancies, this is indicative of an error in one or both definitions. 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. 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 does not adequately cover a particular test discipline or problem the non-ATLAS 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. Previously the non-ATLAS material was bounded by the delimiting statements LEAVE ATLAS and RESUME ATLAS, but this method is now non-preferred.

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 has encountered with if possible a solution. In this way ATLAS will continue to be developed.

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 prescribed orders 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 (15.1). The ATLAS character set was chosen because the characters are commonly available on the input/output devices associated with data processing equipment. The character set of 95 symbols is listed in Table 2.

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	Operators	User-formed strings
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		

Table 2 (continued)

Characters	Usage
{ 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 is used literally as lowercase, a slash is written immediately preceding each affected letter. An example is a manufacturer's pin designation, namely:

socket 1-bb should 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) For statement termination it is essential to use the 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 9). 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 (15.2)

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.* Subdivided into 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 considered in more detail in 4.4.2-4.4.5.

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. 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.

Table 3 shows some typical decimal numbers.

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} .

Some of the many floating-point representations of the value 1234.5 are:

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

Table 3
Examples of Typical Decimal Numbers

(a)*	16393	}	Signed or unsigned	}	Integers	}	decimal numbers
(b)	0						
(c)	-16393	}	Signed or unsigned	}			
(d)	1639.3						
(e)	0.16393	}	Signed	}			
(f)	-16.393						
(g)	+16393	}		}			
(h)	-0.16393						
(j)	+16393.						

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

All these are equally acceptable in ATLAS. However, it is common practice to restrict the 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 1000-1973, SI Units and Recommendations for the Use of Their Multiples and of Certain Other Units). |
| (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)-(3) represents the argument and E the exponent.

The rules for number construction are detailed in ANSI/IEEE Std 416-1984. Deviations, even if they are common national practices such as the 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 are based on numbers with a radix 2. Such numbers are called binary numbers. The digits of a binary number take the values 0 or 1.

Long strings of 1s and 0s tend to be incomprehensible, inconvenient, and error prone. The use of radices that are powers of 2 reduces the size of the number 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.

Although in ATLAS binary numbers can be declared as having an integral part to the left and a fractional part to the right side of a radix point, when a binary number is written explicitly, it always has its apparent radix point at the extreme right-hand end. However, if a decimal number is to be put into a digital store, it is converted so that the radix points coincide, the integral part is converted to an integral binary, and the fractional part is converted to a fractional binary either side of the radix point.

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 digital numbers in single quotes and preceding them by a prefix letter to indicate the type of digital number. The letters are used:

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.

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, and each group represents 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 one of two values, it is not always possible to predict which it will take, either because of inadequate information about the history of the device since it had power applied or because of possible unpredictable pulses in the circuit. In this case it is necessary that the particu-

lar 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 legal digits for each number type are as follows:

Binary	0, 1, X
Octal	0, 1, 2, 3, 4, 5, 6, 7, X
BCD	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, to a degree, 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' to the parity check bit. Dependent on whether odd or even parity checking is required the symbols PO or PE may be written in place of the parity 'field' in a digital word stimulus.

4.4.3.2 Digital Number Descriptors (14.27). 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 when applicable are defined in the DECLARE statement by a digital number descriptor. The ATLAS digital number descriptors are given below:

BNR	<i>Binary Number</i> . Pattern with no sign bit and no negative form.
BSM	<i>Binary Sign and Magnitude</i> . 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	<i>Binary Ones Complement</i> . 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	<i>Binary Twos Complement</i> . 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	<i>Binary-Coded Decimal</i> . The unsigned form has multiples of four bits, one group 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 646-1973, 7-Bit Coded Character Set for Information Processing Interchange.
n ASCII	(15.2). A character from the n-bit ASCII code (ASCII is the ISO code with US national deviations. Since the deviations are not permitted in ATLAS, it is effectively the same as n ISO). 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, and 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 comprise 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 being a distance R from the origin at an angle θ is expressed as R/θ .

(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 being 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. Variables, defined names, and unit under test (UUT) connection designators 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'		
'AC POWER'	}	Unique at the 16th character.
'ACPOWER'		Considered identical
'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 2nd character instead of the 17th 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 Variables

4.5.2.1 General. A variable is the name chosen by the test writer to describe an implementation-time location which holds numerical, literal, or logical data for use in signal-orientated, computational, logical, or input/output operations. Data of a particular type may be held only in locations reserved for that type. The types of variables that are legal in ATLAS are listed in Table 4.

Variables must be declared in either the main preamble or a local preamble. Variables declared in different preambles can use identical labels. The span during which a variable is valid is determined by the type of preamble. Variables declared in the main preamble are said to be global and access to these variables can be made from any subsequent statement except when the same label name has been declared in a local preamble. In this case the local variable takes precedence in those procedural statements directly or indirectly associated with the local preamble containing the identical label.

Two classes of variables are legal in ATLAS. They are described by the term STORE for simple variables and the term LIST for subscripted (indexed) variables. Both classes of variables can hold data of any type for which they have been declared.

Table 4
ATLAS Variables

Type	Contents
DECIMAL	A base 10 numeric value which may have a fractional part
INTEGER	A base 10 numeric value which has no fractional part
DIGITAL	A binary, octal, or hexadecimal value
BOOLEAN	A two-state value: true or false
STEP	A six-digit statement number
CONN	A UUT connection
MSGCHAR	Literal message text for insertion into messages
COMPLEX-POLAR	A complex number in polar notation, for example, 10V/53DEG
COMPLEX-RECTANGULAR	A complex number in Cartesian notation, for example, 10+J12V.

4.5.2.2 Simple Variables. A simple variable holds a single item of data and is identified by a user name of the form store.

ATLAS has one reserved variable, labeled 'MEASUREMENT', which serves the same function wherever it appears. In more complex programs, which involve truly parallel processing, independent 'MEASUREMENT' variables are required for each of the parallel processes.

4.5.2.3 Indexed Variables. A list is an ordered set of data having one dimension. An element in the list is called a subscripted or indexed variable and is identified by a user name and an index, in the form 'list' (index), where the term 'list' names the list and the index is an unsigned integer that identifies the index number, for example, 'FRED' (3).

At declaration time a *list* location is allocated a user name together with an integer, in the form 'list' (integer), where the integer is an unsigned numeral that defines the number of elements in the *list*.

A single *list* element, i.e., 'list' (index) can be considered equivalent to a 'store'.

4.5.3 Defined Names (6.5). 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 the 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.

4.5.4 Connections (14.23, 14.24). 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 character /. 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.

4.5.5 Messages (6.6). 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 of DEFINE messages when these definitions are written.

4.5.6 Character Strings (15.9, 15.10). It may seem unusual to include character strings at this point. However, they have most of the attributes of ATLAS numbers, except that they are not available for use in any calculation. 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 FILL, 'TEXT', C 'THIS STRING MUST USE ONLY
PERMITTED CHARACTERS'\$

Any ATLAS character except the currency symbol and a quote may occur in a character string.

4.6 Classification of the Language Elements

4.6.1 General. The parts of speech 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 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.

(1) *Structure delimiting* (4.0)

BEGIN, ATLAS PROGRAM
BEGIN, ATLAS MODULE
BEGIN, BLOCK
BEGIN, NON-ATLAS MODULE
COMMENCE, MAIN PROCEDURE
END, BLOCK
LEAVE ATLAS (non-preferred)
LEAVE, BLOCK
RESUME ATLAS (non-preferred)
TERMINATE, ATLAS PROGRAM
TERMINATE, ATLAS MODULE
TERMINATE, NON-ATLAS MODULE

(2) *Preamble* (6.0)

DECLARE
FOR UUT
INCLUDE
REQUIRE
DEFINE, 'converter'
DEFINE, 'digital configuration'
DEFINE, 'digital sensor'
DEFINE, 'digital source'
DEFINE, 'drawing'
DEFINE, 'exchange'
DEFINE, 'function'
DEFINE, 'message'
DEFINE, 'procedure'
DEFINE, 'protocol'
DEFINE, 'task'
END, 'converter'
END, 'digital configuration'
END, 'procedure'
END, 'protocol'
END, 'task'
IDENTIFY, EVENT (event)
IDENTIFY, EVENT INDICATOR
IDENTIFY, EVENT INTERVAL

IDENTIFY, EVENT (signal)
IDENTIFY, EVENT (time)
IDENTIFY, TIMER
LEAVE, 'procedure'
LEAVE, 'task'
SPECIFY, 'delay min'
 (3) *Data processing* (8.0)
CALCULATE
COMPARE
DISCARD (non-preferred)
SAVE
 (4) *Input/output* (9.0)
DISPLAY
FILL
INDICATE
INPUT
OUTPUT
PRINT
RECORD
WAIT FOR, MANUAL DATA
 (5) *Program control* (10.0)
BEGIN, DECISION-TABLE
CONTINUE
DECISION-TABLE
DISABLE, ESCAPE
DISABLE, 'event'
DISABLE, MANUAL INTERVENTION
DISABLE, 'task'
ELSE
ELSE, IF
ENABLE, DIGITAL CONFIGURAION
ENABLE, 'event'
ENABLE, ESCAPE
ENABLE, MANUAL INTERVENTION
ENABLE, 'task'
END, DECISION-TABLE
END, FOR
END, IF
END, WHILE
FINISH
FOR, THEN
GO TO
IF, THEN
LEAVE, DECISION-TABLE
LEAVE, FOR

LEAVE, IF
LEAVE, WHILE
PARTITION
PERFORM
RELEASE
REPEAT (non-preferred)
REQUEST
RESUME, DECISION-TABLE
RESUME, FOR
RESUME, WHILE
SUSPEND
WAIT FOR, MANUAL INTERVENTION
WHILE, THEN

(6) *Signal-orientated* (11.0)

ARM	}	single-action verbs
CHANGE		
CLOSE (non-preferred)		
CONNECT		
COUPLE		
DISCONNECT		
FETCH		
OPEN (non-preferred)		
SETUP		
UNCOUPLE		

ADJUST
ALTER (non-preferred)
APPLY
INITIATE
MEASURE
MONITOR
PROVE
REMOVE
READ
RESET
SENSE
STIMULATE
TO MAXIMIZE
TO MINIMIZE
TO REACH
VERIFY

(7) *Timing* (12.0)

DELAY
DISABLE, TIME INTERVAL
DISABLE, TIME-LIMIT
DO, SIMULTANEOUS

```

ENABLE, TIME-LIMIT
END, DO
EXECUTE
PREPARE
START
START WHEN,
STOP
STOP WHEN,
SYNC
SYNC WHEN,
WAIT FOR, EVENT
WAIT FOR, TIME

```

} (non-preferred)

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

DO

DO, TIMED DIGITAL

(9) *Macro statements related to data-bus testing (13.9)*

DO, EXCHANGE

Other criteria could have been used to divide the set of verbs into different groups or subgroups from those listed in (1) through (8), 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.” How big is a step? To overcome this problem some verbs are described in terms of other simpler verbs. This creates the terms multiple-action signal-orientated verbs and single-action signal-orientated verbs. Further subgrouping produces the terms source (and load) type and sensor type. The relationship between single-action and multi-action verbs are illustrated in the state diagrams Figures 6 through 9. These diagrams show the states which can be adopted by a resource and the legal transitions which can be caused by the single-action and multi-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.

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 (16.0, 17.0). In ANSI/IEEE Std 416-1984, listed in Sections 16 and 17, each noun has a list of associated noun modifiers so that semantically inappropriate or nonsensical combinations cannot be formed. Modifiers, together with dimensional quantity or modifier descriptors form statement characteristics which qualify the nouns. The complete list of nouns and noun modifiers is listed in Section 8 of this guide.

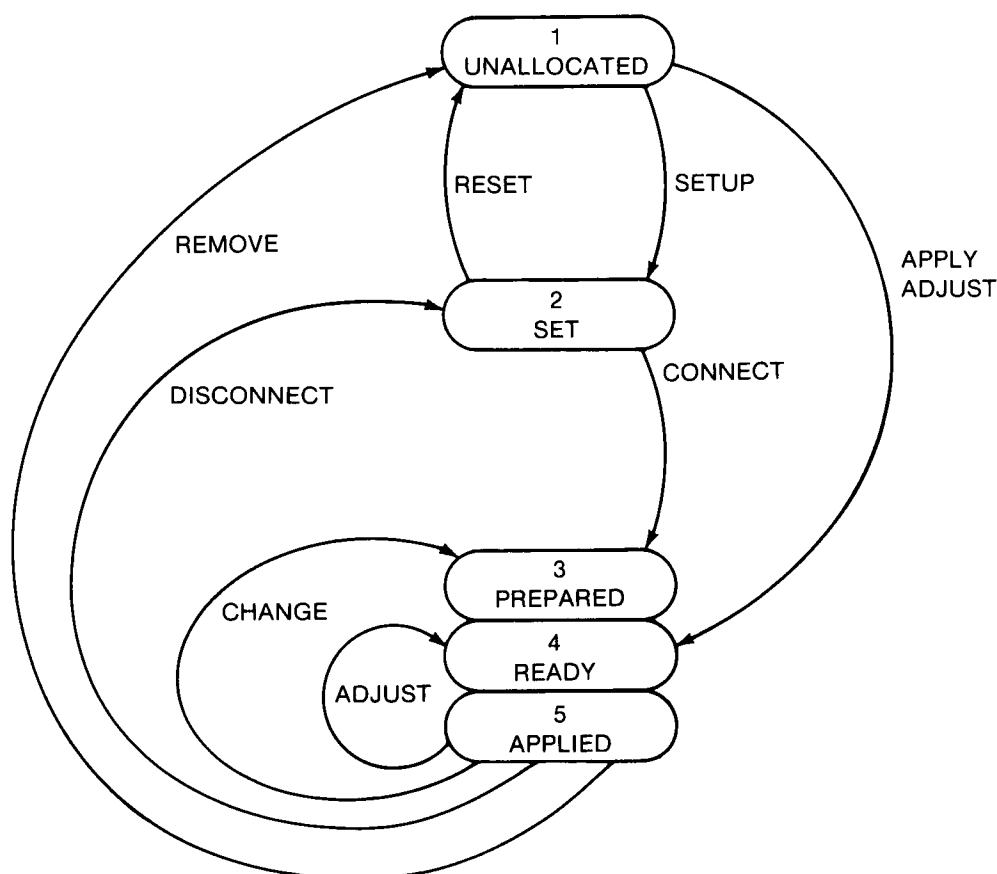


Fig 6
State Diagram for a Source or Load Virtual Resource with
Event Monitor Using Single-Action and Multi-Action Verbs