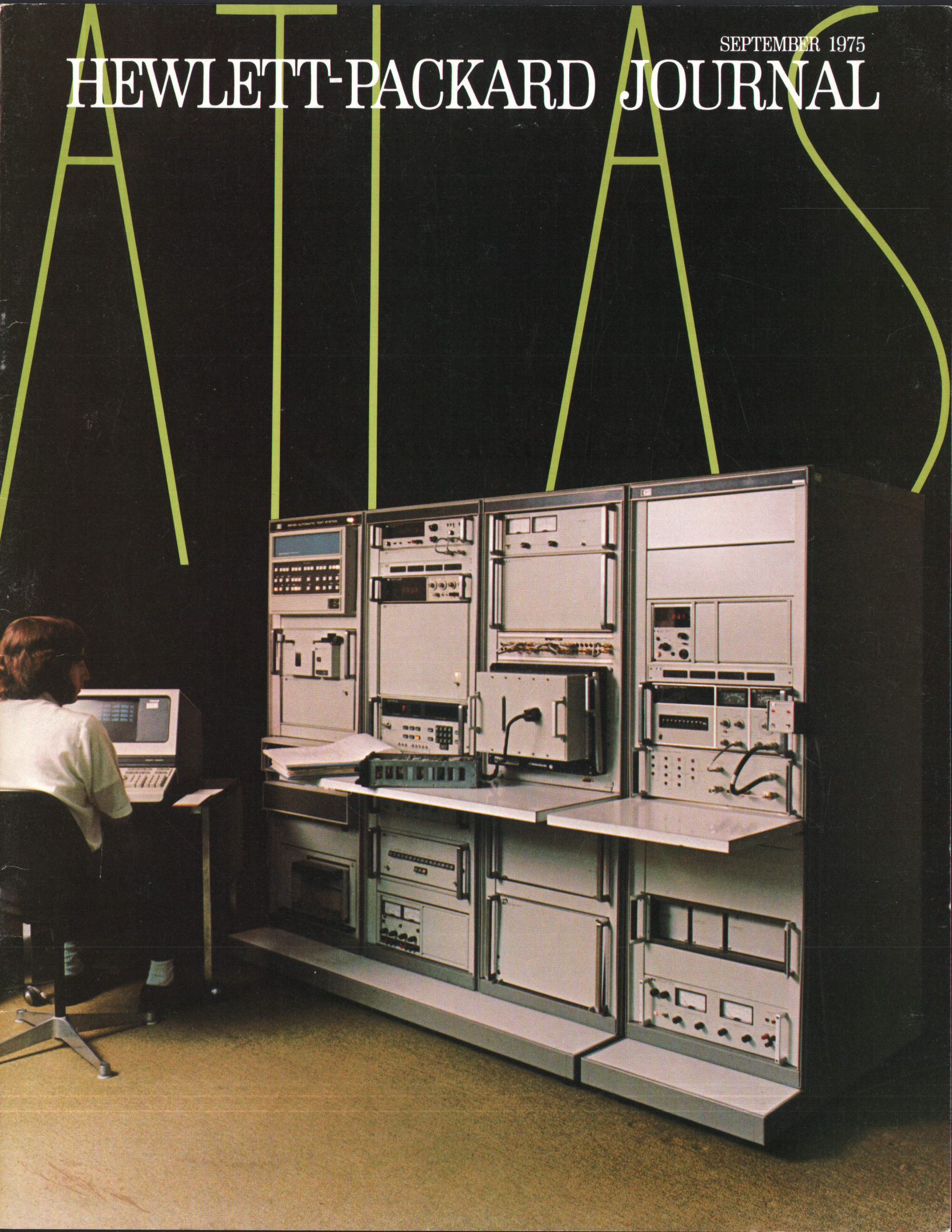
SEPTEMBER 1975

HEWLETT-PACKARD JOURNAL



ATLAS: A Unit-Under-Test Oriented Language for Automatic Test Systems

An engineer can write test procedures in ATLAS without detailed knowledge of the system that will do the testing. HP's new ATLAS compiler is the first comprehensive implementation of what is fast becoming a world-wide standard test language.

by William R. Finch and Robert B. Grady

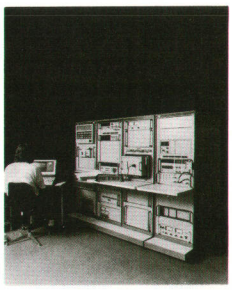
ATLAS is an English-like language for writing test procedures for electronic equipment. It is designed to be easily understood by programmers, engineers, and technicians, and its syntax (grammar) is well structured to guarantee an unambiguous description of the test procedure, one that can be translated by a computer into instructions that control automatic test equipment.

ATLAS is an acronym for Abbreviated Test Language for Avionics Systems. Originally designed, as its name implies, for testing avionics equipment, it is fast becoming a world-wide standard language for general-purpose automatic testing. The official standard for the language is maintained by Aeronautical Radio Incorporated (ARINC). Hewlett-Packard has participated in the standardization effort since 1970.

HP ATLAS

HP ATLAS is a new compiling system designed to run on HP 9500 Series Automatic Test Systems. The principal programming language for these systems has always been a form of interactive BASIC, now evolved to a high level of specialization for control of automatic test systems and called ATS BASIC. HP ATLAS is a higher-level language than ATS BASIC in much the same way that FORTRAN and COBOL are higher-level languages than assembly language. Where the ATS BASIC programmer must specify in detail how to set up individual test instruments and route signals, the ATLAS programmer need only be concerned with the requirements of the unit under test (UUT). ATLAS test procedures are independent of the specific automatic test system that will perform the tests. An ATLAS test procedure for a particular UUT can be employed by many users who may have different system configurations. Clearly this makes it much easier to solve applications software problems that are common to several users, and this is one of the principal reasons for implementing ATLAS on 9500 Series Systems.

The HP ATLAS Compiler analyzes test procedures written in ATLAS and generates segmented test programs in ATS BASIC. It can produce ATLAS listings or mixed listings with ATLAS and ATS BASIC statements properly interleaved to aid in program debugging. Subroutines written in ATS BASIC,



Cover: *Test procedures for Hewlett-Packard 9500-Series Automatic Test Systems like this Model 9510D can now be written in ATLAS (Automatic Test Language for Avionics Systems), a high-level language that relieves the test procedure writer of the need for detailed knowledge of the test system. An ATLAS test procedure for a particular unit under test can be run on any 9500 system capable of executing it, regardless of the precise system configuration.*

In this Issue:

ATLAS: A Unit-Under-Test Oriented Language for Automatic Test Systems, by William R. Finch and Robert B. Grady ..... **page 2**

Automatic 4.5-GHz Counter Provides 1-Hz Resolution, by Ali Bologlu ..... **page 14**

A New Instrument Enclosure with Greater Convenience, Better Accessibility, and Higher Attenuation of RF Interference, by Allen F. Inhelder ... **page 19**

FORTRAN, or assembly language may be called from HP ATLAS.

HP ATLAS is compatible with and meets the standards of ARINC ATLAS 416-10. It is the first implementation of a comprehensive subset of this official standard and is not just a highly adapted pseudo-ATLAS. Because it is unit-under-test-oriented, electronics and avionics suppliers can use ARINC-ATLAS-compliant procedures on HP 9500 Series Systems with a minimum of program rewriting and personnel retraining.

Introduction to ATLAS

An example of an ATLAS statement is:

```
APPLY, DC SIGNAL,
    VOLTAGE 10 V ERRLMT +−0.1 V,
        CNX HI J4-1 LO J3 $
```

The English equivalent of this statement is: apply 10 ± 0.1 Vdc between UUT pins J4-1 and J3. In this example, APPLY defines what to do with the signal, the phrase DC SIGNAL, VOLTAGE 10 V ERRLMT +−0.1 V describes the type and characteristics of the applied signal, and CNX HI J4-1 LO J3 names the UUT input connections.

Another example of an ATLAS statement is:

```
MEASURE, (FREQ ERRLMT +−0.002 MHZ), AC SIGNAL,
    FREQ RANGE 4 MHZ TO 6 MHZ,
    VOLTAGE RANGE 1 V TO 1.5 V,
        CNX HI J1-1 LO J1-2 $
```

The words MEASURE (FREQ) describe the required action, AC SIGNAL specifies the type of signal, the characteristics for frequency and voltage provide the signal description, and the CNX field specifies the UUT output pins. This statement defines the test function as: measure the frequency, with required measurement accuracy of ±0.002 MHz, of an ac signal having expected amplitude and frequency characteristics of 1 to 1.5 Vac at 4 to 6 MHz present at UUT output connector pins J1-1 and J1-2.

The first example above is classified as a source-type ATLAS signal-oriented statement. The statements in this class provide the stimulus function. The second example is a sensor-type ATLAS signal-oriented statement. These provide the response measurement function. A sequence of ATLAS statements must describe the test procedure in sufficient detail to allow the HP ATLAS Compiler to generate an executable program for an automatic test system. The readability of ATLAS statements makes it possible for the same test procedure to be performed manually using bench instruments.

ATLAS uses commas to separate fields within a statement. There will usually be statement numbers preceding each statement. The statement can continue on separate lines at any point except where ending the line would split a word. The symbol $ is the statement terminator. The HP ATLAS implementation does not require commas as separators, although at least one space between words is required where the ARINC specification requires a space or comma.

Other ATLAS statements, called procedure-oriented statements, are used to analyze and output test results and control the execution sequence of a test procedure. These include facilities for variable assignments, mathematical calculations, branching, looping, block structure, test operator interaction, and other program control functions similar to other high-

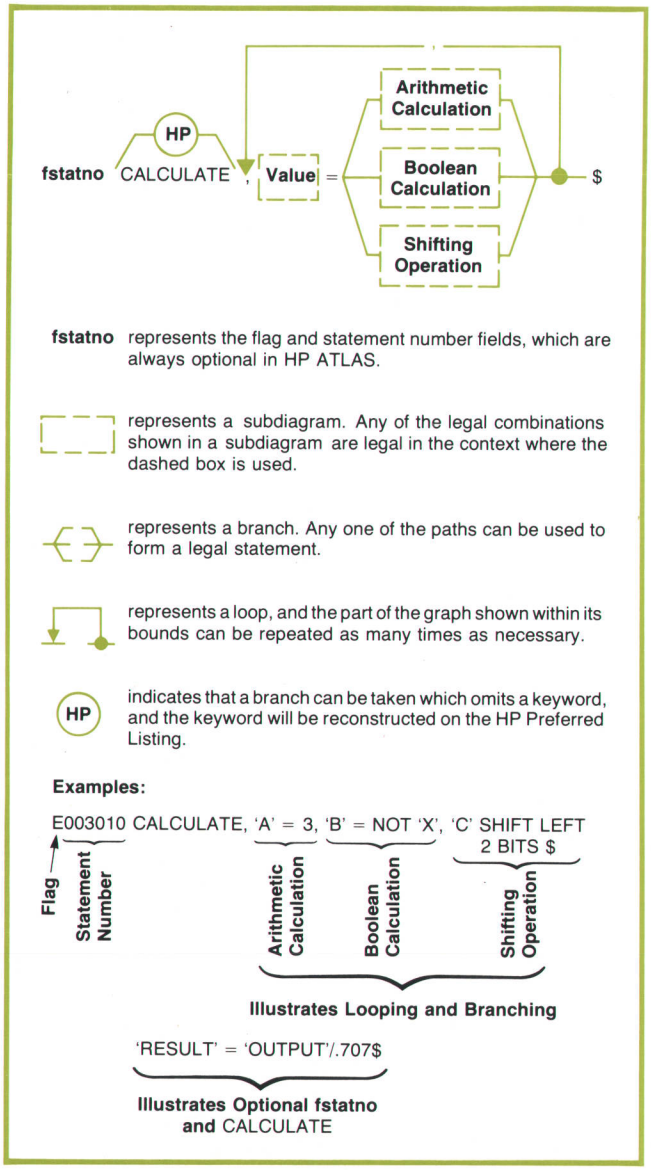


Fig. 1. *A syntax diagram for a* CALCULATE *statement, a typical ATLAS procedure oriented statement. Syntax diagrams specify the rules for constructing all the legal forms of ATLAS statements.*

level computer languages. The complete list of HP ATLAS verbs, nouns, and modifiers is given in the table on page 12.

ATLAS statements have many optional forms. The rules for constructing them are specified by syntax diagrams, such as that shown in Fig. 1 for the verb CALCULATE.

Automatic Resource Allocation

Now let's examine a typical ATLAS statement to understand what the ATLAS compiler is required to do.

```
MEASURE (VOLTAGE), DC SIGNAL,
      VOLTAGE MAX 10V,
            CNX HI J1-15 LO J1-10  $
```

First of all, MEASURE implies that one or more instruments must measure some electronic or non-electronic signal characteristic or event. Compare such a verb with the FORTRAN verb READ, and you can see that both expect a result from a device external to the computer. But where a FORTRAN READ represents simply a data transfer, MEASURE implies additional setups and adjustments to potentially complex instrumentation.

DC SIGNAL and VOLTAGE MAX 10 V provide enough information to determine what instruments in a system can perform the measurement. This implies a choice among a variety of instruments, especially in the simple case in our example of a low-level dc signal. The CNX field implies knowledge of all switching and wiring in the complete test system and adapter/interface, and the ability to select signal paths from the connections J1-15 and J1-10 on the UUT to pins HI and LO on the measurement instrument.

Combining all of these concepts together into a single high-level language statement and taking into account parallel signal activity demands considerable intelligence for a compiler, and the processes it must perform are called automatic resource allocation. It is this capability of the HP ATLAS Compiler that makes it possible to program a test without specific reference to the system instrumentation and switching hardware.

Two Processors Aid Compiler

To translate the signal-oriented statements of an ATLAS test procedure into an ATS BASIC program, the compiler requires two kinds of information in addition to the test procedure (see Fig. 2). First, the capabilities and method of programming the instrumentation and switching hardware of the target automatic test system must be known. Second, the characteristics of the interface between the target system and the specific unit under test must be known. HP ATLAS provides two processors that accept this information in high-level languages and automatically translate it into a form that can be used by the HP ATLAS Compiler. Fig. 3 shows the processor/compiler relationship.

The ATE Processor accepts descriptions of instruments, relays, interface panels, internal wiring, and methods of programming, all in a language very similar to ATLAS (see Fig. 4). Its output consists of data files that model the system's interface, switching,

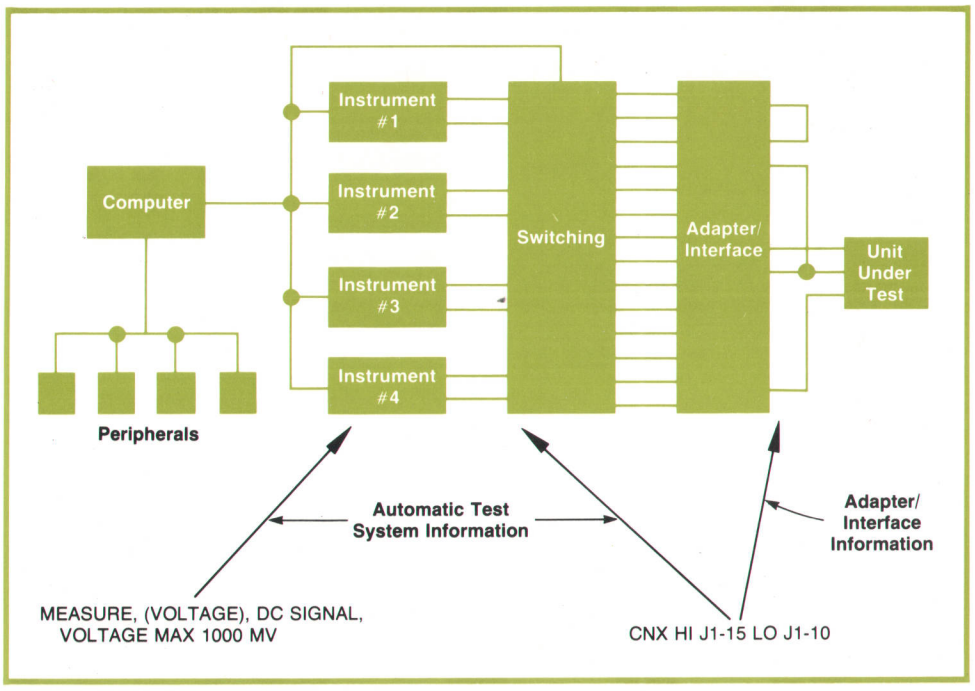


Fig. 2. *A typical automatic test system configuration showing the relationship between parts of an ATLAS statement and parts of the system. The compiler must be given information about the system so it can automatically allocate system resources to execute the ATLAS program.*

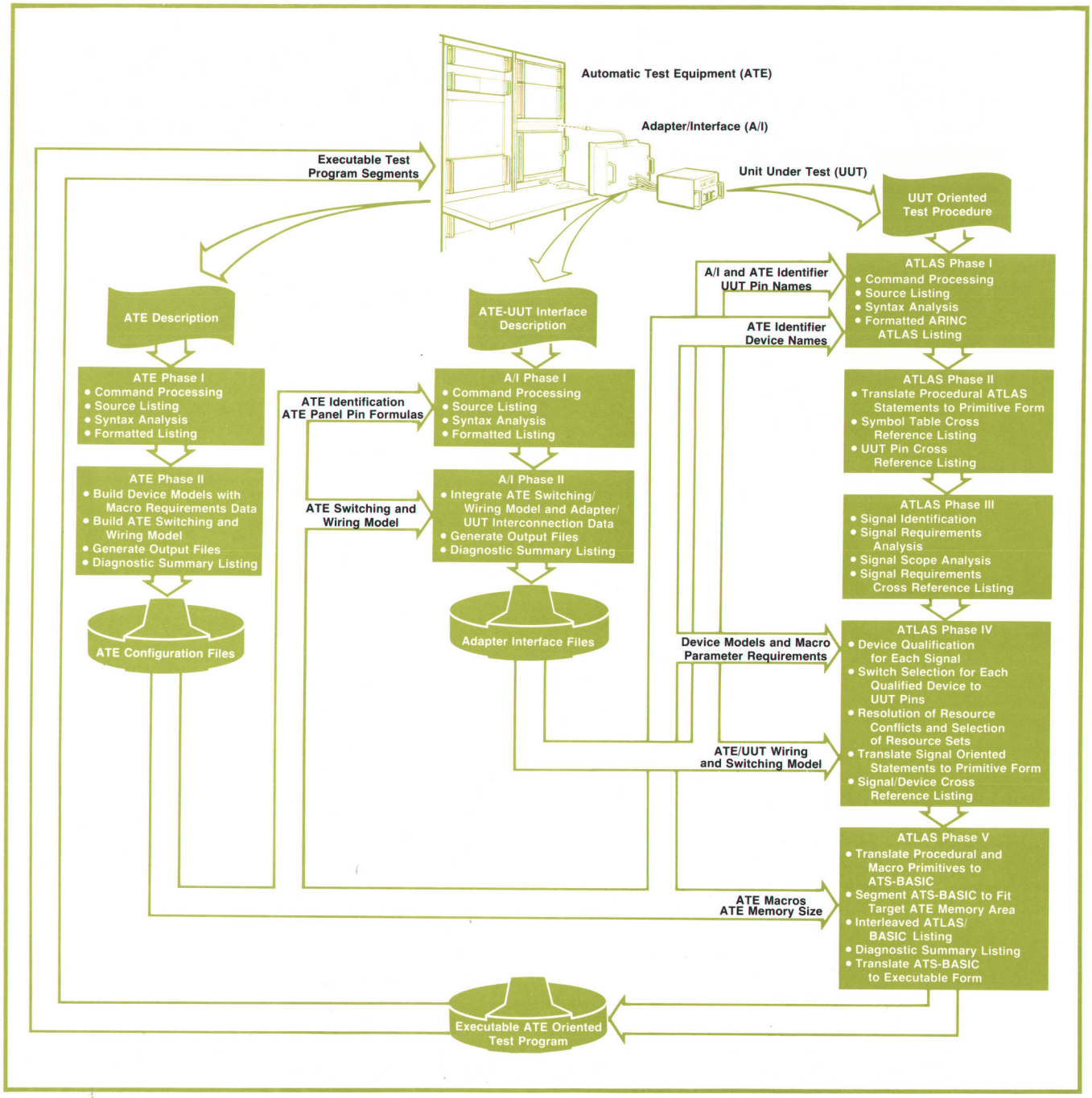


Fig. 3. *The HP ATLAS Compiler has five phases and makes use of information from two auxiliary processors. The ATE Processor accepts a description of system capabilities (supplied with each HP 9500 Series System) and generates disc files for the A/I Processor and the HP ATLAS Compiler. The A/I Processor accepts a user-written description of the adapter/interface between the system and the unit under test (UUT), integrates the description with the system switching and wiring model supplied by the ATE Processor, and generates disc files for the HP ATLAS Compiler. The compiler uses the appropriate ATE and A/I files to translate UUT-oriented ATLAS test procedures to system-oriented ATS BASIC test programs.*

and instrument capabilities and the method of programming the system. The information required by the ATE Processor is supplied with HP 9500 Systems.

The Adapter/Interface (A/I) Processor accepts a description of the connections between the test system and the UUT (see Fig. 4). Its output consists of data files that are a composite model of the system switching capabilities and all the interconnections within the system and between the system interface and the UUT interface.

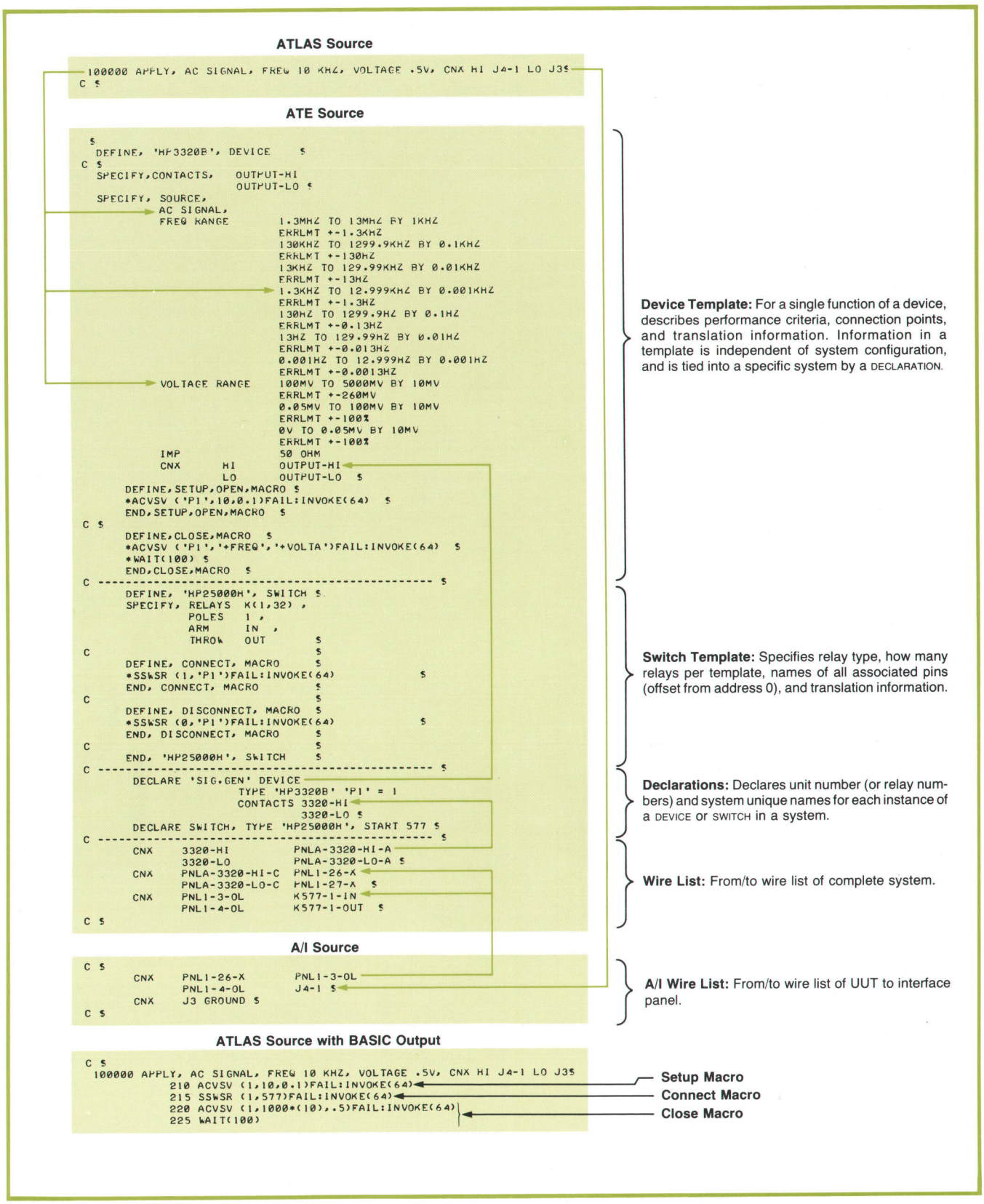


Fig. 4. *Samples of input code for the ATE and A/I Processors showing relationships with ATLAS source code and the equivalent ATS BASIC output. Arrows at the left show how the ATLAS Compiler compares signal characteristics with device capabilities. Arrows at the right show how the compiler traces signal paths between a device and the unit under test.*

Three Compiler Modes

The HP ATLAS Compiler uses the data produced by the ATE and A/I Processors to select instruments and switching hardware and to generate signal-oriented object code. The compiler can be operated in any of three primary modes.

For an initial check of an ATLAS test procedure, the compiler can be operated with neither the target system nor the adapter/interface defined. The compiler performs a considerable amount of analysis on the procedure's syntactic correctness and signal requirements. Operating in this mode, the compiler is really a test procedure analyzer, since it generates no object code. The signal requirements cross reference and diagnostic information that are produced are, however, of considerable value during test procedure development.

If an ATLAS test procedure is compiled for a specific target system, but the adapter/interface is not defined, additional analysis is performed to determine the degree of compatibility between the signal requirements in the test procedure and the available instrumentation in the system description. The output, a signal requirements/device cross reference and diagnostic summary, reflects this compatibility. Operating the compiler in this mode, the user can determine whether or not the target system can perform the test procedure as written, and can determine the system-to-UUT interconnections that are required.

If an ATLAS test procedure is compiled with a defined system and adapter/interface, the compiler will generate the appropriate ATS BASIC program for performing the test procedure.

Compiler Functional Description

The HP ATLAS Compiler is organized into five phases, each of which performs a group of related operations that contribute to the overall translation process (Fig. 3). In general, these subprocesses operate on the internal representation of the ATLAS test procedure, often in conjunction with data from the A/I Processor, the ATE Processor, and the earlier compiler phases. Each subprocess either modifies the internal representation of the test procedure, thus contributing directly to the translation process, extracts and processes related information from the test procedure, thus contributing indirectly to the translation process, or formats information for the purpose of generating printed listings.

Phase I

The primary purpose of the first phase is to process compiler commands and to detect and identify test procedure errors in word construction, required punctuation, statement composition, or statement grouping. Some compiler commands have an immediate effect, directing phase I to obtain source code from particular input devices or disc files, or causing source listings or files to be generated. Other commands that affect later processes are noted for future reference.

The test procedure is input in the form of an ASCII (American Standard Code for Information Interchange) character stream from disc files, input devices, or the operator's console keyboard. Input characters are grouped together to form syntactic units. The compiler replaces each unique syntactic unit by a unique number, called a token, to facilitate efficient processing. The various syntactic units, such as ATLAS keywords (or abbreviations: the compiler requires only three or more leading characters of a keyword that uniquely identify it), punctuation, numeric constants, variable names, UUT pin names, and so on, are assigned token values in numeric groupings or ranges so the type of syntactic unit can be easily identified by the range of the token value. For example, verbs are assigned token values between 100 and 165.

The resulting token stream is then compared with acceptable ATLAS syntactic forms. During this process, omissions of certain parts of strict ARINC ATLAS syntax, considered optional in HP ATLAS, are detected and the missing tokens are inserted into the token stream for conformity to ARINC ATLAS. Statement numbers, commas, and some keywords fall into this category. If required, the statements are renumbered to comply with the ARINC requirement for increasing statement numbers. In addition to statement syntax checking, syntactic requirements that apply to the relationships between two or more statements, or to the test procedure in general, are verified. A listing of the test procedure in preferred ARINC ATLAS format is then generated (see Fig. 5). This listing includes full spellings, punctuation, and block indentations.

Phase II

The second compiler phase translates all procedural statements (non-signal-oriented statements) into an intermediate form called primitive code. This primitive code is independent of any particular object code, but is easily translated into any lower-level language. The translation of procedural statements is quite conventional in many respects, and a detailed discussion will not be given. One unusual aspect of the translation, however, is that ATLAS PROCEDUREs are expanded "in-line" each time they are called by a PERFORM statement, instead of being expanded once and thereafter treated as subroutines to be invoked by PERFORM statements. This is necessary for two reasons. One is that UUT pins may be used as procedure parameters. If the procedures were

```
$LIST BASIC
 000000 BEGIN, ATLAS PROGRAM '09540-90017'$
     01 DEFINE, 'INPUT', SOURCE $
     02 DECLARE, DECIMAL, LIST, 'VOUT'(5), 'FREQ'(5)$
     03 DECLARE, DECIMAL, LIST, 'ULIM'(5), 'LLIM'(5)$
C    $
     04 FILL, 'VOUT', 'ULIM', 'LLIM',
            (1).10, .105, .095,
            (2).05, .055, .045,
            (3).01, .013, .007,
            (4).005, .007, .003,
            (5).001, .003, .00$
C    $
C --------BEGIN TEST PROCEDURE--------- $
C    $
     05 FOR 'INDEX'=1 THRU 5, THEN $
     06      ADJUST, AC SIGNAL,
                 'INPUT',
                 VOLTAGE .10 V,
                 FREQ RANGE 50 KHZ TO 200 KHZ BY 1 KHZ RATE 10 KHZ
                 / SEC,
                      CNX HI J4-1 LO J3$
     07      TO REACH, (VOLTAGE), AC SIGNAL,
                 NOM 'VOUT'('INDEX') V UL 'ULIM'('INDEX') V LL
                 'LLIM'('INDEX') V,
                 VOLTAGE RANGE 0 V TO .12 V,
                      CNX HI J4-8 LO J3$
     08      MEASURE, (FREQ INTO 'FREQ'('INDEX')), AC SIGNAL,
                 VOLTAGE .10 V,
                 FREQ RANGE 50 KHZ TO 200 KHZ,
                      CNX HI J4-1 LO J3$
     09      END, FOR $
C    $
     10 RECORD, 'FREQ', (1 THRU 5) TO EXTERNAL 'D1000'$              EX
     11 FINISH $
     12 TERMINATE, ATLAS PROGRAM $
```

Fig. 5. *Typical ATLAS program listing in preferred ARINC format, produced by compiler phase I. Extensions to ARINC ATLAS are flagged by an EX in the righthand column.*

treated as subroutines, the UUT pin parameters would have to be treated as variables within the procedures. Because signal switching is decided at compile time, implementation of UUT pin variables is not possible. Second, signal-oriented statements must be considered with respect to all other signals that are active concurrently when allocation of system resources is done automatically. Because the signal activity context may differ between one PERFORMANCE of a PROCEDURE and another, the same signal-oriented statement may translate differently in different instances.

At the conclusion of phase II, alphabetized cross reference listings of user-defined symbolic labels and UUT pin names are generated if activated by the $XREF LBL command.

Phase III

The third compiler phase does a detailed analysis of the signal activity that will be present at the UUT/system interface throughout the performance of the test procedure. This analysis results in a highly struc-

Minimizing Test Program Expenses

The benefits of automatic test equipment are well known, the principal ones being faster, more repeatable testing even with relatively unskilled operators.

However, test program preparation can be expensive. There are many hidden costs. Early in the design phase of HP ATLAS every aspect of that expense was examined not only to maximize the value of the final product, but also to minimize the development cost.

The most visible element of a system expenditure is the hardware environment in which the compiler and output programs operate. HP ATLAS operates on HP 9500 Series Automatic Test Systems. Thus computer and peripheral expenses are held to minicomputer levels while additional processing power is provided by using a disc operating system for compilations. The compiler is flexible enough to operate within different configurations and to compile programs for execution on various test station configurations.

A second test preparation expenditure is UUT/system adapter costs and the associated costs of integrating special hardware and switching into the system for special UUT needs. The ability to describe the system configuration and the adapter/interface configuration in high-level language terms helps to minimize the integration costs. Also, one phase of the compiler is dedicated to signal requirements analysis and produces listings that can be used to aid in adapter design, or even in selecting the most appropriate system for testing a UUT.

The third expenditure and potentially the largest is direct programming expenses. This expense is addressed by the ATLAS language, which is English-like, uses engineering terms, and can be used without detailed knowledge of the system hardware, thereby simplifying the writing of test programs. But this alone is not necessarily enough, because high-level languages require more computer processing time than lower-level languages. To minimize this effect, the HP ATLAS Compiler helps reduce the length of compiles by providing partial-compilation capability and several levels of well-designed error diagnostics.

Potentially the greatest source of savings in an ATLAS system is that many common system-dependent errors are minimized by providing common instrumentation and switching routines in preprocessed data files. These errors include

- wrong switching
- incorrect settling delays
- wrong device setups or ranges
- wrong tolerances because of dimensional changes
- device interactions.

The reduction of this broad class of problems reduces costs by shortening the time needed for program verification. Once a program compiles successfully, the probability of execution is good.

Many users of automatic test equipment encounter a fourth cost because they have many different systems that must perform the same tests. Using test-system-dependent languages, a separate program would be required for each type of system. If a system-independent language like ATLAS is used, a program written for one system is transportable to another with only minor modifications, if any.

The last major area of system cost is support. High-level languages generally minimize this cost, because they inherently contain more support documentation value than lower level languages. HP ATLAS further aids this task by outputting a preferred listing using standard ARINC ATLAS spellings, punctuation, and formatting. System-independent ATLAS also reduces support costs by reducing the number of different programs required when different systems perform the same tests.

tured data file which represents the stimulus signals required by the UUT, the UUT signals that are to be measured, and the loads that are to be applied to the UUT. This analysis of UUT signal requirements is done independently from any system configuration information, but results in a precise specification of the system capabilities required to perform a given test procedure.

To determine the UUT signal requirements, all signal-oriented statements are processed to determine how many signals exist and which statements refer to each signal, whether each signal is a SOURCE, SENSOR, or LOAD, the signal type or noun, the signal characteristics, normalized characteristic dimensions or units, range of values, accuracy requirements, and the UUT pins associated with the signal.

In addition, all statements that affect the execution sequence of a test procedure (IF, FOR, WHILE, REPEAT, and GO TO statements) are examined and the range of statements over which each signal is active is determined. This information, called signal scope, is then used to identify signal concurrency, that is, signals that have overlapping periods of activity when the test procedure is executed.

An optional listing, activated by the $XREF SIG command, provides a formatted representation of the signal requirements analysis, which is highly useful for evaluating the signal activity in an ATLAS test procedure.

Phase IV

The primary purpose of the fourth compiler phase is to determine the system resources that are to be used for each signal identified and specified by phase III, and to then translate the signal-oriented statements into primitive code form.

First, for each signal requirement, each system device model that compares favorably with the signal requirement is noted along with device parameter information. This results in a device candidate list for each signal requirement. Second, the system/UUT switching and wiring model is used to determine whether or not the terminals on the device candidates can be connected to the required UUT terminals. If they can, the required switching and signal path resources are recorded, along with the device, as the set of required resources for the candidate. If signal paths from candidate device terminals to UUT terminals cannot be established, the candidate is disqualified from further consideration.

For each signal requirement having one or more qualified candidates (sets of resources that satisfy the signal requirements), a tentative candidate assignment is made. The specific resources associated with the tentative candidates for concurrent signals are cross-checked to detect allocation conflicts. If allocation conflicts occur and either of the concurrent signals has candidate alternatives, the tentative candidate assignment is changed in an attempt to resolve the resource conflict. If a combination of candidates cannot be found which resolves a resource conflict between concurrent signals, one of the signal requirements is identified as being unallocatable. Conflict detection and resolution continues until allocation is completed. All surviving tentative allocations then become actual allocations.

Once resource sets are allocated for all signal requirements, primitive code is generated for each signal-oriented statement, based on the resource set associated with the signal requirement of which the signal-oriented statement is a part. Following the generation of primitive code, an optional signal/device cross reference listing may be generated (Fig. 6). This is similar to the signal cross reference listing of Phase III, but contains additional information about device qualification and disqualification and allocations for each signal requirement.

```
                    SIGNAL REQUIREMENTS CROSS REFERENCE


                              SENSOR CROSS REFERENCE

2: AC SIGNAL (VOLTAGE),
     CHARACTERISTICS:
        VOLTAGE  MIN 0 V; MAX .12 V;
     REFERENCES:
        TO AT 000007; CNX HI J4-8 LO J3 ;
     SCOPE OF ACTIVITY:
        000007
     DEVICE CANDIDATES:
       *'DVM' FUNCTION 2A  SELECTED
        'DVM' FUNCTION 2C  DISQUALIFIED
           NO SIGNAL PATH FOR CNX HI J4-8

3: AC SIGNAL (FREQ),
     CHARACTERISTICS:
        VOLTAGE  .1 V;
        FREQ  MIN 50E3 HZ; MAX 200E3 HZ;
     REFERENCES:
        MEASURE AT 000008; CNX HI J4-1 LO J3 ;
     SCOPE OF ACTIVITY:
        000008
     DEVICE CANDIDATES:
       *'COUNTER' FUNCTION 1 SELECTED
        'COUNTER' FUNCTION 2A  QUALIFIED
        'COUNTER' FUNCTION 2B  DISQUALIFIED
           VOLTAGE NOT SPECIFIED FOR CANDIDATE


                              SOURCE CROSS REFERENCE

1: AC SIGNAL  'INPUT',
     CHARACTERISTICS:
        VOLTAGE  .1 V;
        FREQ  MIN 50E3 HZ; MAX 200E3 HZ; RESOLUTION 1000 HZ; RATE
        10000 HZ
     REFERENCES:
        DEFINE AT 000001;
        ADJUST AT 000006; CNX HI J4-1 LO J3 ;
        FINISH AT 000011;
     SCOPE OF ACTIVITY:
        FROM 000005 TO 000011
     DEVICE CANDIDATES:
        'SIG.GEN' FUNCTION 1 DISQUALIFIED
           VOLTAGE NOT SPECIFIED FOR CANDIDATE
       *'SIG.GEN' FUNCTION 2 SELECTED
        'HFSIG.GEN1' FUNCTION 1 DISQUALIFIED
           VOLTAGE NOT SPECIFIED FOR CANDIDATE
           FREQ MAX OUT OF CANDIDATE RANGE
           FREQ MIN OUT OF CANDIDATE RANGE
        'WORD.GEN1' FUNCTION 1 DISQUALIFIED
           VOLTAGE NOT SPECIFIED FOR CANDIDATE
           FREQ MAX OUT OF CANDIDATE RANGE
        'WORD.GEN2' FUNCTION 1 DISQUALIFIED
           VOLTAGE NOT SPECIFIED FOR CANDIDATE
           FREQ MAX OUT OF CANDIDATE RANGE
```

Fig. 6. *Signal/device cross reference is produced by compiler phase IV.*

Phase V

The fifth and final phase of the compiler translates the primitive code stream into executable object code; specifically ATS BASIC and/or ATS BASIC intermediate code. This process consists of a number of interrelated operations, including expansion of device programming procedures as specified by the primitive code, translation of procedural primitives, mapping ATLAS variables and compiler generated variables into BASIC variables, and segmenting object code to fit the target system core environment. Also generated are an initialization code sequence for initializing variables and opening any required disc files, and an operator interaction sequence for selection of optional starting points in the test procedure.

If activated by a $LIST BASIC command, phase V generates an interleaved listing of ATLAS and ATS BASIC (Fig. 7). Then a summary of any errors or warnings detected during the compilation is produced. If either a $OBJECT <file-name> or $BASIC <file-name> command has been given, a file of commands for the ATS BASIC interpreter is generated for directing the further processing of the segmented test program. Then the input phase of the ATS BASIC interpreter completes the translation process.

Designed for Efficient Use

The HP ATLAS Compiler was designed assuming two types of users: one or more ATLAS test procedure writers who are technicians or engineers and are experts in understanding UUT's, and an ATE and A/I specialist who is familiar with the system and its interfaces and who is responsible for system upgrades, methods of use, and adapter design. These two users can be the same individual, but on a complex system with multiple users it probably makes sense to assign one ATE and A/I specialist.

The compiler offers the user a number of options that facilitate its efficient use. For example, assume the user has prepared a test procedure using abbreviated ATLAS and has punched it on paper tape or cards. To this source program the user can add compiler command statements that control the options available (see table, page 12). This can be done either by typing the commands at the operator's keyboard or by editing the source program. For the initial run the user would probably add the following options (compiler commands begin with the $ symbol):

```
$SAVE ATLAS S20000
$THRU MAX = 1
C     PROGRAM BEGINS HERE $
  BEGIN $
    .
    .  (ATLAS Test Procedure)
    .
  TERMINATE $
```

```
          110 OPEN(7,1000,G(6))
          115 IF G(6)>0 INVOKE(64)
          120 RESET(7,0)
$LIST BASIC
 000000 BEGIN, ATLAS PROGRAM '09540-90017'$
      01 DEFINE, 'INPUT', SOURCE $
      02 DECLARE, DECIMAL, LIST, 'VOUT'(5), 'FREQ'(5)$
      03 DECLARE, DECIMAL, LIST, 'ULIM'(5), 'LLIM'(5)$
C  $
      04 FILL, 'VOUT', 'ULIM', 'LLIM',(1).10, .105, .095,(2).05, .055,
         .045,(3).01, .013, .007,(4).005, .007, .003,(5).001, .003,
         .00 $
          125 LET B(1)=.1
          130 LET D(1)=.105
          135 LET E(1)=95E-3
          140 LET B(2)=50E-3
          145 LET D(2)=55E-3
          150 LET E(2)=45E-3
          155 LET B(3)=10E-3
          160 LET D(3)=13E-3
          165 LET E(3)=7E-3
          170 LET B(4)=5E-3
          175 LET D(4)=7E-3
          180 LET E(4)=3E-3
          185 LET B(5)=1E-3
          190 LET D(5)=3E-3
          195 LET E(5)=0
C  $
C --------BEGIN TEST PROCEDURE--------- $
C  $
E $
      05 FOR 'INDEX'=1 THRU 5, THEN $
          110 LET A(1)=1
          115 GO TO 125
          120 LET A(1)=A(1)+1
          125 IF A(1)>5 GO TO 410
      06 ADJUST, AC SIGNAL, 'INPUT', VOLTAGE .10 V, FREQ RANGE 50 KHZ
         TO 200 KHZ BY 1 KHZ RATE 10 KHZ / SEC, CNX HI J4-1 LO J3$
      07 TO REACH, (VOLTAGE), AC SIGNAL, NOM 'VOUT'('INDEX') V UL
         'ULIM'('INDEX') V LL 'LLIM'('INDEX') V, VOLTAGE RANGE 0 V TO
         .12 V, CNX HI J4-8 LO J3$
          130 ACVSV (1,10,0.1)FAIL:INVOKE(64)
          135 SSWSR (1,577)FAIL:INVOKE(64)
          140 LET T(1)=50
          145  W(3) =.1
          150 IF W(3) <50E-6 LET W(3) =50E-6
          155 ACVSV (1,1000*(T(1)), W(3) )FAIL:INVOKE(64)
          160 WAIT(100)
          165  W(1) = 1
          170  W(2) =1000*(T(1))
          175 DVMSU (2,1000,0)FAIL:INVOKE(64)
          180 SSWSR (1,570)FAIL:INVOKE(64)
          185 DVMSU (2,0,0)FAIL:INVOKE(64)
          190  W(4) = 1
          195 LET T(2)=1E36
          200  W(3) =.1
          205 IF W(3) <50E-6 LET W(3) =50E-6
          210 IF NOT  W(1)  GO TO  225
          215 ACVSV (1,1000*(T(1)), W(3) )FAIL:INVOKE(64)
          220 WAIT(100)
          225  W(2) =1000*(T(1))
          230 WAIT(1/10)*1000)
          235 WAIT(100)
          240 DVMMU (2,V(1),1000)FAIL:INVOKE(64)
          245 LET M(1)=V(2)
          250 LET T(3)=ABS(B(A(1))-M(1))
          255 IF T(3)>=T(2) GO TO 275
          260 LET T(2)=T(3)
          265 LET T(4)=T(1)
          270 LET T(5)=M(1)
          275 LET T(1)=T(1)+1
          280 IF T(1)<=200 GO TO 200
          285 LET T(1)=T(4)
          290  W(3) =.1
          295 IF W(3) <50E-6 LET W(3) =50E-6
          300 IF NOT  W(1)  GO TO  315
          305 ACVSV (1,1000*(T(1)), W(3) )FAIL:INVOKE(64)
          310 WAIT(100)
          315  W(2) =1000*(T(1))
          320 LET M(1)=T(5)
          325 CMPAR(0,M(1),E(A(1)),D(A(1)),N0,H1,L0)
          330 DVMSU(2,1000,0)FAIL:INVOKE(64)
          335  W(4) = 0
          340 SSWSR (0,570)FAIL:INVOKE(64)
 000008 MEASURE, (FREQ INTO 'FREQ'('INDEX')), AC SIGNAL, VOLTAGE
         .10 V, FREQ RANGE 50 KHZ TO 200 KHZ, CNX HI J4-1 LO J3$
          345 CTRSI (3,0,3,0,2)FAIL:INVOKE(64)
          350 SSWSR (1,582)FAIL:INVOKE(64)
          355 CTRSI (0,0,3,0,2)FAIL:INVOKE(64)
          360 CTRSF (3,3,1)FAIL:INVOKE(64)
          365  W(22) =1
          370 WAIT(100)
          375 CTRMF (4,M(1),M(2))FAIL:INVOKE(64)
          380 LET M(1)=M(1)/1000
          385 LET C(A(1))=M(1)
          390 CTRSI (3,0,3,0,2)FAIL:INVOKE(64)
          395  W(22) = 0
          400 SSWSR (0,582)FAIL:INVOKE(64)
      09 END, FOR $
          405 GO TO 120
C  $
      10 RECORD, 'FREQ', (1 THRU 5) TO EXTERNAL 'D1000'$                EX
          410 DRITE(7,0,C(1),5,G(6))
          415 IF G(6)>0 INVOKE(64)
 000011 FINISH $
          420 INVOKE(66)
          425 RESET(7,1)
          430 CLOSE(7,G(6))
          435 IF G(6)>0 INVOKE(64)
          440 STOP
      12 TERMINATE, ATLAS PROGRAM $
          445 INVOKE(66)
          450 RESET(7,1)
          455 CLOSE(7,G(6))
          460 IF G(6)>0 INVOKE(64)
          465 STOP
```

Fig. 7. *Listing of an ATLAS test procedure interleaved with the equivalent ATS BASIC code. This listing is generated by compiler phase V.*

This compilation will execute quickly because the $THRU command will terminate compilation at the first logical point after the first error is detected. This is generally at the end of a compiler phase, or after the preferred listing is generated and the preferred source saved, on disc file S20000 in this case. From the preferred listing, the user can resolve keypunch errors, syntax errors (formal statement structural errors) and undefined statement numbers, and can prepare a correction tape or deck referencing the statement numbers on the preferred listing.

For the second compilation the following options might be used:

```
$SAVE ATLAS S20001
$ATE D315                    These commands will be saved
$A/I D313                    with corrected ATLAS at S20001.
$XREF LBL, SIG
$THRU CONFIG
$UPDATE WITH RDR
$INSERT S20000
$END
```

This compilation will process source file S20000, as modified by the update tape, through the first four compiler phases. Errors involving relationships between statements, incorrect resource specifications, or errors in the A/I definition will be detected by this compilation.

After resolving any errors of this type, the next compilation can be expected to produce code that is ready to test. The commands would be:

```
$SAVE ATLAS S20000
$OBJECT B1000                This command is saved.
$LIST BASIC
$DEBUG
$UPDATE WITH RDR
$INSERT S20001
$END
```

Errors in an ATLAS test procedure that are discovered during execution of the ATS BASIC test program can be easily fixed using BASIC and the corrections noted for a single final ATLAS recompilation.

Top-Down Design

The optimal implementation of ATLAS on HP 9500 Series Systems required three large processors (ATE, A/I, and the ATLAS Compiler). This complex design task was efficiently accomplished by a top-down design which resulted in certain advantages to the user as well as the designer. While the purposes of the three processors are different, their source languages and organizations are similar in many respects. There are two advantages to this. First, the user interface to each processor, in terms of source language, command structure, operating procedures, and terminology, is consistent throughout the system. Second, the highly structured top-down implementation makes use of common modules to accomplish a large portion of the processing.

The two most important areas of commonality in the design are the executive control structure and the I/O interface (input/output to operating system and computer peripherals). The executive control program is the main program in each processor. It controls the execution of subprocessors, creates, opens, positions, and closes files for the subprocessors, and maintains a global data area for communications with the subprocessors. The executive control program for each processor is the same except for tabular data.

Because the ATLAS compiling system is highly dependent on the I/O and file structure of the operating system, the compiling system interfaces not directly with the operating system, but through a standardized interface that can be adapted to different operating systems. A considerable effort has been made to allow the transfer of the ATLAS system to a different environment with minimum effort, because it is anticipated that HP ATLAS will eventually be made available with operating systems other than the disc based operating system presently used by 9500 Series Automatic Test Systems.

Two major results of having common executive and I/O structures are worthy of note. First, new compiler sections or existing ones were easy to rearrange by simply changing tables in the executive. Second, compiler speed was improved during development by better than a factor of two by making changes only in the I/O section, which is where the compiler spends most of its time.

The next level of commonality occurs in the processing and listing sections of the processors. Common software modules use different data (keyword dictionaries and other tabular data) depending on whether the processing is for the ATLAS Compiler or the ATE or A/I Processors. Because of the commonality between the three processors and certain file structures, the same programs are able to generate a variety of listings. For example, the formatted listings from the ATE and A/I Processors, the preferred ARINC ATLAS listing, and the interleaved ATLAS/ATS BASIC listing are all generated by a common program. A common procedure is used throughout the compilation system to handle all error conditions. This processing handles user diagnostics (warnings, syntax errors, semantic diagnostics) as well as system errors (I/O errors, user file reference errors).

A very important compiler technique consists of analyzing source text using syntax graphs, which are

HP ATLAS Language Words and Compiler Commands

Procedure Oriented Verbs

DEFINITION VERBS

BEGIN	Used to start test program or BLOCK
TERMINATE	Used with BEGIN. Last statement of test procedure.
DEFINE	Used to define sources, sensors, loads, messages, and procedures.
END	Delimiter for a PROCEDURE, BLOCK, IF/THEN/ELSE sequence, or FOR/WHILE loops.
DECLARE	Used to declare variable types.
FOR UUT	Enables UUT name and model number to be variables.

COMPILER DIRECTIVES

DISCARD	Used to throw out storage no longer needed
REPEAT	Used to repeat a series of tests or test steps.
LEAVE ATLAS/ RESUME ATLAS	Used to insert non-ATLAS comments in an ATLAS program.
PERFORM	References a defined ATLAS procedure.
PERFORM EXTERNAL	An extension to ARINC ATLAS to enable the user to perform non-ATLAS functions in an ATLAS program. Can reference ATS-BASIC, FORTRAN, or assembly routines.

EXECUTABLE VERBS

CALCULATE	Specifies computation.
COMPARE	Relates a labeled value to a specified limit or limits. Sets GO, NOGO flags.
DELAY	Used to delay execution of test program for a specified interval.
DISPLAY	To present a temporary message or value to operator on CRT or numerical display.
ELSE	Denotes program sequence that is executed when IF statement condition is false.
FILL	Load data in predefined LIST locations.
FINISH	To terminate testing and return test equipment to quiescent state.
FOR	Names a looping structure with a control variable.
GO TO	Used to go to test and/or step out of normal sequence.
IF	Contains conditional expression to be evaluated as true or false.
PRINT	Used to print test results or messages to the operator.
RECORD	To identify and store program variable data in historical records.
SAVE	To identify present value of measurement with another label and save it for later in test procedure.
WAIT FOR	Execute specified statement when specified condition is fulfilled or insert manual direction into test sequence.
WHILE	Contains conditional expression for looping structure.

Signal Oriented Verbs

SINGLE ACTION VERBS

CLOSE	To gate a source, sensor, or load function to the unit under test.
CONNECT	To connect pin connectors of unit under test to source, sensor, or load.
OPEN	Used to open switch at output of power or signal source or at input to measuring instrument.
SETUP	Used to set up power or signal sources or measuring instruments.
DISCONNECT	Used to open a specific connection through a switching matrix.
READ	To read present value of sensor function and retain that value, labeled 'measurement', until a new reading is taken.

MULTIPLE ACTION VERBS

APPLY	For source and load functions in sequence: set up, connect, and close.
MEASURE	Used to set up, close, and read signal from unit under test or stimulus source.
REMOVE	To open, disconnect, and set up source, load, or sensor to quiescent state.
MONITOR	Continuously READS and DISPLAYS measurement value.
VERIFY	Combines MEASURE and COMPARE statements. Compares measurement to predetermined value, sets GO, NOGO flags.

COMPLEX ACTION VERBS

ADJUST	Forms a pair with TO MAXIMIZE, TO MINIMIZE or TO REACH. Creates source-sensor loop that uses sensor feedback to control a stimulus application.
DO	Used with DIGITAL TEST noun to perform combination of digital stimulus, response, mask, save, and compare operations in an iterative manner.
START WHEN/ STOP WHEN	Used to define time interval based on some signal, slope or other value.
WAIT FOR	Causes a suspension of testing until some unit under test sensor criteria is satisfied.
PREPARE/ EXECUTE WITHIN	Forms a pair to set up a condition and trigger a sequence of operations which must occur within a specified period of time.
SYNC WHEN	Defines the synchronizing of signals and/or conditions.

Nouns

AC SIGNAL	LOGIC LOAD	SHORT
AM SIGNAL	LOGIC REFERENCE	SQUARE WAVE
COMMON	MANOMETRIC	STEP SIGNAL
DC SIGNAL	PAM	SUP CAR SIGNAL
DIGITAL TEST	PULSED AC	SYNCRO
EARTH (GROUND)	PULSED DC	TIME INTERVAL
EVENTS	RAMP SIGNAL	TRIANGULAR WAVE SIGNAL
FM SIGNAL	RANDOM NOISE	
IMPEDANCE	RATIO	
LOGIC CONTROL	RESOLVER	
LOGIC DATA	ROTATION	

Modifiers

AC-COMP	IMP	PWR-SP-DENS
AC-COMP-FREQ	IND	Q
ALT-RATE	IN-PHASE	QUAD
AMPL-DENS	MOD-AMPL	REF-VOLT
AMPL-MOD	MOD-AMPL-PP	RES
ANGLE	MOD-DIST	RINGING
BANDWIDTH	MOD-FREQ	RISE-TIME
CAP	MOD-OFFSET	RMS-VOLTS
CAR-AMPL	MOD-PHASE	ROUNDING
CAR-FREQ	NEG-SLOPE	SAMPLE-WIDTH
CAR-HARMONICS	NOISE	SETTLE-TIME
CAR-PHASE	NOISE-P	SKEW-TIME
CAR-RESID	NON-HARMONICS	SLEW-RATE
COUNT	NON-LIN	SYNC
CURRENT	OVERSHOOT	+, −SLOPE
CURRENT-LMT	P-AMPL	TIME
CURRENT-ONE	PEAK-DEGEN	TIME-ASYM
CURRENT-QUIES	PERIOD	TIME-JIT
CURRENT-ZERO	PHASE-A,AB,AC	TRIG
DC-OFFSET	PHASE-ANGLE	TYPE
DELAY	PHASE-B,BA,BC	UNDERSHOOT
DISS-FACTOR	PHASE-C,CA,CB	VOLTAGE
DISTORTION	PHASE-JIT	VOLTAGE-AV
DROOP	PHASE-SHIFT	VOLTAGE-ONE
DUTY-CYCLE	+, −PHASE	VOLTAGE-P
FALL-TIME	POS-SLOPE	VOLTAGE-PP
FORMAT	POWER	VOLTAGE-QUIES
FREQ	PRESHOOT	VOLTAGE-TRMS
FREQ-DEV	PRESS-A, G	VOLTAGE-ZERO
FREQ-ONE	PRF	VOLT-LMT
FREQ-QUIES	PULSE-CLASS	WORD-LENGTH
FREQ-ZERO	PULSE-WIDTH	ZERO-INDEX
HARMONICS	PWR-LMT	

Compiler Commands

Command	Description	Applicable Processor ATLAS	A/I	ATE
$INSERT	Controls source of program input from files or system I/O devices, including optional statement number range specification.	•	•	•
$UPDATE WITH	Provides batch-edit capability based on ATLAS test-step numbers.	•	•	•
$OMIT	Used with UPDATE to delete statements from a program.	•	•	•
$END	Used with INSERT or UPDATE to indicate an End of File.	•	•	•
$BATCH	Provides entry to BATCH control routine to execute ATLAS, A/I, and ATE compilations as specified in a BATCH control file.	•		
$ATE	In ATLAS and A/I Processor, used to specify preprocessed configuration file. In ATE Processor, used to specify where processed configuration information is to be stored.	•	•	•
$A/I	In ATLAS, used to specify preprocessed adapter/interface description. In A/I Processor, used to specify where processed adapter/interface description is stored.	•	•	

$SAVE	Specifies output destination of source or preferred ATLAS (A/I or ATE).	•	•	•
$BASIC	Specifies output destination of BASIC source code. When multiple files are generated, they sequentially increment.	•		
$OBJECT	Specifies output destination of executable object code. When multiple files are generated, they sequentially increment.	•		
$GO	Same as $OBJECT, but execution of first object segment immediately follows completion of compilation.	•		
$SEGMENT	Provides override of segment size specified in ATE configuration file on either a percentage basis or on an immediate segment break if no percentage is specified.	•		
$THRU	Controls partial compilation of ATLAS programs by terminating after a specified phase or number of errors.	•		
$DEBUG	Provides execution of standard debug routine at key points during execution of ATLAS object code.	•		
$LIST	For control of source, preferred ATLAS (A/I or ATE) or mixed ATLAS/BASIC listings.	•	•	•
$NUMBER	Controls test and step number sequencing and increment size.	•	•	•
$XREF	Controls listing of label, pin, and signal requirements analysis cross references.	•		

PRICE IN U.S.A.:
9510D Option 100 or 9500D Option 180: $22,000 first purchase; $12,000 each subsequent purchase. These options add ATLAS capability to 9510D and 9500D Automatic Test Systems. Price includes ATLAS Compiler, ATE Processor, A/I Processor, run-time software, and configured ATE file.

MANUFACTURING DIVISION: AUTOMATIC MEASUREMENT DIVISION
974 East Arques Avenue
Sunnyvale, California 94086 U.S.A.

generated off-line by a special processor. Often a certain amount of translation is also performed during this syntax checking. In the HP ATLAS Compiler this technique is extended and used throughout the system. Every process that uses the internal form of the ATLAS (or ATE or A/I) program as its primary input data is controlled by the way the input token stream (see "Phase I" above) corresponds with a grammar specifically defined for that process. In the HP ATLAS Compiler there are seven such "syntax driven processes, differing only in their specific grammars and some low-level subroutines that are invoked as a result of comparing the input token stream with a specific grammar. Most of the software for these seven processes consists of modules that are common to all of them. Once these were designed, implementation of each separate process was reduced to dealing only with its unique aspects, a great simplification.

Finally, the same development language and debugging tools were used by all members of the project team, and each was improved during the project to achieve the maximum leverage possible.

Acknowledgments

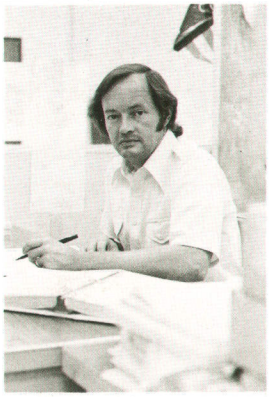
In a complex project such as the HP ATLAS Compiler, which took over two years to develop, the cooperation and enthusiasm of a large number of people from marketing, manufacturing, and R&D areas alike was necessary to provide the push we needed to get to completion. To the many individuals who provided this support, we express our grateful appreciation.

The ATLAS team members displayed a fantastic degree of cooperation and dedication. In particular, Fred Warren did both the front-end processing (Phase I scanner and commands) and the Phase V translation and segmentation. Kathy Hahn did the Phase II procedural translation and expansions, the various cross reference listings, and digital ATLAS, and helped almost every other team member out of schedule problems. Gene Baena designed the switch modeling and did the Phase III signal requirements analysis and the renumbering processing of Phase I. Jack Cooley did the ATE and A/I Processors almost single-handedly.

Finally, our thanks to Mike Chambreau for his special support and encouragement of our project organization and designs.

Robert B. Grady

Bob Grady is HP ATLAS project manager. With HP since 1969, he has designed self-test programs for avionics test systems and served as project manager for a 1400-program instrument-calibration software project. Before joining HP he was involved in the design of digital data acquisition and data transmission systems. A native of Chicago, Illinois, Bob received his BSEE degree from Massachusetts Institute of Technology in 1965 and his MSEE degree from Stanford University in 1969. He and his wife, who works for HP as a programmer, have one small son and live in Los Altos, California. Bob plays basketball and softball in local city recreational leagues and enjoys mountain vacations—hiking, camping or skiing.



William R. Finch

Bill Finch was ATLAS project manager in the investigation phase of the project and chief software designer in the development phase. He's active in the ARINC ATLAS Subcommittee and is chairman of the special working group on structured programming for ATLAS. Born in Saginaw, Michigan, Bill attended Central Michigan University, graduating in 1961 with a BS degree in liberal arts. He came to HP in 1970 with nine years' experience in software development for automatic test systems. His leisure-time activities are primarily family-oriented and include sailing, youth soccer, and camping. Bill and his wife have four daughters and live in San Jose, California.

Automatic 4.5-GHz Counter Provides 1-Hz Resolution

This new frequency counter offers high performance for telecommunications and other applications at a modest cost. Systems compatibility and built-in diagnostics enhance its value.

by Ali Bologlu

AUTOMATIC MICROWAVE frequency measurements to 18 GHz and beyond are the special capability of Hewlett-Packard's Model 5340A Frequency Counter.[1] The general-purpose HP Model 5345A[2] measures frequency automatically to 4 GHz when equipped with the 5354A Automatic Frequency Converter plug-in. It also measures time interval, period, ratio, and other parameters.

Now a new counter, Model 5341A, strikes a middle ground for the user who needs to measure frequency to 4 GHz or so, but doesn't need the universal capabilities of a general-purpose plug-in counter. Model 5341A (Fig. 1) is a 4.5-GHz automatic frequency counter that provides a high level of performance at a modest cost. An optional version that has a 1.5-GHz upper frequency limit offers even greater economy.

The counting principle used in the new counter is an automatic heterodyne frequency converter technique similar to that used in the 5345A plug-in.[3] The user has a choice of automatic mode, in which the counter searches its entire range and measures the lowest-frequency signal, or manual mode, in which the search is restricted to one of ten narrower bands as selected by the user.

Model 5341A is compatible with the HP interface bus (HP-IB).[4] As many as fifteen devices can operate on the bus, so the new counter can be part of an easily implemented automatic measurement system. Equipped with the appropriate options, Model 5341A can communicate digitally with printers, calculators, card readers, computers, and other devices. All of its front-panel controls except power on/off can be remotely programmed.

Front-Panel Inputs and Controls

The new counter has two inputs, a 50Ω, 50-MHz-to-4.5-GHz input and a 1-MΩ, 10-Hz-to-80-MHz input. Specified sensitivity of the 4.5-GHz channel is −15 dBm in automatic mode and −20 dBm in manual mode. Fig. 2 shows measured sensitivities for a typical counter. Sensitivity of the 80-MHz channel is 10 mV

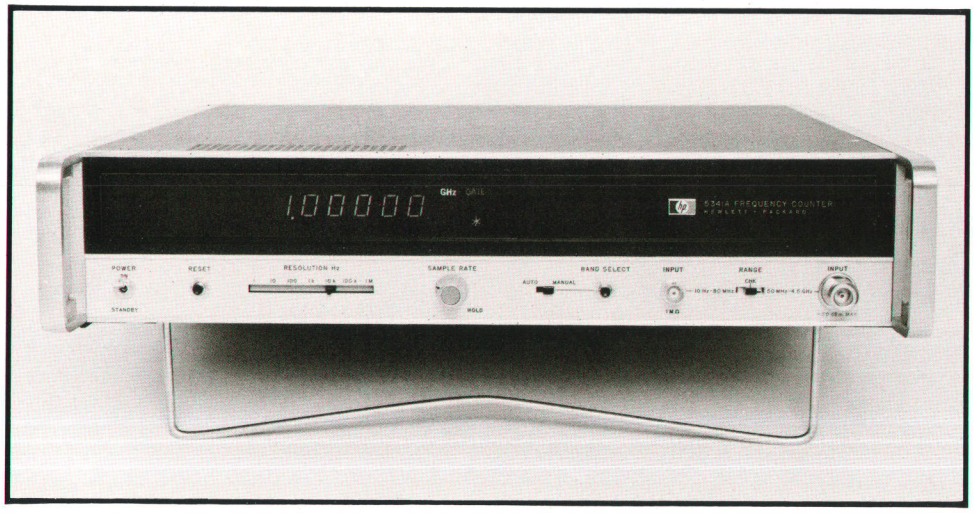


Fig. 1. *Model 5341A measures frequencies from 10 Hz to 4.5 GHz with maximum resolution of 1 Hz. Operation can be manual or automatic. An optional 1.5-GHz version is also available.*

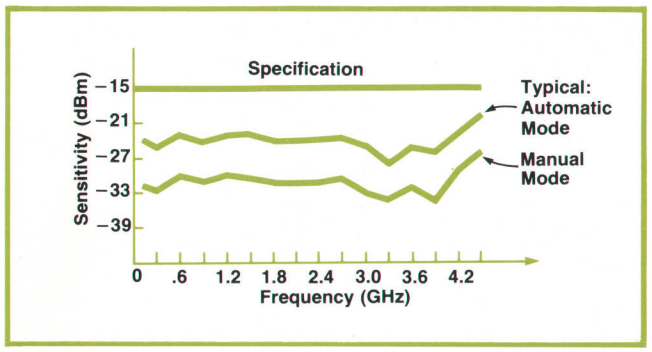


Fig. 2. *5341A sensitivity at 50Ω input.*

rms sine wave.

A range switch on the front panel selects the input channel to be used. Also on the range switch is a self-check position. Another switch selects the resolution of the measurement in decade steps from 1 Hz to 1 MHz. The counter can display ten digits with appropriate annunciators, enough to provide 1-Hz resolution over its full range. A mode switch selects either automatic or manual mode, and a band-select push-button is provided for switching bands in the manual mode.

Counter Organization

Fig. 3 is a general block diagram of the new counter. 10 MHz from the crystal time base oscillator goes to the time base generator and to a multiplier chain that produces all the comb lines (local oscillator frequencies) needed for heterodyne mixing. Lines at 500, 750, and 1000 MHz are produced by printed-circuit-board multipliers. Lines at 1.5, 2.0, 2.5, 3.0, 3.5, and 4.0 GHz are generated in a hybrid module. In the optional 1.5-GHz version of the counter the hybrid module is deleted.

The IF module has a switchable upper frequency limit; it is nominally 530 MHz except for the two lowest-frequency comb lines, for which it is 280 MHz. Also in the IF chain are two peak detectors and a frequency discriminator that control the automatic acquisition algorithm. The output of the IF module is prescaled by 20 and sent to the counter chain.

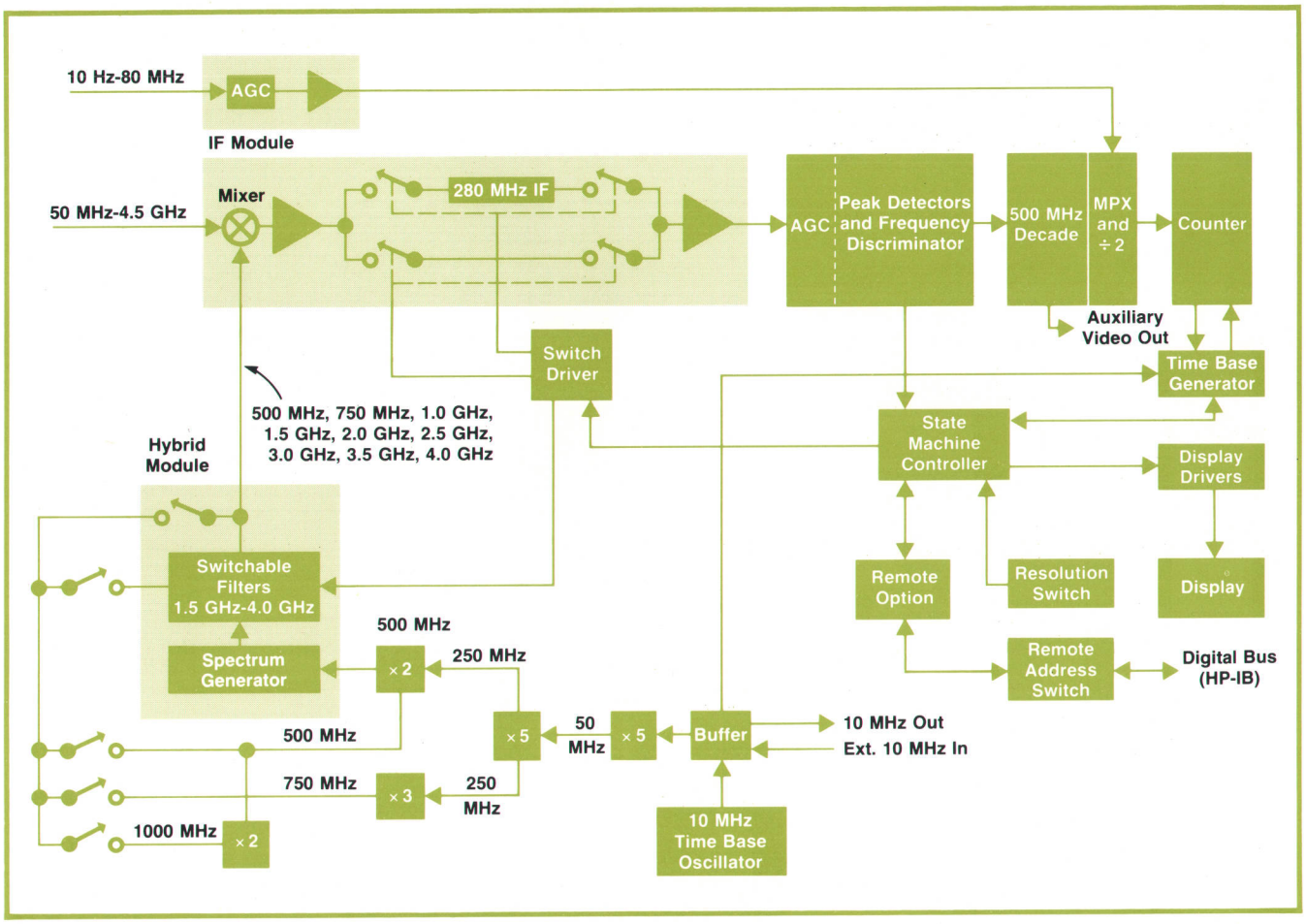


Fig. 3. *50-MHz-to-4.5-GHz frequency range is divided into ten bands by switching different local oscillator frequencies to the input mixer. Band switching can be manual or automatic. Two peak detectors and a frequency discriminator control the automatic acquisition algorithm; they also provide diagnostic information.*

All digital control of the counter is vested in a state machine. Its algorithm is shown in Fig. 4.

Counter Operation

In the automatic mode, the counter continuously sequences through its ten frequency bands by switching appropriate comb lines to the input mixer. This proceeds until a signal is detected in the IF channel by the fact that both peak detectors and the frequency discriminator are triggering. These conditions assure that the intermediate frequency signal is of the proper magnitude to be counted (−1 to +6 dBm) and is within the frequency range of the counting chain, which is 500 MHz. The counter then resets to its lowest band and reacquires the signal to assure that the intermediate frequency is an upper sideband of the comb line used. The frequency of the IF signal is then counted. The state machine identifies the comb line frequency the search has stopped on, adds it to the counted IF, and displays the result.

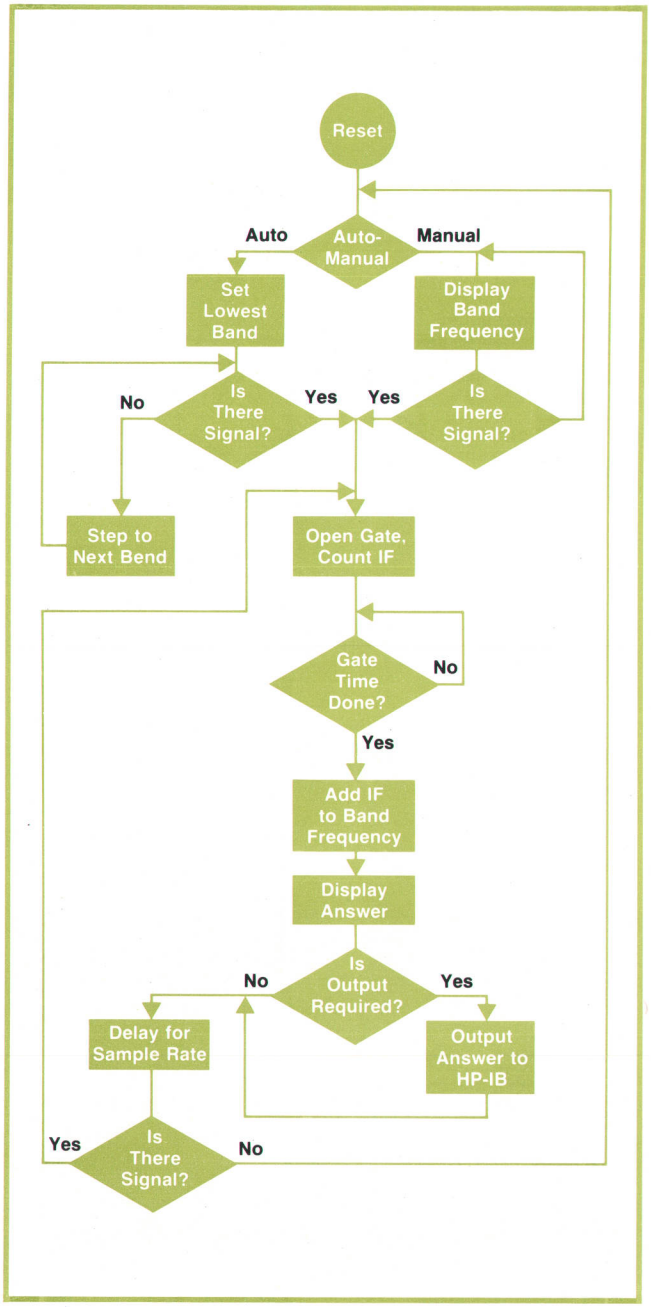


Fig. 4. *Algorithm for the state machine that controls counter operation.*

Overlap between bands is a minimum of 30 MHz, which determines the counter's tolerance to frequency modulation on input signals near the band edges. FM peak-to-peak deviation tolerance rises to ±250 MHz, the bandwidth of the IF channel, for signals near the band center.

In the hybrid structure that generates the upper comb lines, each line is activated by a PIN diode switch (see Fig. 5). This allows fast switching between bands, making it possible for the counter to acquire a signal in 600 microseconds, worst case, in the automatic mode. In the manual mode, where the search is over a narrower range, worst case acquisition time is only 100 μs.

Automatic and Manual Modes

In the automatic mode the counter will measure and display the lowest-frequency CW signal that exceeds its minimum sensitivity. If the signal level is insufficient for the automatic mode but sufficient for the manual mode an asterisk appears in the display.

In the manual mode the correct frequency is found by first pushing the reset button and then sequentially pressing the band select pushbutton. Should no signal be present in a band, the comb line frequency associated with that band is displayed. When a signal is encountered it is measured and displayed. The first frequency displayed is the correct one. Pressing the band select button once more would yield the lower sideband of the next comb line, the image response, which would be the wrong answer.

Manual mode enables the user to search within any of the ten frequency bands. This is advantageous when attempting to find and measure several signals in a complicated spectrum, or for systems applications where low acquisition times are needed.

Built-In Diagnostics

When the range switch is moved to the check position a 500-MHz signal is applied to the mixer assembly, IF module, prescalers, and counter chain. Thus the counters are exercised at their highest operating frequency instead of the customary 10 MHz. With the instrument in automatic mode the 500 MHz is also added to the contents of the counter chain to check for proper operation of the acquisition algorithm.

An internal switch causes the states of the two peak detectors and the frequency discriminator to be

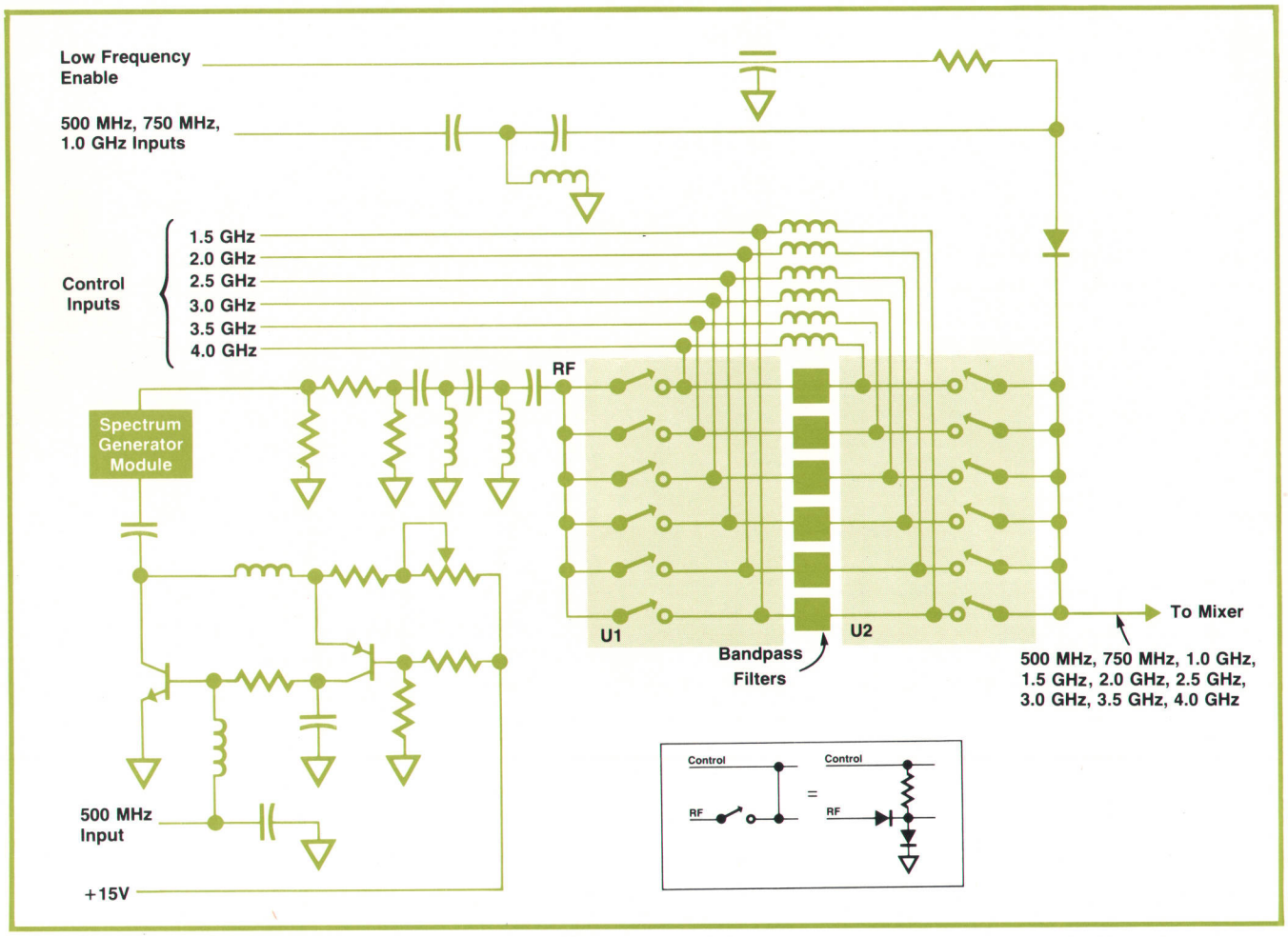


Fig. 5. *Hybrid module generates the upper local oscillator frequencies and provides for rapid switching from one to another. Worst-case acquisition time for an automatic search of the full frequency range is less than 600 microseconds.*

displayed, as shown in Fig. 6. This information and a flow chart under the top cover of the instrument allow the user to diagnose failures to the subassembly level. Fig. 7 shows the flow chart.

If the instrument operates properly in the self-check mode but not in use, an external source and the displayed qualifiers can be used to check the comb lines and the input mixer. This is done in manual mode. If the three qualifiers are active in the image band instead of the proper band the problem is the absence of the associated comb line. If no count occurs in two adjacent bands and self-check has been satisfactorily completed, the input mixer is probably faulty.

Acknowledgments

Digital design of the 5341A was capably handled by Tom Coates. RF designs were by Chuck Shinn and Al Barber. Product design was by Glen Elsea assisted by Dick Goo. Thanks are due Dick Harris for production support and Duncan MacVicar for product introduction.

References

1. R.F. Schneider, "A High-Performance Automatic Microwave Counter," Hewlett-Packard Journal, April 1973.

Fig. 6. *An internal switch causes the counter to display the states of the two peak detectors and the frequency discriminator in the IF channel (three digits at left). These are used with the flow chart of Fig. 7 for troubleshooting.*

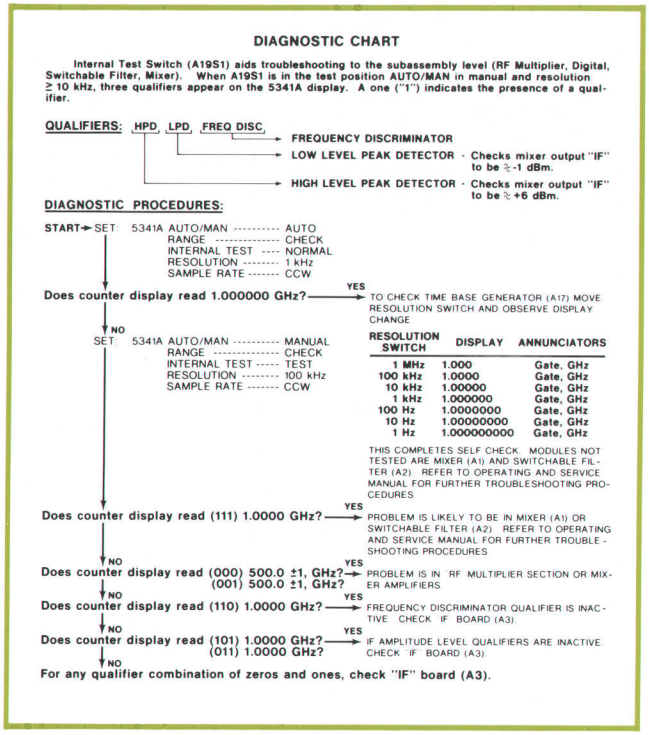


Fig. 7. *Diagnostic flow chart is under top cover of counter.*

2. J.L. Sorden, "A New Generation in Frequency and Time Measurements," Hewlett-Packard Journal, June 1974.
3. A. Bologlu, "A Completely Automatic 4-GHz Heterodyne Frequency Converter," Hewlett-Packard Journal, June 1974.
4. Hewlett-Packard Journal, January 1975.

Ali Bologlu

Ali Bologlu received the BSEE degree in 1962 and the MSEE degree in 1963, both from Michigan State University, and the degree of Electrical Engineer in 1965 from Stanford University. At HP since 1963, Ali has contributed to the design of the 5100A, 5102A, 5103A, 5105A, and 8660A/B Frequency Synthesizers, and was project leader for the 5354A Frequency Converter and the 5341A Microwave Counter. He's a member of IEEE. Born in Istanbul, Turkey, Ali now lives in Mountain View, California, where he devotes much of his free time to the local youth athletics program.

SPECIFICATIONS

HP Model 5341A Frequency Counter

Input 1

RANGE: 50 MHz to 4.5 GHz
IMPEDANCE: 50Ω nominal
CONNECTOR: Precision Type N
SENSITIVITY: −15 dBm (AUTO operating mode); −20 dBm (MANUAL operating mode)
MAXIMUM INPUT: +20 dBm
DAMAGE LEVEL: +30 dBm
OPERATING MODES:
AUTO: Counter automatically selects and displays lowest frequency within its sensitivity range;
MANUAL: Measurement band is selected manually, and counter measures within a 525 MHz range above displayed band number (in the 500 MHz and 750 MHz bands, counter measures within a 250 MHz range).
MEASUREMENT TIME: Acquisition time + gate time
ACQUISITION TIME: 600 μsec (AUTO operating mode); 100 μsec (MANUAL operating mode)
FM CHARACTERISTICS: Tolerates ±250 MHz maximum deviation (0-500 MHz and 1.0-4.5 GHz) and ±125 MHz maximum deviation (500 MHz-1.0 GHz) in center of bands; bands overlap 30 MHz at band edges.

Input 2

RANGE: 10 Hz to 80 MHz
IMPEDANCE: 1 MΩ shunted by 50 pF
CONNECTOR: Type BNC female
COUPLING: ac
SENSITIVITY: 10 millivolts
MAXIMUM INPUT: 5 volts peak-to-peak
DAMAGE LEVEL: 400 volts dc; 250 volts rms ac, 10 Hz to 100 kHz, decreasing 6 dB per octave to 80 MHz

Time Base

CRYSTAL FREQUENCY: 10 MHz
STABILITY:
AGING RATE: $<1 \times 10^{-7}$ per month
TEMPERATURE: $<\pm 1 \times 10^{-6}$ over the range 0°C to 50°C
LINE VARIATION: $<\pm 1 \times 10^{-7}$, ±10% from nominal
OUTPUT FREQUENCY: 10 MHz, ≥2.4V square wave (TTL compatible) available from rear panel BNC.
EXTERNAL TIME BASE: Requires 10 MHz approximately 1.5V p-p sine wave or square wave into 1 kΩ via rear panel BNC.

Optional Time Base (Option 001)

Option 001 provides an oven-controlled crystal oscillator time base with an aging rate near that of a time standard. This option results in better accuracy and longer periods between calibration. A separate power supply keeps the crystal oven on and up to temperature when the instrument is turned off as long as it remains connected to the power line.
FREQUENCY: 10 MHz
AGING RATE: $<\pm 5 \times 10^{-10}$/day after 24-hour warm-up, for less than 24-hour off time, and $<1.50 \times 10^{-7}$/year.
SHORT TERM STABILITY: 1×10^{-11} for 1 s avg. time; 1×10^{-11} for 10 s avg. time; 2×10^{-11} for 100 s avg. time
LINE VARIATION: $<1 \times 10^{-10}$ for ±10% change from nominal. A 10% voltage change will cause a frequency change of $<1 \times 10^{-8}$ for <2 minutes.
TEMPERATURE: $<7 \times 10^{-9}$ frequency change over the range 0° to 50°C.
WARM-UP: Within 5×10^{-9} of final value 20 minutes after turn-on, at 25°C.
FREQUENCY ADJUSTMENT RANGE: $>\pm 1 \times 10^{-6}$(>±10 Hz from 10 MHz) with 18-turn control.
FREQUENCY ADJUSTMENT RESOLUTION: 1×10^{-9} (0.01 Hz)

General

ACCURACY: ±1 count ±time base error
RESOLUTION: Front panel switch selects 1 MHz, 100 kHz, 10 kHz, 1 kHz, 100 Hz, 10 Hz, or 1 Hz.
DISPLAY: Ten-digit sectionalized LED display and appropriate measurement units of kHz, MHz, or GHz.
SELF CHECK: Counts and displays 1 GHz for resolution chosen.
SAMPLE RATE: Continuously adjustable from 40 msec to 10 seconds and HOLD.
OPERATING TEMPERATURE: 0°C to 50°C
REMOTE PROGRAMMING AND DIGITAL OUTPUT: Optional (Option 011) via 24-pin, series 57 Microribbon connector. Program and output information are 7-bit ASCII code.
PRICES IN U.S.A.:
5341A Frequency Counter, $3,600
Option 001 High-Stability Time Base, $500
Option 002 Rear Panel Input Connectors, $105
Option 003 1.5 GHz Frequency Range, less $1,000
Option 011 Remote Programming-Digital Output, $390
MANUFACTURING DIVISION: SANTA CLARA DIVISION
5301 Stevens Creek Boulevard
Santa Clara, California 95050

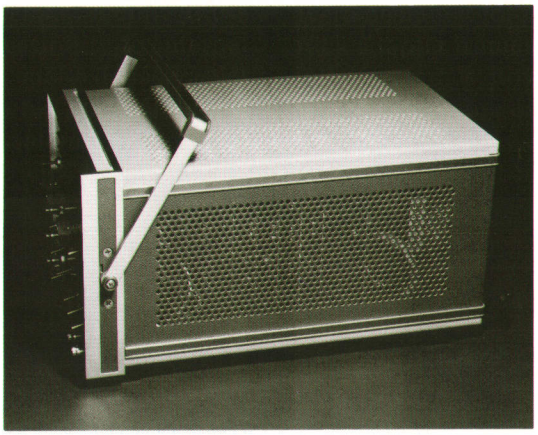
A New Instrument Enclosure with Greater Convenience, Better Accessibility, and Higher Attenuation of RF Interference

Evolutionary changes in the way electronic circuits are packaged have called for a new approach to enclosure design. Described here is the result of a corporate-wide effort to meet customers' changing requirements.

by Allen E. Inhelder

A NEW DIRECTION IN ENCLOSURE design was taken some fourteen years ago with the development of a universal instrument enclosure system at Hewlett-Packard. This system (Fig. 1) directly addressed the problem of how to group instruments, whether on the bench or in a rack, a problem that had been growing more acute as the number of instruments used in measurement setups continued to grow.

This enclosure system made it practical to stack instruments neatly for bench use while at the same time providing a convenient means for mounting the instruments directly in a rack. It was also esthetically more appealing than the simple boxes that had been the norm, and it provided more convenient access to internal parts and more efficient use of space than the earlier chassis-deck-slipped-into-a-box approach. During subsequent years, the basic concept was adopted by a number of other manufacturers, attesting to the wide acceptance of this approach to enclosure design.



Changing Requirements

As time went on, however, continuing changes in the nature of electronic instrumentation created new needs in enclosure systems. Foremost among these was the need for even better accessibility to internal parts as circuits were packed in more densely. Not only was access needed from the top and bottom, as provided by the 1961 enclosure system, but from the sides, front and back as well.

Front-panel space was growing scarce. As more capability was concentrated in smaller spaces, the area for mounting a growing number of controls, displays and nomenclature shrank, so it became more and more difficult to arrange controls for convenient operation.

Radiated electrical interference was also becoming a significant problem. As the transition times of digital signals were shortened to the nanosecond region, instruments were radiating a greater amount of high-frequency energy, creating potential problems for users operating sensitive instruments in close proximity. Cracks between a top cover and a front panel or along the edges of plug-ins provided leakage paths for RF energy, requiring expensive metal gasketing to keep the radiation confined to the inside of the instrument.

At the same time there was a growing need for a small universal enclosure design. As the circuits within instruments were miniaturized to an increasing degree, more and more instruments were housed in enclosures that did not fill the standard 19-inch rack width. There were very few of these smaller instruments when the 1961 cabinet was developed, so small instrument enclosures at that time were designed for grouping in combining cases or in rack adapter frames. These combining cases and rack adapter frames, however, began to assume costly proportions as more of them were required to house the increasing variety and number of small instruments, and they did not make maximum use of space.

Fig. 1. *System I cabinets introduced in 1961 were designed to allow the same instrument to be used either on the bench or mounted in a rack. Plastic feet fit over a cabinet below, enabling instruments to be stacked conveniently for bench use. Removal of the feet and installation of simple angle-iron brackets in the space provided enabled installation of the same instruments directly in a standard relay rack.*

A Fresh Start

Clearly, there was a need for a new approach to the kind of enclosure that instruments could be put into. Because of numerous other problems that designers in HP's various divisions had been experiencing in anticipating customer needs—such as the limited number of standard sizes that sometimes caused an instrument to be larger than it really needed to be—it was decided to bring together industrial designers from the major HP divisions to work as a team designing a new enclosure system. In this way it was anticipated that various customer requirements related to the packaging of a product would be accommodated in the new design.

The basic design approach was to be "inside-out", in which all the servicing, manufacturing, electrical mechanical, and thermal needs would be met first, after which the esthetics of the design would be considered.

From this effort, a new cabinet system, known within the company as System II, was developed. It has greater strength but is lighter than the earlier design, it provides better accessibility for servicing and more versatility in configuration and it inherently provides significant attenuation of unwanted RF energy.

Multipurpose Framework

The framework of the new enclosure is shown in Fig. 2. This frame is rigid by itself and does not depend on decking, front and rear panels, or covers for strength as the older design did. It gives the designer greater freedom in the product design.

The heart of the design is the front-panel frame, an aluminum die casting that has integral pads for mounting the side members, mounting holes for fastening the front panel, recesses for links that lock adjacent enclosures together, slots for plug-in latches, and narrow channels for holding top, side, and bottom covers. In addition, mounting holes are provided for dividers that may be used for plug-in compartments or other front-panel subdivisions, either vertically or horizontally.

A key part of the design is the narrow channel for mounting covers (Fig. 3). These U-shaped slots serve as wave traps that reduce the radiation of (or susceptibility to) unwanted RF energy. As a further precaution, small ridges aligned in the direction of cover insertion provide high-pressure points for establishing good electrical contact every inch or so. Only RF energy at wavelengths much shorter than those of concern can escape between these contact points. Channels on the side covers provide the same kind of RF seal along the sides, as does a similar arrangement under the lip of the covers at the rear. The covers, however, are each retained by a single captive screw, enabling quick removal for servicing.

The sizes of other holes in the external envelope, such as those needed for mounting the feet, were reduced to practical minimums. The overall RF attenuation achieved by these measures is indicated in the graphs of Fig. 4.

The plastic feet are compatible with the earlier

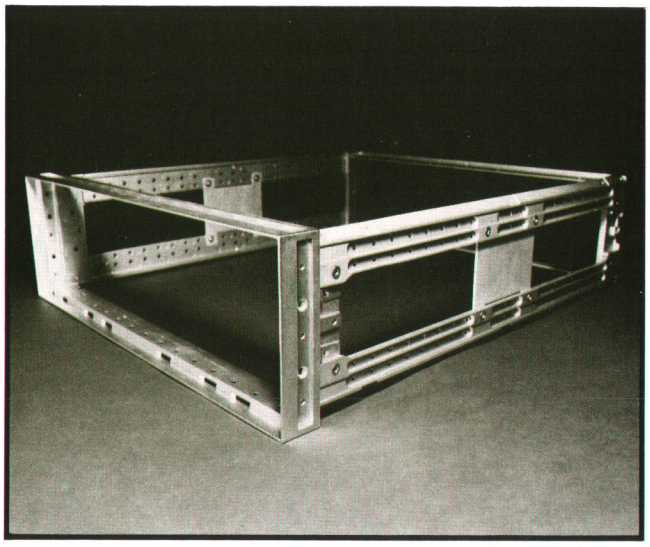


Fig. 2. *Frame of the new enclosure system. Countersunk holes are for mounting internal parts that have captive nuts. Threaded holes are for attachment of external parts, such as handles, rack-mount brackets, and other fasteners. Slots along the bottom inner edge of the front frame are receptacles for latches that retain plug-ins.*

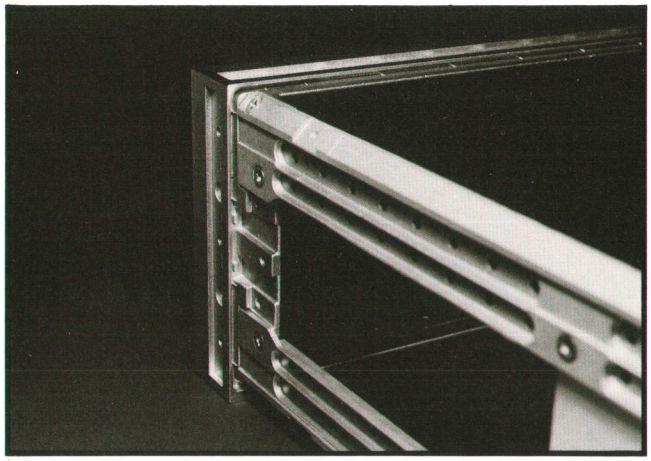


Fig. 3. *Rear view of the front-panel frame shows the RFI-trap channels into which the top, side and bottom covers are inserted. Ridges provide high-pressure contact points for limiting the wavelength of RF energy that may leak out.*

cabinet design so there is no problem when products in the new enclosure are stacked with those in the earlier style cabinet. Positive instrument location is provided for the plastic feet and the tilt-up bails. Both front and rear feet accept the tilt-up bails so an instrument can be tilted up at the rear to give it a more convenient "tilt-down" stance when it is placed above eye level.

Maximized Panel Area

Unlike the earlier design, the front-panel frame uses all the available area in full multiples of vertical EIA/IEC increments (multiples of 1¾ inches). The earlier design allowed vertical space for accommodating rack shelving, so a filler strip was added to fill in the resulting gaps when instruments were rack mounted. In the new design, the front-panel frame overhangs the lower side members, completely fill-

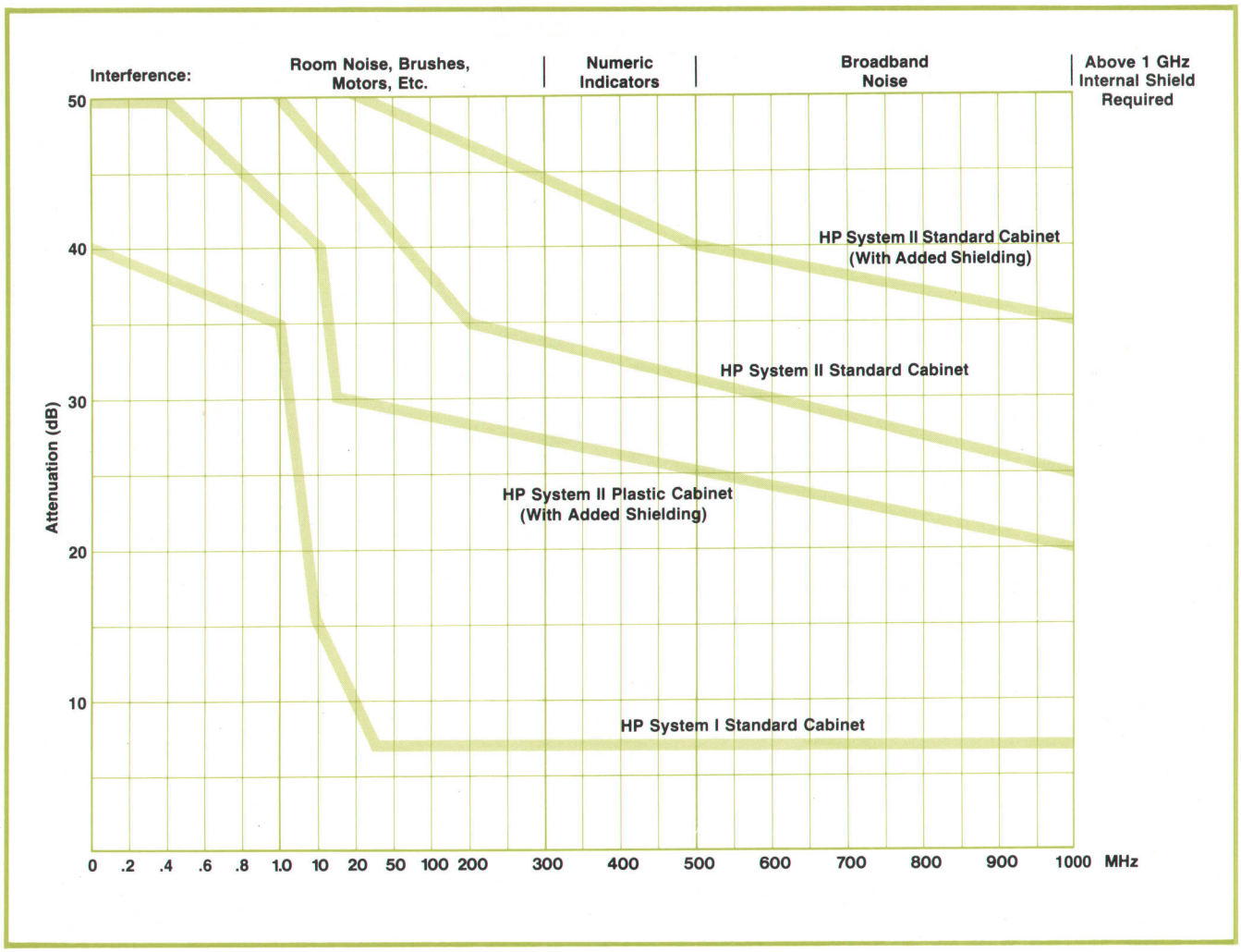


Fig. 4. *Attenuation of RF energy achieved by the new enclosure design. The values represented are conservative, actual measurements made with a bare enclosure being 10-15 dB greater. The performance actually achieved by any given instrument, however, depends on such matters as the shape and size of holes cut into the panels, on whether metal or plastic control shafts are used, and on the signals present on open connectors.*

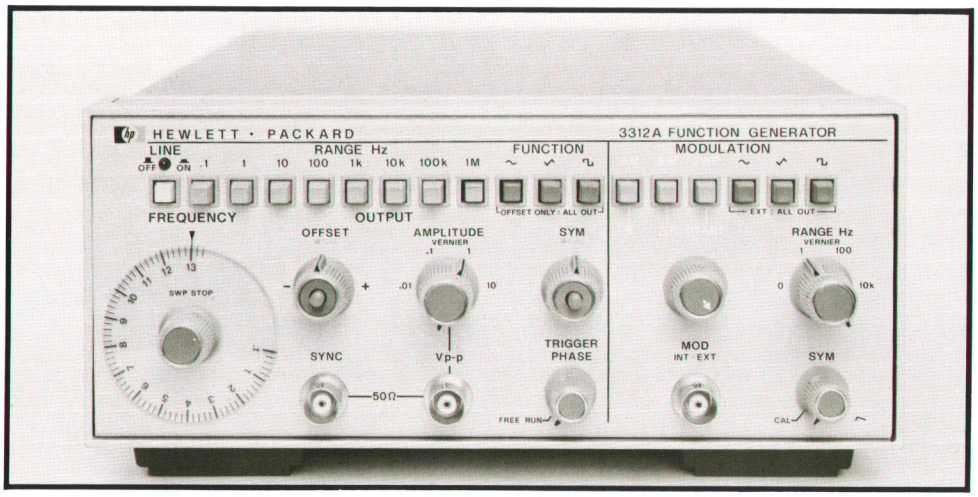


Fig. 5. *New cabinet design maximizes the available front-panel area for uncrowded control lay-out.*

ing the allotted rack space while still allowing room for the rack shelves.

The front panel mounts to the framework with screws that are accessible from the outside. The panel may therefore be designed for removal, thus providing access to panel-mounted components without requiring the disassembly of any other part of the instrument. The system thus has the potential for reducing service time significantly.

Because the front panel does not serve as a structural member, it does not need rounded edges for strength, thus increasing the amount of usable panel space (Fig. 5). This reduces the crowding of controls so instruments can be easier to operate.

The rear panel also mounts in a die-cast frame, simplifying its removal. Front- and rear-panel frames are joined by die-cast side struts, leaving plenty of open space for accessibility to internal components (Fig. 6). The four struts are identical, and the same type is used for all cabinets regardless of height or width. Thus it is economical to manufacture the struts in five lengths, giving a wide assortment of available cabinet sizes. Also, since all the main structural members are die cast, the dimensional tolerances are tighter than is possible with sheet metal, making parts replacement easier.

All screws used in the cabinet assembly are of the self-locking type with an inserted plastic patch on the threads, per MIL-F-18240C, that prevents the screw from working loose when subject to vibration. As a result, the assembly is much more rigid than would have been possible using lock washers.

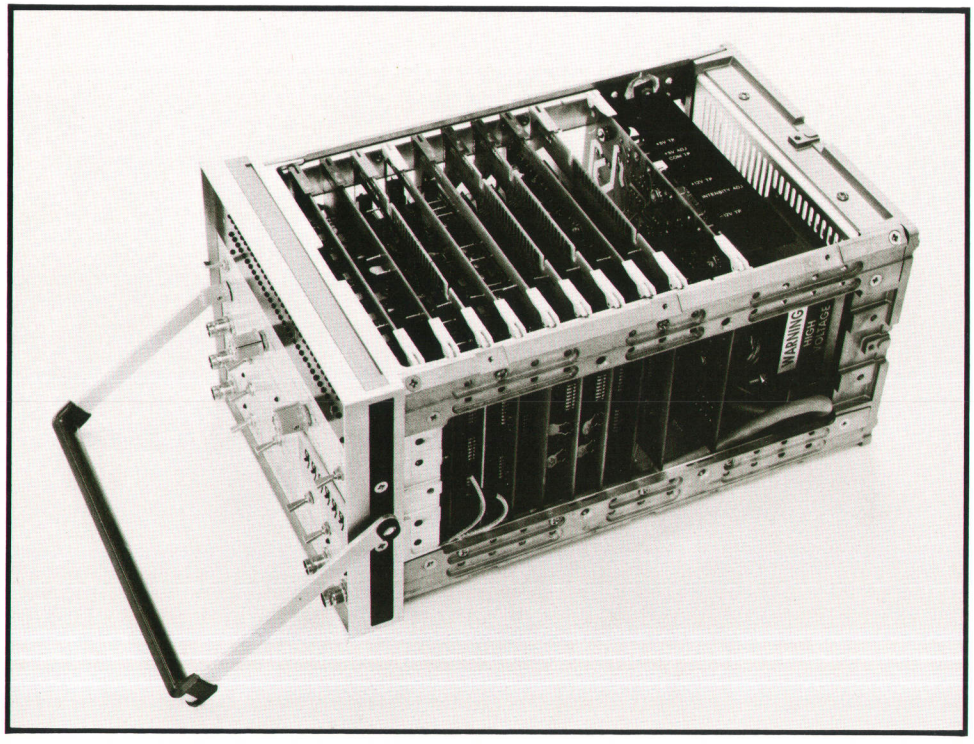


Fig. 6. *New enclosure makes maximum use of interior space. Circuits are accessible for trouble-shooting from the top, bottom and sides.*

Easier Carrying

Front-panel handles, now an optional item, cant outwards. This has two benefits. Access to controls along the edge of the front panel is improved, and when the instrument is carried by the handle, the angle is the proper one for the hand (Fig. 7). The handle was also reshaped to provide a more comfortable fit to the hand. The optional rack-mounting brackets may be installed with or without the handles in place.

A bail handle is available for the smaller instruments (see Fig. 6). The side handle, which is fitted to the top cover of submodules, is a long strap. This provides more freedom in finding a balance point. The cover panels used with the strap handles have a shallow depression for the strap. This performs an alternate function by providing a place for mounting rack slides.

Modular Submodules

The smaller enclosures in the new system are dimensioned to be exact submultiples of the standard rack width design. Rack mounting adapters are therefore not required for the smaller instruments—a simple extender to reach full rack width is all that is needed (Fig. 8).

Instruments can be fastened together horizontally and vertically with simple links (Fig. 9). Because the submodules are dimensioned as part of the total system, standard full width and full height cover panels and handles can be used where smaller instruments are to be combined more or less permanently in a grouped arrangement.

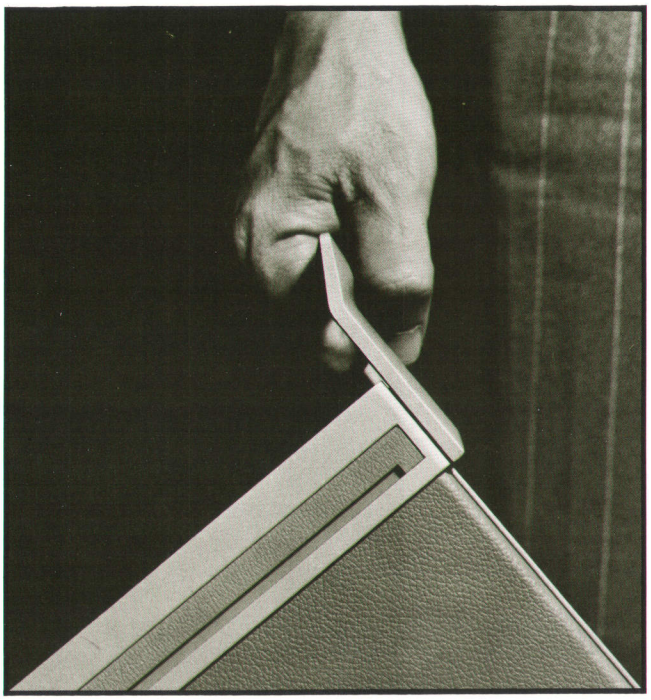


Fig. 7. *Canted handle provides a more comfortable grip for carrying an instrument.*

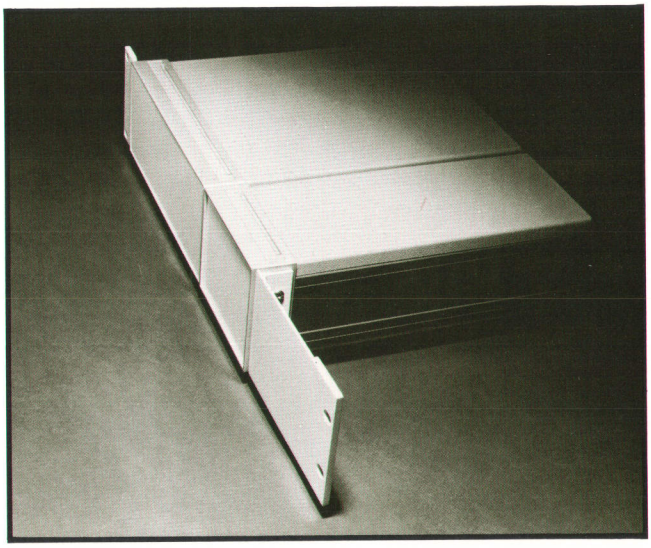


Fig. 8. *Smaller instruments either singly or in groups, fit standard EIA rack dimensions with the addition of simple extensions.*

Where instruments are to be grouped temporarily, links that allow quick assembly and separation (Fig. 10) can be installed by using threaded holes already provided in the framework.

Environmentally Evaluated

The thermal characteristics of the new cabinet design have been thoroughly evaluated and guidelines supplied to HP instrument designers to aid in maintaining heat rise within conservative limits, thus obtaining improved reliability and assuring

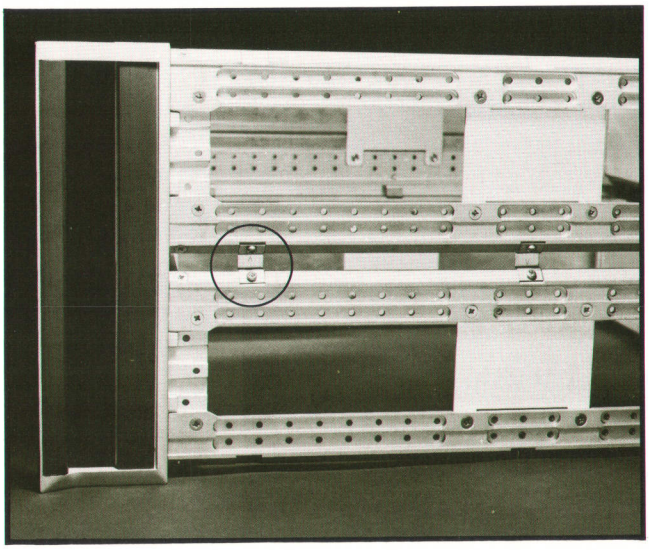


Fig. 9. *Instrument frames are linked together with universal brackets that can be used for either horizontal or vertical instrument groupings. Linked frames can be fitted with full-size covers and handles to make a single, unified package.*

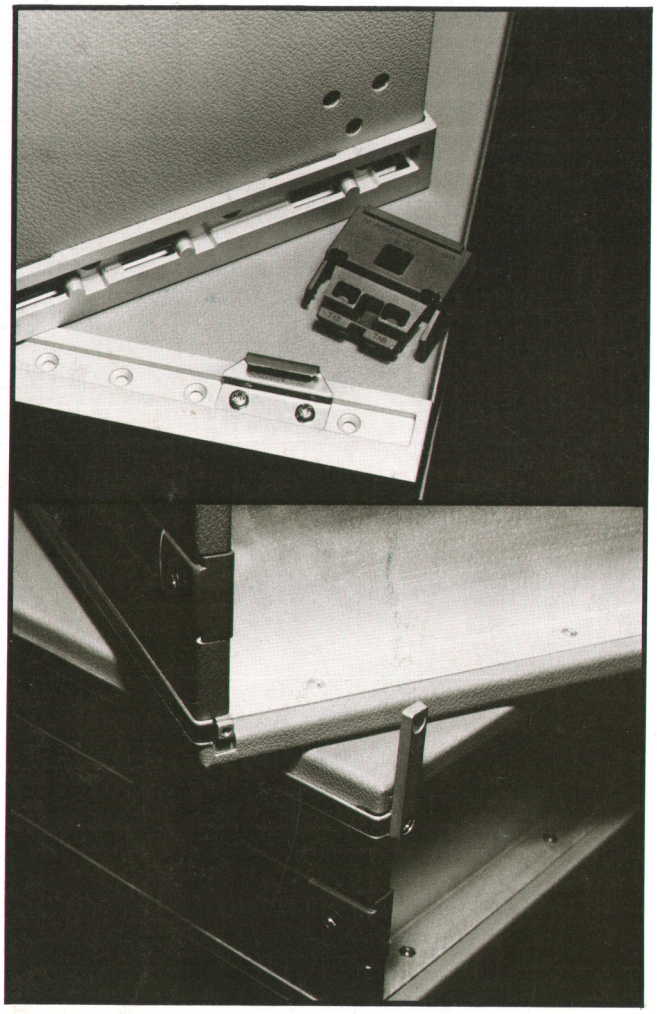


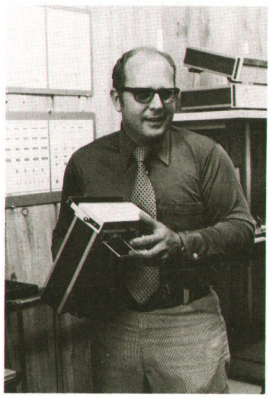
Fig. 10. *Temporary groupings of instruments are enabled by quick-disconnect links that are fitted to the lower instrument and insert into slots in the front-panel frame of the upper instrument. The rear is retained by straps that bolt to builtin pads. The links also work with horizontal groupings.*

longer product life. The cabinet design is in compliance with various environmental and safety specifications such as the Underwriters Laboratory proposed specification 1244, MIL-S-901 shock test, and MIL-STD 167 Type I vibration test, among others.

All in all we believe that we have arrived at a superior design that offers many benefits to the users of instruments packaged in this enclosure.

Acknowledgments

Industrial designers and engineers contributing to the System II enclosure design were Dick Anderson, Rich Hoogner, and Paul Rasmussen of the Corporate Industrial Design Department, Andi Aré of the Colorado Springs Division, Bernie Barke of the Santa Clara Division, Chuck Dodge, Bob McCaw, and Bill Benham of the Stanford Park Division, and Arnold Joslin of the Loveland Division. Project manager was Roy Ozaki.



Allen E. Inhelder

Schooled at the Art Center in Los Angeles, Al Inhelder designed automobiles for 2½ years before the lure of the west coast brought him back to California in 1957 and a position on Hewlett-Packard's corporate industrial design staff. He has headed the department since 1963, directing a staff in bringing coherence to design elements such as color schemes, human factors, and enclosure form as some 24 HP divisions independently design their own products. Outside of working hours, Al relaxes with his family at home in a wooded area on the fringes of Portola Valley, California.

Hewlett-Packard Company, 1501 Page Mill Road, Palo Alto, California 94304

HEWLETT-PACKARD JOURNAL

SEPTEMBER 1975 Volume 27 • Number 1

Technical Information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard S.A., CH-1217 Meyrin 2 Geneva, Switzerland
Yokogawa-Hewlett-Packard Ltd., Shibuya-Ku Tokyo 151 Japan

Editorial Director • Howard L. Roberts
Managing Editor • Richard P. Dolan
Art Director, Photographer • Arvid A. Danielson
Illustrator • Sue M. Perez
Administrative Services, Typography • Anne S. LoPresti
European Production Manager • Michel Foglia

Bulk Rate
U.S. Postage
Paid
Hewlett-Packard
Company

CHANGE OF ADDRESS: To change your address or delete your name from our mailing list please send us your old address label (it peels off). Send changes to Hewlett-Packard Journal, 1501 Page Mill Road, Palo Alto, California 94304 U.S.A. Allow 60 days.