The Hewlett-Packard Digital Symposium & Exhibition

HEWLETT
PACKARD

1. A Logic Analyzer Look at High Level Software

HEWLETT PACKARD

Gail Hamilton of Colorado Springs R&D Lab designed the 8085 and 6502 personality modules for Model 1611A Logic State Analyzer. Gail has a B.S.E.E./C.S. from U. of Colorado and is working on a Masters in Engineering Administration from Stanford.

A Logic Analyzer Look at High Level Software

by Gail Hamilton

Abstract

Computer system software is distinctly hierarchical in nature and measurement interest ranges from the instruction level to the system level. The first generation of logic state analyzers had the ability to look at the digital signal levels on the address and data buses. With the second generation, the capability to analyze software was extended through sequential triggering techniques and assembly language program tracing. Relevant information concerning the execution characteristics of software at these various levels can be determined using one of these second generation analyzers, the HP Model 1610A/B Logic State Analyzer. In addition, the utilization of a controller and HP-IB provides powerful post-processing, as well as automated setup. This paper describes methods of measuring higher level software modules in a computer system with currently available instrumentation.

Introduction

In a computer system there are usually many discrete levels of the software hierarchy that can be observed. Depending on the complexity of the system, there could be only one or many levels. A system that could only be programmed in machine code might possibly have one level, whereas an operating system such as UNIX* would have five. In a system that has both an operating system environment and a user environment, the software hierarchy of interest would have the same basic elements with a slightly different emphasis. As an example, many operating systems have the hierarchy depicted in Figure 1. In the domain of the user, the lowest level of interest would probably be the instruction level, while the highest level might be the program or job level. The software hierarchy is built on the reduction of detail and the increase of abstraction.

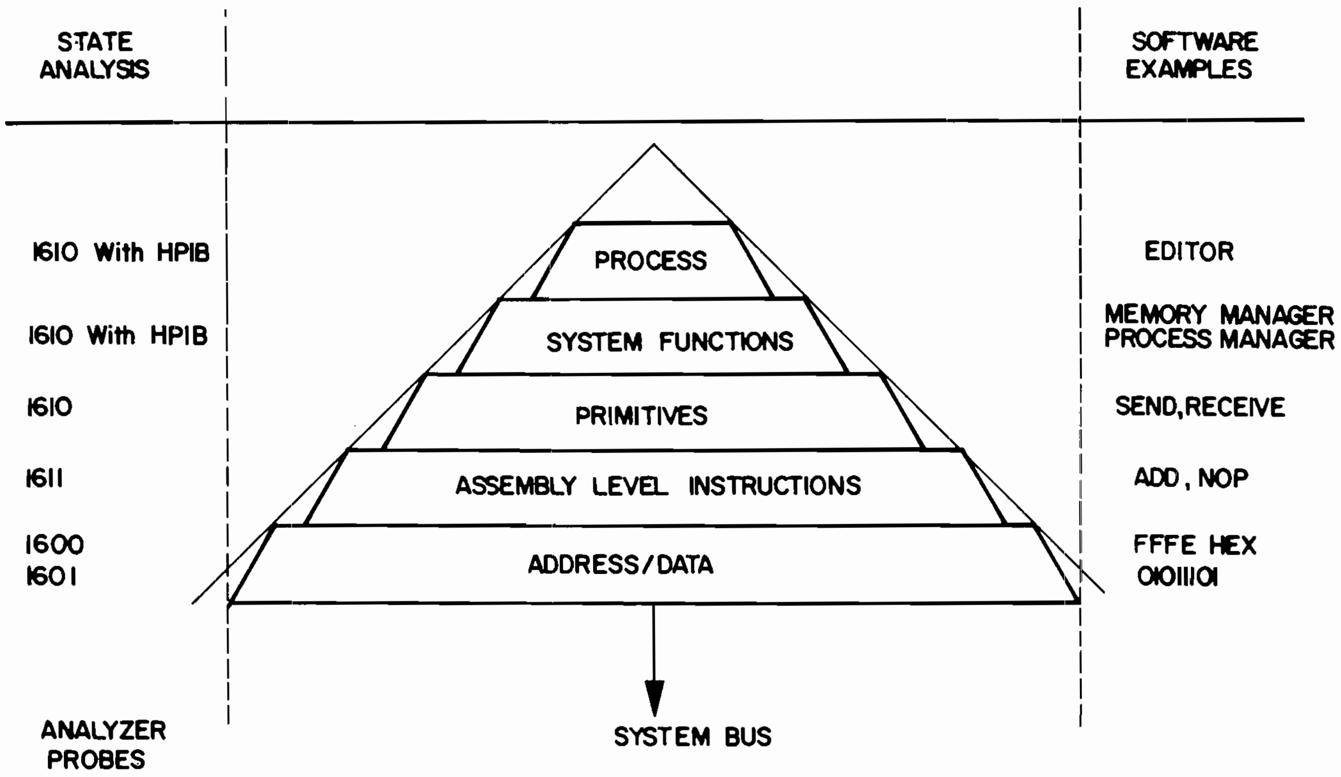


FIGURE 1 Example of Software Hierarchy in a Computer System

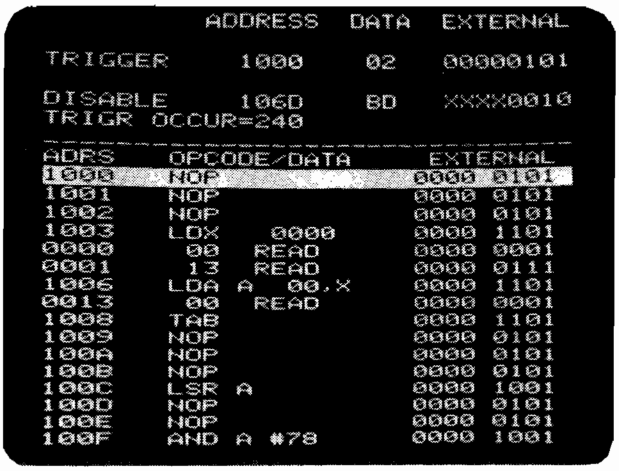
In order to look at these various levels of abstraction, logic state analyzers have evolved with a parallel hierarchy in the measurement domain. They too must reduce the data by removing the nonessential detail for the appropriate level of abstraction. The ability to analyze software depends on how well the instrument can look at two major areas. The first is qualitative and is more of the "what" is the software doing. For this type of information, a dynamic trace or sequential time list of events is appropriate. The second aspect is quantitative and answers the question "how": how much, how often, and how long.

State analyzers provide passive probing at the hardware level, but they must provide sufficient data to decode the activity that occurs at the higher levels. The lowest level in the hierarchy are the addresses and data that can be derived directly from signal levels on the bus. For measurements in the software domain, a logic analyzer must translate the digital signal levels on the computer system's bus into three fields. The first field is the control signals to and from system devices, the second field is the addresses which refer to specific locations, and the third field is the data that is associated with the address.

It was in 1973, when the first parallel logic state analyzer was introduced, that computer system designers had a measurement tool that gave them a view of their software that many had never seen before. The 1601 had the ability to capture a sequence of digital words, but didn't attempt to transform them in any way, but instead displayed the sequence of events as a table of 1's and Ø's. The 1601 wasn't able to make any quantitative measurements.

With the introduction of the 1600, the capabilities were broadened by adding more bits of data, but the data display did not change significantly. It is interesting to note, that as early as the time of the 1600, there were attempts to recognize these higher levels. The MAP mode was added to the 1600 which gave a more comprehensive view of the system. In the MAP mode, each event on the system bus had a unique position on the CRT display. As a sequence of events was observed, the instrument traced lines from dot to dot. The MAP mode also gave an indication of the relative frequency or occurrence of any state - the more often a state is repeated, the brighter its dot. This was the first time that an attempt was made to provide a quantitative measurement. For example, the mode gave a highly visible indication of which memory addresses were used most often and which were not used at all. A user, then, could recognize a particular pattern of word sequences as being significant. Even the ability to transform the binary code into a more compact form, such as octal or hexadecimal, was not available until the introduction of the 1611A microprocessor analyzer at the end of 1976.

The 1611 was the first logic state analyzer to actually transform the sequence of digital words into something more meaningful than a tabular form of signal levels. The major contribution of this instrument was its ability to convert the machine language of a microprocessor into assembly language (refer to Figure 2). In other words, the 1611 focuses at the second level of the software hierarchy, namely, the instruction level. The instrument displays the microprocessor's instructions and makes it easier to interpret snapshots of executing program steps. Two quantitative measurements are provided in the 1611. The 1611 has the ability to measure the time between two events and count the number of states between two events. With the instrument in the continuous trace mode, it also provides statistical parameters, such as minimums and maximums.



```
                ADDRESS   DATA   EXTERNAL

TRIGGER          1000      02    00000101

DISABLE          106D      BD    XXXX0010
TRIGR  OCCUR=240
-------------------------------------------
ADRS     OPCODE/DATA             EXTERNAL
1000     NOP                    0000 0101
1001     NOP                    0000 0101
1002     NOP                    0000 0101
1003     LDX    0000            0000 1101
0000      00   READ             0000 0001
0001      13   READ             0000 0111
1006     LDA  A   00,X          0000 1101
0013      00   READ             0000 0001
1008     TAB                    0000 1101
1009     NOP                    0000 0101
100A     NOP                    0000 0101
100B     NOP                    0000 0101
100C     LSR  A                 0000 1001
100D     NOP                    0000 0101
100E     NOP                    0000 0101
100F     AND  A  #78            0000 1001
```

FIGURE 2

It certainly was becoming apparent that hardware measurements of a complicated system had little to offer the software designer unless they could be related to events and states at a software level. Similar to all of the logic state analyzers, the 1610A/B collects data at the lowest level, but much of the upper level activity can be inferred using the techniques presented in the paper. Contributions of the 1610A/B, in terms of sequential triggering, selective trace, sequence restart, and the time/state count are crucial to making these higher level measurements. The time/state count of the 1610 is similar to what is available in the 1611, but these metrics are available for every event that is observed.

All of the current state analyzers, then, are optimized for making measurements at different levels in the hierarchy. By qualifying the incoming data, the unnecessary detail is suppressed for the appropriate level. Essentially, the micro-events are being probed in all these cases and the only difference lies in the data reduction and data display portions of the instrument.

What Measurements and Why

One of the first and most important questions that needs to be answered is what quantitative measurements need to be made, given that these software levels can be recognized. In analyzing software modules of a computer system, the two most significant quantities to be measured are time and frequency. Especially of interest is the time required to perform various functions or the time spent waiting to perform those functions. For example, you might want to know the instruction mix of a particular routine; how many ADD's, how many NOP's, etc. It is possible that a frequently used instruction could be implemented in microcode differently so as to provide faster execution. The knowledge of the execution time and frequency of usage of critical subroutines might be important as well. Another measurement that has value is a memory map, similar to what was being done qualitatively with the 1600 MAP mode. By partitioning regions of memory into buckets of address ranges, the total time or frequency that a software module spends executing in each partition can be accumulated.

The second question, and of equal importance, is why do these measurements need to be made and of what importance are they? If the utilization of these routines can be monitored, then a profile of system activity can be created. Bottlenecks occur when the relationship among the various resources becomes unbalanced. By knowing where these areas are that create inefficiencies, a programming team can concentrate resources on those regions where the greatest benefit will be gained from the smallest investment. It should be recognized that the implications of being able to maximize a programmer's utility are overwhelming. Current trends in programming methodology, such as the increasing use of structured programming and modular design, have amplified the need for tools that can detect and tune regions critical to performance. The question, "Where is the time spent?" can be answered by these measurement techniques. Even though these questions can be answered, a subjective judgement must still be made as to whether the amount of time spent on a given function is acceptable. Knowlege of where system overhead is located allows a designer to minimize those areas so that the time spent executing programs is maximized. By inferring the occurrence of significant logical events from the myriad of bus activities, higher level software can be observed and analyzed during actual execution.

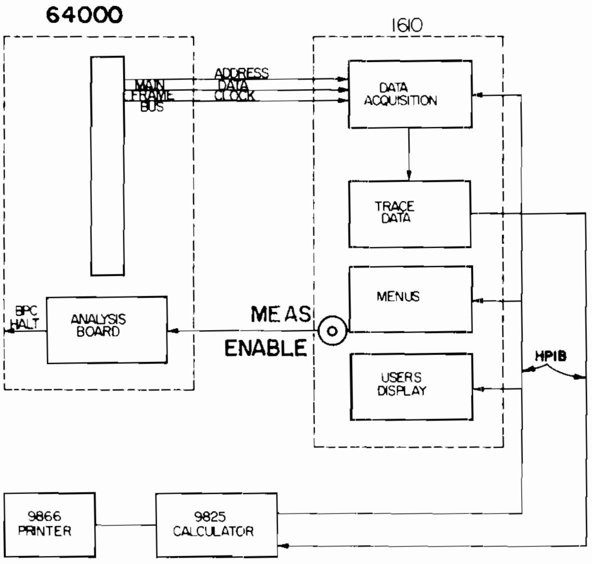
How To Make the Measurements

A few years ago HP responded to the need for a standard bus for their instruments with the Hewlett-Packard Interface Bus (HP-IB), which has since been adopted as an industry standard, IEEE-488. Many of the second generation analyzers are HP-IB controllable via a short mnemonic language. The ability of logic analyzers to communicate over HP-IB allows programmable control of the analyzer, and combining this with an intelligent controller provides considerable flexibility. Many of the measurements that have been discussed previously, can be made using this capability.

There are many advantages to having an intelligent controller. The calculator can reduce data and provide statistical summaries with means, standard deviations,minimums, and maximums. It can also compile distribution data and calculate parameters such as correlation coefficients. The calculator can reduce the collected data before it is stored or displayed and also facilitate on-line modification of measurements.

There are two ways that the measurement system can observe the system under test. One choice is to have the measurement system be completely transparent to the system that is being observed and the other is to have the measurement system play an interactive role. For example, the measurement system could have the ability to halt the processor to make some real-time calculations or make changes in the setup based on the current input.

The block diagram in Figure 3 is one example of how the measurement system could be configured. In this case,the system under test is the HP 64000 Logic Development System. Two instruments, the 1610A/B and the HP Model 9825 Calculator are required; the 9866B printer can be used to interface to either the 1610AB or the 9825. The printer provides hard copy of Trace Lists as well as Format or Trace specifications if it is directly connected to the logic analyzer.



Software Measurement System

FIGURE 3

In general, the sequence of events needed to make some of these measurements are as follows:

1. The calculator sets up the 1610A/B Format and Trace Specifications.

2. The calculator tells the 1610A/B to Run (begin looking for the Trace points).

3. When the trace points are found, the 1610A/B generates a signal telling the processor in the system under test to halt and displays the results in a Trace List. If there were no interaction with the system under test, the 1610A/B would only display the Trace List.

4. The calculator then reads the data from the Trace List, stores the pertinent information, and depending on the particular measurement, may output to the 9866B printer.

5. The calculator then sets up 1610A/B to make another measurement. The new setup could be dependent on the data obtained in the previous trace.

6. After all the measurements are completed, the user display capability of the 1610A/B is employed to graph the results.

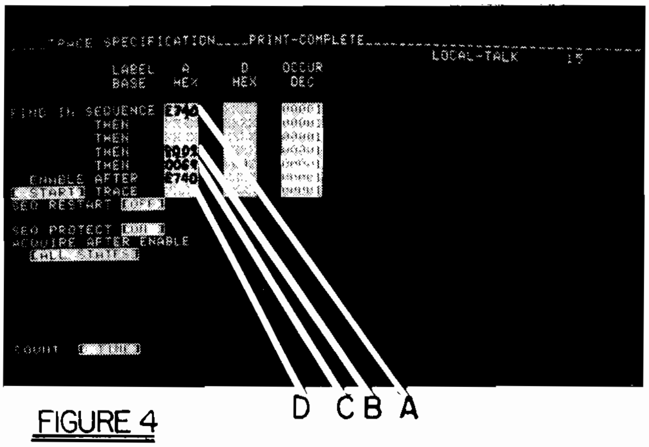
The 9825 calculator is only one example of the type of controller that could be used in this measurement system. The only requirement is that it be able to communicate over HP-IB. This type of measurement system can be a valuable tool for the optimization and analysis of software.

Two Measurement Examples

Two measurement examples will be illustrated in this section. The first emphasizes how the analyzer recognizes these higher level software modules. The second describes measurements made on a system that led to considerable performance improvement in an operating system.

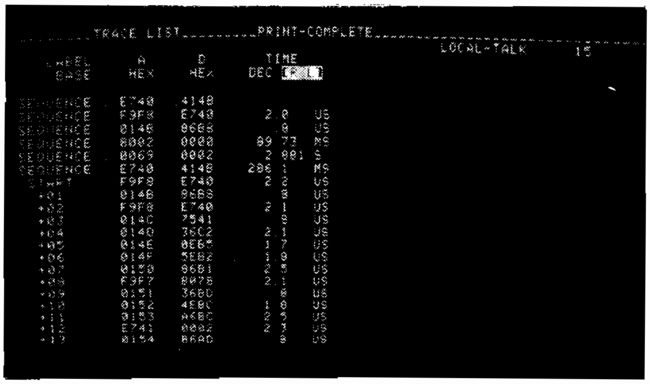
In the first example, the computer system under test is the HP 64000 Development System. The HP 64000 uses a 16-bit processor, called the BPC, that was designed in-house. The method used in this operating system to map external storage to physical main memory is that of manual segmentation, otherwise known as an overlay technique. With this type of memory management, program modules must be mutually exclusive and each program module must call the operating system to load and execute the next overlay.

The Assembler for 8080 code was chosen as one of the high level software modules to be analyzed. Some measurements were also made on the operating system module called LOADEXEC. The 8080 Assembler written for the 64000 System has five overlays. Depending on the options that were called for by the user, it is possible that not all five overlays would be brought into memory. The task of the 1610A/B was to identify the modules, know when they were executing, and measure the execution time. Format specifications on the 1610A/B were set up to look at the 16 bits of address and the 16 bits of data available on the BPC bus. Shown in Figure 4 is an example of the Trace Specification to identify Pass 2 of the assembler. (A) is the beginning address of the LOADEXEC routine and (B) is the starting address of the overlay. (C) is an address that is written to by the assembler module and identifies which module is being executed. For example, the data word would be 0003 if the third overlay were being executed. This was a software testpoint that had been inserted by the designer. It is also possible to identify the module by its name, since the name is passed as a parameter to the LOADEXEC routine. Three sequence terms would be needed on the analyzer to recognize the routine in this way since the name is packed into three 16-bit data words. In this case the three sequence terms for the name recognition would come before (B). The last term (D) signals the end of the overlay since it is the starting address of the LOADEXEC routine which is called at the end of each assembler module. A Trace List that was obtained for the second overlay is shown in Figure 5. Since the Count



```
____TRACE SPECIFICATION___PRINT-COMPLETE_____________________
                                                LOCAL-TALK     15
            LABEL     A      D     OCCUR
            BASE     HEX    HEX     DEC

FIND IN SEQUENCE   E740           00001
            THEN                  00001
            THEN                  00001
            THEN   0002           00001
            THEN   0069           00001
    ENABLE AFTER   E740           00001
  START    TRACE
SEQ RESTART [OFF]

SEQ PROTECT [ON ]
ACQUIRE AFTER ENABLE
  [ALL STATES]

COUNT  [OFF]
```

FIGURE 4



```
______TRACE LIST_________PRINT-COMPLETE____________________
                                                LOCAL-TALK     15
   LABEL      A      D        TIME
    BASE     HEX    HEX     DEC [REL]

SEQUENCE    E740   4140
SEQUENCE    F9F0   E740       2.0    US
SEQUENCE    0140   0660        .8    US
SEQUENCE    0002   0000     99.73    MS
SEQUENCE    0069   0002     2.001    S
SEQUENCE    E740   4140     206.1    MS
   START    F9F0   E740       2.2    US
     +01    0140   0660        .8    US
     +02    F9F0   E740       2.1    US
     +03    014C   7541        .9    US
     +04    014D   36C2       2.1    US
     +05    014E   0E65       1.7    US
     +06    014F   5E82       1.9    US
     +07    0150   8681       2.5    US
     +08    F9F7   8078       2.1    US
     +09    0151   3680        .8    US
     +10    0152   4E8C       1.0    US
     +11    0153   A68C       2.5    US
     +12    E741   0002       2.3    US
     +13    0154   8680        .8    US
```

FIGURE 5

function of Time was selected in the Trace Specification, the trace list displays a relative time between each sequence term. This time information, as well as the data word identifying the particular overlay, is what is used by the controller. Figure shows the flow of the 9825 program that was written for this example. After the measurement is completed, the results are displayed in the form of a time histogram (see Figure 7) which is done with the user display capability of the HP-IB interface for the 1610A/B. The execution times of these software modules were measured during the assembly of a single program. An accumulation of these execution times was also done during the assembly of ten different 8080 programs.

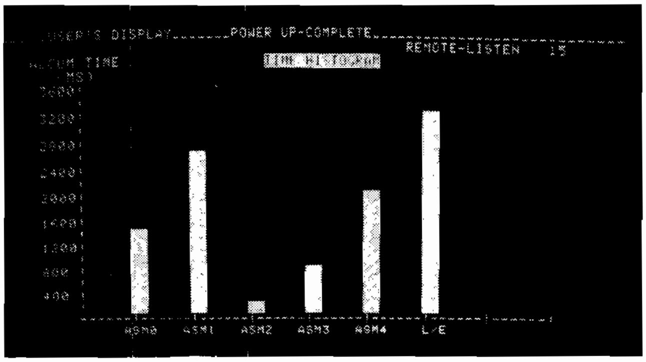
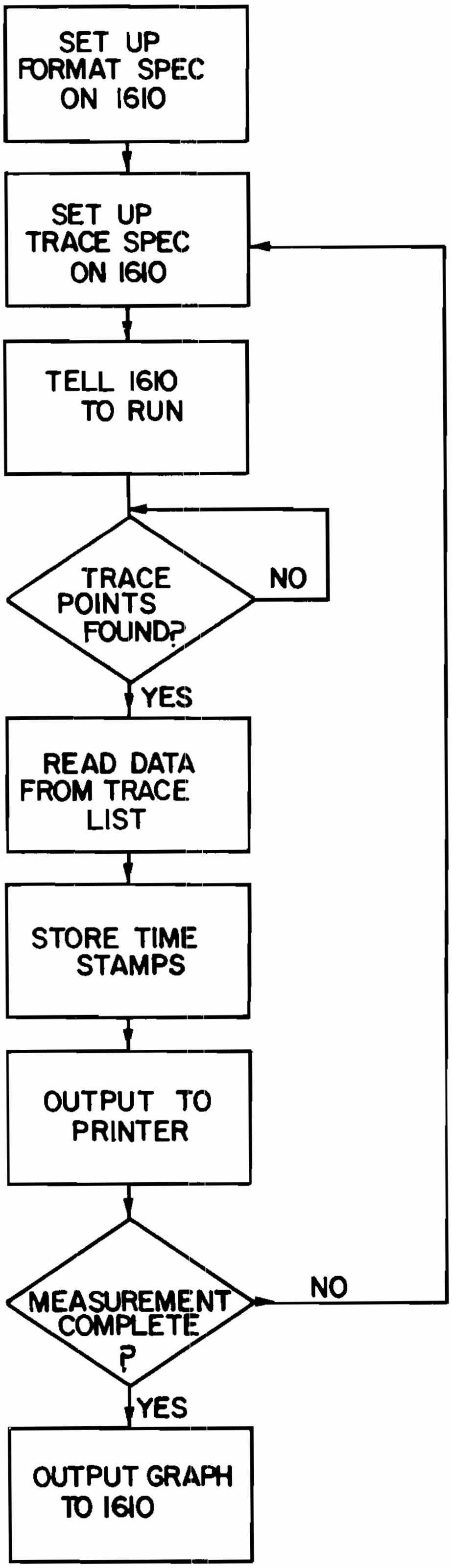


FIGURE 7



9825 Program Flow

FIGURE 6

Measurements were also made on particular subroutines that were called during the assembly process. For these software modules, the frequency of usage was an important parameter. Time measurements were still made, but the distribution of times for the modules was of special interest. Figure 8 shows a distribution of times for five software modules that was printed on the 9866B.

Another measurement that has been made on the 64000 System with a similar setup is a memory map of address space which partitions addresses into eight different buckets. The address range of the buckets is user-definable and the display is again a histogram (see Figure 9). The display shows the total number of addresses acquired, the exact number of addresses which fell within the specified range, and the number of addresses as a percentage of the total acquired addresses for each bucket. For this measurement the 1610A/B traces on the address = XXXX (Don't Cares) and the analyzer takes samples of what is happening on the address bus. There is no interaction with the BPC processor in this case.

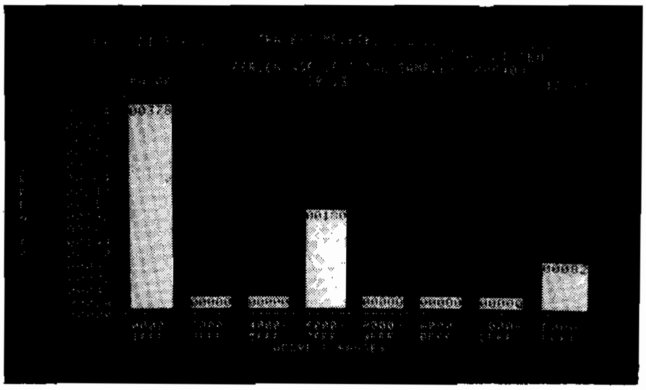
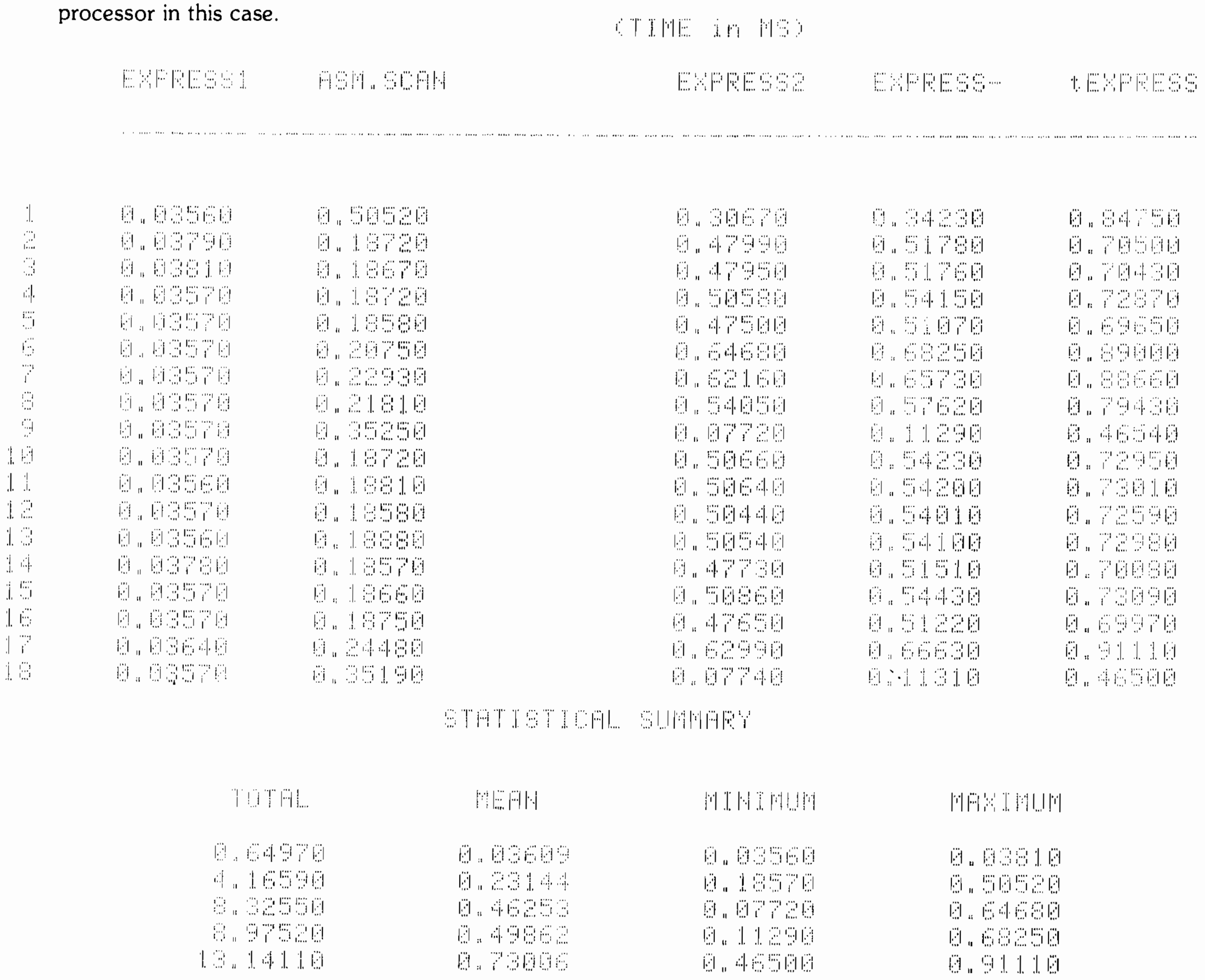


FIGURE 9



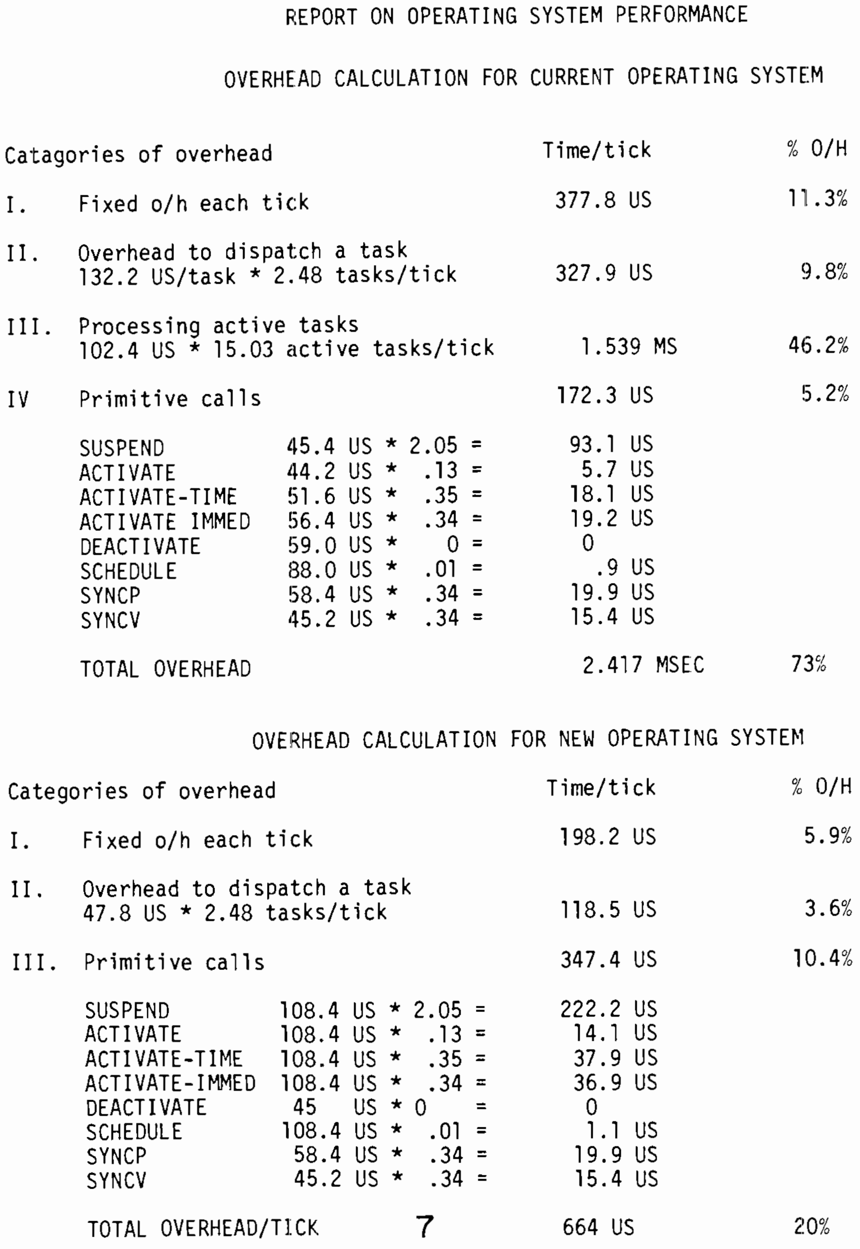
```
                                      (TIME in MS)

      EXPRESS1     ASM.SCAN                  EXPRESS2     EXPRESS-     ↑EXPRESS

 1    0.03560      0.50520                   0.30670      0.34230      0.84750
 2    0.03790      0.18720                   0.47990      0.51780      0.70500
 3    0.03810      0.18670                   0.47950      0.51760      0.70430
 4    0.03570      0.18720                   0.50580      0.54150      0.72870
 5    0.03570      0.18580                   0.47500      0.51070      0.69650
 6    0.03570      0.20750                   0.64680      0.68250      0.89000
 7    0.03570      0.22930                   0.62160      0.65730      0.88660
 8    0.03570      0.21810                   0.54050      0.57620      0.79430
 9    0.03570      0.35250                   0.07720      0.11290      0.46540
10    0.03570      0.18720                   0.50660      0.54230      0.72950
11    0.03560      0.18810                   0.50640      0.54200      0.73010
12    0.03570      0.18580                   0.50440      0.54010      0.72590
13    0.03560      0.18880                   0.50540      0.54100      0.72980
14    0.03780      0.18570                   0.47730      0.51510      0.70080
15    0.03570      0.18660                   0.50860      0.54430      0.73090
16    0.03570      0.18750                   0.47650      0.51220      0.69970
17    0.03640      0.24480                   0.62990      0.66630      0.91110
18    0.03570      0.35190                   0.07740      0.11310      0.46500
                          STATISTICAL SUMMARY

         TOTAL          MEAN           MINIMUM        MAXIMUM

        0.64970        0.03609        0.03560        0.03810
        4.16590        0.23144        0.18570        0.50520
        8.32550        0.46253        0.07720        0.64680
        8.97520        0.49862        0.11290        0.68250
       13.14110        0.73006        0.46500        0.91110
```

FIGURE 8

The second measurement example was performed on the operating system of a product that is currently being developed at another of our HP divisions. A 1610A logic state analyzer and the System 45 Computer were used in the measurement configuration. The measurements were taken while the system was running in RAM and certain software test points were inserted to generate counters that could be sampled with the analyzer. The assumption was made that these test points were a minimal perturbation to the system.

Measurements were made on three different parts of the operating system. They were: the fixed overhead of the operating system, the overhead to dispatch a task, and various primitives. Not only were the execution times noted, but a count was made of the number of times each module was called. Other parameters of interest were (1) the number of tasks that became ready each clock tick, (2) the number of active tasks each clock tick, (3) the execution time of high priority tasks as a percentage of total execution time, and (4) the execution time of all tasks as a percentage of total execution time. With the numbers that were generated for the current operating system, estimates were made for a new operating system. It was predicted that the total O/S overhead could be decreased approximately 67% by modifications that could be made on particular software modules. Figure 10 points out the overhead calculations that were made on the current operating system as well as the calculations for the new system. The current system uses 73% of the total available time and based on the estimates, the new system would only require 20% of the time. The new operating system has been implemented, and it appears that the original estimates were valid.

The histograms and distribution data were produced with the System 45 graphics capability. The on-screen graphics were then transferred to a built-in thermal printer for hard-copy output.



```
                 REPORT ON OPERATING SYSTEM PERFORMANCE

              OVERHEAD CALCULATION FOR CURRENT OPERATING SYSTEM


Catagories of overhead                            Time/tick          % O/H

I.     Fixed o/h each tick                         377.8 US          11.3%

II.    Overhead to dispatch a task
       132.2 US/task * 2.48 tasks/tick             327.9 US           9.8%

III.   Processing active tasks
       102.4 US * 15.03 active tasks/tick            1.539 MS        46.2%

IV     Primitive calls                             172.3 US           5.2%

       SUSPEND          45.4 US * 2.05 =            93.1 US
       ACTIVATE         44.2 US *  .13 =             5.7 US
       ACTIVATE-TIME    51.6 US *  .35 =            18.1 US
       ACTIVATE IMMED   56.4 US *  .34 =            19.2 US
       DEACTIVATE       59.0 US *    0 =            0
       SCHEDULE         88.0 US *  .01 =              .9 US
       SYNCP            58.4 US *  .34 =            19.9 US
       SYNCV            45.2 US *  .34 =            15.4 US

       TOTAL OVERHEAD                               2.417 MSEC       73%


                OVERHEAD CALCULATION FOR NEW OPERATING SYSTEM

Categories of overhead                            Time/tick          % O/H

I.     Fixed o/h each tick                         198.2 US           5.9%

II.    Overhead to dispatch a task
       47.8 US * 2.48 tasks/tick                   118.5 US           3.6%

III.   Primitive calls                             347.4 US          10.4%

       SUSPEND         108.4 US * 2.05 =           222.2 US
       ACTIVATE        108.4 US *  .13 =            14.1 US
       ACTIVATE-TIME   108.4 US *  .35 =            37.9 US
       ACTIVATE-IMMED  108.4 US *  .34 =            36.9 US
       DEACTIVATE       45   US * 0    =            0
       SCHEDULE        108.4 US *  .01 =             1.1 US
       SYNCP            58.4 US *  .34 =            19.9 US
       SYNCV            45.2 US *  .34 =            15.4 US

       TOTAL OVERHEAD/TICK                         664 US            20%
```

FIGURE 10

Summary and Conclusion

This paper has described what measurements can be made on higher level software constructs in a computer system and why anyone would want to have this type of analysis available. The ability to relate data on a system bus to various levels of software was shown to be possible with a computer or calculator controlled logic state analyzer. Two actual systems were characterized in the paper; one example related more of the mechanics involved in making some of these measurements, and the other described how the measurements actually led to the optimization of an operating system.

Even though these new measurement techniques have provided considerable insight into the software domain, the possibility for extending these measurements is extremely exciting. We have the tools available today! Clearly, much more work is needed to explore this new measurement domain, as it appears that we have only scratched the surface.

* UNIX is a trademark of Bell Laboratories.

REFERENCES

1. W.A. Farnbach, "The Logic State Analyzer - Displaying Complex Digital Processes in Understandable Form", Hewlett Packard Journal, January 1974

2. J.H. Smith, "A Logic State Analyzer for Microprocessor Systems", Hewlett Packard Journal, January 1977

3. G.A. Haag, "A Logic State Analyzer for Evaluating Complex State Flow", Hewlett Packard Journal, February 1978

4. C.T. Small and J.S. Morrill, Jr., "The Logic State Analyzer, A Viewing Port for the Data Domain", Hewlett Packard Journal, August 1975

2. Better Use of Logic Analyzers in Dedicated Environments.

HEWLETT PACKARD

Rick Palmer, Colorado Springs R&D Lab, designed the 1802 module for Model 1611A and Internal Analysis for 64000. Rick attended MIT where he earned B.S.E.E./M.S.E.E. degrees.

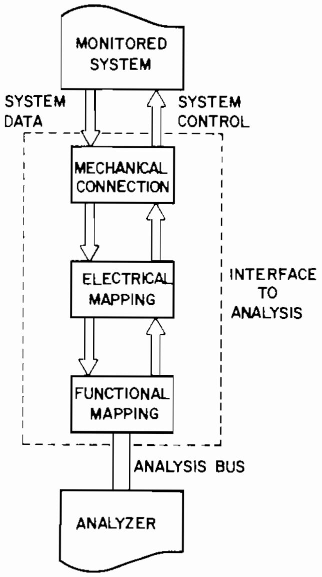
BETTER USE OF LOGIC ANALYZERS IN DEDICATED ENVIRONMENTS

INTRODUCTION

Monitoring the operation of digital hardware with logic analysis tools requires an interface which can transform the representation of information used in the hardware to one that an analyzer can understand. In this paper, it is shown that the definition and realization of this interface should be a consideration during the design of digital system components. The degree to which this should be done depends upon the nature of the design and the associated analysis needs.

Focusing on the interface to analysis is one way to determine the analysis requirements in a specific situation. An understanding of the characteristics of general purpose probes, as well as the ways that a dedicated probing interface can enhance measurement capability, allows an evaluation of the difficulties that will be encountered when analysis tools are used. By supporting analysis within a design, or by implementing a dedicated interface for analysis, these difficulties can be avoided. In addition, specific measurements can be supported by the interface which could not be made otherwise.

The increasing complexity of machine design, the optimization of these designs for specific tasks, and the internalizing of information due to higher levels of integration have lead to a greater need for more capable logic analysis tools. Concurrently, there are more problems in establishing the analysis interface. It is becoming less practical to build a single analyzer which is able to conform to the specific requirements of all bus protocols. If a greater part of an analyzer's resources is spent on this task, then more sacrifices must be made in the measurement set. For these reasons, the trend appears to be in the direction of supporting analysis with dedicated interfaces to overcome the complex mechanical, electrical, and functional probing problems.



THE INTERFACE TO ANALYSIS - FIGURE 1

Figure 1 is a model which describes the major functions of an interface to analysis. There must be a way to mechanically connect to signals. This may be individual probes, as in the case of general purpose probing. It may be a mass termination supplied on a board for cable connection. A plug-compatible board often performs this interface function for dedicated preprocessors.

Once a connection to the signals is made, these signals must be electrically and functionally interfaced to the analyzer. These tasks are accomplished in the electrical and functional mapping blocks. The specific ways in which these tasks are performed are described subsequently. Notice that in the diagram there are arrows shown that travel toward the monitored system. These represent the control functions which can be implemented into a preprocessor or embedded interface.

It is hoped that this paper will supply ideas and inspiration for the design of analysis interfaces for the new bus architectures now being developed. Specific techniques are shown which have been used for interfaces that have been built, and thoughts for making analysis more convenient are presented.

Interfacing Levels

The interface to analysis can be thought of in three levels: general purpose probing, preprocessing, and embedded analysis.

General purpose probing schemes are made available with general purpose logic analyzers to allow connection to signals without the need for a dedicated interface. These schemes consist of probes for data and clock (in the case of a state analyzer) with provisions for mechanical connection to the desired signals. The electrical translation of the waveform to a digital representation for the analyzer is done by comparing the probed signal with a set threshhold.

Primary uses for GP probing are making quick measurements, and making measurements when no dedicated interface is available. When an analyzer is to remain connected for a period of time, it is often desirable to implement some form of dedicated interface, even if only to solve the mechanical problem of keeping many wires connected. When used, however, there are characteristics of GP probes which should be known, although they may usually be ignored. These include required signal swings, active signal and threshold range, loading of the signals, and timing requirements. The causes and effects of these factors will be discussed in the section on general purpose probing.

A dedicated preprocessor is an interface which performs the electrical functional, and mechanical mapping required to transfer information from a specific system to a logic analyzer. It often consists of a board which can plug directly into a connector in the monitored system. Preprocessors have been built for standard buses with known protocols, and has made probing these buses much more convenient. It has been possible to add analysis features on these interfaces, as they operate in a well defined environment. It is also possible to do this for unique, nonstandard architectures to handle complex protocols. Techniques used in building existing dedicated interfaces which have general applicability will be described in the section on preprocessors.

In a third section, the idea of embedding the interface to analysis within system design is examined. Building analysis capability into a design has been commonly done in the past. Front panel controls, debug monitors, and system diagnostic routines are examples of this. The availability of logic analyzers, however, provides an opportunity to obtain control over system activity in real time with more capability for recognizing and interpreting complex events.

GENERAL PURPOSE PROBING

When a dedicated interface is not available for the specific logic that is to be monitored, GP probes must be used. These probes must translate information from an unknown environment to a form which an analyzer can accept. Figure 2 is the model for a GP probing system which will be used in this discussion. It is desirable to keep any disturbances of the signals probed to a minimum, and the restrictions placed upon the signals for correct passage of information should be nonobtrusive to the target system.

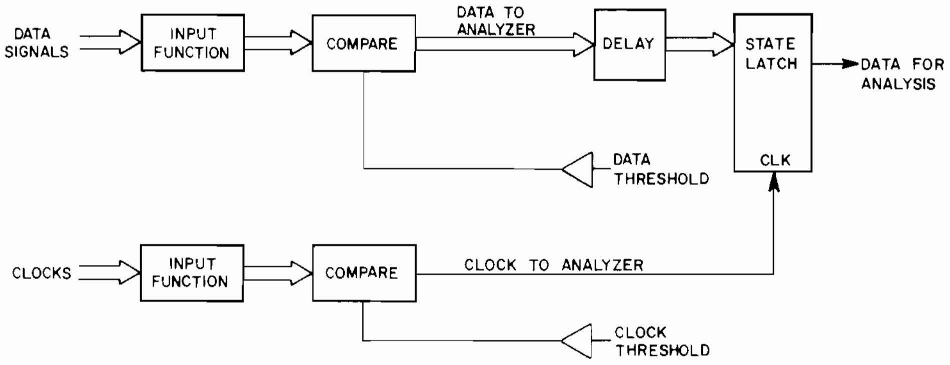


FIGURE 2: GP PROBING

It is the information represented by the waveforms, and not the detailed waveforms themselves, which are important for state analysis. There is an inherent loss of knowledge about the signal levels and swings, as well as the exact timing relationships between the probed signals. The mapping of information from the external domain to the analyzer domain should be 1:1 over a large range of these parameters to avoid limitations in the applicability of the probing system.

It is not possible to make the mapping of information 1:1 over the entire range of parameters, such as signal swing. The specification of the electrical and timing characteristics of a general purpose analyzer (referenced to the probes) describe over what range of the input variables this translation is done without degradation of the information content. A setup and hold specification, for example, describes the required timing relationship between data signals and the analysis clock to guarantee correct data transfer.

Probing specifications, then, define the limitations on the operating range of an instrument. It is necessary to know which parameters are critical, as well as their specified limits, so that a judgement can be made as to whether or not an instrument will operate in a given application. It is often possible to overcome some limitations when the component factors leading to a nonideal specification are known.

Parameters of probing which will be discussed are:

Setup and Hold
Signal Swings and Levels
Signal Loading
Threshold Range

Setup and Hold

The setup and hold specification for a logic analyzer gives the timing requirements for passage of information into the analyzer without distortion of data content. It would be most desirable to know exactly when the input data is sampled with respect to the clock edge. This would allow some timing information about the waveforms to be maintained. It would then be possible to determine if the system under scrutiny was not meeting its own setup and hold requirements. This could be done by skewing the clock and data by an amount which makes the analyzer sample at the minimum setup time to insure that the data was stable.

The actual specifications given are the same as for any logical storage element. A window of time about the transfer clock edge during which the data signals must remain stable is given. Because of this, the signals in a design which can be probed may be limited. See figure 3. This is a common structure found in pipelined, microprogrammable machines. Address and data registers are clocked simultaneously. The clock period is limited by the delay through the address register and the memory, as well as the setup time required for the data register.

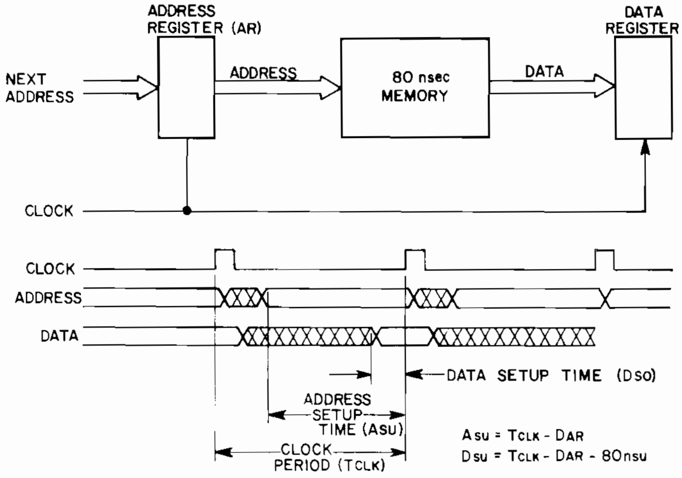


FIGURE 3: DELAY AND SETUP TIME

Assume that an analyzer is available with a setup time of 20 ns and a hold time of zero. The output of the address register can be probed, since the setup time is the clock period minus the delay of that register, which is about 90 ns. Hold time is derived by the positive delay through the register. The signals at the input to the data register, however, do not have the required setup time with respect to the clock, because of the delay in the memory. Since the register only requires 10 ns setup, the circuit operates correctly, while the analyzer will not necessarily see the same data as the register.

It should be noted that the reason a zero hold time specification for logic analyzers is necessary is to insure that at least the output of storage elements can be probed. Whether or not any other signal can be probed must be determined by its timing relationship with the clock.

Components of setup and hold are:

- Probing point capacitance variations
- Analyzer's internal register setup and hold
- Compensation delay for zero hold time
- Channel skew
- Delay variations over temperature
- Comparator variations over threshold

Each of these contribute directly to the specified setup and hold. If temperature is held constant, the only contributor to the window required is channel skew. Other components will remain constant for a particular measurement environment. The actual channel skew for a particular instrument can be calculated by measuring the required timing window at a constant temperature and threshold, and may be found to be considerably less than 20 ns.

Signal Swings and Levels

The minimum signal swing and the maximum input level must be given since there is always some limit for these as a result of nonideal components. The maximum input is determined by the breakdown voltage of the comparators used and the input function applied to the signals. The minimum signal swing is more subtle.

Components of minimum swing are:

- Comparator offset voltage
- Comparator offset current
- Offset skews between comparators
- Threshold amplifier error
- Input divider error
- Hysteresis used for noise immunity

An important point to make is that the minimum swing specification is affected by the measurement error of setting the threshold. The swing required about the actual threshold that the comparator sees is affected only by the offset skew between comparators, the divider error differences between channels, and the hysteresis used. As an example, a 600 mV swing requirement can be reduced to about 200 mV minimum swing if the threshold is adjusted to be centered on the actual signal limits. This is done by tweaking the threshold while observing the data displayed by the analyzer.

Signal Loading

The ideal probe would appear as an infinite impedance to the signal being probed. In reality, some capacitance and resistance to ground is normally seen. The capacitance will cause a change in the actual (nonprobed) timing by reducing edge speeds. The resistance will have some timing effect, but its main influence is an increase in DC current at any voltage other than ground.

Probing capacitance is undesirable when the signal being probed has cirtical timing, or if the signal driver is very sensitive to this type of load. When the last nanosecond is used to advantage in a design, any incremental delays must be accounted for. The degree to which the additional capacitive load due to probing will affect the timing in a specific case should be determined.

The resistive component is a consideration when the existing load presented to a driver is its maximum fanout. Since TTL is specified fairly conservatively on their high level fanout specs, this is rarely seen as a problem with that family. In logic families with less DC drive capability, the resistive loading should be considered. The sensitivity of gain and phase of analog circuits to load resistance made this a more critical parameter for oscilloscope probes. In logic circuits, the capacitive component is more apt to create problems.

Threshold Range

The active signal range over which the threshold applies should be known when logic families other than TTL or ECL are to be used. All general purpose probing schemes with variable threshold will cover the range required for these. A range of +10 volts to -10 volts is common, and covers all of the standard logic families. If a signal which switches between 24 volts and 48 volts is to be probed, an external translation to the active range of the analyzer must be performed.

Other Considerations

There are other characteristics of general purpose probing systems which often affect their usefulness in specific applications. Some of these are mentioned here.

It is not practical to have separate probes and thresholds for each channel, as for an oscilloscope, when there are many channels. The logical division of channels, both mechanically and with respect to threshold, will determine whether widely separated signals can be probed and whether signals from different logic families can be analyzed together. A reasonable tradeoff must be made in the grouping of channels.

The clocking function of an analyzer specifies when data is sampled in the system and when to transfer this data to the analyzer. Multiclock schemes are useful for demultiplexing buses, and for aligning data in pipelined architectures. These capabilities generate more specifications on the required signal timing, such as minimum clock-to-clock time and the setup and hold of data with respect to each clock. Any qualifier bits, which enable and disable clocks, also have timing requirements. The clocking functions at the front end of an analyzer should be known since they can be useful in unravelling bus protocols.

A variety of mechanical hookup techniques have been developed which make connection of many channels to an analyzer more convenient. Accessories made available for multiple channel hookup should be considered when a connection is to be made for more than a one-time measurement.

Comments

To make the best use of general purpose probing, operating ranges of the important input parameters should be known. If it is necessary, in a particular situation, to operate an analyzer out of its specified range, external compensation can sometimes provide a solution. In other cases, knowing the components of the error can generate a technique to overcome the specified limitation (as in the case of tweaking the threshold to reduce minimum signal swing).

It is also possible to increase the usefulness of GP probing by supplying some hooks for analysis in the original design. For example, an additional register could be added to the circuit of figure 3 which latches the output of the address register. This would allow passing address and data to an analyzer in parallel without a setup time problem. This is exactly the kind of function which could be built into a preprocessor interface for that architecture. Many other functions can be performed in this type of interface, and these are presented in the next section.

PREPROCESSORS

The function of a preprocessor is to transform the representation of information in a known environment to a representation appropriate for an analyzer. Because the environment is well defined, this type of interface to analysis can compensate for the complexitites of the representation, and also perform functions which are particularly useful for the given architecture. Figure 4 shows the basic functions of a preprocessor.

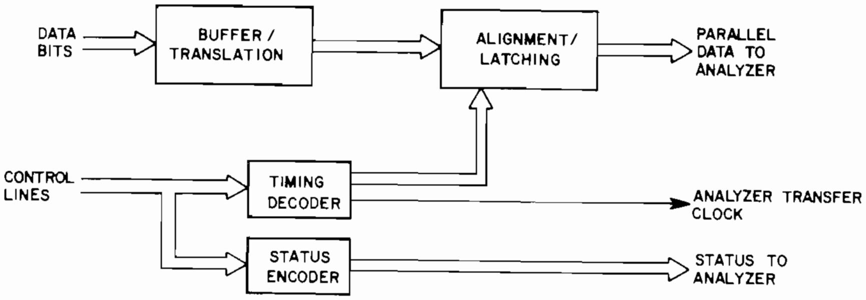


FIGURE 4: MODEL OF PREPROCESSING INTERFACE

Buffering the data and control signals minimizes loading of the monitored bus to standard levels. For a specific bus, the maximum loads that can be applied are usually specified, and the preprocessor must conform to these. If the logic family used on the monitored bus is different than that of the interface, then a level translation is also performed.

The interface must determine when to sample data from the bus, and when to send this data to the analyzer with a transfer clock. In the simplest case, all data on the bus is valid with sufficient setup and hold, with respect to a single bus strobe. Then this strobe can be the transfer strobe and no latching of bus data is needed. More commonly, data on the bus is valid at different times, with more than one control line required to determine validity. Each set of data bits must then be latched when they are valid, and after all of the bits have been gathered, a transfer strobe is derived to send them to the analyzer. This function is performed in the alignment/latching and timing decode blocks of figure 4.

Control lines are assembled into a status word by the status encoder block. This word is then sent to the analyzer as part of the parallel data word. Typical types of bus cycles are OPCODE FETCH, READ, WRITE, INPUT, OUTPUT, DMA IN, DMA OUT, INTERRUPT, etc. Identifying the type of bus transaction which occurred is used by the analyzer in such functions as storage qualification and trace point recognition. As an example, it is then possible to store only OUTPUT cycles and ignore all other cycle types. This is a common measurement need for debugging I/O problems.

The actual codes which are used to encode the status can affect analysis features. If a single bit were used for each type of status, any combination of cycle types can qualify analysis events. If the eight types given above were coded into three bits, however, only certain combinations of the eight can be derived, determined by the coding of bits. It is often desirable to encode status information to reduce the number of bits required to pass this information to the analyzer. The affects on analysis of the coding of status should be considered in the design of a preprocessor.

Buffering, data alignment, timing decode, and status encoding may be considered the basic functions of a preprocessor. In addition to these, many other functions can be added to the interface which enhance its usefulness.

Data Selection

An analyzer typically has a fixed number of bits upon which it performs analysis functions. If the bus to be monitored has more bits of information than the analyzer can accomodate, then the decision must be made as to which bits to ignore. An alternative is to select particular bits for the analyzer, either statically or dynamically.

Static selection means that for one measurement a certain set of bits are analyzed, and the others are not. This is useful if the sets of bits are really independent, such as in monitoring two buses with one interface and either bus can be selected without physically moving the interface. Another case is incorporating the ability to monitor different sets of status bits on a bus that are independent. It may be possible to monitor interrupt control lines or DMA control lines, for example, but not at the same time.

Dynamic selection is a technique of selecting the important sets of bits as a function of the status of the bus and sending codes to the analyzer which specify which set of data is being presented. For a processor with multiple I/O ports, for instance, a preprocessor may be able to determine which port is active and switch that port's data onto the analysis bus. The status information sent to the analyzer would include bits to identify the port through which the data was transferred. Static selection could also be used, in which case only one port could be monitored during a measurement, but the status information would not then be required.

Error Detection

It is desirable to perform error detection in a preprocessor for two reasons. First, a state analyzer cannot detect timing or electrical errors. An opportunity exists in the design of a preprocessor to pass information about these types of problems to an analyzer via status bits. Second, by letting the preprocessor do error detection, state measurements can be done in parallel by the analyzer. More complete use can then be made of the analyzer's feature set.

The detection of bus protocol errors in a preprocessor can often be done with a simple state machine monitoring activity on the control lines. An asynchronous circuit may be required for detection of timing errors, while a synchronous implementation will detect state flow problems. When an error is found, the condition should be latched until the next transfer clock to the analyzer. This guarantees that the analyzer will be able to see that the problem occurred.

Reconstructing Internal Information

With the increasing capabilities for integration of circuits, buses which concisely represent the overall activity have been internalized. Such is the case for the 8086 processor. The external bus activities are isolated from the real process by the on-chip bus interface unit. For analysis, the ideal place to probe the 8086 is on the execution bus, which is inside the chip.

A preprocessor can be imagined which reconstructs the operation of the bus interface unit, and derives the actual instruction flow. This data could then be passed on to the analyzer. This type of preprocessing function will become more important as more and more of the convenient nodes of information are internalized.

Analysis Features

In some cases, it may be found that the available analyzer cannot perform a particular measurement. It may be possible in this case to add measurement capability to the preprocessor to perform the desired function.

A simple example is adding the ability to monitor only every Nth state, to obtain a general picture of state flow. See figure 5. A counter can be used to derive a clock to transfer data to the analyzer only when the carry is true. By making the divide constant programmable, the analyzer will receive every Nth state. This type of preprocessor function is called prereduction. Analysis features included in preprocessors usually perform some form of prereduction of the state flow information for an analyzer. Other examples of prereduction would be passing only certain types of bus cycles to the analyzer, or generating complex status information, such as when a processor is in user vs supervisory modes.

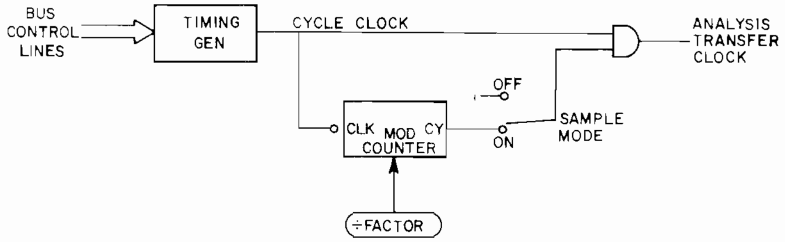


FIGURE 5: PREREDUCTION OF DATA BY SAMPLING

Control

General purpose logic analyzers typically offer outputs which can be used to stimulate the monitored system. These may be used to halt the ongoing process, interrupt the process, or drive other dedicated functions made available to the interface. The ability to halt when the analyzer memory is full, for example, allows paging through the process while obtaining a record of all activity between halts. This is like single stepping, but the relationship between many cycles can be seen.

Comments on the Complexity of Information

The design techniques used to get more throughput and features from a given amount of hardware have become increasingly complex. The effect on the requirements for analysis is twofold. First, there is a greater need for analysis due to the higher likelihood of bugs in more complex designs, and the difficulty of monitoring activity without dedicated tools. Second, the analyzer must be tailored to the dedicated environment of the design since a general purpose front end cannot begin to encompass all of the new techniques. This leads to a greater need for dedicated interfaces to analysis.

So, in some cases a dedicated interface is needed just to unravel the way that the information is represented. Another source of the need for preprocessors is the increasing tendency toward internalizing of information. As IC technology advances, the real nodes of information are moving inward, where standard analysis tools cannot reach them. Such is the case for analysis of some of the new 16-bit processors. Effective monitoring of the activities of such processors requires that the internal execution bus be reconstructed. The external bus not only has the data scrambled in order, but some of it may not even be valid, and the validity is a function of the sequence of fetched instructions and data. Surely, a dedicated analysis tool would greatly improve analysis of such a machine.

And so, with the ever more complex architectures being designed for more efficiency and throughput, the analysis problem becomes correspondingly more complex and demanding. The need to consider analysis problems during design has thus become more critical.

EMBEDDED ANALYSIS

Embedding analysis tools within a system's design has been done since the first computers were designed. It was, in fact, the complexity made possible by such organized and regular designs that led to the need for analysis tools. The availability of general purpose logic analyzers, however, provides an opportunity for supporting analysis in all designs.

Consider use of a logic analyzer as an integral part of a system. The hardware interface to analysis could then include functions which allow passing information about the activity of the system to an external instrument, under software control. A debug monitor, resident in a separate analysis memory, could allow application programs to access the analysis features through a monitor level interface. A diagram of this type of system integration of analysis is presented in figure 6.

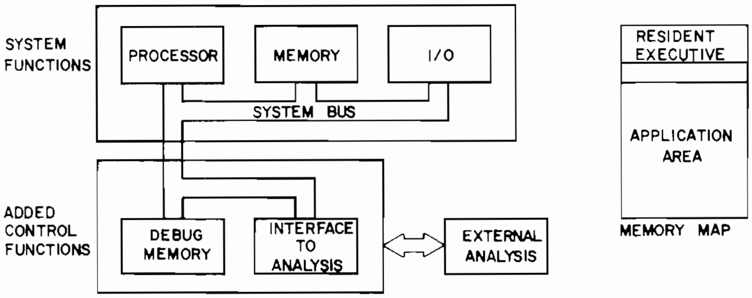


FIGURE 6: ADDING CONTROL OF A SYSTEM FOR ANALYSIS

Some of the hardware added to the basic design performs the same functions that have been described for external preprocessor interfaces. Other tasks can be performed by the hardware interface, however, when it is built into the design. The debug memory, for example, is memory space added specifically for the use of analysis interface routines. The memory space available for application programs is therefore not reduced. The analysis interface may also be made hardware-addressable by the system so that the software interface can pass specific information for display to the analyzer.

The software interface, as previously mentioned, resides in the debug memory. The operating system must support these external programs, so it is not possible to make the existence of the integral analysis interface transparent. The function of the debug software would be to allow setting breakpoints via the analyzer's control outputs, to program the operating mode of the interface to analysis, and to offer a standard way to send information to the analyzer. Other utilities could be offered in the debug space, such as programs which will do prereduction of the information sent to analysis from an application program or display and alteration of application memory contents.

An example of integrated analysis is the design of the emulation and analysis modules which are available for standard microprocessors.

In these tools, the analysis capability is directly coupled to functions for control over program flow, giving essentially the same architecture as that shown in figure 6. A general purpose logic analyzer can be used to provide the same conveniences in a dedicated environment. The requirement is that the analysis functions must be supported in the system design, and the benefits that can be obtained by such an approach must be weighed against this task.

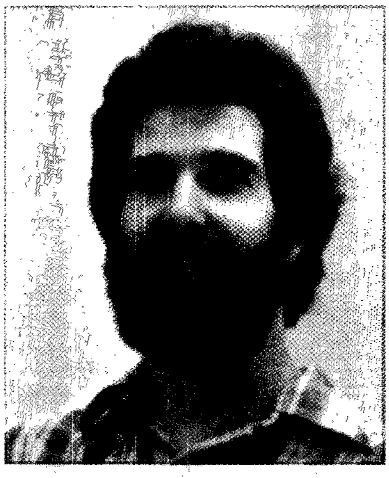
CONCLUSIONS

The need for dedicated interfaces to analysis tools becomes greater as the complexity and diversity of new designs increases. It will become even more difficult to define a general purpose logic analyzer which can solve the majority of probing and measurement problems. Through consideration of analysis needs during design, and through implementing a dedicated interface to analysis, a general purpose analyzer can be used very effectively to solve specific measurement problems.

This paper has presented some of the techniques which have been used to employ analysis more effectively. This has been an attempt to generate an awareness of the possibilities, and is not meant to be all-inclusive. The hope is that the viewpoint of considering the interface to analysis as a separate entity will lead to the development of innovative solutions to specific measurement problems using logic analysis tools.

3. Minicomputer Development Using Logic Analyzers.

HEWLETT PACKARD



Greg Hansen holds a B.S.E.E. from U. Colorado and MBA from University of Santa Clara. A hardware and firmware engineer for HP computers, he is a Section Manager of the Data Systems Division.

#3

Minicomputer Development Using Logic Analyzers

By
Greg Hansen
Hewlett-Packard Company
Cupertino, Calif.

INTRODUCTION

The design of a new minicomputer is a long process, involving the efforts of many engineers, and a process that brings together many disciplines — product design, analog design, digital design, LSI design, and software design. There is no one tool that can meet the needs of all of these designers; at best we try to use one to cover as much as possible.

The logic state analyzer has been one tool that has met the needs of most of the people on our latest effort, being used by our analog, digital, LSI, and software design engineers throughout the development phase. These uses are best illustrated by example; hence this paper. This paper will attempt to highlight the major uses to which we have applied logic analyzers in the course of developing one of HP's new processors, the HP 1000 L-Series.

THE HP 1000 L-SERIES

The HP 1000 L-Series is the newest addition to the 1000 family product line. It is based on two custom Large Scale Integrated (LSI) circuits which were designed as part of the overall project. These LSI devices were designed using HP's advanced Silicon-on-Sapphire (SOS) technology, to implement a high-performance processor in the microcomputer area of the market. Our use of LSI has greatly enhanced the L-series computer, but also posed additional design problems, as seen later.

A simplified block diagram of the L-series is shown in Figure 1. The bus structure chosen has sixteen data lines, fifteen address lines, and 36 control lines. These lines are contained on two 50-pin connectors on one end of the board; the other may be used for connection to external devices, in the case of the I/O cards.

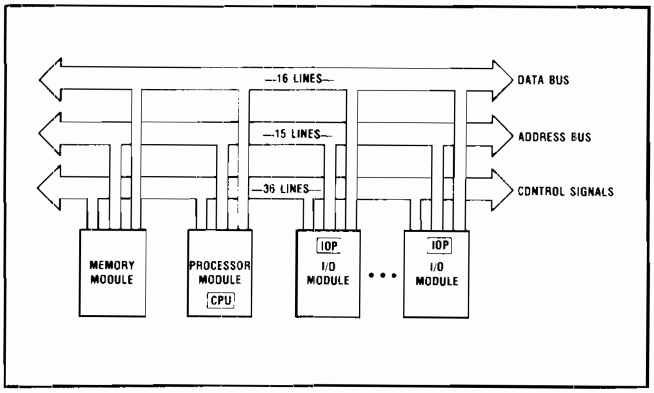


Figure 1. Simplified L-series block diagram.

There are four major types of bus activity:

1. Instruction and operand fetches.
2. Direct Memory Accesses (DMA), reads or writes.
3. I/O instruction processing.
4. Interrupt processing.

Instruction and operand fetches are initiated by the processor in the normal course of instruction execution. The backplane protocol is identical to a Direct Memory Access by an I/O card, with the exception that the signal RNI (Read Next Instruction) is asserted, to alert all I/O processors that an instruction is being fetched for possible I/O processor execution.

DMA accesses are initiated by any of the I/O modules, since each CMOS/SOS I/O Processor contains all necessary DMA logic as part of its standard circuitry, a feature unique in the small computer market. The accesses are prioritized in daisy-chain fashion, and are the primary method of data transfer into and out of an L-series computer.

I/O instruction processing may involve interaction between the processor module and the I/O module, as in the case of a transfer of the desired word count on a DMA transfer to the I/O processor chip, or it may be accomplished solely by the I/O processor chip itself (as in a start command). If CPU to I/O processor communication is required, it is interleaved with the DMA activity on a lowest-priority basis.

Interrupt processing is a cooperative effort between the processor and the highest-interrupting-priority I/O device, where the processor initiates an instruction fetch from the memory location supplied by the I/O processor chip. This implements the HP 1000 vector interrupt feature.

By monitoring the flow of data over the backplane and the associated control signals, the user has visibility into the step-by-step operation of the computer.

USE OF LOGIC ANALYZERS IN EACH DEVELOPMENT PHASE

Every project follows a sequence of well-defined (and, hopefully, well-implemented) design phases:

- Inital product definition.
- Breadboarding to verify functionality.
- Conversion to prototypes.
- Verification of integrated circuits, if any.
- System integration.
- Release to manufacturing.

Logic analyzers were used during each phase of the L-series, contributing heavily to the speed with which new designs were checked out. An example of the use of logic analyzers during each phase has been selected, showing their power and flexibility.

A. Initial Design-Architecture Definition.

The architecture of the L-Series evolved from its major objective of implementing a very efficient I/O structure that would allow the Real Time Executive (RTE) operating system to spend dramatically less time servicing input-output related events. Experience with our existing processors showed that as much as thirty to forty percent of the processor execution time could be spent on I/O-related activities, primarily related to interrupt servicing. We attacked the problem in both hardware and software: The software was designed to more efficiently handle bus-oriented interfaces (such as the HP-IB* interface), and the hardware was designed to allow every interface to directly transfer data to memory, only generating an interrupt on the completion of a transfer.

Figure 2 shows the difference in overall processor throughput when this concept is utilized.

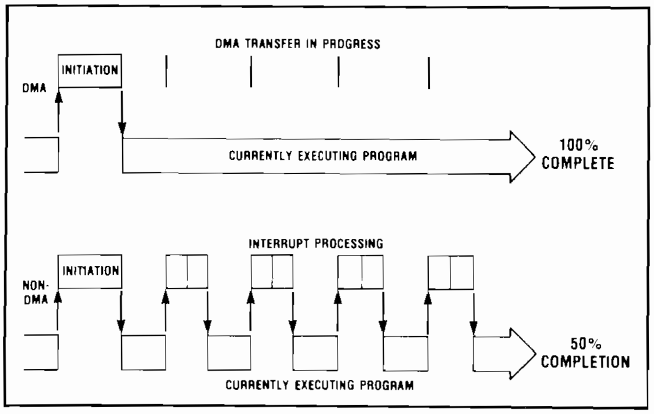


Figure 2. Direct memory access = high performance.

To accomplish these objectives, a standardized interface to the L-series backplane was defined, and the I/O Master and I/O Processor concepts formed. The I/O Processor is a custom-designed CMOS/SOS device that incorporates logic to do I/O instruction execution, as well as to handle all Direct Memory Access (DMA) functions. The I/O Master incorporates the I/O Processor and its associated TTL support circuitry into an identical circuit on each I/O board. Each I/O board has its basic functions provided by the I/O Master, so the software sees a very uniform structure to the interface on such widely disparate cards as a 16-bit parallel interface and an asynchronous serial interface. This uniformity of interface allowed the I/O hardware and software designers to concentrate on the peculiarities of their particular I/O function, while guaranteeing that overall system performance was maintained.

Once the backplane definition had solidified, we worked with the designers at our logic analyzer division to produce an interface that would allow that division's logic analyzers to quickly be connected to the computer using a plug-in interface. This has proven to be a valuable debug aid.

B. Breadboarding

Once the architecture was defined, the individual designers then produced the detailed designs to realize the desired functions. In order to produce a working prototype of the computer, it was necessary to generate at least a memory, a processor board with the CPU chip being emulated by its TTL breadboard, and one I/O interface with the I/O processor being emulated by its TTL breadboard.

The CPU chip was brought up first. Its block diagram is shown in simplified form in Figure 3. It was designed as an algorithmic state machine with a clocked asynchronous interface to the rest of the system, to allow memory operation to occur in any amount of time from 2 to N cycles (where N is as big as you are willing to wait for!). The logic analyzer was connected to the output of the state machine to monitor the transitions from state to state, and also connected to the data bus and as many address bus bits as we could. In this fashion we could manually provide inputs for instructions and verify that the CPU was correctly executing these instructions (see Figure 4).

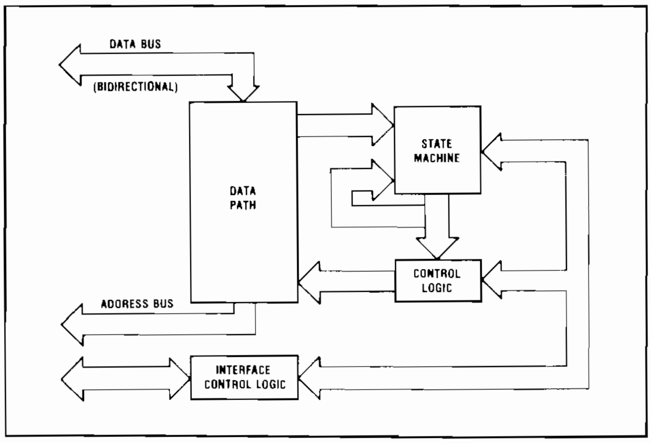
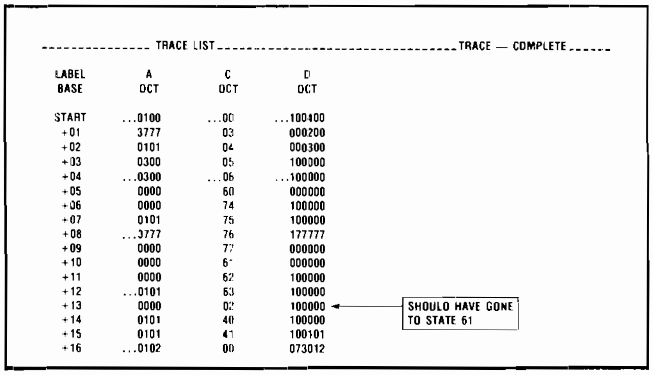


Figure 3. Simplified CPU chip block diagram.

*HP-IB is Hewlett-Packard's implementation of the IEEE 488-1975 interface standard.

This configuration was used on all instructions, but was particularly helpful in finding out why the divide algorithm was failing — it was possible that the control structure was incorrect, or that the data structure was at fault. By having the state sequence and the data bus available, the problem was quickly isolated to an incorrect state branch — the correct state sequence for Figure 4 should have been states 63, 64, 65, 66, 67, and then 40! Further investigation revealed that a don't care on a Karnaugh map was not actually a don't care.



```
--------------- TRACE LIST -------------------------------- TRACE — COMPLETE ------

LABEL      A        C        D
BASE      OCT      OCT      OCT

START   ...0100    ...00   ...100400
 +01       3777       03      000200
 +02       0101       04      000300
 +03       0300       05      100000
 +04    ...0300    ...06   ...100000
 +05       0000       60      000000
 +06       0000       74      100000
 +07       0101       75      100000
 +08    ...3777       76      177777
 +09       0000       77      000000
 +10       0000       61      000000
 +11       0000       62      100000
 +12    ...0101       63      100000
 +13       0000       02      100000 ◄────── SHOULD HAVE GONE
 +14       0101       40      100000         TO STATE 61
 +15       0101       41      100101
 +16    ...0102       00      073012
```

Figure 4. CPU chip state machine operation.

As soon as the CPU chip emulator was ostensibly "working", our attention then focussed on plugging it into the processor board with a memory and getting the processor-memory interface established. This involved operation of circuitry both internal and external to the "chip", so we left the one logic analyzer connected to the chip emulator and attached a second analyzer to the backplane, giving us complete visibility as to the internal and external workings of the machine [Figure 5]. Since the processor-memory interface is very straight forward and synchronous, the checkout was done using simple functional checks on the handshake lines at slow speeds. A sample processor-memory handshake is shown in Figure 6.

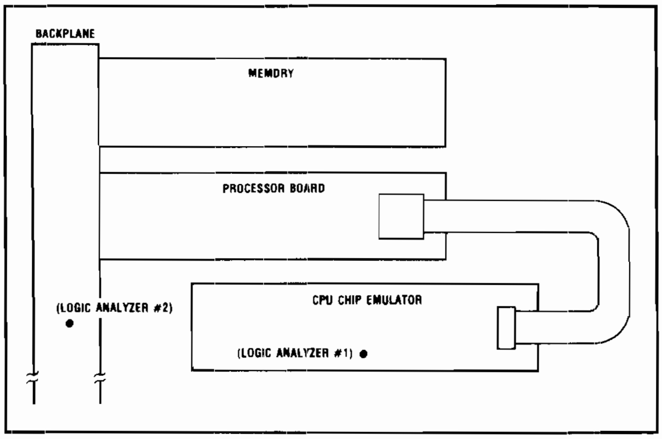


Figure 5. Chip and backplane logic analyzer.

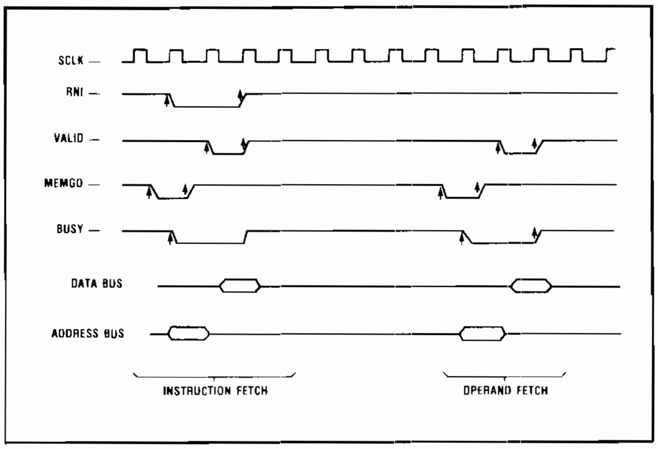


Figure 6. Sample processor-memory handshake.

At this point we were ready to attach our first I/O interface and device — a serial input/output card to allow connection to a terminal. The I/O processor emulator had been initially checked out manually, in a fashion similar to the CPU chip (again using a logic analyzer), so again the interface between functional modules was the focus of our attention. The I/O processor is unique in that it serves two roles in the system, concurrently. The first role is that of an I/O instruction executer — when I/O instructions are received from the backplane, it is the responsibility of the I/O chip that that instruction addresses to see that the instruction is executed. At the same time, the I/O chip may be performing its direct memory access function, moving data from the I/O device to memory or vice versa. Since both operations are conducted over the same backplane lines, they are not simultaneous, but may be interleaved (see Figure 7). This led to the second example of logic analyzer use — finding out why interleaved operation destroyed seemingly random memory locations.

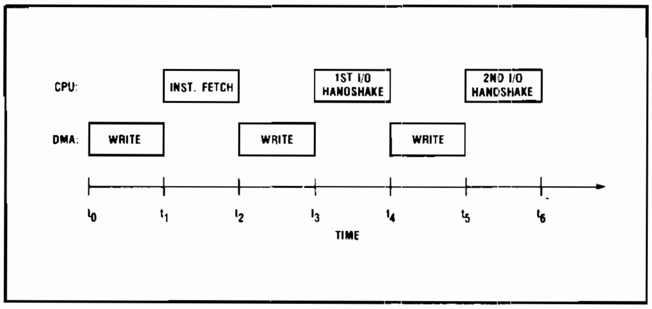


Figure 7. Interleaved instruction and DMA processing.

To check out what was happening, the logic analyzer was set to record each successive DMA write to memory, both address and data. One trace that was observed is shown in Figure 9. Only one channel of DMA was active, so each address should have been the successor to the previous address. As can be seen, on one line this was not the case; an interesting observation is that on the next write the address was correct again. Something was temporarily destroying the address external to the I/O processor chip, since internally it seemed to be correct. The logic analyzer was then shifted to the I/O Processor chip bus (Figure 8), and the attendant enables.

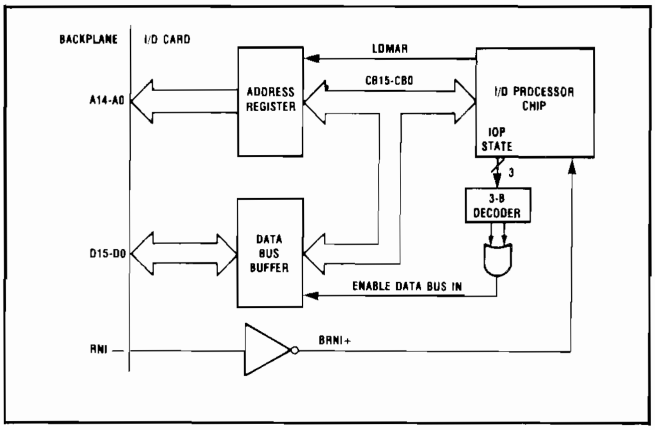
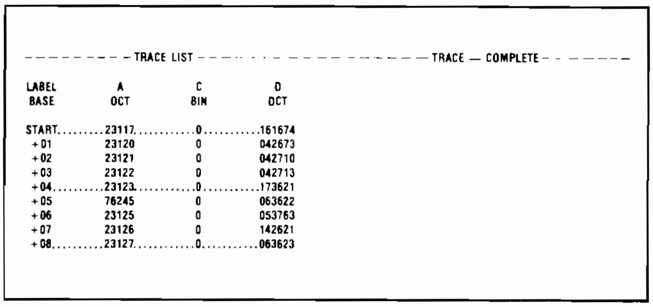


Figure 8. I/O processor chip bus.



```
--------- TRACE LIST ----------------------------- TRACE — COMPLETE ---------

LABEL       A          C          D
BASE       OCT        BIN        OCT

START.........23117...........0..........161674
  +01        23120            0          042673
  +02        23121            0          042710
  +03        23122            0          042713
  +04........23123............0..........173621
  +05        76245            0          063622
  +06        23125            0          053763
  +07        23126            0          142621
  +08........23127............0..........063623
```

Figure 9. Trace list showing bad DMA write (see start +05).

This very quickly revealed that the signals LDMAR (load memory address register) and Enable Data Bus In overlapped for one backplane clock just prior to the defective write (time t_2 of Figure 7), with the data bus buffer overpowering the I/O Processor chip (see Figure 10). Further paper analysis revealed that a related signal, MRQ, was the source of the problem; MRQ had to be removed when RNI appeared, so that the I/O processor DMA state machine would not assert the DMA address and the signal LDMAR while the data bus was still enabled onto the chip bus.

C. Prototype Phase

After some amount of time spent with the breadboard, running first diagnostics, then the preliminary operating system, it was time to commit the designs to actual printed circuit board implementations that could replace the somewhat awkward wire-wrap boards; similarly, the two LSI devices were laid out (actually, the chip layouts overlapped the breadboard phase, due to the relatively long lead times on LSI designs versus printed circuit boards, but up until then it had been largely at the block-level planning stage). The printed-circuit boards arrived well before the chips, and replaced their respective wire-wrap boards; we were ready to receive the chips so that we could proceed to the environmental part of our testing and verify our design margins.

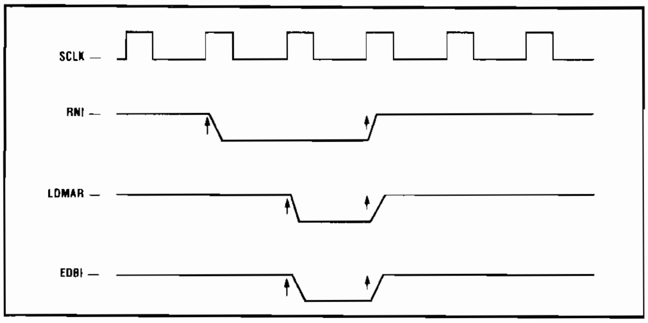


Figure 10. Trace showing LDMAR and enable data bus in overlapped.

1. Chip Checkout:

At this point one of the project engineers suggested a way to use the logic analyzer to get visibility into the inner workings of the CPU chip — an LSI designer's dream!

Since the CPU chip is bus-oriented (see Figure 11) and the chip designers had fortuitously chosen the default state of the data bus transceivers to be enabling the internal data out, a properly connected logic analyzer and a copy of the CPU chip state diagram (see Figure 12 for an example of this) allowed the engineers to "see" what was happening, even in intermediate states of execution, in the packaged part.

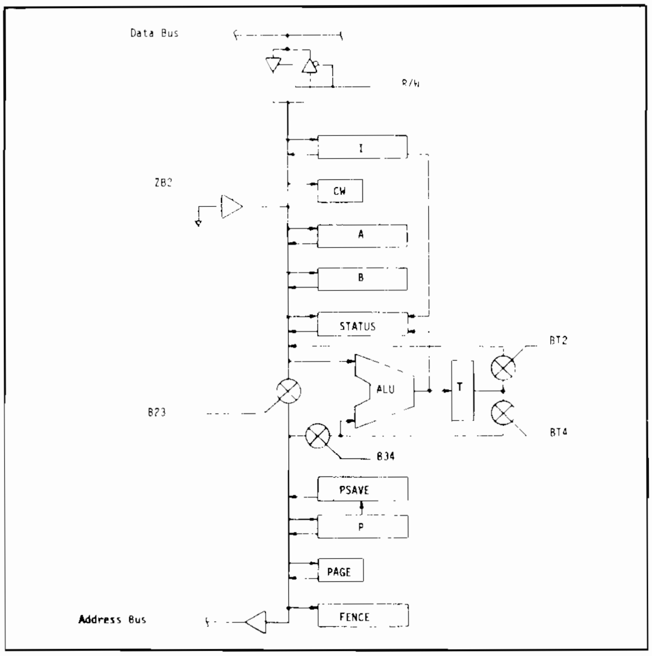


Figure 11.

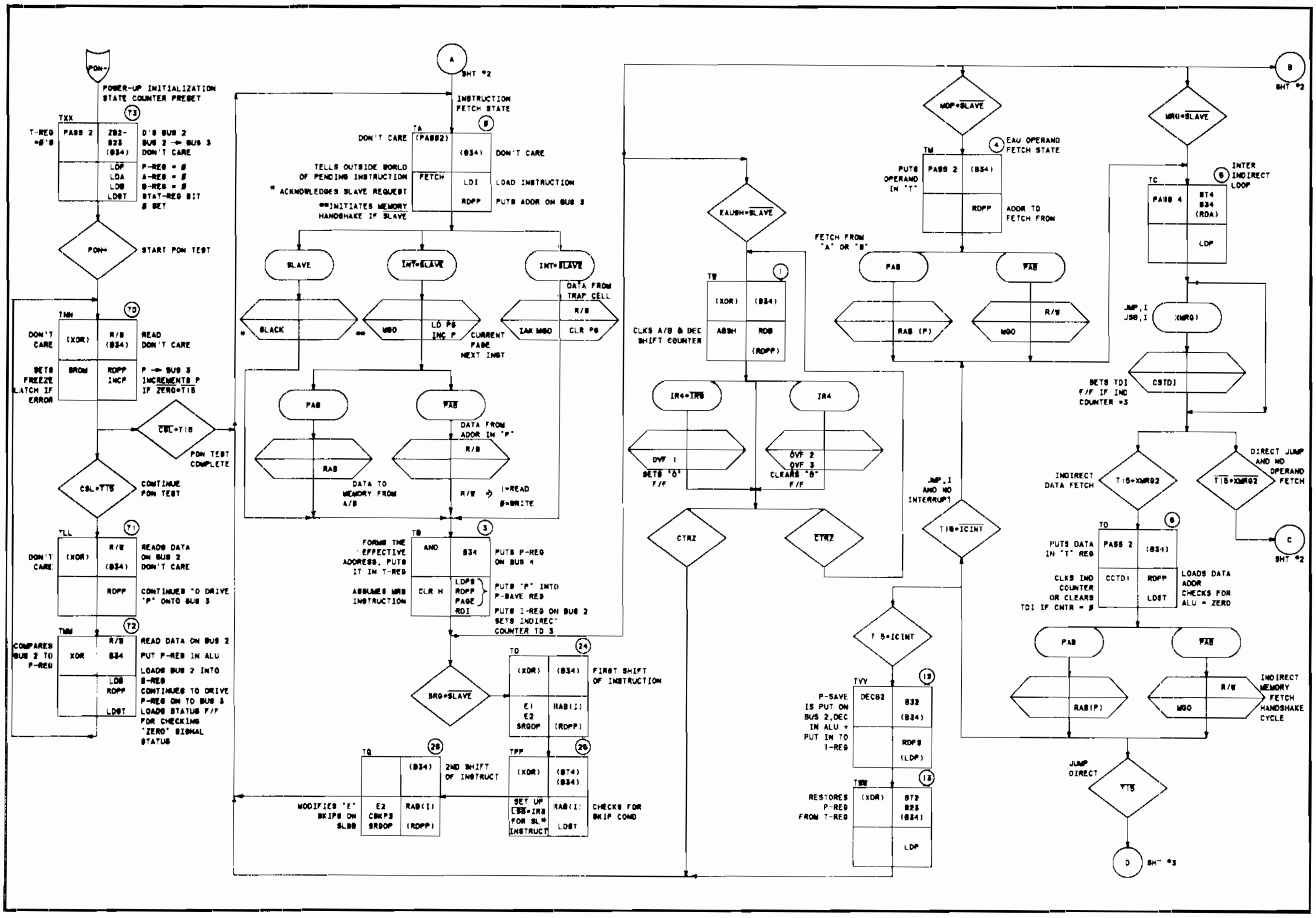


Figure 12. CPU chip state diagram.

An example of the use of that was to discover the cause of the failure of several arithmetic and logical instructions. At first it seemed as if the ALU was performing incorrectly — instead of adding, it appeared to AND the first operand with the two's-complement of the second operand. Closer examination of the bus waveforms, however, revealed that the bus from the latch at the output of the ALU was changing value on both edges of the clock! (See Figure 13.) This problem was, after much pondering by the LSI designers, attributed to coupling between the clock to the ALU output latch and the data in the latch. No logical error — it was a topological one.

2. Environmental Testing:

As part of the design cycle at HP, every new product must undergo what some of our engineers irreverently call "shake and bake" testing — subjecting the unit to temperature, humidity, altitude and vibration extremes which are more stringent than what will appear on the final product specifications. We used this opportunity, also, to verify that at each combination of extremes (high temperature and humidity, low voltages or high voltages on the DC supply) that we had sufficient timing margin to allow operation with a system clock of higher than normal frequency, and to analyze where the circuitry stopped working to see what the critical paths were. Occasionally (should I say usually?) this testing also reveals a device that does not quite operate the way it should. An example: Part of the testing requires operation at elevated temperature for 72 hours continuously. A computer was undergoing this procedure, running diagnostics and on-line functional tests using our Real-Time-Executive (RTE-L) software. This was set up to cycle from initial power-on test, through the diagnostics, to the system on-line tests.

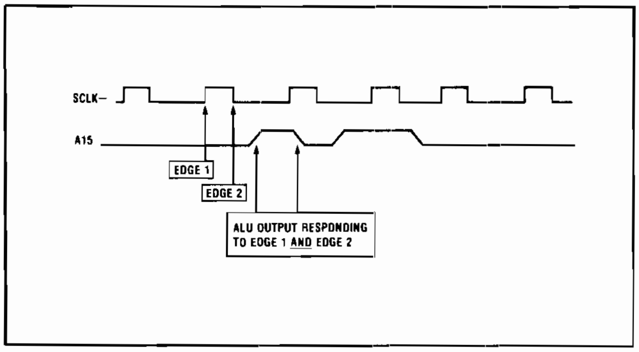


Figure 13. Trace showing incorrect CPU chip operation.

This cycle was to have run repeatedly overnight, but was interrupted sometime during the night by the detection of a "break" condition on the Asynchronous Serial Interface which was connected to the system console. This break condition activated the computers' Virtual Control Panel, the equivalent on other machines of pressing the Halt button.

The logic analyzer was attached to several points in the path from the RS-232 line received from the terminal to the input of the I/O processor chip which requested the break (see Figure 14). Since the terminal was quiescent during the entire test, the logic analyzer was set to trigger on any break indication; the test was restarted. The trace of Figure 15 was the result!

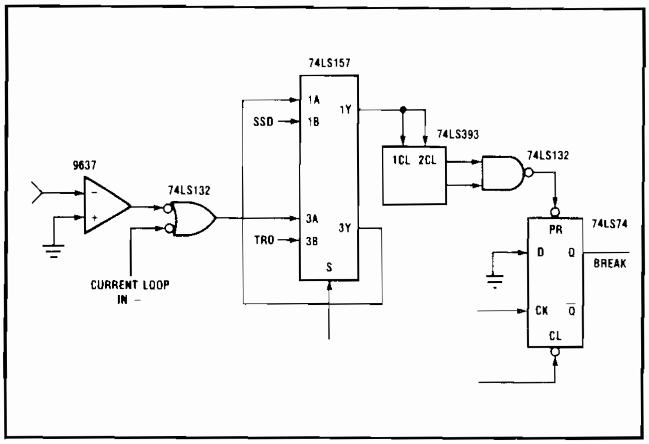


Figure 14. Path of the break signal on the ASI card.

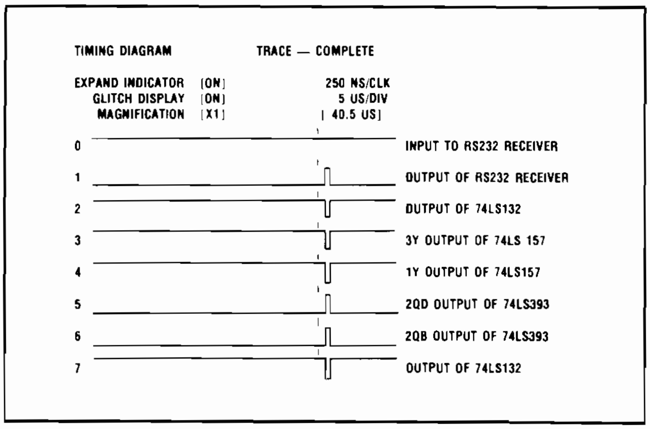


Figure 15. Trace of bad break signal.

The RS-232 receiver had signalled a logic 0 level when its input remained at a logic 1 level! Further investigation showed that noise on the RS-232 line, even though well above the threshold of the input, got coupled through to the output — but this mainly showed up only after the device had been heated to a high temperature. A call to the manufacturer confirmed the problem, and obtained the data code of the revised parts. Caveat Emptor!

3. System Integration:

Now all the individual parts of the computer had been tested individually — packaging, power supply, and logic designs; it was time to integrate them into the final form. And, true to our luck, there was a problem or two.

An example:

The L-Series, like most contemporary computers, uses semiconductor RAM for its main memory, necessitating the use of a battery backup card to sustain the memory during temporary AC power loss.

Some signalling is necessary to both alert the processor that a power failure is impending, and to switch the battery backup card from charge mode to sustain mode. During the interval between the signalling of the impending power failure and the time that the power supplies are no longer valid, the operating system does as orderly a shutdown as possible, then waits for power to be re-asserted. As a precaution on power-up, the software checked to see if a flag had been set indicating that the power-down routine had run to completion. If not, the system is probably in some form of disarray, and needs to be reloaded.

This is exactly the indication we observed when we simulated a power-outage situation — except that there should have been no reason that the power-down routine could fail to finish!

The software engineers called in the logic analyzer and its backplane connection so as to trace instruction flow (the backplane connection of Figure 5). They quickly observed that the power-fail routine did, indeed, finish; the power-up routine saw that the flag was set, but the power-up routine was being entered a *second* time, without an intervening power-down! This was traced back to a transistor being at the low end of its beta specification, providing insufficient gain in the control circuitry. The +5V supply was assuming the waveform shown in Figure 16. The first upswing only took the supply to 1.5V, which was just enough for the power-on signal PON to be asserted twice within a very short time, an infraction of the supply specifications. The specification for that transistor is being tightened.

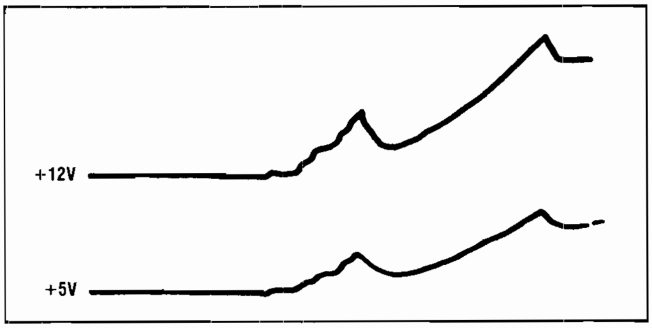


Figure 16. +5V turn-on caused by marginal component.

SUMMARY

This paper has tried to show that today's logic analyzers are powerful tools, fulfilling the needs of a broad range of engineers, from software designers to system integrators. The logic analyzers are used in a variety of modes, to solve:

— asynchronous problems	("break" problem)
— synchronous problems	("LDMAR" problem)
— state flow problems	(CPU chip debug)

This paper tried, through examples, to have lent some credence to the wild-sounding claims of the logic analyzer manufacturers as to how much you can do with a logic analyzer — they are true!

4. The True Costs of Owning a Microprocessor Development System.

HEWLETT PACKARD

John Marshall draws on his board background with HP, from R&D Lab to marketing, as the Promotion Manager for the 64000 Logic Development System. He earned his B.S.E.E. at Texas Tech.

SOFTWARE DEVELOPMENT: WHY USE A DEVELOPMENT SYSTEM?

INTRODUCTION

We are working today in an environment of rapid change. Products are changing, design methods are changing as well as the basic materials that we use. It was only a very few years ago that most design labs were primarily analog orientec and exclusively staffed with hardware design engineers. Few were skilled in software design techniques and there was little incentive to learn them. Computers were used primarily for modeling and computation.

New developments have drastically altered all phases of the electronics industry and fostered whole new segments of the industry as well. The plunge in cost of memory and dramatic increase in processing power has made it essential to consider the use of a microprocessor in new designs. (Figure 1)

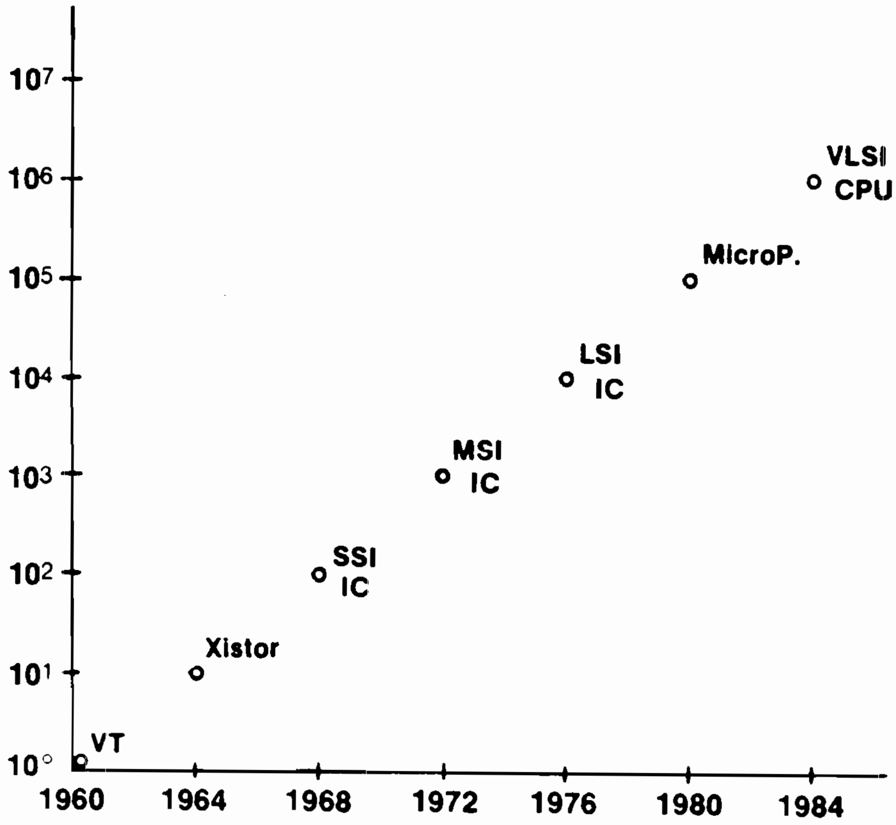


FIGURE I

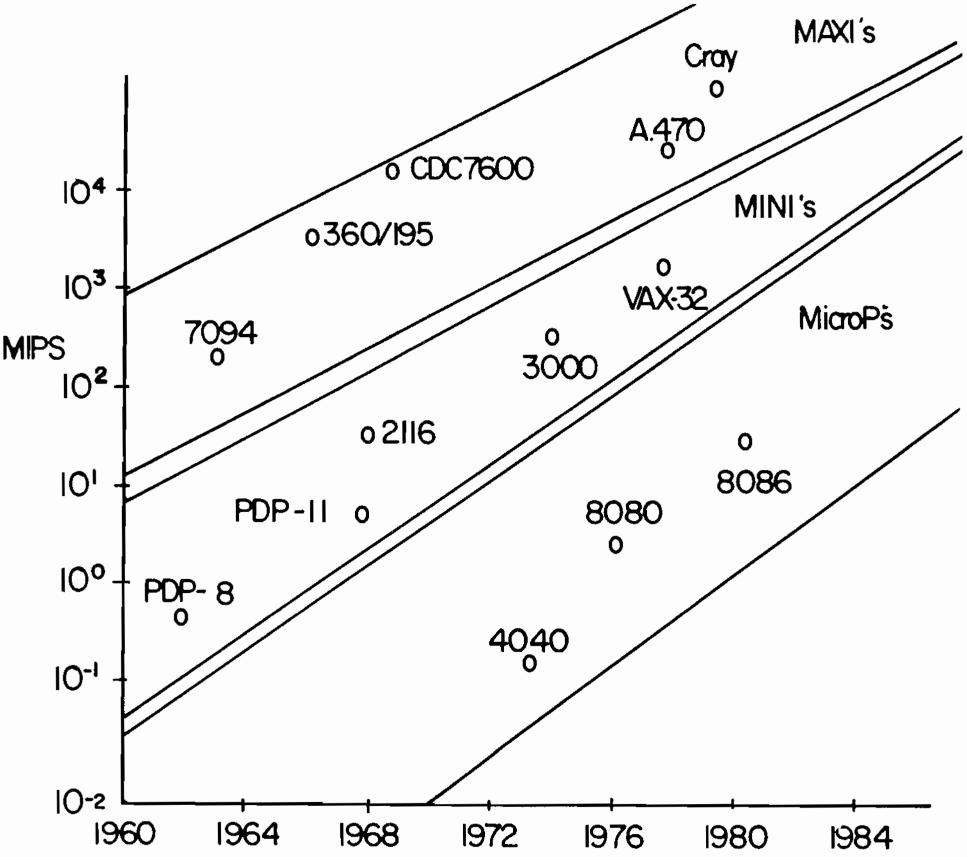


FIGURE I

At Hewlett Packard, for example, we have no new designs that do not include a microprocessor. As a result, the configuration of our design labs has changed markedly. Most of the hardware design engineers are now digital circuit oriented, with skill sets that make them proficient in designing both at the gate level and in systems type work where the functional blocks provided by semiconductor manufacture are integrated into a desired functional unit. This doesn't obviate the need for good analog skills to anticipate and analyze problems such as glitches, coupling, ground loops, etc. In addition to the new breed of hardware designers, a significant number of software types are needed on new project design teams. At HP, software designers comprise about one half of our lab staff.

The nature of the business climate has also changed significantly since the advent of microprocessors. With today's easy availability of inexpensive and powerful semiconductors, almost anyone can design, and at least announce, if not successfully produce, a sophisticated processor-based product. Even sophisticated products are easily copied and perhaps obsoleted in a short period of time. Competition is intense and often the largest market share goes to the producer who introduces a product first.

As a result of these changing parameters, the design process has changed as well. In most present day design projects, two distinct, but closely related, design areas are readily identifiable: the hardware design that includes circuit design, layout, breadboard and test, and software design where program code is written to drive the hardware under design (Figure 2). These two design tasks are usually accomplished simultaneously in order to minimize total design time. Of the two tasks, the software job is probably the least understood and most subject to delays and cost overruns. The hardware job, although far from trivial, is usually well understood and well supported by test and development equipment.

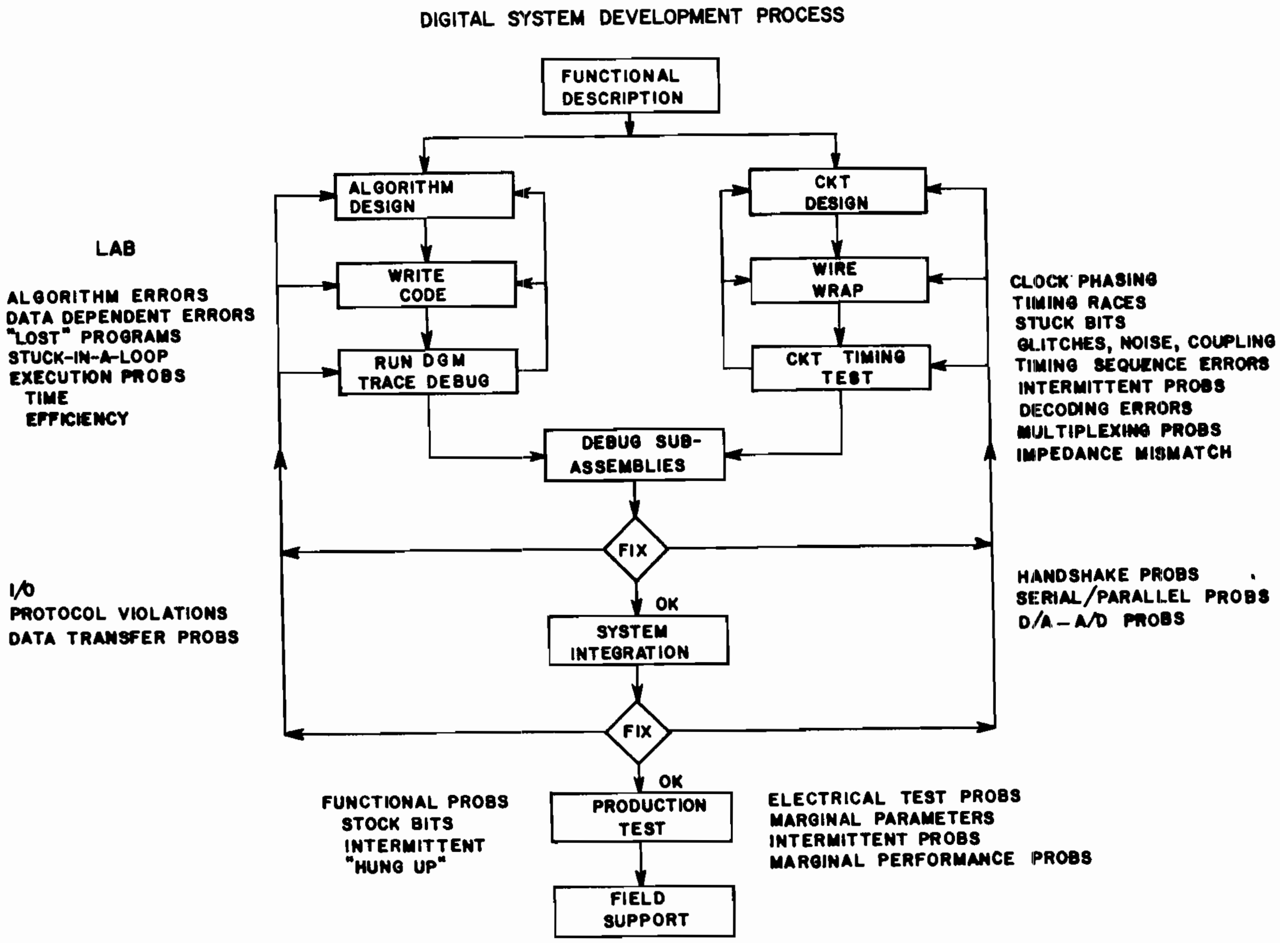


FIGURE 2

IDENTIFYING THE PROBLEMS

Since software design often restricts and limits the overall design goals of a project, it might do well to look at some of the roadblocks that often doom a project to costly delays, overruns and introduction problems.

- Inexperienced staff: The winds of change have blown so fast that engineering staffs are not yet trained in the skills of efficient programming. Also, many nontraditional companies are drawn into the business without having enough time to accumulate adequate experience.
- Improper tools: There are a number of ways in which the software design task may be approached;but,without adequate tools or design aids, the project can quickly get mired down in a mass of poorly documented code.
- Staff turnover: This can be a real problem if the design aids being used do not promote good documentation or if they are difficult to use and thus consume valuable time to train new personnel.
- Low productivity: This may be the result of inexperienced staff, poor planning and management, improper tools or some combination of these factors. The problem is often compounded when, as things get bad, more staff is added which usually makes things worse.
- Down time: No matter how efficient the design aids are, if they are not reliable or cannot be serviced quickly, the design team is idled while the equipment is down.

THE BASIC ELEMENTS

In order to better understand how the previously mentioned problems can be minimized, it is essential that we understand what is needed to complete a software design task and then consider the alternatives available to accomplish the most cost effective solution.

The complete software design task consists of three basic elements: (Figure 3) program development, downloading,and analysis or debugging. There are a number of alternatives available to accomplish the tasks. Doing it all by hand is possible,but not practical beyond a program sizє of 1k byte. Given that the job is not to be done by hand, some program development tools are needed -usually in the form of application programs.

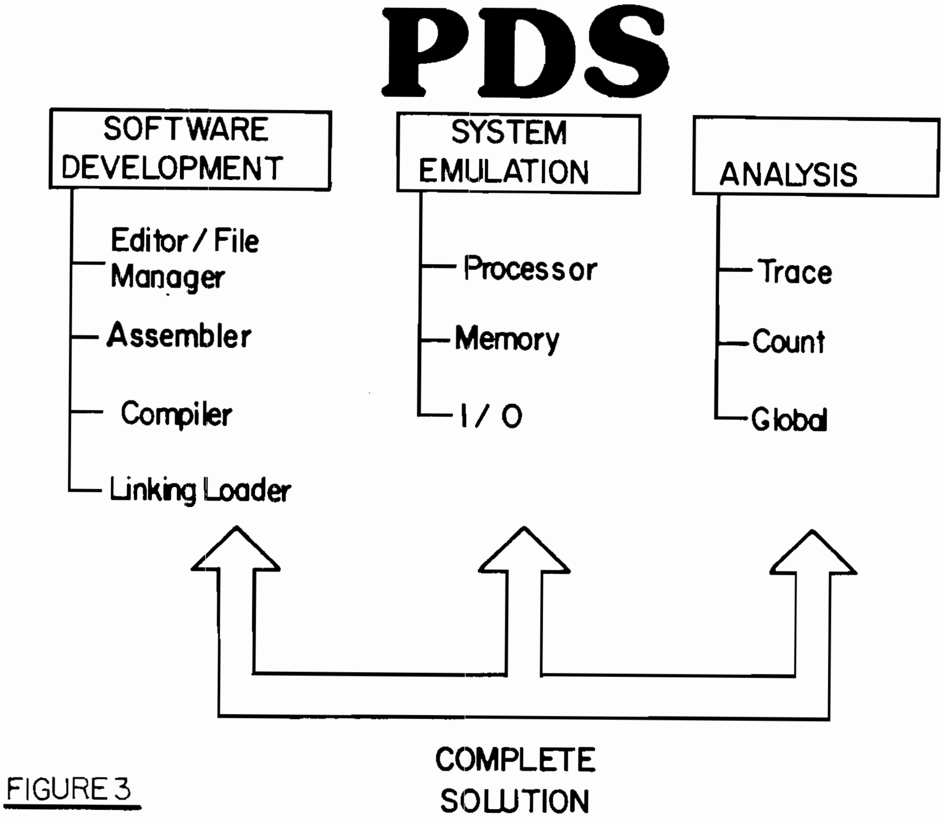


FIGURE 3

Editor: provides the ability to enter, modify and store program modules under development.

Assemblers and Compilers: enables the user to write in more convenient and understandable language than the basic executing machine language.

Linkers: allows the user to write programs in modules without regard to absolute addresses. These relocatable modules are easier to handle and usually identified by function. They are then linked together and assigned absolute addresses at link time.

These application modules can be found on timeshare systems, mainframe computers, and minicomputers, (Figure 4) all of which provide the necessary console for user interface, mass storage, and I/O. The question then becomes one of what to do with the edited, assembled, and linked code in the machine--the download problem. It can be executed and tested functionally on a simulator, which still leaves open to question the I/O related or time dependent problems. The code can be downloaded through a PROM programmer into a PROM and inserted into the target system and tested using logic analyzers--a very effective solution in many cases. The difficulty is that there is no link between the target system and the software development facility. Correction of errors can be a

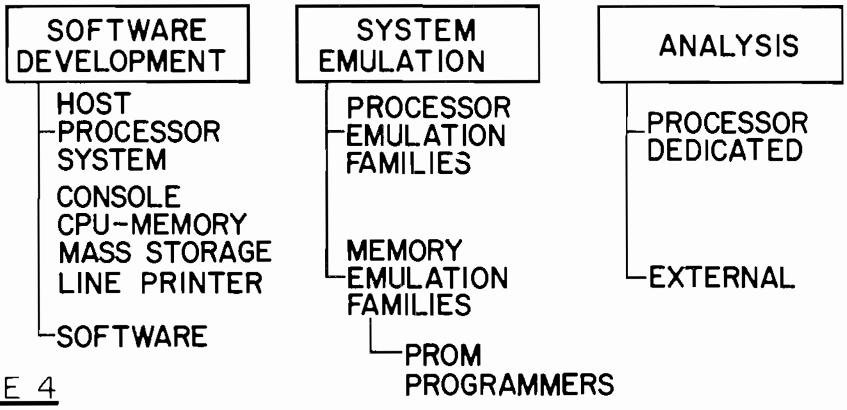


FIGURE 4

were,for the most part,products of semiconductor houses and were intended to support the manufacturer's silicon. This forced the user to make a capital investment with a very narrow application and reduced his flexibility on new design starts.

As a result of these limitations, many design groups who wanted to apply the contributions of development systems still could not justify their use. The needs and desires of the industry were articulated by Jerome W. King and Daniel F. Ferriola of General Electric, Bridgeport, Connecticut, in a paper presented at Electro 78. (Figure 5) In their paper they summarized what was needed:

FUTURE REQUIREMENTS OF AN MDF

1. Support chip sets from various manufacturers
2. Support multiple users
3. More extensive file systems to permit better software engineering features
4. High speed rigid disc with matching system software
5. More extensive emulator facilities
6. Standard interface to all peripherals
7. More comprehensive utility software to increase designer efficiency

A NEW SOLUTION

These needs have for the most part, been addressed by a new introduction from Hewlett Packard. The HP system is based on a hard disc with more than 10 megabytes of storage. Full advantage is taken of this fast mass storage with the implementation of a high performance operating system and file manager. A very friendly user interface is maintained using "soft key directed syntax" where the available commands are displayed on the CRT and selected via keys whose definition changes as the command structure progresses. New people can be trained quickly since there is no exhaustive syntax or complex time consuming process and documentation may suffer. It is this link between the target system and the source program that represents the major contribution of microprocessor development systems through emulation. With properly designed emulation, the program code may be executed in real time in the target system. As errors are uncovered, the user can quickly access the source code, modify it with the editor, then assemble, link, and reload and begin execution with a simple command file. With the HP systems, memory modifications may even be made on the fly, enabling ideas to be tried immediately. As program modules are perfected, they can be downloaded into PROM and mapped over to the user system. In summary, the microprocessor development system provides two major advantages: downloading and microprocessor run controls. This tight coupling of the software writing and debugging, if properly implemented, can provide substantial productivity and cost benefits to the user.

DEVELOPMENT SYSTEM LIMITATIONS

It would seem then, that a development system might be the ultimate answer for today's software design problems, but, until recently, their performance was such that many elected to stay with their computers.

In order to keep costs down, they were usually floppy disc based, which severely limited the quality of operating systems and overall performance. Execution of application programs tended to be very slow. Since disc space was limited, each user kept his files on individual diskettes leading to fragmentation of the design effort. Since operating systems were primative, the system tended to be hard to use and required a significant investment in time to teach newcomers the syntax and procedures--a severe limitation on design team productivity. Another limitation of the floppy based system was that they don't support one development station very well, and certainly not multiple stations--again, because of speed and capacity. This meant that as the design team grew, new systems had to be added which further fragmented the design effort. Another problem was created by the fact that developments procedures to learn. The development station executes application modules in its own 64k host memory but also uses the disc as a virtual memory. Multiple stations can share the disc--creating a tightly coupled design team environment where user files and libraries can be shared and multiple tasks performed simultaneously. (Figure 6) The user wanting to download and emulate is not held back by the one needing to edit his source files. Sharing of the disc by multiple stations reduces the cost per station and thus the overall system to less than the cost of an equivalent number of floppy-based systems which provide far less performance. (Figure 7)

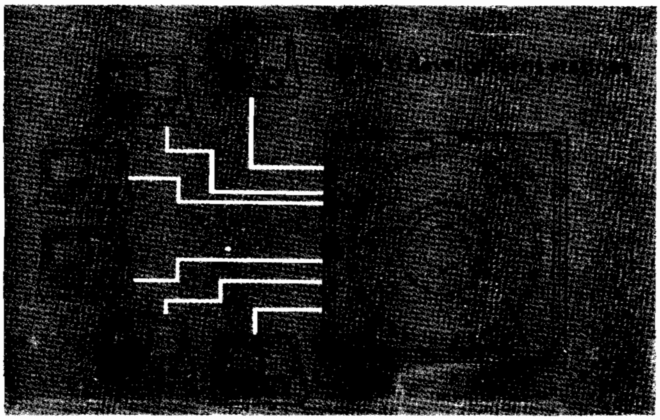


FIGURE 6

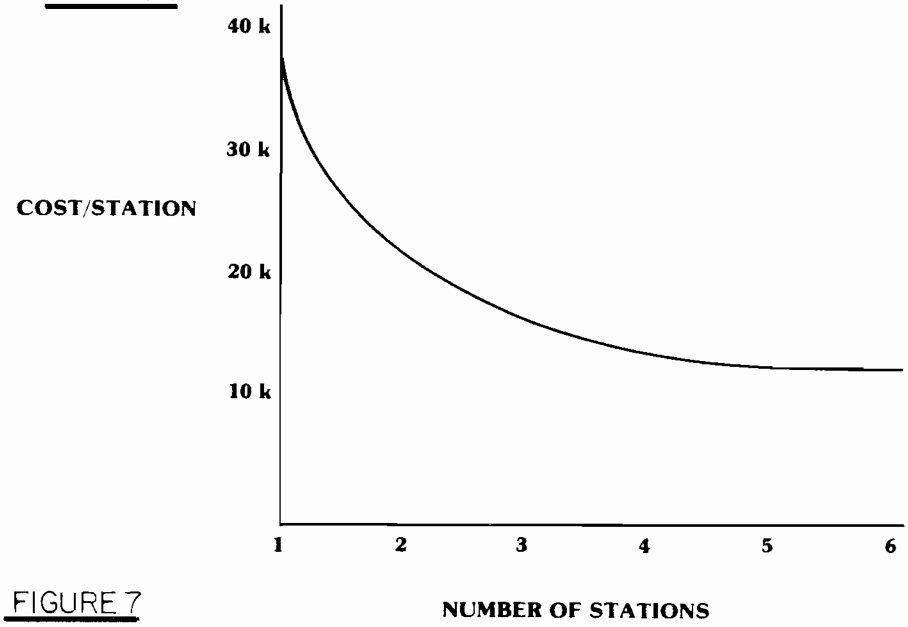


FIGURE 7

The development station architecture is such that the emulator subsystem is independent of the host processor and memory,eliminating bus and memory contention problems. (Figure 8) The architecture is also processor bus width and speed-independent,permitting emulators for all types of processors to be designed and installed.

LDS DEVELOPMENT STATION ARCHITECTURE AND BUSES

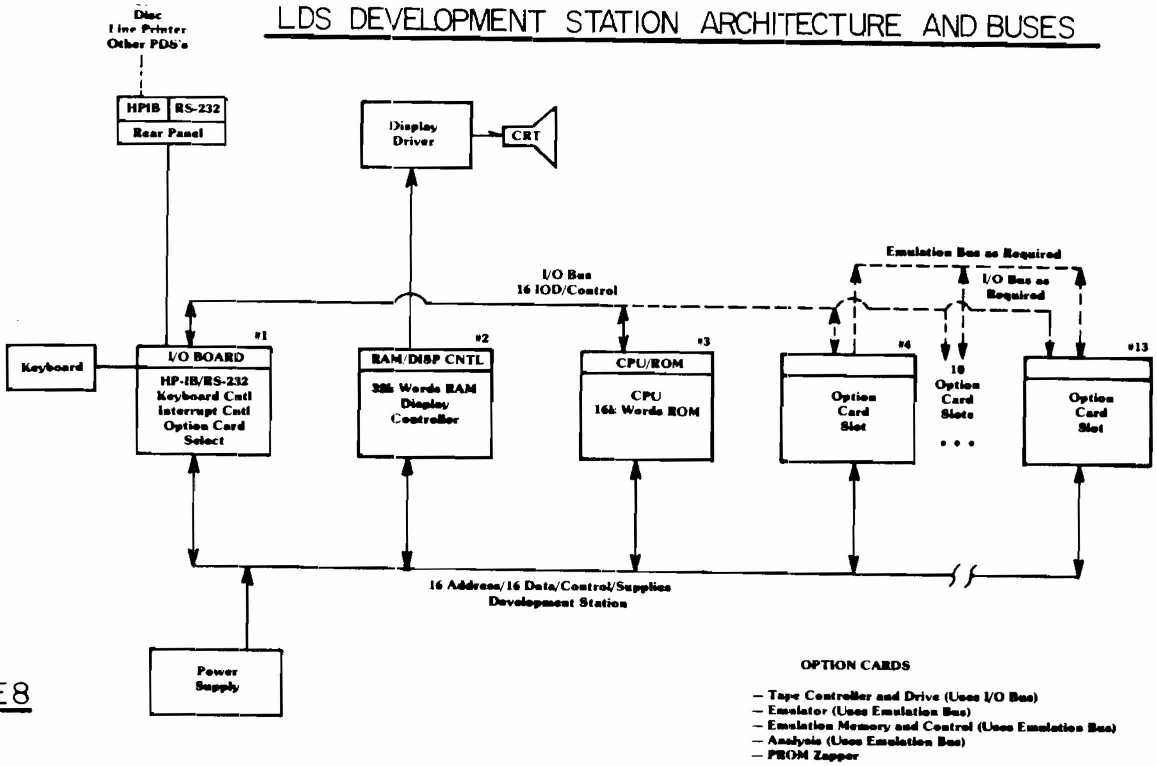


FIGURE 8

Emulation may be accomplished in real time and transparently due to the independent emulation memory implemented with fast static RAM and a special background memory where emulation functions are executed. (Figure 9) Use of background memory will permit such activites as examination and modification of memory while the target system is running.

Application modules are also greatly enhanced through the use of a hard disc. The editor is fast, easy, and comprehensive; and the assembler runs at 4000 lpm on any size program.

64000 EMULATION ARCHITECTURE AND BUS

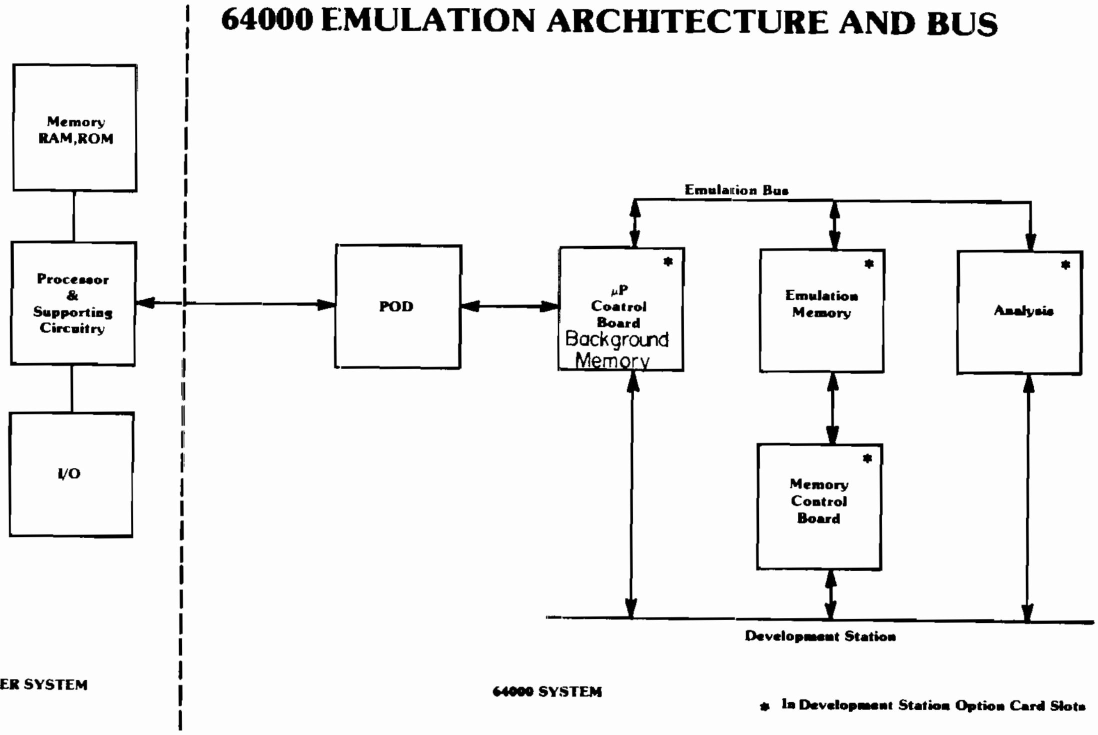


FIGURE 9

SUMMARY

A system such as this overcomes many of the objections formerly raised about development systems. It offers the necessary high performance peripherals to provide a sophisticated operating system,and application programs to simplify and enhance the software design effort. These features, plus enhanced emulation and the encouragement of teamwork through a multi-station network,all work to increase design team productivity and reduce development time.

5. Advanced Microprocessor Emulation Techniques.

HEWLETT PACKARD

Chris Jones prepared software for the new 1610B Logic State Analyzer and was part of the team that developed the operating system and monitor for the 64000 Logic Development System. Chris has a B.S.E.E. from Rice.

ADVANCED MICROPROCESSOR EMULATION TECHNIQUES

by Chris Jones

INTRODUCTION

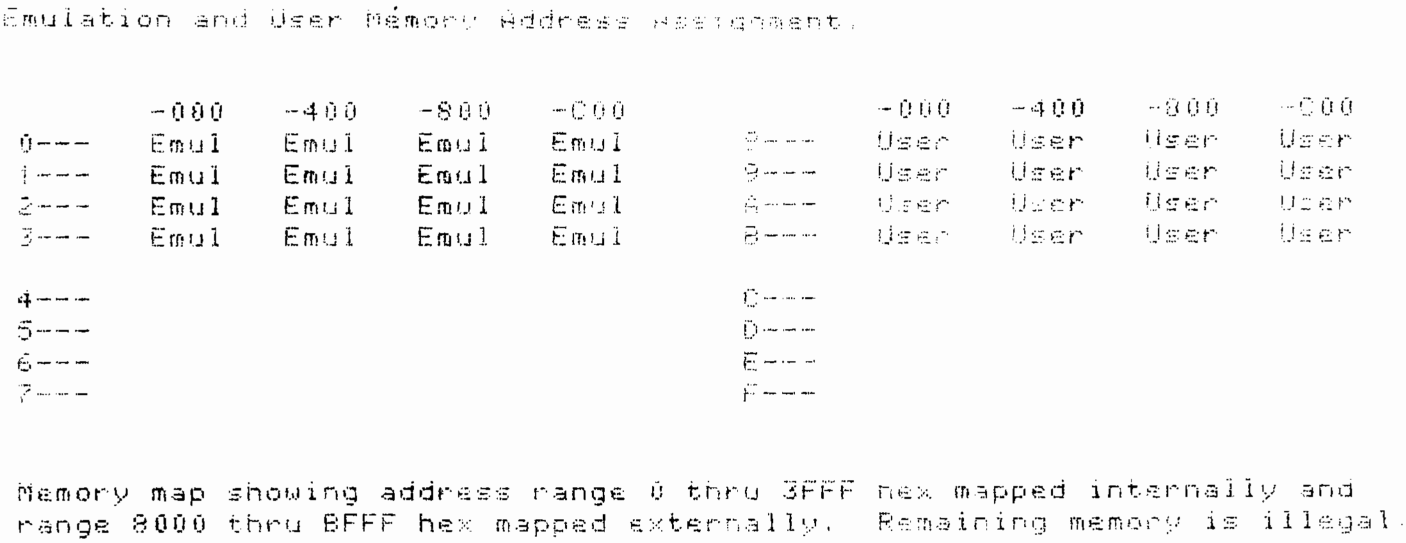
Taken from the standpoint of microprocessor based system developer's needs rather than known solutions, the current techniques of in ciruit emulation and real time analysis will be explored as they meet and fail to meet developer's growing demands. Support requirements are addressed in terms of the separate and overlapping requests of the system software and hardware developers as a target system is developed. Implementations designed to meet these current needs are discussed and, lastly, a look is taken at what characteristics of microprocessor development system of today will be important for their continued viability in the future.

SOFTWARE DEVELOPERS' EMULATION AND ANALYSIS NEEDS

Host and Target System Memory Referencing

Because the software design cycle runs in parallel with that of the hardware, it typically becomes necessary for software to develop in two phases. One is very basic "test software" developed soley to aid the hardware designer in checking out prototype hardware. The other is designed in a top-down manner and carefully thought out in an overall system framework before coding of actual segments begins. This second phase software is the system software and is the product of software engineers. Usually, after the system has been software architected the specific coding tasks will be assigned individuals and the process of creating and debugging software begins. As modules of the final system are developed, they must be executed and debugged on the processor with which the software will eventually be used. Many of these software modules will have no interaction with the system hardware at all and will serve to provide system utilities or user interfaces to the system. Such modules require emulation capabilities which allow execution of code prior to, or independent of, any actual system hardware being present. The phase one, or test, software will provide intermediate software support to hardware developers and will eventually be upgraded to provide the interface between the system software and hardware (i.e., the hardware drivers). This software must be supported by emulation techniques which provide executing and debugging of software on the actual target system hardware.

The solution of these two distinct needs is the technique known as "memory mapping". At its best, the user selects a mapping from the emulated processor's address space to physical memory residing either within the emulation system or within the user's system. Once defined this mapping should become transparent to the user such that references to a specific address access memory directly in either area.



Emulation and User Memory Address Assignment.

	-000	-400	-800	-C00
0---	Emul	Emul	Emul	Emul
1---	Emul	Emul	Emul	Emul
2---	Emul	Emul	Emul	Emul
3---	Emul	Emul	Emul	Emul
4---				
5---				
6---				
7---				

	-000	-400	-800	-C00
8---	User	User	User	User
9---	User	User	User	User
A---	User	User	User	User
B---	User	User	User	User
C---				
D---				
E---				
F---				

Memory map showing address range 0 thru 3FFF hex mapped internally and range 8000 thru BFFF hex mapped externally. Remaining memory is illegal.

Input and Output from Emulation

In terms of the two varieties of software defined above, the system software and the hardware drivers, the system software is almost surely the bulk of the effort in modern microprocessor based systems. For it to be adequately supported in development, the developer will need to have available a mechanism for providing output from his code prior to integration with system hardware. This output capability actually falls into two categories: output from the code which results in a change to data in the system (which may be observed by emulation techniques) and output which is explicitly coded into the software (such as that provided by a WRITE command in a high level language). This latter capability is far more a convenience than a necessity to the software developer as ultimately the machine's output will be that of the final system's hardware. On the other hand, with this capability, one more software and hardware effort can be carried out in parallel since hardware driver software may temporarily be replaced with emulation output software to allow the system to develop further before hardware is completely developed.

The minimal solution to this need is to provide the user system a means of output to the emulation system's display. Such output is immediately observable by the user and thus eminently available for debug. The complete solution would also allow the user a means of output to the system's printer and mass storage medium for permanent storage of user's output.

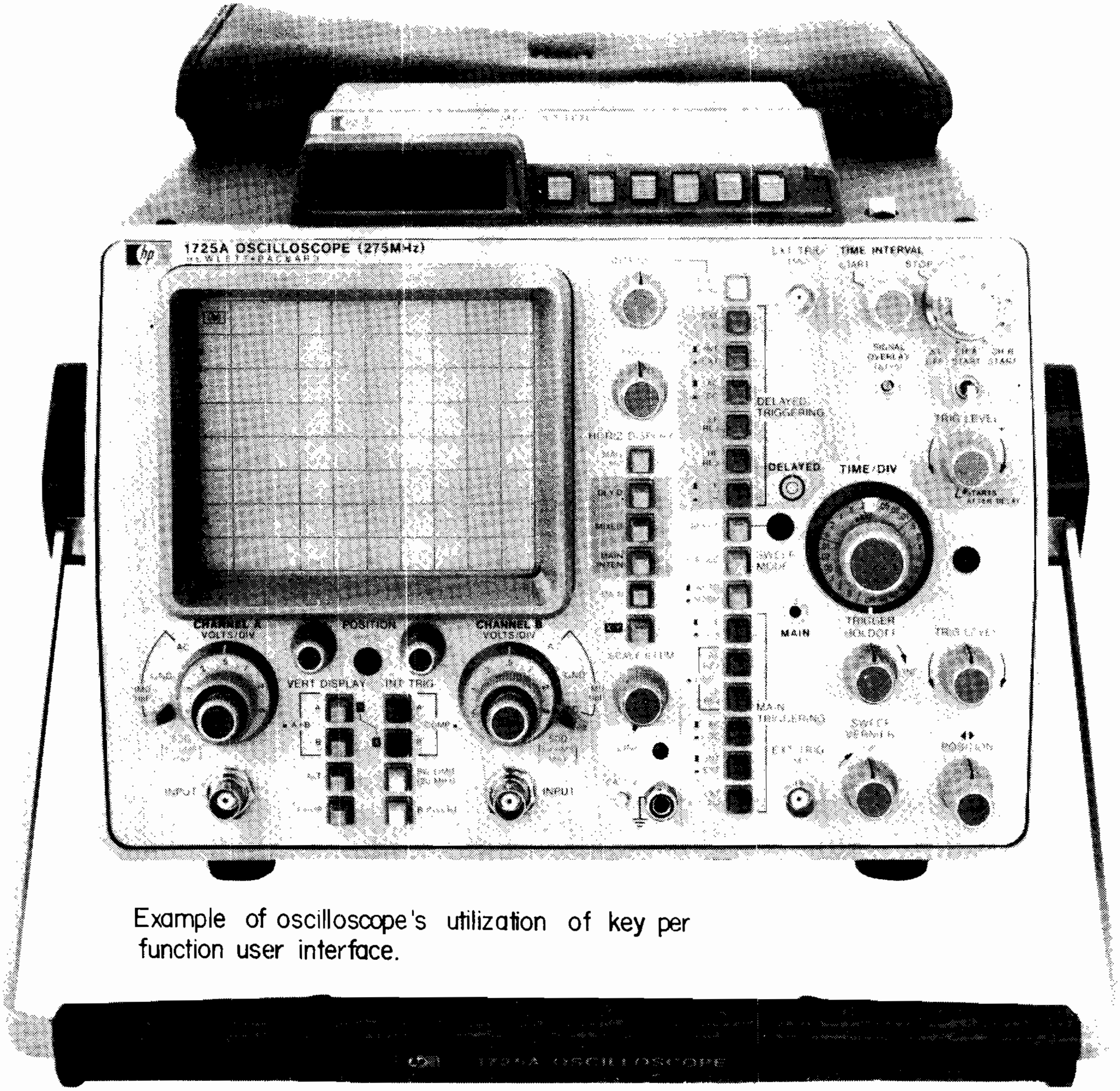
Carrying the emulation of system output one step farther allows consideration of emulation providing complete system I/O. Again this increases the possibility of more system software development before completion of hardware. Now the system might be able to fully implement the user interface prior to hardware being complete. The implementation of "simulated I/O" is valuable to the user but certainly with the caveat that no I/O simulation will exactly duplicate the user's hardware and thus will require specialized software to be rewritten prior to use with actual I/O devices. This implies that the communication should be as simple as possible so as not to require a large software effort to provide an interim solution.

Easy User Interface

Of course the software developer is very much in need of sophisticated logic analysis capabilities in debugging all code. Much of the logic analysis support necessary for software debug is the same as that which will be required in order for the hardware designers to adequately debug their design. These analysis capabilities will be referred to as real analysis since current technology allows them to be performed in real time and since hardware development requires analysis up-to-speed. Software developers require an interface which is easy to use. For them this implies that the emulation system must speak their language which means several things. First the developer must be able to reference memory locations by the names given them in the source programs. That is to say, using the symbol FIRST-BLOCK which is assigned by a program to location 1000 hex must be understood by emulation in the same manner as entering the hex constant. The software developer may also prefer to refer to a line of interest by the line number in the source listing rather than by a memory address. Both are examples of the need that emulation speak the developer's language in its interactions.

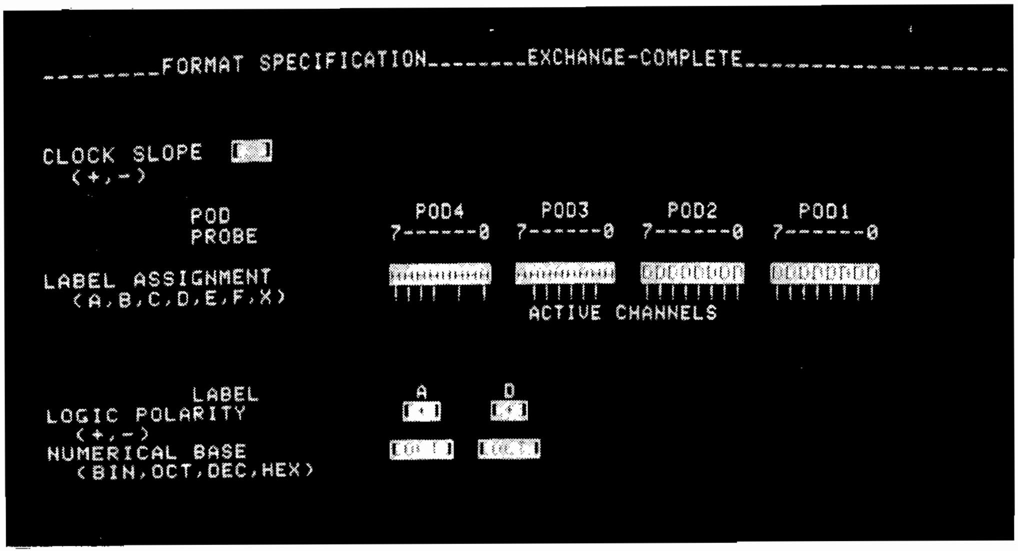
This deceptive need for an easy-to-use interface is encountered throughout computer systems and has neither an accepted nor obvious solution. Until voice actuated systems come of age, the user will always be faced with learning how a system wants user inputs. So the first concern in user interface solutions is ease in learning. The second concern is for the user who has learned the system and now desires that it be convenient for his purposes. To date interfaces have taken three distinct forms, namely: 1) Key per function 2) Menu 3) Command language. The key per function interface is a classical instrumentation approach to the problem. Here the user selects keys which have dedicated functions in the system. The oscilloscope is a good example of this approach. It is arguable that the new user must learn the location of each function but certainly labelling can decrease this time, and convenience, once learned, is very great. The limitations of key per function are associated with physical constraints as the number of functions becomes large. Also a design criterion of flexibility is severely limited since expanded features require physical redesign. The menu interface approach has several of the advantages of key per function without the extent of constraints. The learning ease is still accomplished by displaying to the user the choices available to him and, similarly, the experienced user is not hampered by any lack of convenience in instructing the machine. Here the user may be assisted by reasonable default menu choices included in the system. Ideally the user will be able to use the system in the "default mode" for many of the more frequent measurements employed. The menu may share a drawback in common with key per function interfaces. That is, as choices become large in number, the menu may become cluttered with more information than the user desires. This has been combatted with more menus and, as features expand, this software interface approach allows easily implemented changes.

More common in the computer interface world is the idea of command languages to direct the machine. Typically this approach involves cost to the beginning user, who must first learn the machine's language, with the benefit of ease of use and great flexibility to the experienced user. Here the interface has lost one of the



Example of oscilloscope's utilization of key per function user interface.

valuable attributes of the other two approaches, the ability to show the user the functions available on the system. The user must know the words, or abbreviations, understood by the system to evoke responses. Once again, by utilizing convenient defaults in the language, the system may make communication more easy for the human involved.



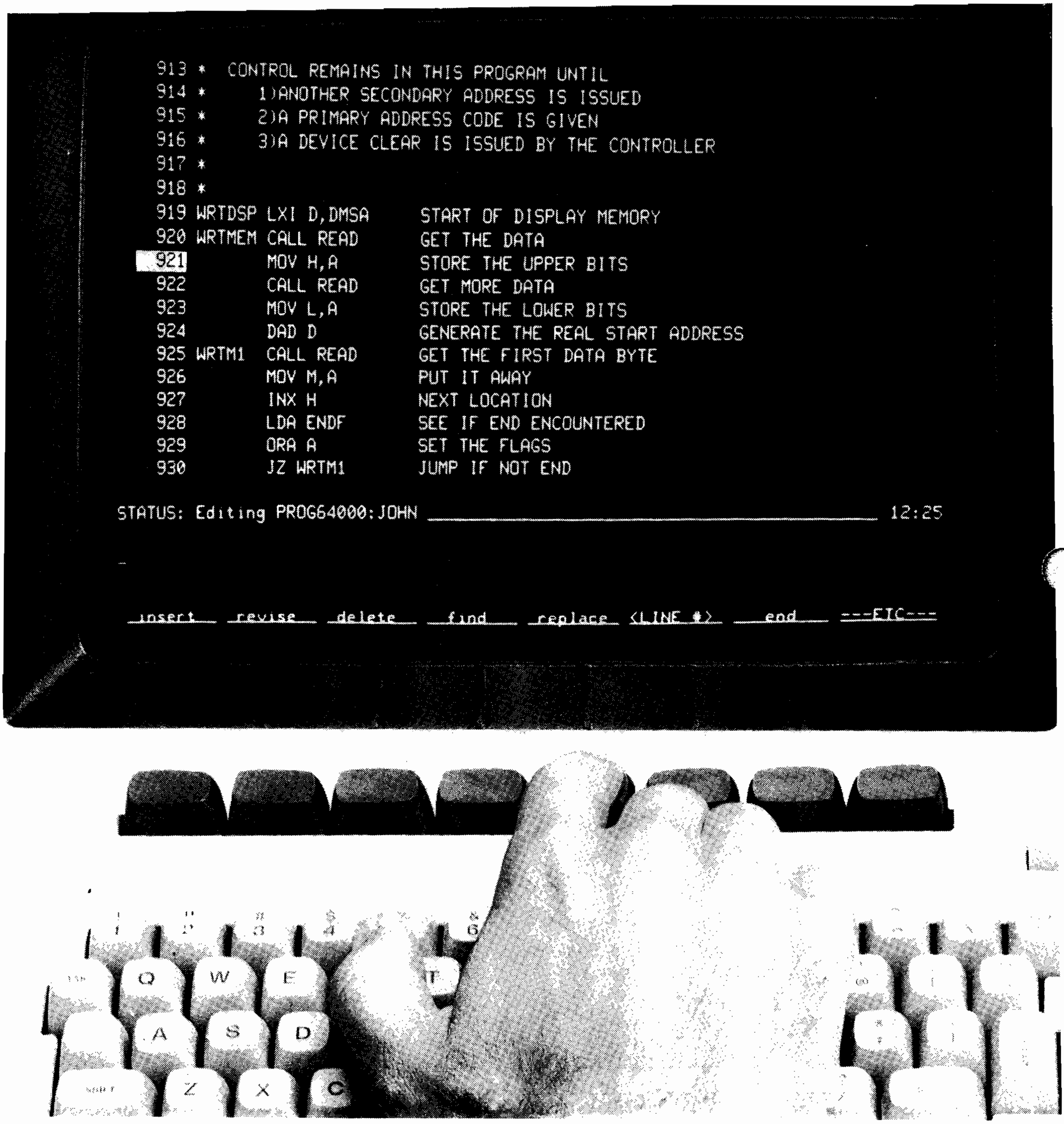
```
_______FORMAT SPECIFICATION_______EXCHANGE-COMPLETE_________________

CLOCK SLOPE
  (+,-)
            POD              POD4      POD3      POD2      POD1
            PROBE          7------0  7------0  7------0  7------0

LABEL ASSIGNMENT
  (A,B,C,D,E,F,X)                    ACTIVE CHANNELS

            LABEL            A         D
LOGIC POLARITY
  (+,-)
NUMERICAL BASE
  (BIN,OCT,DEC,HEX)
```

Example of logic analyzer using menu user interface

```
(a)          RU,ED,,99,1                  (11)

             F FILE,*LP,FILEB             (16)

             J FILE(2/20)                 (12)

      -----------------------------------------------

(b)          edit FILE                    (5)

             assemble FILE                (5)

             merge FILE from 2 thru 20    (10)
```

(a) Classical command language instruction examples
(b) Softkey driven command examples
Numbers in parentheses indicate number of keystrokes required.

Recently Hewlett-Packard has developed a hybrid interface incorporating aspects of each of the above interfacing techniques. This improved interface is fundamentally a command language approach enhanced to provide the user with information describing both the capabilities of the system and the words to be used in instructing it. At the heart of this interface are "soft keys". Used for years in smart terminal products, a soft key is a keyboard key which has a redefinable meaning at different points in the system's operation. In particular, the software of the system may change the key's meaning, hence "soft keys". These soft keys are used to display to the user all command word choices available at any one time in operation. Now a decided disadvantage to command language interfaces has been overcome. The soft keys may teach the language of the system to the new user. Also, because the keys are software controlled, they may print whole English words at a single keystroke. This makes possible commands which make better sense to the user ("assemble FILEA" rather than "RU,ASMB(FILEA)" for example). In fact, the soft keys have been further used to direct the user as to the syntax of each command. They provide word choices to the user as well as prompt for data to be entered and even describe its format.



```
*  CONTROL REMAINS IN THIS PROGRAM UNTIL
*      1)ANOTHER SECONDARY ADDRESS IS ISSUED
*      2)A PRIMARY ADDRESS CODE IS GIVEN
*      3)A DEVICE CLEAR IS ISSUED BY THE CONTROLLER
*
*
WRTDSP LXI D,DMSA     START OF DISPLAY MEMORY
WRTMEM CALL READ      GET THE DATA
       MOV H,A        STORE THE UPPER BITS
       CALL READ      GET MORE DATA
       MOV L,A        STORE THE LOWER BITS
       DAD D          GENERATE THE REAL START ADDRESS
WRTM1  CALL READ      GET THE FIRST DATA BYTE
       MOV M,A        PUT IT AWAY
       INX H          NEXT LOCATION
       LDA ENDF       SEE IF END ENCOUNTERED
       ORA A          SET THE FLAGS
       JZ WRTM1       JUMP IF NOT END

STATUS: Editing PROG64000:JOHN ______________________________ 12:25

 insert   revise   delete   find   replace  <LINE #>   end   ---ETC---
```

Softkeys are labeled on screen to reflect command word choices

User interfaces must be judged according to some standard. Such a standard may be described by several criterion which should be listed in order of priority, as trade-offs exist between them. A good standard for evaluating a system might be:

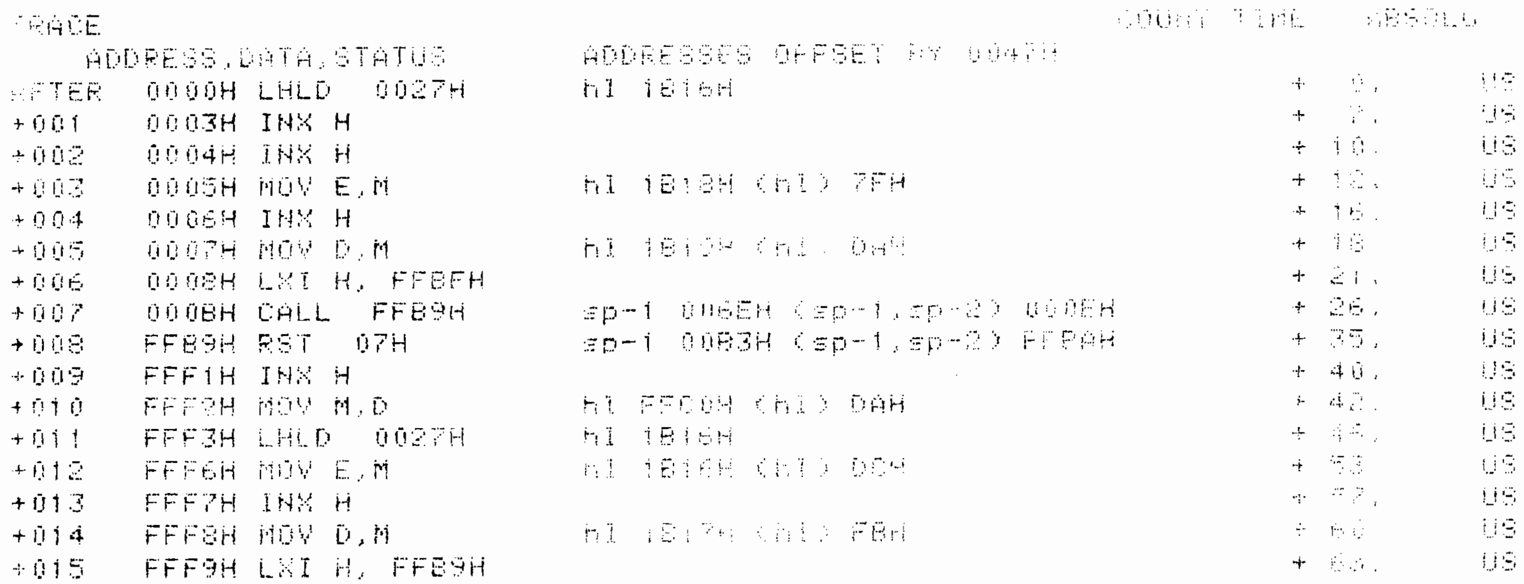
1) System accuracy, "what you see is what you get"
2) Ease in performing frequent operations
3) Convenient for experienced user
4) Easy for first time user
5) Consistency of system operation
6) The easier the function, the easier it is to perform
7) Meaningful error recovery (words not numbers)

Users will have their own additions, ordering, and ideas regarding their standards for interfaces. Each is as valid as the next and should be applied to systems uniformly in comparing them.

Real Analysis Capability

Perhaps it goes without saying that the user must be able to direct analysis to save states (i.e., address, data, and status information) which are of interest. The user must be able to qualify the information stored as to states of interest in terms of their addresses, data content and status of the processor. For example, gather all of the states in the address range 100 - 500 hex of data value nonzero and status of write. Further the output of the analysis must again be in the user's language. This involves its being "disassembled" at a minimum. In the case that the user coded in a high-level language, the output should relate the user back to his high-level code. As well as being in words which the user understands, the analysis output should "look like" the compilation/assembly listing. This implies, in the instance of relocatable code, that the addresses be offset by a constant to allow them to appear as in the unrelocated source listing.

Many implementations attempt to solve these needs. Some significant contributions in this area include: the ability to refer to memory locations by their symbolic name or by source text line number, offsetting address displays to correspond to unrelocated compiler/assembly listings, providing mnemonic as well as absolute analysis displays. All are attempts to produce output more closely related to the software developer's way of understanding data.



```
TRACE                                                                    COUNT TIME   ABSOLU
    ADDRESS,DATA,STATUS          ADDRESSES OFFSET BY 0047H
AFTER   0000H LHLD   0027H       hl 1816H                                   +   0,      US
+001    0003H INX H                                                         +   7,      US
+002    0004H INX H                                                         +  10.      US
+003    0005H MOV E,M            hl 1818H (hl) 7FH                          +  12.      US
+004    0006H INX H                                                         +  16.      US
+005    0007H MOV D,M            hl 1819H (hl) 0AH                          +  18       US
+006    0008H LXI H, FFBFH                                                  +  21.      US
+007    000BH CALL   FFB9H       sp-1 006EH (sp-1,sp-2) 000EH               +  26.      US
+008    FFB9H RST   07H          sp-1 00B3H (sp-1,sp-2) FFBAH               +  35.      US
+009    FFF1H INX H                                                         +  40.      US
+010    FFF2H MOV M,D            hl FFC0H (hl) DAH                          +  42.      US
+011    FFF3H LHLD   0027H       hl 1816H                                   +  44.      US
+012    FFF6H MOV E,M            hl 1816H (hl) DCH                          +  53       US
+013    FFF7H INX H                                                         +  57.      US
+014    FFF8H MOV D,M            hl 1817H (hl) F8H                          +  60       US
+015    FFF9H LXI H, FFB9H                                                  +  63.      US
```

Real Analysis Display formatted for software developer.
State information is mnemonically disassembled and
addresses have been offset for correspondence to assembly source listing.

Real analysis for the software user also needs to flag unusual occurrences, preferably optionally, which may appear in executing code. Examples of such occurances include showing illegal opcodes when they occur, breaking the processor's execution, and listing states leading up to it. The same support is needed for illegal memory references, such as writes to ROM, or vectoring to undefined memory spaces. In this same category comes support for observing states leading up to catastrophic circumstances, such as a slow processor clock.

Execution Overview and Performance Monitoring

Another function which emulation should provide the user is some form of overview of program execution. The software developer must go through an extensive process in debugging which involves observation of unexpected results, hypothesis as to the cause, and testing to determine the accuracy of the hypothesis. The logic analyzer is invaluable in the last phase and the emulation capabilities described above go a long ways toward aiding the observation of unexpected results; however, the user is left much to his own insight in formulating a hypothesis as to cause of failure. An overview of program execution may provide the necessary information to assist in this area. For example, the knowledge that an unexpected code segment is being executed may lead to significantly faster debug and can be quickly observed by an overview of program counter values.

Implementations of overview information are wide ranging, from maps of memory address space showing references to maps showing states versus time, processor register data overview displays, and memory data at given locations displayed over time. All of these functions have then been shown on a sampled or continuous real time basis and provide different feedback to the user.

A last real analysis need of software developers, that is largely unaddressed currently, is that of providing functions appropriate to performance monitoring once the system is regarded as complete. Here the user needs both real-time information regarding the execution time of code segments as well as sophisticated post-execution time data processing to provide in a meaningful format the performance monitoring information.

Code Patching and Processor Control

Once having analyzed programs under execution, the software developer needs write access to processor memory allowing patching of invalid code as well as modifying data values. This is best when implemented with symbolic referencing and use of assembler mnemonics when patching instructions. Additional desirable capabilities include saving modified copies of memory for later use or further debug.

In the area of processor control, users need to be able to run, single step, and stop the processor under test. Once the processor is halted a variety of new information is available which cannot be obtained otherwise. Namely, once halted, a processor's internal register values may be included in the state information. Since such values only appear on buses internal to the processor they cannot adequately be emulated in real time by discrete logic without drastically effecting processor performance. Thus, such information is only available in non-real time (i.e., when the processor execution is stopped). To the software user, the ability to analyze system performance with this added non-real time information may be more important than executing the code at full speed. When code is not working at all, it is more critical to be able to find out why than to restrict its operation to real-time. Then, for software a new category, non-real analysis,becomes desirable. Such measurements are a logical extension of real analysis to include non-real state information. The interface is best made to be the same as that for real analysis. Once made available, the user may now specify a non-real event (such as the value of register A becoming 7) as a trigger to allow information to be stored. The user also needs to be able to update registers and to observe them in an overview fashion, even at the expense of real-time execution.

HARDWARE DEVELOPERS' EMULATION AND ANALYSIS NEEDS

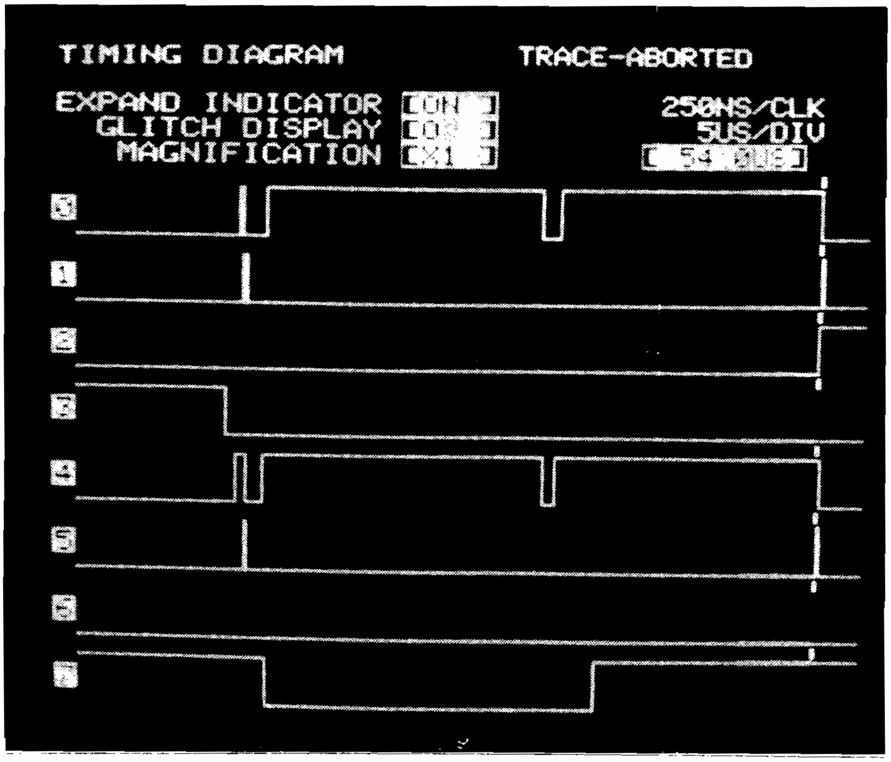
In Circuit Emulation

Whereas software development may not necessarily need the interface to external hardware to begin development, this is absolutely necessary to hardware engineers. For them, the capability to execute the phase one, or test, software described above on their hardware is mandatory. Additionally, two cases exist in this instance. In the first case, the memory destined to be ROM or RAM in the target system may not be completed, or ready for test, at the time that another portion of the hardware, say the processor board, is to be checked out with test software. Here the capability of executing, with the target processor code which resides in memory outside the user's system is required. In the second case in which target memory is now available, the emulation system must provide emulation support for memory inside the user's system.

This need of hardware and software designers has been answered by emulation systems which provide means for selectively mapping the address space of the emulated processor to memory physically internal or external to the emulation system. It is clearly desirable, once this mapping has been made, to have processor execution occur in either physical memory in the same manner and to make it unnecessary that the user be concerned with which is being used. This transparency of user versus system memory is constrained by the inevitable fact that the type of memory in the two systems may have different characteristics.

Timing Analysis of External Data

As was mentioned above, hardware designers are only interested in analysis of their system at speeds which will ultimately be those of the completed system. That is, real analysis is the only tool of interest to them. The states which this user considers are slightly expanded over those of the software designer. In addition to address, data, and status information, the user now desires information obtained by probing the target system at points of interest and displaying the logical value of this external data. Because of the digital nature of the data, this approach satisfies most of the digital designer's needs. This, together with glitch (multiple transistions on a line between clock times) detection, may be sufficient to debug an entire system. Again the hardware designer wants a system which "talks his language" and, thus, must have displays formatted in timing diagrams as well as in formats needed by software designers.



TIMING DIAGRAM TRACE-ABORTED

EXPAND INDICATOR [ON]
GLITCH DISPLAY [ON]
MAGNIFICATION [X1]

250NS/CLK
5US/DIV
[54.0US]

External data displayed in timing diagram format with glitch detection.

This desire on the part of hardware developers to include external data for analysis closely parallels the software developer's need to have access to register data for analysis. Here, as before, the external data is a logical extension of the state of the system as described in terms of address, data, and status information. Such an extension need not impact user interface to analysis since more state information is readily enterable and, due to the real time nature of external data, this analysis extension may be accomplished in real time,which is clearly necessary in hardware debug.

Hardware Stimulus Capability

The hardware designer needs not only to observe the user system, but to provide input to it as well. Certainly the address and data imformation is user system input and this together with the ability to provide specific logical stimuli to selected test points in the target system is necessary for complete hardware debug. In particular,the user needs the ability to program a hardware stimulus, run with this input, and analyze the results.

In the past,stimulus to systems under development, as well as sophisticated logic state analysis, has only been available in instruments separate from the emulation system. In order to simplify both the number of interfaces with which the developer needs to become familiar and the instrumentation necessary to support development, it is increasingly desirable to incorporate design support in a single package. To this end, use of the same probing techniques to observe as well as perturb the system under test will become very advantageous.

Support for External Measurements

Lastly, despite all the efforts of emulation and analysis, some portion of hardware debug will undoubtedly require some external instrumentation. The user now needs the ability to use the analysis to determine some operation of external instruments. Such a communication means needs to be provided to complete the debug. This means is provided by system outputs which provide TTL level signals to trigger other instruments based on conditions tested in real time by sophisticated state analysis available in emulation systems.

EVOLVING NEEDS AND SOLUTIONS

Universal Development Systems

As technology continues to provide an expanding and diversifying line of microprocessors, the developer of processor based systems is faced with the problem of inventing or purchasing tools to support development. As this occurs, an increasing amount of the developer's time is spent in learning the capabilities and idiosyncracies of new development systems. Clearly a need has arisen for systems which are designed with support of yet to be revealed microprocessor advancements in mind. In the area of software development support, software technology currently admits to the design of compilers, assemblers, and linkers so as to be very readily upgraded to support processors with varying instruction sets, address spaces, addressing modes, and architectures. Similar foresight in emulation software and hardware architectures allows for adaptability to new processors. In emulation software designs, systems can be constructed in a top-down manner so as to exhibit processor dependence only at the lowermost, hardware driver level. This allows handling of new processors with a minimal design impact, and keeps the user interface to changing target processors familiar and unchanged. In hardware architecture, the use of general purpose interfaces to real analysis and memory controlling units allows hardware emulation support for other processors without redesign of anything more than processor control.

Emulation of Multiprocessor Systems

Another area of growing importance is emulation support for the developer of multiprocessor based systems. To date such a developer has been asked to work with multiple emulation systems or to resort to having full emulation control of only one processor at a time. Emulation systems based on the concept of independent buses for emulation and host processors promise the means for emulating multiple processors simultaneously with user control or analysis of either processor by instructions affecting each processor separately. As the two or more processors need not contend for a shared resource, in terms of address or data buses, both may execute at full speed and real analysis be performed on interactions between processors.

Maximizing System Utilization

As development systems become more universal design aids, the need to maximize utilization of this resource becomes increasingly important. Certainly such systems must admit to multiple users sharing the same data base. No large scale development effort today consists of a single software developer and, thus, systems to support software development must admit to use by more than one designer simultaneously. Similarly, such systems must minimize conflict between the use of the system for real time emulation for hardware development and use by software developers. The implementation, mentioned above, of systems in which the target processor need not use the host system processor's address and data buses to operate makes possible the simultaneous use of the system to emulate a processor and execute system developed software in real time (using only the target processor buses) and use of the system for software development (using the system's host processor buses). This implies that now the hardware developer may debug, with external instrumentation, an emulated system running real time while the emulation station itself is being used by a software developer to continue code production.

CONCLUSION

Overall, today's microprocessor based system designer faces an expanding need for support tools which will not be obsoleted as new technology continues to become available. Much has been accomplished toward understanding how to support these needs. Emulation and development systems continue to anticipate and be designed with expansion and enhanced features in mind and promise to allow the developer more freedom in choosing new technology processors without fear of lagging development support. In short, much has been accomplished in the field of microprocessor software and hardware development support and there is equally as much to be accomplished in the future.

6. Making Your Data Communication Network More Available.

HEWLETT PACKARD

Garn Nelson of the Delcon R&D Lab received his B.S.E.E. from Brigham Young University. An experienced lecturer in data communication testing, Garn is co-authoring a book on data communications.

MAXIMIZE AVAILABILITY OF YOUR DATA COMMUNICATION NETWORK

Introduction

The goal of data communication testing is to increase system availability. End users of a data communications network demand a certain level of system availability. In many instances, being able to quickly and accurately test the network is the only way to provide an adequate level of system availability.

As service industries become more dependent on data communications, their need for high system availability increases. As an example, consider the resulting customer frustration and dissatisfaction if an airline reservation system was "down" for several hours, or if an automatic bank teller could not communicate with the bank's main computer.

One national drugstore chain has a large national data communication network to assure availability of prescription drugs. If any store has run out of a drug or has need of an uncommon drug, the drug is delviered within 24 hours. Delivery schedules and inventory management at the warehouses are controlled via two large mainframe computers in California and Michigan.

The use of computers by manufacturing companies for inventory control is another area that is becoming increasingly dependent on data communications. Managers must be able to control inventory and schedule in Real Time in order to reduce costs and improve delivery time.

A major semiconductor company uses a worldwide network to improve delivery times and manage inventories and production schedules. Sales orders are collected at a central location in Europe and transmitted to the United States over a wideband channel to the central computer site in Santa Clara County, California. Domestic orders are also transmitted in Real Time to Santa Clara over an extensive network. The computer processes the orders, adjusts manufacturing schedules, prepares shipping information, invoices and acknowledgements. Much of this information is sent to the manufacturing facilities in the Far East over a wideband channel. This very elaborate network gives a competitive advantage to this firm by lowering inventory costs and speeding delivery.

As these and other applications become more critical to the way companies do business, there is an increased need for high system availibility. Many companies now require system availability at 98% or better.

System Responsiblity

Responsibility for maintaining a data communications system in a multi-vendor environment remains with the Data Communications Manager and not the vendors. The vendor can repair the equipment, but the vendor cannot be responsible for pinpointing the system component that is not functioning properly. Only the Data Communications Manager knows how each component of the system relates to the entire system. The Data Communications Manager is responsible for isolating network problems to the system component level. Once this is done, the appropriate vendor can be called to fix the problem.

Vendors expect the Data Communications Manager to find the faulty system component. Most vendors charge about $50 per hour for a service call, whether or not the vendor's equipment is faulty. Many carriers charge extra for service calls when there is not a carrier problem.

Survey of Data Communication Managers

Hewlett-Packard has completed a survey of 50 Data Communications Managers. The survey ascertained how the ability to test a data communication network increased system availability.

One finding of this survey is that each network has unique maintenance problems. Networks are different, testing philosophies are different, vendor's maintenance capabilities are different. Even the same vendor has varying abilities to maintain system components at different locations in the country. Some Data Communications Managers claim that their carriers MTTR (Mean Time To Repair) is about one hour. Others complain that it takes two or three days, on the average, for the carriers to return their lines to a serviceable condition; some examples were given where it took the carrier weeks to restore service.

A second finding is that there is an extremely high correlation between maximum system availability and the amount of test equipment owned. In fact, on all networks that had 98% availability, test equipment is used to maintain the network.

An interesting sidelight is that a return on investment analysis is seldom used to justify purchase of test equipment. There is generally no attempt made to quantify end-user dissatisfaction or loss of revenue when the system is down and use that number to justify the expense of test equipment. The justification for the purchase of test equipment is that end users expect or require a certain level of system availability. In order to achieve a high level of system availability, the Data Communication Manager needs test equipment.

Survey Results

1. The most important way testing capability increases system availability is to positively isolate faulty system components. Calling a vendor in to fix equipment that is working properly wastes a significant amount of time. Most vendors are able to restore equipment and service in four to eight hours.

The following is a hypothetical example of the problem calling the wrong vendor first:

Example

Day 1

10:00 a.m.: A user complains his terminal is not working.

10:15 a.m.: The carrier is called to check the line.

11:30 a.m.: The carrier calls back, "No trouble found".

11:45 a.m.: The modem manufacturer is called.

Day 2

10:00 a.m.: The modem repair person arrives, verifies that the modems are working properly and leaves a bill for $100.

11:00 a.m.: The host computer service engineer is called.

1:00 p.m.: Computer service engineer arrives, makes some tests, and says the computer is working properly.

2:00 p.m.: The terminal vendor is called to test the remote terminal.

Day 3

10:00 a.m.: The terminal repair technician arrives, finds a problem in the terminal and makes the repair.

11:30 a.m.: The system is up and working properly.

Since in many networks, the mainframe computer, modem, line, and terminal vendors are all different, simply guessing which vendor to call will decrease system availability. By looking at the symptoms of a problem and using common sense, the Data Communication Manager can find the faulty system component better than 50 percent of the time. Having testing capability improves the accuracy of troubleshooting. One Data Communication Manager with some test equipment could isolate the correct vendor 90% of the time. This was insufficient to provide adequate service and he was planning to upgrade his testing capability so he could find the problem the first time 95% of the time.

Several of the Data Communication Managers surveyed felt that it was necessary to be able to isolate the faulty system component the first time at least 95% of the time to maintain system availability at 98% or better.

2. The most difficult problems to find are intermittent failures. By the time the vendor repairman arrives, the problem goes away. The best way to find intermittent problems is to have test equipment on site at all times so when the problems occur they can be found immediately. Otherwise, these annoying intermittent problems can go on indefinitely and are a major factor in determining system availability.

3. Many Data Communication Managers feel it is necessary to have control of their network in order to provide adequate system availability. With no testing capability, the Data Communication Manager is not in control but is at the mercy of his vendors. The vendor's competency, honesty, and work ethic determine system availability, not the Data Communication Manager. For example, a number of Data Communication Managers thoroughly test new lines immediately after the carrier has brought them up. Usually, one line in ten does not fully meet specifications. By testing new lines, the Data Communication Manager is in control of the network and is not dependent on the carrier.

4. Starting up new applications and installing new equipment always has problems. Having the capability to completely test lines, modems, terminals and software individually, shortens the start-up time. Systematically testing the new line, then the line and the modems followed by the line, modems and terminal, and finally, the line, modems, terminal and software, is an orderly approach to installation which can decrease installation time and startup problems.

   Also at the time of installation, the test results of the digital and analog links are recorded. These measurements serve as a benchmark for future testing needs. For example, when an end user reports a problem, quick and simple loop-around tests can be made and compared to the original test results. Often, this will quickly point to the faulty network component.

   The technicians who install modems and terminals of the San Francisco Bay Area divisions of Hewlett-Packard carry with them an HP 4944A Transmission Impairment Measuring Set for testing the line and an HP 1640A Serial Data Analyzer for testing the modem and terminal. They are able to quickly isolate problems when they occur and, thereby, maximize the use of their time and travel, as well as speed up the installation time.

5. Perhaps the most difficult and uncomfortable problem a Data Communication Manager has is fingerpointing squabbles among vendors. Fingerpointing is also very costly in lost system availability. Fingerpointing problems may last several days or even weeks.

One company in Oregon had a problem with a link going to a remote site in California. The carrier was called to check the line. The next day, the carrier tested the line and reported "no trouble found" On the third day, a modem repairman was flown to the remote site at the expense of the Oregon-based company. The modem repairman did some extensive testing on the modem and concluded the modem was working properly. The fourth day was spent negotiating with the carrier to retest the line. Finally, on the fifth day, the carrier did some more complete testing and found that frequencies around 2200 Hz were severely attenuated. The signaling frequency of the modem was 2200 Hz. A day later, service was restored to the remote site. This company periodically has problems of this nature and has just recently upgraded its analog testing capability to minimize these problems in the future.

These costly outages are minimized or eliminated if the Data Communication Manager can positively identify the problem in the first place. This is another example of how a Data Communication Manager profits by being in control of the network.

6. Analog testing can improve the carrier's mean-time-to-repair. When the Data Communication Manager is able to consistently and accurately isolate the line problem, his credibility and reputation with the carrier improves. When a manager calls in a trouble report, the carrier will not be prone to "finger-pointing" and the carrier technicians immediately go to work.

Usually, when a trouble call comes into the carrier, the first testing that is done is a simple continuity, level and noise test between central offices. If the line passes these tests then the carrier reports "no trouble found". If the customer still reports trouble, then more sophisticated testing is done. But if the Data Communication Manager can identify a specific problem, such as "20 degrees of phase jitter", the carrier technician might test phase jitter first, which may save hours of down time.

If preliminary testing from central office to central office does not uncover the problem, the carrier typically will dispatch a technician for on-site testing. Data Communication Managers are expediting this procedure by doing the on-site testing with the carrier and eliminating the need for the several-hour wait for a technician to arrive.

Many Data Communication Managers are able to develop a team relationship with the carrier as opposed to an adversary relationship. Since they have the ability to make measurements, they can talk to the carrier in his own language. Carriers are becoming much more open about the way they operate, even to offering visits to carrier facilities. Also, many Data Communication Managers go out of their way to develop friendships with Carrier Technicians. These three things

1) Being able to test as well as the carriers,
2) Being able to understand the problems and workings of the carrier,
3) Developing personal relationships

engender a feeling of teamwork and mutual trust and respect. When the team spirit is accomplished, great improvements in mean time to repair occur. In fact, several Data Communication Managers strongly feel that this cooperation is by far the most important factor in maintaining a high system availability.

7. In spite of good relationships with vendors, sometimes it is necessary to escalate the problem to a higher level of management. Even though this is the last resort for the Data Communications Manager, it may be necessary to get proper service. To get proper service from a vendor, a Data

Communications Manager may ask the upper-level management to call the upper-level management of the vendor. Before the upper-level manager calls the vendor, he wants to be absolutely certain the vendor is at fault. Adequate testing capability gives the data communication manager the assurance needed to begin the escalation procedure.

8. One way to prevent down time is to catch the problem before it impairs communications. Routine maintenance testing can spot a degrading line or modem. When a system component is found to be deteriorating, it can be fixed at a convenient time to minimize disruption to the end user. Routine maintenance testing is used by Data Communication Managers to derive the maximum availability out of their data communication network.

CONCLUSION

The way companies do business is changing; they are using data communications to improve service and better manage resources. As data communication becomes increasingly important to the performance of the company, system availability becomes increasingly important. A survey of Data Communication Managers indicates that the use of data communication test equipment significantly increases system availability.

7. Three Domains of Data Communication Testing.

HEWLETT PACKARD

John Wetzel of the Delcon Division, with a B.S.E.E. from Princeton and an MBA from Harvard, is Product Marketing Manager. John has an R&D background in data communication.

David Novotny has served on design teams for the 4943A TIMS and 1640A Serial Data Analyzer. He holds a B.S.E.E. from California State Polytechnic and an M.S.E.E. from Stanford.

NETWORK ANALYSIS
UTILIZING PROTOCOL, DIGITAL, AND ANALOG TESTING

Introduction

Protocol, digital, and analog testing all work together to help a Datacom Network Manager maximize the availability of his network. Each of these testing techniques has its strengths which enable it to isolate certain types of faults, but the real value comes in using all three of them together to be able to quickly and efficiently isolate network problems. Protocol testing is testing on the EIA (Electronic Industry Association) digital interface. Its prime characteristic is that the information content of messages and protocol control characters is retained and analyzed. Digital testing also tests on the EIA digital interface, but it involves quantitive tests of error rate or distortion using a Pseudo Random Bit Sequence (PRBS). It is typically referred to as Bit Error Rate Testing (BERT). Analog testing is performed on the analog telephone circuit typically provided by the telephone company. It involves the characterization of channel parameters that measure how well a modem should transmit over that channel.

The general attributes of each of these testing techniques needs to be understood before examples of using them together can be discussed. First, protocol testing is probably the most common testing technique. Protocol test instruments vary significantly in terms of price, capability, flexibility and ease of use. Protocol analyzers start at about $3,000 and extend up to about $25,000. All units have the ability to monitor data information on a channel without interrupting the data flow. In addition, the more expensive units have abilities to simulate system components in the network to other system components and to make performance measurements such as response time and percentage of blocks having to be retransmitted. These can be very useful in bringing up new system components or troubleshooting more sophisticated problems. The strengths of protocol analyzers are their ability to monitor the traffic on an operating data communications channel and to analyze the information traveling between two logical devices (computer and a terminal). This allows a technical control operator to look at a channel when a user reports a problem.

By being able to look at the problem in Real Time, often the situation can be quickly analyzed and corrective action taken. The weaknesses of the protocol analyzers are that most of them are not as portable as one would like, they are somewhat confusing to use if they are not operated regularly, and they can be expensive if the more sophisticated units are needed.

Digital testers or BERT testers are strong where protocol analyzers are weak. They are less expensive with some units available for under $500 and the more expensive units costing about $4,000. The more expensive units offer additional features. They can include a breakout box, and the capability of making EIA digital interface timing measurements. They can also include longer PRBSs that test for pattern sensitivity of modems. Some of the more expensive units will make simultaneous measurements of several parameters to show whether errors are occurring in bursts, are due to a clock timing problem, or are biased indicating a possible modem threshold problem. The strengths of BERT instruments are that they are inexpensive, they are portable, very easy to use and give a single, quantitative number for a measurement result. It is possible with a BERT instrument to talk an inexperienced operator through the operation over a telephone. The weakness of BERT testing is that it is normally a static test. It does not excercise the EIA digital interface leads or turn the datacommunications channel around in half duplex operation. When used by itself, it is difficult to isolate modem problems from line problems. Also, it is an intrusive test which involves taking a channel out of service.

Analog testing addresses only one component of the datacommunications network - the telephone line. But since this is such an important building block for the network, it can merit such focus. Analog test instruments vary in price from about $1,000 to more than $10,000, with the major difference being how many analog channel parameters are measured. A general rule is the faster a network is trying to transmit data, the more analog parameters that must be measured to assure reliable operation. The strengths of an analog instrument are its ability to unambiguously identify a faulty telephone line and the probable channel parameter which is at fault. This can improve the telephone company's restoration time. The weakness of analog testing is that the analog parameters are difficult for many network personnel to understand, since they operate in a primarily digital environment. To some network personnel, the instruments are confusing to use, probably due to the basic confusion over the parameters themselves. This can be minimized by using instruments which have Master/Slave capability (the ability for an instrument at a central site with a skilled operator to automatically control an instrument at a remote site where personnel may be less skilled). Finally, the more complete instruments which measure most of the channel parameters are rather expensive (more than $7000) and are less portable than desired.

When a Network Manager combines instruments from these three domains of data communications testing (analog, digital and protocol), and adds circuit access patching, he has created a technical control center. Such a center is the key element in keeping network availability high. Just establishing a technical control center will not magically do this. The center must be efficiently operated to obtain maximum benefit from the test equipment. When the equipment benefits are maximized, network downtime is minimized. How does the Network Manager maximize the benefit from test equipment? When should measurements be made? What measurements should be made? How should the results be interpreted? These are areas on which this paper will focus.

Three Phases of Testing

To obtain full benefit from a well-equipped technical control center (one that has instruments capable of making measurements in all three domains of data communication testing), specific testing phases and testing methods need to be identified. Three such phases are commissioning, troubleshooting, and preventive maintenance. Commissioning testing is done after the analog circuit has been turned over by the Telephone Company, but before it is merged into the network. It also includes incoming inspection of other on-line equipment. This is the Network Manager's opportunity to use his test equipment to evaluate the analog circuit and equipment for himself and to insure that they meet all tariffs and specifications. Once all of the equipment is assembled, a benchmark or "fingerprint" of each layer of the datacommunications channel is made. The troubleshooting test phase is done under high pressure. Part of the network is down and the prime objective is to restore as much of the network as possible, to isolate the faulty network component, and to get that component repaired. The preventive maintenance test phase can be time well invested. Every element in the network can be periodically checked, under controlled conditions, to insure it meets all network specifications and tariffs. If network personnel can spot a potential problem before it occurs, they can keep network availability high.

By putting these three test phases together, the Network Manager creates a combined test concept. Commissioning tests the network from the inside out, network component by component, starting with the analog circuit. Each successive network component is then added to a proven, solid block until the data channel is assembled. This prevents the network from being built on weak links. Troubleshooting proceeds in the opposite direction, that is outside in. Network components are tested and eliminated from the suspect list, starting with the FEP (Front End Processor) and terminals and then moving in toward the analog circuit. This testing method starts with quick tests and proceeds to the more thorough and time-consuming ones. Preventive maintenance is really a repeat of commissioning testing on the installed equipment and, therefore, starts the inside out cycle again. To keep all of the test data organized, a good record-keeping system must be established and maintained at the technical control center. Organized on a datacommunications channel basis, these records form the backbone of the combined concept.

Initially, a 3002 grade, point-to-point, private line will be considered. See Figure 1. After looking at the three testing phases in this basic situation, special applications to different network architectures involving multipoint, DDD, DDS and multiplexers will be discussed.

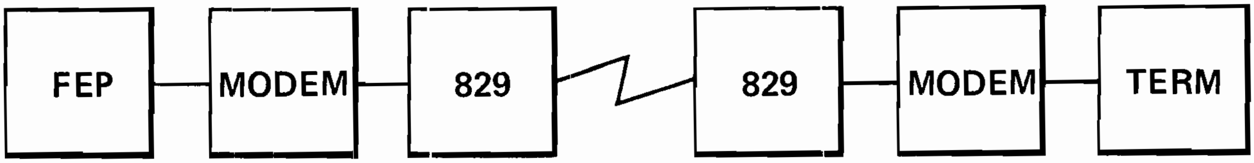


Figure 1. A simple data channel.

Commissioning Testing

Commissioning testing is the Network Manager's opportunity to analyze an analog circuit and the on-line equipment before the circuit is integrated to the network. The objectives of commissioning testing are threefold. First, the Network Manager has the opportunity to measure and record all of the analog circuit parameters, end-to-end. By doing so, the circuit can be compared to the applicable tariffs to insure that the circuit meets the specifications which have been ordered. Second, loop-around measurements can be made. Although the loopback measurements are not subject to tariffs, they provide an excellent benchmark or "fingerprint" of the circuit. This fingerprint can be very useful in tracking down future analog circuit problems. Third, complete diagnostic tests on the equipment can be run to insure all equipment is operating and has the proper options installed.

Since the tariff checking portion of commissioning testing must be done end-to-end (this is how the tariff is written), equipment capable of making the required measurements must be at each end of the circuit. This equipment should include an analog test set and a digital test set (BERT). Some analog test sets offer a Master/Slave capability which allows the personnel at the central site unit to control the measurements and to collect the data from the remote site as well as the central site. See Figure 2. Access to the analog circuit can either be made at the analog patch field or the Bell DAS 829 Channel Interface Unit or equivalent. The standard test tone is 1004 Hz at 0dBm power level[1]. The following measurements should be made on each circuit and recorded for future reference:

- Received level of test tone
- Bandwidth (Attenuation Distortion)
- C-Message Noise
- C-Notched Noise (or signal-to-noise ratio)
- Impulse Noise
- Envelope Delay Distortion
- P/AR

and if the data rate is greater than 2400 bps:

- Phase Jitter
- Non-linear Distortion
- Gain Hits
- Phase Hits
- Drop-Outs

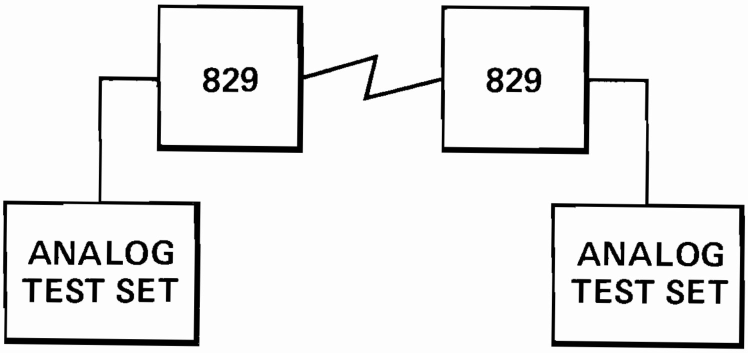


Figure 2. End-to-end testing using Analog test sets.

Although the P/AR (Peak to Average Ratio) is a non-tariffed parameter, it is a good quick quantitive figure of merit for both bandwidth and envelope delay. Typical P/AR numbers for lines are:

Basic 3002 Circuit	>45 P/AR Units
C1 Conditioning	>48 P/AR Units
C2 Conditioning	>78 P/AR Units
C3 Conditioning	>87 P/AR Units
C4 Conditioning	>95 P/AR Units

D Conditioning has a minimal effect on P/AR values.

All of the transmission objectives for these measurements are available in Bell System Technical Reference, PUB 41004. More detailed information on making analog measurements is available in Bell System Technical Reference, PUB 41008, PUB 41009, and operating manuals for the various analog test equipment.

Looping the analog portion of a 4-wire 3002 circuit can be done in several ways. Since, in this case, the circuit is a non-multidrop, the easiest method would be to loop the circuit at the channel interface, normally a Bell 829. The 829 can be looped remotely by applying a 2713 Hz tone on the line for 5 seconds, then removing it. The 829 will put a 16dB gain amplifier into the loop to keep the signal at its appropriate data level. The following analog loopback measurements should be made and results recorded for future use:

- Received level of test tone
- Bandwidth (Attenuation Distortion)
- Noise
- Signal-to-Noise Ratio
- Envelope Delay Distortion
- P/AR

[1] A companion to this paper is "Analog Testing and its Benefits to the Data Communications Manager" by Donald A. Dresch and Thomas R. Graham. Analog measurement techniques are expanded here.

See Figure 3.

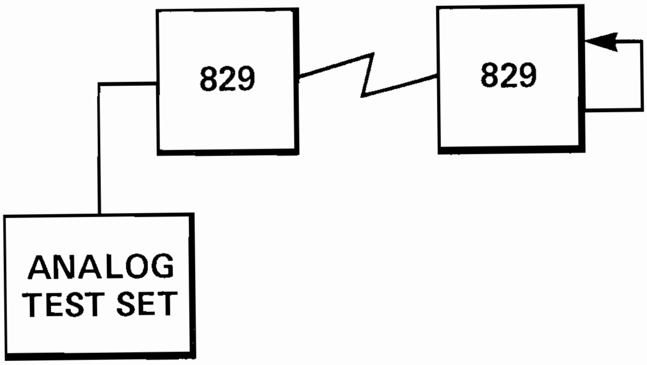


Figure 3. Testing the Analog line in a loop around configuration using Analog test equipment. This is a "Fingerprint" test.

To release the loopback on the 829, the 2713 Hz tone should be applied for two seconds, then removed. The operator can confirm the loopback is released by measuring a loop-around loss greater than 35dB.

A few words of caution are appropriate here. First, the measurements made in the loopback configuration, though not subject to tariffs, are extremely useful for troubleshooting. They provide a quick check of the analog circuit that can be made easily when trouble occurs on the channel. By knowing what the loopback measurements were at the time of commissioning, a reference point is established. Similar measurements made during a trouble call can indicate the source of the problem quickly. Second, many of the loopback measurements will not be the linear sum of the end-to-end measurements. Some add nonlinearly, like noise, for example. Others may cancel completely, such as phase jitter, if the circuits are routed through the same facility. Again, it must be emphasized that at the time of commissioning, the telephone company should be called only if the end-to-end parameters are out of specification. Once the analog tests on the line have been completed, the first block of the data communications channel is proven and established. It is time to proceed outward and look at the modems.

If possible, the modems should be pre-tested and "burned-in" before installation. This is essential if the far end of the intended circuit is not close to a service facility. This test is akin to incoming inspection; the Network Manager should get the quality for which he is paying. A good test is to run a 24-hour BERT test on the modems in the analog loop configuration. This allows the modem transmitter to talk to the receiver. A zero error count is the goal in this test. If the manager desires to set a less stringent count as the goal, then the modems with better performances should be sent to the remote sites. It will be easier to substitute modems at the central site as opposed to the remote site.

At installation, the modems are added to the tested analog circuit and end-to-end BERT tests performed. See Figure 4. This test should be done over a fifteen-minute period. The results should be recorded. Next, place the remote modem in digital loopback and perform a loop-around BERT from the central site. See Figure 5. This is another line fingerprint. Digital loopback on the modem connects the received data to the transmit data. This loop-around test is a good measure of performance for the modem-line combination.

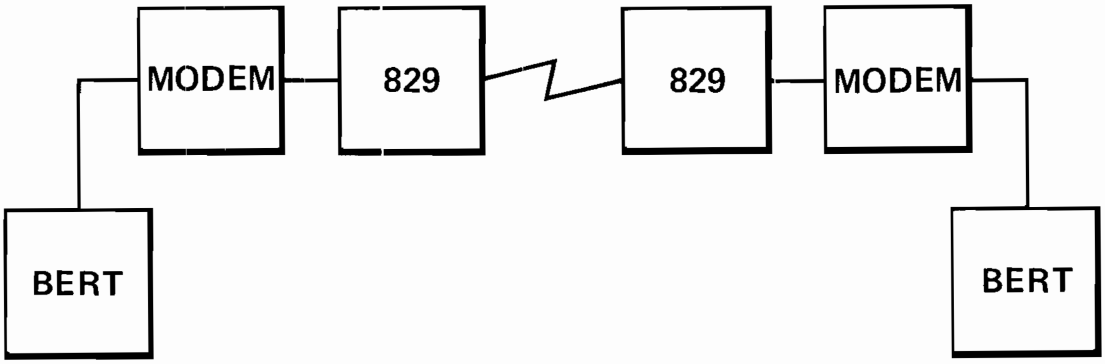


Figure 4. End-to-end BERT test.

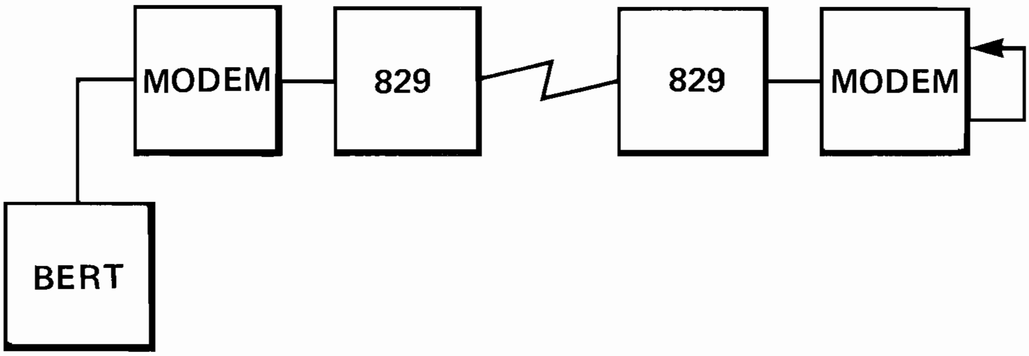


Figure 5. Loopback BERT test with the modem in digital loopback. This is a "Fingerprint" test.

A typical minimum acceptable BERT for both of these tests is one error in 10^5 bits transmitted or zero errors in fifteen minutes. Experience with particular modems may require even different limits.

The prime reason for making each of these loopback tests is to establish a benchmark or "fingerprint" for each circuit. Knowing the loopback circuit parameters at the time of commissioning (assuming the line meets tariffs) gives a basis for comparison when similar measurements are made during troubleshooting. Notice that the fingerprint is used only in comparison which is a relative term. The objective is to see if anything has changed, and if so, what.

The terminals and FEP (if new) are the final items to be added to the datacommunications channel. Again, pretesting of the individual units should be done, if logistics permit. To perform the terminal pretest, a protocol analyzer with simulate capabilities is required. These are the more expensive models, but worth the money spent in this application. The objective of the terminal pretest is to insure the device is configured with the proper options and functions as desired when stimulated with the protocol in use. The protocol analyzer is programmed to simulate the FEP.

The final installation step is attaching the terminal and FEP to their respective modems and exercising the terminal. This, of course, should be done before installation people leave the remote site. If one of the network components is not ready to come on line (terminal or FEP), the protocol analyzer should be used to simulate the missing component.

The final step has now been completed. The channel has been assembled, tested and fingerprinted from the inside out. All installation data has been acquired and recorded for future reference. The network is brought up and running smoothly until a trouble call comes in. It is now troubleshooting time.

Troubleshooting

The troubleshooting phase is done under high pressure. Part of the network is down and it is the responsibility of the Network Manager to restore the network as quickly as possible. With data communication networks constantly growing and the supply of qualified data technicians decreasing, the Network Manager should rely on good test equipment and well-documented troubleshooting procedures. Troubleshooting testing begins on the outside of the network and moves in toward the analog telephone circuit.

Upon notification of a network problem, be it either through a phone call or an indication from the TP (Telecommunications Processor) monitor, network personnel patch a protocol analyzer onto the datacommunications channel. See Figure 6. Key items to analyze are polls and responses. If transmit data is badly garbled, or non-existent, the problem is toward the FEP. Network personnel should be sure that the terminal in question is still in the polling sequence. If the receive data is badly garbled or non-existent, the problem is toward the terminal. If the terminal is not responding to the poll, network personnel should swap in a spare modem at the central site. This will identify and eliminate central site modem problems quickly and easily. Now that the

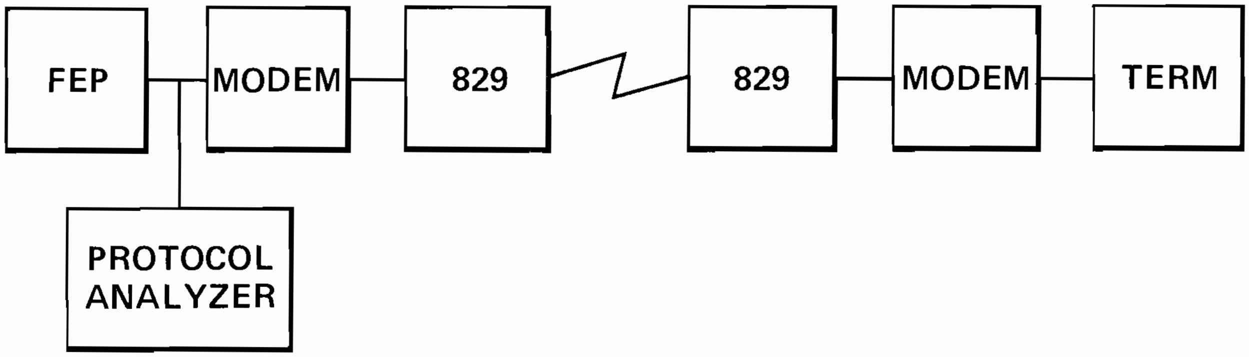


Figure 6. Monitoring the data channel using the protocal analyzer.

computer, FEP and central site modem are known to be good, the next step is to move in one layer in the troubleshooting scheme and check the line-modem combination using digital test equipment. To do this, the remote end modem is placed in digital loopback. The loop-around BERT test is performed from the central site. See Figure 7. The results are then compared with the channel's fingerprint. If the BERT test compares favorably, then the terminal is suspect. If the BERT test is bad, then the remote site 829 should be placed in loopback and another BERT test run. See Figure 8. A successful test indicates remote site modem problems. So far, the higher two layers of the channel have been tested. If poor results are still obtained, then analog loopback tests should be performed on the telephone circuit. See Figure 9. If the analog loopback measurements show that the analog circuit has poorer circuit characteristics than when it was commissioned, then the telephone circuit is probably at fault. If the loopback analog measurements show no change in the telephone circuit, then the remote modem is most likely at fault.

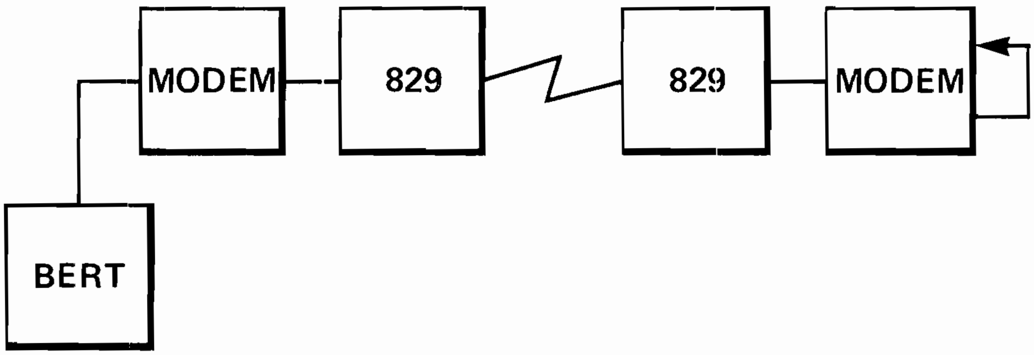


Figure 7. Loopback BERT test with the modem in digital loopback. This test isolates the terminal.

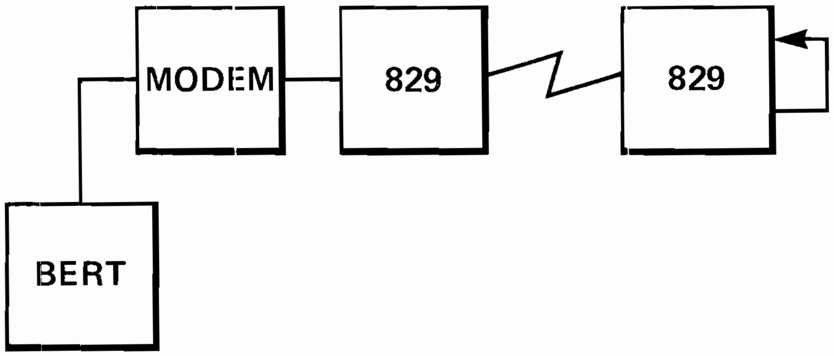


Figure 8. The BERT test with the remote CIU in loopback mode, used to isolate a faulty modem during troubleshooting.

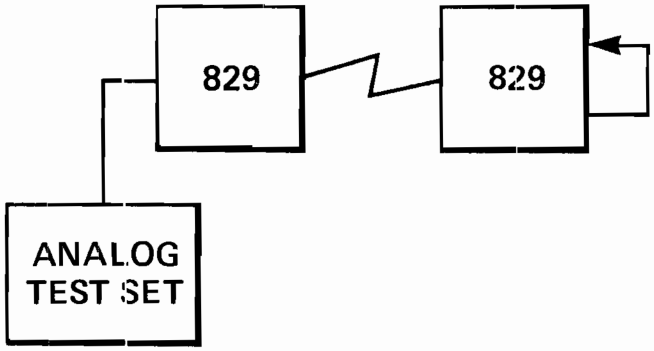


Figure 9. The final troubleshooting test is running the analog "Fingerprint" test.

Preventive Maintenance

The preventive maintenance phase gives the Network Manager an opportunity to identify marginal data channels before an outage on that channel occurs. A marginal channel may be the reason behind slow response time and decreased throughput. By doing regular preventative maintenance, a good rapport with the telephone company can be established. When telephone company personnel see network personnel willing to devote time, money, and equipment in preventive maintenance, both parties gain confidence in the other's ability to make accurate measurements.

What is the preventive maintenance phase? It is really the commissioning phase all over again. If network time and personnel availability permits, complete end-to-end testing should be done, making all applicable measurements for the data rate in use. The Master/Slave capability on the analog test sets can decrease the trained personnel requirements and greatly speed the data collection as can calculator or computer-controlled measurement systems. If end-to-end tests are not possible, then loopback tests are acceptable. If time is critical, network personnel should make at least: 1) received level at 1004 Hz, 404 Hz and 2804 Hz, 2) C-message noise, 3) signal-to-noise ratio and 4) P/AR. If time permits or an automatic data collection system is used, then the complete set of measurements should be made. Finally, the transient measurements of impulse noise, phase hits, gain hits and dropouts should be made. These are time-consuming tests (5 or 15 minutes). The loopback fingerprint sequence should be run to check for any variation from the original fingerprint. The BERT test should also be included in this sequence. All results should be recorded.

As can be seen, one of the keys to this troubleshooting scheme is the fingerprint. For a network that is currently operating, and commissioning tests were not made, there are two alternatives for fingerprinting the network. The first is to assume that since all lines are operating, they are, by definition, good and a benchmark fingerprint can be taken. The second is to go through all of the commissioning tests to insure the integrity of all network components and then fingerprint the network. Obviously, the second method is more comprehensive but takes more network time and is more expensive.

Record Keeping

So far, many references have been made to recording readings, but where are these to be recorded? What should be in the circuit records? What format should be used? The Network Manager must answer all of these questions. One possible method of record keeping uses a looseleaf notebook. It can be organized by user terminal or remote modem. Each entry should include such things as:

- Make, model and location of all equipment on the channel
- Telephone numbers of vendors for service
- Circuit routings
- Telephone Company designators for lines
- FEP Port
- Polling address
- Location of circuits on patch panels

Immediately following this identification information should be the current channel fingerprint followed by the commissioning and preventive maintenance data. Each measurement session should have its own circuit record card, similar to that in Figure 10. Finally, there should be a channel history section where all problems that have occurred on that particular channel are recorded. This information can be valuable in tracking down long-term or periodic problems.

TIMS
TRANSMISSION LINE
PARAMETER RECORD

CIRCUIT DESCRIPTION ____________________

____

REASON FOR TEST ____________________

____

TEST NO. ________

TEST DATE ________

FROM ________ TO ________

1. **LEVEL & CONNECTION VERIFICATION**
   1000 HZ RCV ________ DBM 60 HZ FILTER ________
   TRMT TLP (FROM SIDE) ________ FREQ SHIFT ________
   RCV TLP (TO SIDE) ________

2. **LOSS** ________ DB

3. **P/AR** ________ UNITS

4. **NOISE**

	C MSG	3 KHZ	DESCRIBE NOISE
MSSG CRCT	________DBRN	________DBRN	
WITH TONE	________DBRN	________DBRN	
S/N RATIO	________DB	________DB	
TO GROUND	________DBRN	________DBRN	

5. **PHASE JITTER** ________

6. **NONLINEAR DSTRTN**

	2 ND	3 RD
TEST	________DB	________DB
CHECK	________DB	________DB
CORRECTED	________DB	________DB

7. LEVEL/FREQUENCY			
300	________DB	2000	________DB
500	________DB	2200	________DB
600	________DB	2400	________DB
800	________DB	2500	________DB
1000	________DB	2600	________DB
1200	________DB	2700	________DB
1400	________DB	2800	________DB
1600	________DB	3000	________DB
1800	________DB	3200	________DB

8. ENVELOPE DELAY			
500	________ μS	2000	________ μS
600	________ μS	2200	________ μS
800	________ μS	2400	________ μS
1000	________ μS	2500	________ μS
1200	________ μS	2600	________ μS
1400	________ μS	2700	________ μS
1600	________ μS	2800	________ μS
1800	________ μS	3000	________ μS

9. **IMPULSE NOISE,** PHASE HITS, GAIN HITS, DROP OUTS (BELL STD., C MSG)	______DBRN	LO COUNTS	________
	+4DB	MID COUNTS	________
	+8DB	HI COUNTS	________
	______	PHASE HITS	________
		DROP OUTS	________
________CNT TIME	______ DB	GAIN HITS	________

FROM ________ TO ________

1. **LEVEL & CONNECTION VERIFICATION**
   1000 HZ RCV ________ DBM 60 HZ FILTER ________
   TRMT TLP (FROM SIDE) ________ FREQ SHIFT ________
   RCV TLP (TO SIDE) ________

2. **LOSS** ________ DB

3. **P/AR** ________ UNITS

4. **NOISE**

	C MSG	3 KHZ	DESCRIBE NOISE
MSSG CRCT	________DBRN	________DBRN	
WITH TONE	________DBRN	________DBRN	
S/N RATIO	________DB	________DB	
TO GROUND	________DBRN	________DBRN	

5. **PHASE JITTER** ________

6. **NONLINEAR DSTRTN**

	2 ND	3 RD
TEST	________DB	________DB
CHECK	________DB	________DB
CORRECTED	________DB	________DB

7. LEVEL/FREQUENCY			
300	________DB	2000	________DB
500	________DB	2200	________DB
600	________DB	2400	________DB
800	________DB	2500	________DB
1000	________DB	2600	________DB
1200	________DB	2700	________DB
1400	________DB	2800	________DB
1600	________DB	3000	________DB
1800	________DB	3200	________DB

8. ENVELOPE DELAY			
500	________ μS	2000	________ μS
600	________ μS	2200	________ μS
800	________ μS	2400	________ μS
1000	________ μS	2500	________ μS
1200	________ μS	2600	________ μS
1400	________ μS	2700	________ μS
1600	________ μS	2800	________ μS
1800	________ μS	3000	________ μS

9. **IMPULSE NOISE,** PHASE HITS, GAIN HITS, DROP OUTS (BELL STD., C MSG)	______DBRN	LO COUNTS	________
	+4DB	MID COUNTS	________
	+8DB	HI COUNTS	________
	______	PHASE HITS	________
		DROP OUTS	________
________CNT TIME	______ DB	GAIN HITS	________

A second method of record keeping is a logical extension of the first. Since the data communications center is generally co-located with a large computer, the computer could maintain the records. This is a natural application for data-base management programs. Everything from trouble ticket generation to vendor response time evaluation is available using this type of record-keeping system. There is virtually no limit as to what can be done.

Multipoint

Up to this point, discussion and examples have been limited to networks having only 4-wire private line (3002), point-to-point, datacommunications channels. Maintaining a network with other types of channels will require some changes in the techniques already discussed.

The multipoint data communications channel is often used to help reduce data communications network expenses. In the process of reducing the network expenses though, the Network Manager has increased the maintenance problems of the network. First of all, the severity of an individual failure is increased. For instance, with a streaming terminal, it is possible for one user's failure to be felt by everyone who shares that circuit with him. In cases such as this, the network maintenance personnel must begin to troubleshoot the problem without even knowing on which remote station they should be focusing.

Commissioning testing should be stressed even more in multipoint situations than in point-to-point situations. Later, troubleshooting will be based on the process of elimination and inference from the problem symptoms. The more solid the network is at commissioning, the less likely it is to fail later and, also, the less likely it is that there will be the complex problem situation where two marginal situations are interacting with one another.

One of the big differences from the analog testing standpoint is that the network operator can no longer use loopback testing. This was a critical measurement in point-to-point channels. It enabled network personnel to positively identify situations due to telephone line problems. If the network operator were to place a 2713 Hz tone on the multipoint analog line, he would activate multiple remote loopbacks making any loopback measurement meaningless. Even worse, he might activate only some, but not all, of the loopbacks. Now when he alternates back and forth with applications of 2713 Hz, he will always have some interface channel units looped back. Therefore, a firm rule is do not use 2713 Hz activation of remote loopback on multipoint circuits.

The Network Manager should try to maintain circuit record cards on all of his multipoint circuits which show the telephone company's

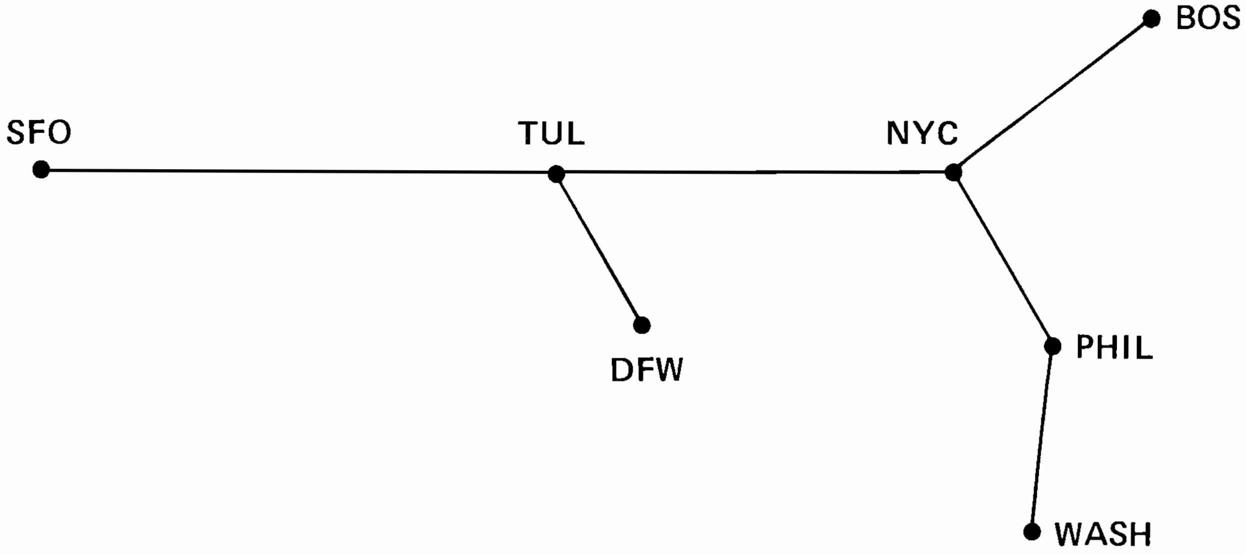


Figure 11. Multipoint communication circuits.

routing for his circuit. In most cases, the telephone company can be convinced to give the Network Manager that information. This allows a network operator to infer a line problem from a situation's symptoms; for example, in the multipoint circuit shown in Figure 11.

Failure of terminals in Philadelphia and Washington to respond to a poll would fit the symptoms of a New York to Philadelphia line problem. However, failure of terminals in New York and Washington to respond to a poll would not fit a line problem for a circuit routed this way.

In the example just discussed, the network operator is using some additional information that he did not use in the point-to-point situation. He used circuit routing, who is having problems, and who is not having problems to help him reason a problem through. He must use this extra information to compensate for his inability to make loopback tests.

Remote (secondary) modem problems should always show up as a single station being out of service. Remote terminal problems should show up either as only one terminal having problems or all terminals having problems (streaming terminal). Analog circuit problems will tend to take groups of terminals out of service, either all upstream or downstream from the problem, depending on whether the problem is on the transmit or receive side of the line.

If the symptoms of the problem do not allow the problem to be resolved with any certainty and if protocol monitoring (using the techniques discussed in the basic point-to-point instruction) does not provide any further information, the critical test will be a loopback BERT test to each secondary modem. This is an intrusive test and requires that the network operator has equipped all of his modems with analog and digital loopback. To run this test, all remote modems need to be quieted (place them all in analog loopback). Then each remote modem is placed, in turn, in digital loopback for a 10-15 second BERT. If a station shows errors and stations located beyond it (as referred to actual circuit routing) are not showing errors, then the remote modem is likely at fault. If all stations beyond a certain point on a circuit station show errors, then the circuit is likely at fault. If the station is showing errors and it is the last station on the line, then a modem self-test will have to be used to isolate a modem problem from a circuit problem. If all stations are in error, then it is either the primary modem or the circuit leg to the primary modem which is at fault. This can be resolved by utilizing the same technique that was used to identify a defective central site modem in point-to-point networks (spare modem substitution).

Direct Distance Dial (DDD)

Still another arrangement often used in a datacom system is the use of the DDD network. This is the everyday telephone system that people use to talk to one another. Networks using this type of architecture gain flexibility in talking to many stations for short periods of time very economically. However, in this type of network, maintenance procedures need to be changed. The main differences are that the telephone line is no longer a full duplex, 4-wire circuit that can be looped back for testing,and that generally every time a new call is placed to the remote station, a new circuit will be assigned to complete the call. Each of these differences will be analyzed separately.

First, DDD circuits are 2-wire circuits and generally half duplex (the recent 2-wire, 1200 bps, full duplex modems are an exception to this). From a testing standpoint, this means that the Network Manager cannot make analog and digital loopback (BERT measurements) to troubleshoot a network problem. The Network Manager should add a maintenance technique to make up for this. He should insure that his modems and DAAs (Digital Access Arrangements) are equipped with certain options. The DAA at all remote sites should be equipped with a switch-activated tone generator and a speaker. The remote modems should be equipped with the ability to send a random bit pattern or at least some kind of an idling pattern (to be able to energize and modulate the remote modem carrier).

With these options installed, the Network Manager will be able to perform the following checks. First, the remote modem can be set into a transmitting mode (sending a random pattern or idling pattern) and the telephone line can be bridged at the central site with an analog test set. Network personnel can measure the received level and listen to the signal on the analog test set's speaker. Network personnel can observe if the remote modem is transmitting properly and if the telephone circuit's loss is acceptable. If the incoming signal is weak, the switch-activated tone generator on the DAA at the remote site can be activated. However, if the circuit still shows high loss in the telephone circuit, then the telephone line would be at fault. If the circuit shows acceptable channel loss with the tone, then the remote modem would be at fault.

To test the telephone circuit in the other direction, central site to remote station (just because it is a 2-wire circuit at the analog circuit interface does not mean it is not 4-wire elsewhere), either a tone can be placed on the circuit at the central site, or the central modem can be set to transmit. Then an operator at the remote station can turn up his speaker and listen to the signal. If the remote operator is not trained to do this, he can hold a telephone close to the speaker and let network personnel at the central site interpret the sounds. This check should allow a defective telephone line to be identified if it affects only the transmit side of the circuit.

The second difference is that the telephone circuit will generally change every time the remote station is redialed. This means variability in analog circuit characteristics. On the positive side, many circuit problems will simply go away if the call is redialed and another transmission facility is selected to route the call. This is a straightforward DDD maintenance procedure. One of the first steps will be to have the network personnel redial the call. During light traffic periods (the middle of the night), redialing can consistently get a first-priority route and, consequently, the same routing and circuit. To force another circuit, the network operator can place a call to another telephone in the remote office and while it rings (thus busying the primary circuit), the computer can dial through and receive another circuit.

The negative aspect to getting a new line each time is that if a transmission problem is suspected, it is more difficult to get the telephone company to identify and fix it. If good rapport is established with the telephone company and if the network personnel are patient, the network personnel can hold the defective line for 15 minutes until the telephone company can trace it. Then the telephone company personnel will proceed to further pinpoint the problem.

There is one portion of the DDD circuit that will never change. This is the local loop or the portion of the circuit between the last telephone central office and the network's location. If this section of the line has problems, then redialing will not be able to fix the problem. This section of the line can be checked by dialing a DAA in the same office. This will allow loop-around analog measurements to be made. As was discussed in the point-to-point, private line situation, a benchmark of analog parameters when a circuit was functioning properly will allow measurement-by-measurement comparison to indicate transmission degradation.

Testing of remote local loop changes from the central site is not possible. It will appear as a consistent telephone line problem and the Network Manager will have to depend on the telephone company to isolate the problem.

Digital Data Service (DDS)

In the last several years, the use of DDS has increased dramatically. When a Network Manager chooses to use DDS, again he must modify his network maintenance approach. Luckily, the maintenance task in this situation is simpler than with other network architectures. With DDS, the customer demarcation point is normally the digital (RS232) side of the telephone company's D.S.U. (Digital Service Unit). This eliminates the modem vendor from the finger-pointing situation because the Network Manager is purchasing a telephone service that includes modem equivalent functions.

With DDS, there is no need for the network operator to do analog testing. First of all, it is not his responsibility to pinpoint faults to either a digital or analog problem. This is the telephone company's job. Secondly, analog measurements are not possible between customer demarcation points. Analog continuity is not maintained through the network. Giving up analog testing on this type of network should not leave the network in a more vulnerable maintenance situation. The telephone company has automatic maintenance systems on the DDS network. This enables them to provide a higher quality of service and faster restoration time. In many cases, they will be aware that a network's circuit is in trouble and be working to correct the situation even before network personnel report it.

With DDS service, the network should emphasize commissioning testing of terminals and use his protocol analyzer to perform logical problem isolation. Digital BERT instruments can be used to clearly prove the responsibility for a network problem to the telephone company. The use of modem and analog end sections on a DDS circuit to extend it to areas where DDS is unavailable is not recommended. This architecture provides maintenance complications that outweigh any benefit derived from using DDS for a portion of the circuit.

Multiplexers

The prices charged for multiplexers have significantly decreased in the last two years. Low-cost statistical multiplexers are now available, as well as very sophisticated network processors that multiplex into high-speed interfaces (greater than 9.6 Kbps). At the protocol and digital levels, there is not that much difference in maintaining a network with multiplexers than a network using simpler architectures. There is an additional network component that can contribute to the fault and, possibly, another vendor involved. However, at the analog level, the situation has changed. There is no longer analog continuity between the central site and the remote station location. Therefore, end-to-end or full analog loopback tests cannot be performed. Remote multiplexer locations should be viewed as remote network nodes, not just stations. This increases their importance from a network standpoint.

The Network Manager should take greater care in locating remote nodes than he would remote stations. To be on top of his network maintenance, a Network Manager will need to consider small technical control centers for the remote nodes. This is the only way he can be assured of maintaining the same network availability that is possible with a point-to-point network and a central technical control center. Also, patch access and, possibly, A/B port switching should be provided on both sides of any multiplexer.

The overall maintenance approach, with appropriate patch access and small technical control centers at remote network nodes, uses all three types of testing equipment. First, protocol analyzers are used to monitor and analyze individual ports. They will generally not be used to look at the high-speed digital interface of the multiplexer. On this interface, the user's message is buried in multiplexer protocol and is confusing to interpret.

BERT instruments are used on two interfaces. First, they can be used on an individual port of the multiplexer to verify if the port channel is intact. Many channel problems can be identified by troubleshooting just one port because the problems will affect all ports on the channel. Secondly, BERT instruments can be used on the high-speed digital interface to help isolate a problem to either a multiplexer or the remainder of the channel (the analog circuit) and modems.

Analog testing is used for the telephone line. However, with small tech control centers at remote sites, end-to-end testing can be performed on the central site to remote node section of the circuit. Then the remote tech control center will need to assume responsibility to troubleshoot the remote analog sections (remote node to final customer station). If the size of the network is not big enough to support test equipment and trained maintenance operators at remote nodes, the remote technical control center can patch the analog line through (jumper it around the high-speed modem, multiplexer, and end-section modem). This saves on test equipment, but requires that all ports on the multiplexer be taken out of service to troubleshoot a problem that was affecting only one port of the multiplexer. The network operator will have to evaluate the economics of increased equipment investment vs. decreased network availability.

CONCLUSION

There are many reasons why Network Managers are choosing to go with multivendor data communications systems. For whatever the reasons, the Network Manager will need to consider technical control centers and test equipment if network availability is critical in his system. By utilizing good test equipment in conjunction with good maintenance practices, the Network Manager can maintain high availability of his system.

8. Benefits of Analog Testing for Data Communications.

HEWLETT PACKARD

Thomas Graham is Engineering Section Manager for data communiction instrumentation at Delcon Division. Tom holds a B.S.E.E. from Minnesota, and M.S.E.E. from Stanford, and has completed a year of graduate work at MIT.

Donald Dresch has an extensive background in communication and sonar systems, and developed the nonlinear distortion measurement for the 4940A TIMS. Don earned both his B.S.E.E. and M.S.E.E. at Minnesota.

BENEFITS OF ANALOG TESTING FOR DATA COMMUNICATIONS

INTRODUCTION

Analog transmission is fundamental to most data communications systems, and careful, well planned analog testing can help keep datacom systems fundamentally sound. Analog testing involves some complications that don't arise in digital testing. There are more parameters to measure, and the parameters are sometimes not easily related to error rate and throughput.

The goal of the Datacom Manager is to keep the system up and operating at peak efficiency. Impairments that reduce efficiency must be identified and corrected. Circuit outages must be analyzed accurately to identify the fault and restore service as quickly as possible. Different strategies and test methods can be used to achieve this goal depending on the system function, datacom network and analog test equipment involved.

The purpose of this paper is to discuss analog impairments and their measurements and to help the Datacom Manager decide which test methods are most likely to lead to an early identification of the problem. This is the first step in restoring proper system operation.

DATA TRANSMISSION

Consider a simple link between a terminal and a computer. Digital pulses are passed between the two via an interconnecting cable. This is no problem, but when we extend the distance between the two pieces of equipment beyond about 50 feet, the received data pulses are degraded, and the receiving equipment begins to have difficulty distinguishing a one from a zero. Transmitting data over long distances at an acceptably low error rate requires two things: an adequate transmission path, or link, and the proper equipment to match the data signal to the link. The solution is obvious: there exists a huge network of telephone lines and facilities that can connect your location to almost any other place in the country, if not the world.

Some problems are encountered when the telephone network is used because it was originally designed to transmit voice signals, not data signals. The 300Hz to 3300Hz frequency range of a telephone channel will neither support the baseband nature of the data signal at the low end nor the fast rise times at the high end. The data signal must be adapted to match the voice frequency range of the channel.

The equipment that is used to adapt data signals to the telephone channel is called the MODEM. It performs basic functions that resolve the two problems just mentioned: 1) frequency translation (MOdulation) of the data signals at the transmitting end to remove the baseband. A reverse translation (DEModulation) at the receiving end returns the data signals to the baseband frequency range. 2) frequency spectrum limiting (filtering) to fit the 300 to 3300Hz range.

The three types of modulation used in data communications are amplitude modulation (AM), frequency shift keying (FSK), and phase shift keying (PSK). A combination of AM and PSK which is popular at higher bit rates is called quadrature amplitude modulation (QAM).

FSK modems are low cost, low speed asynchronous modems. They are relatively immune to most impairments, primarily because they are used at low bit rates. Most FSK modems are Bell compatible.

AM modems are inefficient in terms of bandwidth and signal power. Single sideband or vestigial sideband techniques increase the bandwidth efficiency but at higher price and increased sensitivity to impairments. AM modems are rarely used today.

PSK modems are used in medium speed applications from 2400 to 4800bps. They are synchronous modems; that is, data timing signals (bit clock) are recovered from the received signal. PSK modems are fairly immune to most impairments, but phase jitter and phase hits may be a problem.

QAM modems are used almost exclusively in high speed, synchronous data applications above 4800bps. QAM modems from different manufacturers are rarely compatible.

PSK and QAM modulation lend themselves to multi-level encoding which allows 2, 3 or 4 bits to be encoded into each signal element. In this manner, 9600bps transmission can utilize 16 different levels (4 bits for each signal level); the signaling rate in this case is 2400 per second. This conserves bandwidth but requires higher signal to noise ratios for acceptable error rates. As bit rates increase so does the susceptibility to all impairments.

There is a special class of MODEMS called the Limited Distance Modem, or LDM. LDM's are a particular type of modem that keep the transmitted data in the baseband frequency range, from DC to 30kHz or above, depending on the bit rate. Because the data is transmitted directly, no modulation of a carrier frequency takes place. Therefore, there must be a metallic connection between the transmitting and receiving ends.

Although a type of modulation or an encoding scheme can be chosen to provide protection against a single particular impairment, every transmission line has most impairments present at all times and almost always has several dominant impairments simultaneously. So it is impossible to select a particular modem that provides superior performance under all conditions. Generally speaking, the higher the bit rate, the higher the S/N ratio required to maintain the same error rate.

Because data sent at higher bit rates is more susceptible to impairments than data at lower bit rates, automatic equalizers are used in all high speed data sets and most medium speed sets. Automatic equalizers reduce the effects of envelope delay and limited bandwidth, and track slow changes in these parameters. But automatic equalizers do not eliminate your concern with line conditioning; they only reduce it.

IMPAIRMENTS

There are at least eleven impairments to data transmission over voice channels. These impairments are either steady state or transient in nature. They are listed here in roughly the order they are considered when troubleshooting a circuit.

- Loss
- Noise
- Impulse Noise (transient)
- Amplitude Distortion
- Envelope Delay Distortion
- Phase Jitter
- Intermodulation Distortion
- Phase Hits (transient)
- Gain Hits (transient)
- Drop Outs (transient)
- Frequency Shift

The first two on the list are almost always checked first to answer the question, "Do I have a good clean signal?" After that, the next parameter to be measured depends on the symptoms, the type of modem and the channel make-up.

Every transmission path has loss, but as long as it is within reasonable bounds the signal can be maintained at a usable level. If loss does not vary with time it is not a problem, but it sometimes degrades slowly until the signal is unusable. Or you may lose the circuit altogether.

Most carriers measure loss in voice channels at 1004Hz. The test signal is applied to the line at a level of about 0dBm using a 600 ohm signal generator for most private lines and 900 ohms for dialed lines. The exact level to use depends on the circuit and should be carefully checked before testing. At the receiving end the signal power into a 600 or 900 ohm termination is measured. The difference between the transmitted and received levels is the loss (or gain) in the total circuit. End-to-end loss on most data circuits is typically 16dB.

There are many phenomena that are measured as noise: cross-talk, echo, single frequency interference, quantization noise, thermal noise, power line pick-up, etc. Noise, like loss, is an omnipresent phenomenon.

To have any significance a noise measurement must specify the bandwidth over which the noise is measured and the frequency response (weighting) within that bandwidth. The C-Message filter is used to measure noise in voice channels in the North American Telephone systems. This is a bandpass filter that limits and shapes the frequency band to approximate the frequency response of the 500 type telephone in conjunction with the hearing of the average person. A second noise filter used for voice channels is the 3kHz flat filter. This is a simple low pass filter with a 3dB point at 3.0kHz; being low pass, the effects of power line pick up are included in this measurement.

The noise specification for 3002 voice channels is 50dBrnc maximum, where "dB" specifies that the number is in dB, the "rn" specifies that the 0dB reference is -90dBm, and the "c" specifies the C-Message filter (thus 50dBrnc is -90 + 50 or-40dBm).

Noise measurements can be made with several different equipment configurations. Differential noise across the pair of wires can be measured either with or without tone. To measure noise with tone, the transmitter sends a 1004Hz tone at the appropriate level. At the receiver this tone is notched out, and the C-Message noise is measured. This is called C-NOTCHED noise. The purpose of the tone is to keep telephone equipment operating with the same gain that is used when data signals are present. To measure noise without tone, the line is simply terminated at the transmitter end with a 600 or 900 ohm resistor.

Longitudinal noise, or noise to ground is measured with the transmitter end terminated as before but at the receiver end the noise on the two wires is summed and measured with respect to ground. Very high noise to ground readings are usually indicative of poor shielding or shield grounding of the telephone cable or close proximity to a strong noise generator, typically a power transmission line.

Impulse noise has many sources; the most common are switches, relays, electric motors, automobile ignition, lighting, etc. In voiceband circuits an impulse is counted when an excursion of band-limited noise (usually C-MESSAGE filtered) exceeds a preset threshold. The rate of counting is limited to 7 counts per second. Some impulse noise counters allow faster counting, but the tariffed impulse noise specification is based on the 7 count per second rate. The counting threshold is usually set about 68dBrnc. The exact value in dBrnc depends on the circuit.

Amplitude distortion results when loss is a function of frequency. This impairment is caused primarily by filters in carrier systems and to some degree by the telephone cable itself. In carrier systems, band pass filters are used to separate one channel from another. These filters introduce amplitude distortion at the upper and lower ends of the frequency band. Telephone cable, because it is predominantly capacitive, attenuates the signals at the upper end of the frequency band. Amplitude distortion is measured in the same manner as loss; the only difference is that the test signal frequency is varied, usually over a band of about 200 to 3400Hz. The amplitude distortion is the difference in level at any frequency compared to the level at 1004Hz.

In the past, envelope delay distortion (EDD) has been one of the worst impediments to fast, reliable data transmission. However, with the advent of automatic equalizers it has become less predominant, as long as it remains steady state and within the limits guaranteed by the channel specification. Envelope delay distortion, like amplitude distortion, is seen mostly at the edges of the frequency band. EDD occurs because the phase changes that a signal undergoes as it passes through a telephone channel is not proportional to its frequency. The result is that different frequency components of the signal arrive at the receiver at different times so that components of one pulse smear over into components of other pulses. This is referred to as intersymbol interference and can be corrected by circuitry that introduces phase compensation.

The measurement of EDD is probably the most complex of all impairment measurements. It is accomplished by comparing the phase of a test signal that is transmitted over the line being tested with the phase of a reference signal. In North America when measuring a four wire circuit, the pair of wires being tested carries the test signal, a variable frequency amplitude modulated with 83-1/3Hz, and the other pair of wires carries the reference signal, a fixed frequency amplitude modulated with 83-1/3Hz. The fixed reference frequency is usually chosen to be 1804Hz because this is near the frequency of minimum delay. The variable frequency is swept or stepped from about 200 to 3400Hz as in the amplitude distortion measurement. EDD is measured in microseconds and is the difference in the delay at one frequency compared to the delay at the reference frequency.

Phase jitter is another impairment that originates in carrier transmission. Noise interference can be converted to phase modulation (phase jitter) on the data signal as the signal is multiplexed and demultiplexed in the carrier system. The jitter frequencies are usually ringing (20Hz) and power line (60Hz) and their harmonics. Phase jitter is measured in degrees, peak-to-peak. Phase jitter is usually measured in a frequency band of 20 to 300Hz but in the last several years, low frequency phase jitter components in the region below 20H have been found to affect certain high speed modems.

Phase jitter is measured by examining the phase disturbances of a transmitted test tone of 1004Hz. This can be done qualitatively using a simple oscilloscope or quantitatively using a digital phase jitter meter.

Intermodulation distortion also referred to as nonlinear distortion is caused by a variety of equipment in the telephone network, notably compandors (COMpressor-exPANDOR) used in PCM systems or other active devices in the channel and is usually considered to be a problem only with bit rates of 4800bps and above. A harmonic distortion measurement may not give valid results. If several sources of distortion are present in a channel, but separated by a link with a particular phase shift characteristic, a harmonic distortion measurement may show no harmonic products are present because the phase shift can cause one source of distortion to cancel the other one. To overcome this problem the transmitted signal for the intermodulation distortion measurement is a set of 4 tones, two centered at 860Hz and separated by 6Hz and two centered at 1380Hz and separated by 16Hz. The measurement is a three step process. First, the 4 tones are measured. Second, the 2nd order distortion (centered at 520Hz and 2240Hz) is measured using narrow bandpass filters and referenced to the level of the 4 tones. Third, the third order distortion (centered at 1900Hz) is measured using another narrow bandpass filter and again referenced to the level of the 4 received tones.

Phase hits and gain hits are caused primarily by carrier systems; automatic channel switchover is an example. Selectable thresholds are provided for gain hits (2, 3 or 6dB) and phase hits (10^o to 45^o, in 5^o increments). Either transient must be at least 4ms long to be counted. The counting rate is limited to 7 counts per second. As with impulse noise, higher counting rates may be used, but the channel specification is based on 7 cps.

A dropout is defined by the Bell System as a 12dB or greater loss of signal for 4 milliseconds or more. Dropouts are caused by microwave fade, circuit disconnections or equipment failures. The count rate for dropouts is also limited to 7 counts per second.

Frequency shift or frequency offset is caused by transmission through a non-synchronous carrier system. If the demodulation frequency at the carrier receiver is different from the modulation frequency at the carrier transmitter, the entire received frequency spectrum is offset or shifted by the difference. Nearly all carrier systems today are synchronous, the modulation and demodulation frequencies are derived from a single source. In rural areas some non-synchronous systems still exist but it is rare that the frequency offset exceeds 2 or 3Hz, and most modems are unimpaired by this small amount.

Frequency shift can be measured using a stable signal generator and a frequency counter. It is simply the difference between the transmitted and received frequencies.

TROUBLESHOOTING METHODS AND BENEFITS

In the past the Datacom Manager was reluctant to do analog testing. The argument was, "This modem is specified by the manufacturer to operate over a C2 line, and I've got a C2 line so why should I have to test it?" The answer became painfully obvious the first time the system failed and the Datacom Manager was told by the telephone representative, the modem representative and the terminal equipment representative (usually each from a different company), "There's nothing wrong with our equipment, it must be the other guy's". To test or not to test is no longer the question, rather what should be tested and when?

There are two philosophies that are followed: one is periodic testing or preventative maintenance, the other is demand testing or testing only when a failure occurs. Each has its merits and is influenced somewhat by the particular system and its application.

In either philosophy, it is critically important to record the analog test results on each line prior to its being put into service and immediately following any major network change or repair. The greater variety of analog tests made, the more valuable the benchmarks or comparison points obtained for future preventative maintenance or demand testing.

These benchmarks should include measurements on each transmission path, with end-to-end as well as loopback test results. Loopback tests can be easily implemented by a single operator at the control center, and although they may mask the parameters of each individual link, loop back benchmarks can serve as powerful tools to isolate problems. End-to-end benchmarks are needed to aid in nailing down the offending link after the fault has been isolated to a suspected area.

Preventative maintenance coupled with good record keeping practices can provide these benefits:

1) Early problem detection - before it gets painful.
2) Measurements are made with no duress - fewer human errors are introduced.
3) Records are updated - knowing the time interval when problems recur can be a valuable clue for the troubleshooter.
4) Personnel do not have to relearn the equipment - this is especially valuable when a service outage has occurred and the operator is under stress and more prone to error.

5) Shorter repair times - if you can identify the faulty parameter or tell which parameters changed, the telephone company can restore service more quickly.

6) It enhances your credibility with telephone personnel - if they know you are keeping tabs on their system on a regular basis, they are more likely to believe you when you call in for service.

The disadvantages of preventative maintenance in analog testing are:

1) Analog testing is intrusive testing - data transmission must be interrupted.
2) Inevitably as more human interaction with the system takes place, the chances of human error are increased.
3) Preventative maintenance analog testing is time consuming and therefore relatively expensive in the short run.

The advantages and disadvantages of demand testing, on the other hand, are pretty much the converse of those for preventative maintenance. Generally, what you lose with one, you gain with the other.

If the datacom system has built in back-up capability the argument in favor of demand testing is strengthened. Another aspect is that of the tolerance to an increased error rate or a service outage. A low tolerance might dictate that a preventative maintenance program be followed.

Eventually, the system will fail, and at some point in the troubleshooting process, analog testing will be required. A recommended series of steps to follow when performing either loop around or end-to-end testing is:

1) Establish signal continuity - measure 1004Hz. Is the received level correct? If your test set is equipped with a monitor amplifier and speakers, listen carefully for interfering tones, distortion, hum, or high noise levels. Most Datacom managers will agree that loss of signal continuity is the most frequently occurring problem. This is a convenient time to check the signal level at the band edges, 404Hz and 2804Hz, to get a quick estimate of the channel bandwidth.
2) Measure signal-to-noise ratio. With some test sets, this is a single step process; signal and noise are measured and the ratio is calculated automatically. With other sets, you must measure signal and noise levels separately, then calculate the ratio.

3) Check the line quality - perform a P/AR measurement. Although P/AR is not a tariffed measurement, it provides a good, fast check on several parameters simultaneously. P/AR is primarily a measure of intersymbol interference and is therefore most sensitive to envelope delay distortion; however, P/AR also responds to attenuation distortion, severe noise and high levels of intermodulation distortion. A very low P/AR reading (45 P/AR Units) or a change in the P/AR reading of 12 units or more, in either direction is a clue that more detailed testing is required.

4) Measure additional parameters as indicated by the nature of the problem. For example, a high error rate on a 9600bps circuit that includes PCM carrier in the channel make-up indicates that intermodulation distortion should be checked. Or, any link showing error bursts should be checked for transient impairments.

Some analog problems can be difficult to identify. Obviously the more insidious ones are those that are either soft failures or sporadic, and the worst problems are those that are both soft and sporadic. With these types of problems, a preventative maintenance program coupled with good record keeping is your best defense and gives you the best chance of early identification and protection. The longer the history over which these records are kept, the better established are the bounds for these impairments and the better the "fingerprint" for each section of your system.

Analog testing can be made easier and more effective through the use of two innovations: Master/Slave operation and desk top calculator control of the test sets. Master/Slave provides remote operation with nothing more than the properly equipped test sets. At the flick of a front panel switch an analog test set can become a master control unit or a remote slave unit. Only one qualified test person is needed to do detailed end-to-end testing. Once the test set is connected to the far end of the circuit and the switch set to SLAVE, all tests are controlled from the master test set, and all results are displayed there also.

Further benefits can be derived by operating the test equipment under the control of a desk top calculator. This type of operation is possible in test sets that provide the IEEE 488 option. With this option, analog tests can be automatically sequenced and all test results formatted and stored as desired.

CONCLUSION

Analog testing is an important phase of an overall network maintenance and management program. Data transmission channels cannot be replaced as quickly as other components of the network, therefore, faulty channels must be identified and repaired as quickly as possible to maintain system integrity. Analog testing is just as successful when it proves that a suspected line is not the problem as when it proves that it is.

Benchmark testing is a very important part of an analog testing strategy. Even if benchmarks do not presently exist, they can be established by analog measurements on lines that are known to be trouble free.

The optimum solution to each particular system problem may vary greatly depending on system use, network topology, symptoms, etc., and it is up to the Datacom Manager to know when and where and which tests and which methods of testing to apply in each particular case.

The trend for data communication systems has been, and will continue to be, toward larger, more complex and sophisticated systems with higher bit rates and fewer errors. This in turn increases the need for more powerful and more efficient analog testing capability. Analog test equipment should be accurate, reliable and easy to use.. Master/Slave operation can benefit the Datacom Manager by more efficient utilization of personnel, less chance for error and easier record keeping.

The economic trade-off in establishing a preventative maintenance program is the cost of the analog testing and analog test equipment versus the cost of lower system availability or efficiency.

Appendix

Table 1

Table 2

TABLE I Tariffed Parameters.

	Non-Conditioned 3002 Channel		With C1 Conditioning		With C2 Conditioning		With C4 Conditioning	
Frequency Range in Hertz (Hz)	300-3000						300-3200	
Attenuation Distoration (Net Loss at 1000 Hz)	Frequency ,Range	Decibel Variation	Frequency Response	Decibel Variation	Frequency Response	Decibel Variation	Frequency Response	Decibel Variation
	300-3000	−3 to +12	300-2700	−2 to +6	300-3000	−2 to +6	300-3200	−2 to +6
	500-2500	−2 to +8	1000-2400	−1 to +3	500-2800	−1 to +3	500-3000	−2 to +3
			300-3000	−3 to +12				

Delay Distortion in
Microseconds (μs)
	Less than 1750 μs
	from 800 to 2600 Hz.
			Less than 1000 μs
			from 1000 to 2400 Hz.
			Less than 1750 μs
			from 800 to 2600 Hz.
					Less than 500 μs
					from 1000 to 2600 Hz.
					Less than 1500 μs
					from 600 to 2600 Hz.
					Less than 3000 μs
					from 500 to 2800 Hz.
							Less than 300 μs
							from 1000 to 2600 Hz.
							Less than 500 μs
							from 800 to 2800 Hz.
							Less than 1500 μs
							from 600 to 3000 Hz.
							Less than 3000 μs
							from 500 to 3000 Hz.	

			With D Conditioning					
Signal to Noise (dB)	24		28					
Non-Linear Distortion Signal to 2nd Harmonic	25		35					
Signal to 3rd Harmonic	30		40					

TABLE 2 Non-tarrifed parameter limits.

Parameter	Specification	
1. C-message noise	Facility miles	Maximum noise at modem receiver (assumes standard design channel)
	0 50	28 dBrnc
	51 100	31
	101 400	34
	401 1000	38
	1001 1500	40
	1501 2500	42
	2501 4000	44
	4001 8000	47
	8001 16000	50
2. C-message notched noise	At least 24 dB below received 1004 Hz test tone power	
3. Impulse noise	15 or less counts in 15 minutes with a threshold set 6 dB below level of received 1004 Hz test tone	
4. Single frequency interference	At least 3 dB below C-message noise limits	
5. Frequency shift	No more than ± 5 Hz	
6. Phase jitter	20-300 Hz, 10° maximum peak to peak; 4-300 Hz, 15° maximum peak to peak	
7. Peak to average ratio (P/AR)	Greater than or equal to 48 P/AR units	
8. Phase hits	8 or less hits in 15 minutes of greater than 20°	
9. Gain hits	8 or less hits in 15 minutes of greater than 3 dB	
10. Dropouts	1 or less dropouts in 30 minutes of greater than 12 dB	

Signature Analysis for Board Testing

John R. Humphrey
Loveland Instrument Division, Hewlett-Packard

INTRODUCTION

When Hewlett-Packard developed the Model 5004A Signature Analyzer[1], the objective was to provide a way to substantially reduce repair costs on microprocessor and ROM based products. The board-exchange approach commonly adopted for field service support of such products has a number of economic drawbacks. Signature Analysis was viewed to be a viable, component level repair alternative for LSI circuit-based equipment that could stand a few hours of repair downtime. During the past three years, Hewlett-Packard has introduced more than 50 major products which include provision for testing with Signature Analysis. Furthermore, by early 1979, more than half of the top 100 U.S. electronic companies (by sales volume) had invested in Signature Analysis as a measurement tool.

Implementation of Signature Analysis (SA) capability into an automated test system with bed-of-nails circuit visibility extends the practicality and power of the technique from the realm of field service of LSI-based products to that of production testing of individual circuit boards. The stimulation, measurement, and fault-isolation aspects of SA within an automated test system environment are discussed. Examples of SA testing of representative memory and microprocessor boards are presented with sample stimulus approaches, tester interconnection and programming requirements, and troubleshooting strategies.

LSI BOARD TESTING - PROBLEMS AND SOLUTIONS

Considerable space has been allocated within the spectrum of literature directed toward test problems to the challenges posed by boards designed around LSI technology.[2] [3] [4] [5] [6]

Chip complexity has reached the point that a single microprocessor may contain more gates than a 50 to 100 IC board of MSI devices. The board on which the processor is placed may contain several other chips of LSI complexity as well as the original 50 MSI devices. Such complexity makes the task of generating effective test sequences more difficult. The simulator-aided programming techniques which have been successfully applied to MSI technology may also be used for LSI boards. However, modeling LSI devices may be very difficult and time consuming.

For simulators which utilize algorithmic pattern generation, complex circuits modeled in terms of nand gate equivalence can easily exceed the 4K to 20K gate capacity of the simulator-even if the detailed, gate level specifications of the LSI chips in question are available from the manufacturer. An approach which reduces the severity of the chip complexity problem is to employ a simulator which works with a functional rather than gate level model of complex LSI devices. Functional modeling reflects overall data transfer within the device based upon operational data published by manufacturers. In any case, modeling of LSI devices is not a trivial task and tester manufacturers have not updated libraries at pace with the development of new LSI chips.

After the modeling of the LSI devices is complete, a simulator will model the board and help develop a pattern set for testing the board. The task of pattern development typically encompasses several steps involving both automatic and semi-automatic operations. Manually generated inputs may be required to account, for example, for sequential functional requirements.

The next problem is stimulation and detection of digital activity on the board as specified by the simulator-generated data. Boards containing LSI typically exhibit greater susceptibility to timing related parameters. Furthermore, many such devices function only at dynamic conditions and cannot be statically exercised. This means that an LSI board tester must work at rates corresponding to the sub 1μsec cycle level. Functional testers currently on the market may offer high speed pin options to meet this requirement. Such options are generally configured around a specialized controller with RAM memory at each (high speed) driver/sensor pin and software considerations for loading RAM from and dumping RAM data to mass storage between test sequences. Depending upon the degree of sophistication of both tester hardware and software, as well as cost, dynamic test capabilities of commercially available systems may include such features as:[6]

- The capability to change pin function between driver/sensor modes during program execution without reducing system test rates.
- Capability to synchronize the tester to different clock phases on the board.

The need for such tools must be considered in view of tester cost and programming requirements to achieve a level of fault coverage. The objective of testing is optimization of fault coverage and fault isolation based upon customer satisfaction and warranty cost parameters factored by production throughput and production cost.

The question of fault coverage depends upon the adequacy of the functional test sequence in exciting the various fault modes of the board under test. Simulators normally provide a calculation which relates faults detected by the test sequence to the total number of faults modeled for a particular board. Because LSI devices are complex, board functions are more complex and it is more difficult to evaluate the quality of an LSI board test program. Functional modeling techniques or the use of comparison chips in a reference-system tester[7] provide a hardware model, but they model the correct operation of the LSI chip and do not predict faults. The use of other test tools, such as in-circuit inspection for shorts and opens in combination with digital functional tools may significantly enhance tester fault coverage capabilities. In the final analysis, test quality is measured and test approaches are tuned on the basis of customer feedback.

Fault isolation on many commercial testers is based upon guided probe and IC clip techniques. The structural characteristics of LSI boards as related to bi-directional busses and specialized I/O devices can make diagnosis of faults to the component level difficult. Although a given node may be detected faulty, the problem of determining which of five to ten IC's connected to the bad node is generating the fault is significant. For a bi-directional bus structure, inputs to one device become outputs for another and the probing problem becomes one of resolving a feedback loop condition. Furthermore, because a guided probe method must examine all inputs associated with several devices which are connected to a failing node, it generally takes longer to diagnose a fault on a LSI board.

LSI Boards feature extraordinary functional density. These are simply too many functions, too many states, and too much memory to test the entire spectrum of design operation. Furthermore, there is the problem of visibility. As chip complexity has increased, visibility has decreased. Functional capacity has increased, but the number of edge pins on boards has decreased in proportion to functionality.

Complexity, at-speed test requirements, visibility, effective fault coverage, fault isolation, tester cost effectiveness - these are the problems associated with testing today's LSI boards. Yet circuit designs are already evolving from LSI to VLSI technology. Testing may very well become the major cost associated with the use of these technologies. In line with the emphasis on programming support and peripheral chip development which were required to make circuit design with LSI practical, designed in testability will have to be seriously addressed both by the chip manufacturers and end-product designers.

Signature Analysis is one proven method to help solve these problems in a cost-effective manner and it offers alternatives to simulation, RAM backed test pins and dynamic reference test systems.

SIGNATURE ANALYSIS - TECHNICAL DETAILS AND CHARACTERISTICS

SA is conceptually simple. It is a synchronous process, whereby activity at an electrical node, referenced to a clock signal, is monitored for a particular stimulus condition during a measurement time period. The result of the SA nodal monitoring process is based upon a unique data compression technique which reduces long, complex data stream patterns into a 16 bit, 4 digit "signature." Correct signatures for a particular circuit are determined empirically from a known good product. Testing is performed by probing interdependent nodes to determine the functional origin of bad signatures.

Stimulus Considerations

The original Signature Analyzer manufactured by Hewlett-Packard requires that the stimulus for SA testing be generated by the product being tested. For many boards, the most effective stimulus is indeed derived internally, within the framework of "designed-in" SA testability. However, when considering SA with respect to board rather than full product testing and in conjunction with an automated test system rather than a portable field service capability, it may be desirable, practical, and even necessary that stimulus be generated as part of the test environment.

SA testing relys upon the principle of "exercising" circuit nodes - changing logic nodes from one state to the other - stimulating the various fault conditions (stuck-low, stuck-high, pin faults, etc.) that may exist for a particular circuit. For combinational circuits it may not even matter "what" specifically causes a node to be exercised to provide useful diagnostic information. In fact, psuedo-random pattern stimulation may be effectively applied in conjunction with SA testing of combinational logic. However, for circuits such as microprocessor-based controllers, sequential operational characteristics may reduce the validity and effectiveness of random stimulation.

One approach to LSI board stimulation for SA testing is to "free-run" the board. Free-running, for purposes of SA, involves getting the circuit to run in a repetitive loop with only a minimum number of the circuits logic elements required to control the process and causing the maximum number of logic nodes to be exercised. In the case of microprocessors, controllers, sequencers, and algorithmic state machines, free running is often accomplished by opening the data (or instruction) input bus and forcing in an instruction or control that causes a continuous cycling through the entire address or control field. The circuitry performing this cycling function is referred to as the kernel and is functionally the "heart" of the system. Taking signatures while in the free-running mode can verify the kernel, much of the combinational circuitry associated with address and control functions (especially address decoders and ROM data), and the operation of the data bus. Free-running alone may not sufficiently exercise all devices and circuit nodes.

To test board functions not exercised by free-running, specific test algorithms must be generated to emulate the functional modes of the board's applications environment. The algorithms applied must also be designed to make fault isolation as effective as possible. Often the test algorithm may incorporate subroutines used in normal application of the product or as part of a self test procedure. In the case of a microprocessor board, the SA stimulus algorithms are written into a portion of on-board or externally connected ROM (as part of the auto test fixture, for example). Stimulus for the test is thus generated internally which is easier than trying to stimulate devices surrounding a processor from the board's edge connector I/O ports. Furthermore, the intrinsic data manipulative capabilities of microprocessors make this a powerful stimulus control approach, yet a relatively straightforward task to implement in the design.

Consider, for example, the stimulation of peripheral I/O devices that accept inputs from a processor and output data to another board. A short program can be written to increment an accumulator in the processor and output the data to all devices. Signatures can be taken on each output line to test not only the devices that are connected to the processor but also the processor itself for several specific instructions (increment accumulator, test accumulator, jump, etc.).

Signature Measurement

The process by which the signature of a data stream is measured is controlled by three signals. A START signal is used to trigger the beginning of a signature measurement time interval. The CLOCK signal synchronizes the SA data sampling circuitry to signals to be monitored on a test node. The STOP signal is used to trigger the end of a signature measurement time interval.

The START and STOP signals which define the data sampling interval can be taken from address lines, state pointers, software controlled output ports or any other signals that identify the presence of a unique data stream. A sophisticated tester may allow the measurement to be made on the basis of a START signal and a specific number of clock pulses following START. Special circuitry may also be designed into the production test fixture to generate measurement window signals.

As with SA stimulus discussed previously, SA was originally designed to allow the unit under test to run at speed by its own clock with the tester synchronized to that clock. In the production test environment, it is likely that some boards will be tested independently of other boards which normally supply such signals as a clock. In such a case, the tester may be utilized to generate the clock required to make the board function. The SA measurement process is likewise synchronized to the tester-generated clock. The test fixture can be designed to furnish such signals as a clock as well.

When utilizing SA for field service, both the CLOCK and START/STOP signals are generally derived from circuit test points which are not changed through many or all signature data probings. In such a case, with the START/STOP and stimulus running repetitively, the fault isolation process is a matter of physically moving the Signature Analyzer input probe from node to node on the circuit under test - searching for components with good signatures on the input but bad signatures on the output. The test system implementation of SA can provide not only automated multiplexing of the data input connections but also of START, STOP, and CLOCK connections. This capability in conjunction with properly designed stimulus signals can simplify the fault isolation process since specific measurement windows can help isolate devices functionally. Figure 1 illustrates the timing involved in a signature measurement window.

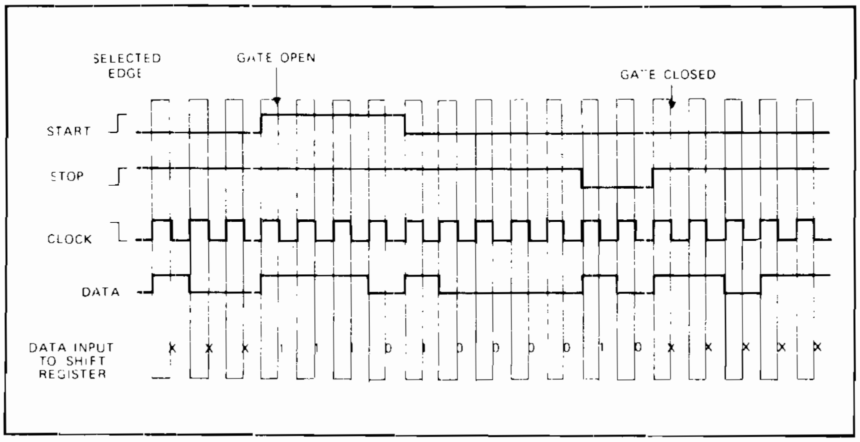

SELECTED EDGE
GATE OPEN
GATE CLOSED
START
STOP
CLOCK
DATA
DATA INPUT TO SHIFT REGISTER
X X X 1 1 1 0 1 0 0 0 0 1 0 X X X X X

Figure 1. Typical data input to the Signature Analyzer illustrating the timing relationships between the control signals and the measured data stream.

Processing of the nodal activity measured by SA is accomplished using a compression technique known as cyclic redundancy check (CRC) which has been commonly used in the communication's industry for error checking.[8] Compression of potentially long streams of data into a unique 16-bit result (the signature) obviates the need for RAM and mass storage for performing dynamic tests. This reduces usage complexity and tester cost.

Signatures are captured using an n-bit linear shift register with multiple feedback taps that are modulo-2 summed with the input data as shown in Figure 2. Feedback is selected so that the shift register produces deterministic and maximal length bit sequences. The deterministic characteristic refers to the fact that if the shift register is initialized with a particular word and shifted using a particular bit sequence, the remaining residual word (signature) is always the same. For a shift register with feedback that is characterized by the maximal length property, any bit stream that is shifted into the register will exhibit all 2^n-1 possible states before repeating a state. (See reference 9 for a discussion of shift register properties.)

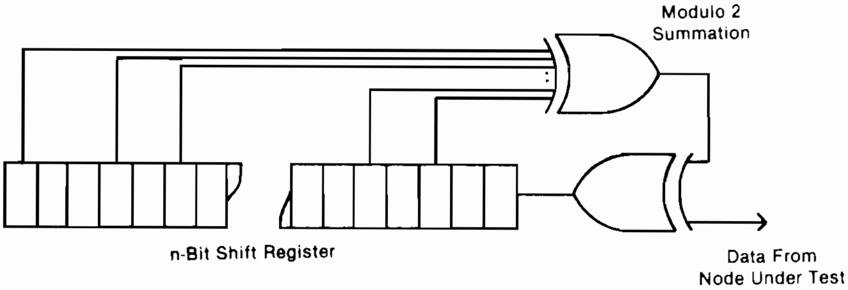


Figure 2. A linear shift register with feedback is utilized to [illegible] the [illegible] data stream

The selection of shift register feedback taps to establish these properties is not unique and various choices for different length shift registers are described in reference 9. For SA, a 16-bit register with feedback from the 7th, 9th, 12th, and the 16th bits has been selected.

This CRC implementation yields a 100 percent detection of all single-bit errors in a bit stream and 99.998 percent detection of multiple faults in a data stream regardless of length.[9] Note that these percentages indicate probabilities of detecting faults that may exist in the data under test. The effectiveness of the fault coverage is a function of the stimulus pattern and propagation of faults to a measurement node.

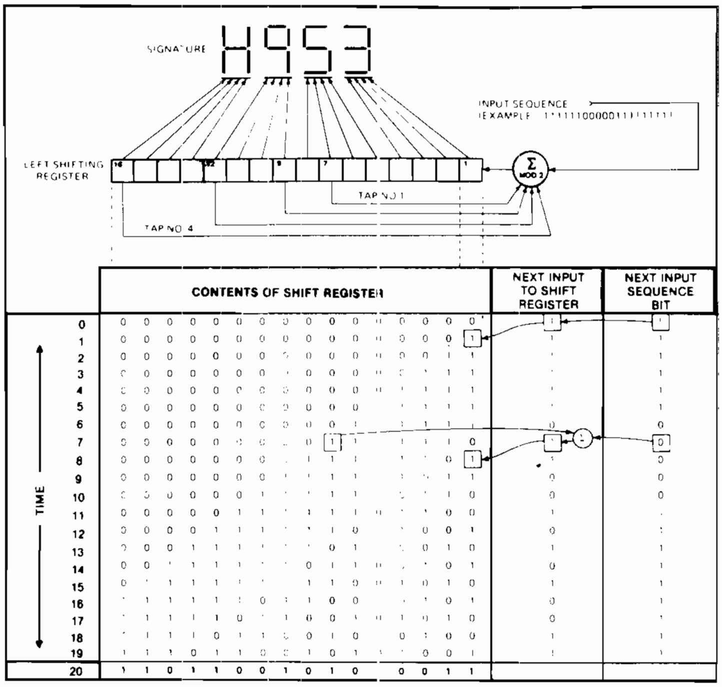
The 16-bit shift register residue from a data stream measurement is displayed and processed in a hexadecimal format. This representation of the result is what is referred to as the "signature" of the measured bit stream. HP Signature Analysis products utilize a nonstandard hexadecimal character set (0123456789ACFHPU) which was chosen for easy readability and compatibility with 7-segment displays.

The table in Figure 3 shows how a signature is generated from the 20-bit sequence 11111100000111111111. Initially (time 0 through 7) the register acts merely as a shift register. At time 7, the first 1 of the input sequence has reached the first feedback tap (tap 1, Figure 3). It is fed back and mixed with the input 0, with the result that a 1, not a 0, is next clocked into the register (time 8). This behavior continues until the end of the measurement when a residue of 16 bits, 1101100101010011 (time 20), is all that is left from the 20-bit input sequence. (Note the total dissimilarity in appearance between this residue and the original 11111100000111111111 input sequence.) This residue is displayed in hexadecimal format as H953, the signature of the 20-bit sequence.[10]

Fault Isolation With Signature Analysis

Signature Analysis is a nodal analysis technique. As such, it is very effective at yielding pass/fail decisions. In fact, SA can help minimize the cost of testing good circuitry. This is due to the fact that a single signature measurement at one node can accurately reflect the correct or incorrect operation of a logic structure consisting of a large number of devices and many nodes. This is dependent upon effective fault stimulation of the various nodes feeding the cardinal measurement nodes. Examples will be presented later to show how this may often be accomplished in a straightforward manner.

In the event of a nodal signature fault, the effectiveness of using SA to isolate the device(s) causing the fault is dependent upon the logic



	CONTENTS OF SHIFT REGISTER																NEXT INPUT TO SHIFT REGISTER	NEXT INPUT SEQUENCE BIT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0	0	0	0		1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0	1		1	1	1	1	0	0
7	0	0	0	0	0	0	0	0	0	1	1		1	1	1	0	1	0
8	0	0	0	0	0	0	0	0	1	1	1		1	1	0	1	1	0
9	0	0	0	0	0	0	0	1	1	1	1		0	1	1	1	0	0
10	0	0	0	0	0	0	1	1	1	1	1		1	1	1	0	0	0
11	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0	0	1	1
12	0	0	0	0	1	1	1	1	1	1	0		1	0	0	1	0	1
13	0	0	0	1	1	1	1	1	1	0	1		0	0	1	0	1	1
14	0	0	1	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1
15	0	1	1	1	1	1	1		1	1	0	0	1	0	1	0	1	1
16	1	1	1	1	1	1	0	1	1	0	0		0	1	0	1	0	1
17	1	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1
18	1	1	1	1	0	1	1	0	0	1	0		0	1	0	0	1	1
19	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	1	1
20	1	1	0	1	1	0	0	1	0	1	0		0	0	1	1		

Figure 3. The table shows how a 20-bit input sequence is processed to a four digit (modified) hexadecimal signature.

structure of the node itself factored by testability features designed into the board. In the ATE system test environment, fault isolation effectiveness also depends upon test system hardware and software tools.

For combinational logic, without feedback, fault isolation is largely a simple comparison of the measured and known good signatures for the board. If a bad signature is found, the signatures of lower order nodes are checked until a component can be located with good signatures on the input but bad signatures on the output. For an ATE system, with either bed-of-nails or guided probe visibility, the tracing algorithm may be easily built into lookup tables which reflect circuit topology. Tracing may be based upon binary half-splitting, inside-out checking of all signatures, or by straight back-tracing. These methods of fault probing are illustrated in Figure 4.

Isolation of faults to the component level on circuit boards which employ feedback connections is dependent upon hardware and software capabilities for breaking the feedback paths. This consideration applies both to the test system and the board under test. For example, the most common loop associated with microprocessor-based boards consisting of the processor, address bus, memory elements, and data bus may be broken by including a data bus jumper plug capability within the design. During production test, the board may be tested before the jumper is installed. After the kernel is tested using a free-run stimulus as described earlier, test system relays can be closed to emulate the normal "jumper-in" data bus connections. Then, additional test cycles in which typical operations of the board are verified may be run.

Feedback could also be disabled electrically by designing the board with buffers to tristate selected paths upon command from the test system. This approach may dictate extra hardware cost for the product and the need to test the states of the devices in question. However, significant return may be realized on such testability investments.

Feedback loop failures may also be analyzed by measuring signatures during periods when the feedback signal(s) take on constant zero or one states. This may be accomplished by designing stimulation software so as to provide sequences in which feedback signals are constant. In some cases, IC's may permit shorting to ground for the brief period in which signatures are measured in order to set feedback paths to zero.

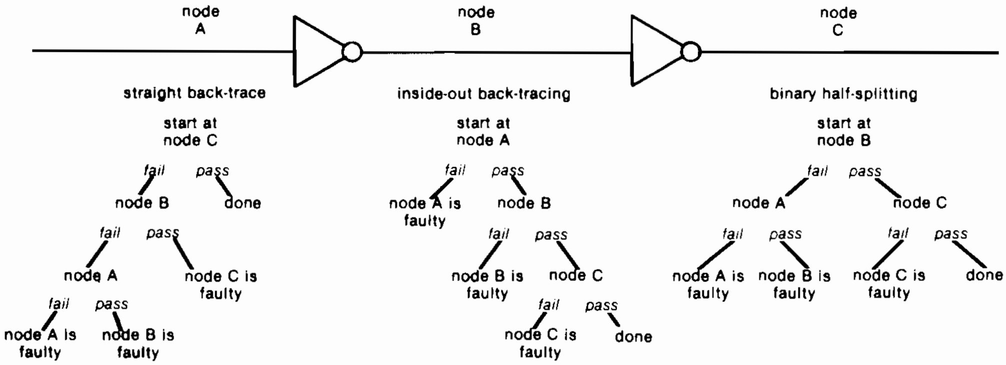


Figure 4 Methods of fault probing using Signature Analysis.

Within a bed-of-nails test environment, the capability to select START, STOP, and CLOCK signals automatically from anywhere on the board and to perform measurements using START and CLOCK pulse count can contribute significantly to effective fault isolation. Use of such tools is illustrated with the examples in the following section.

SA IN PRODUCTION TEST ATE

In this section, the general concepts of SA testing which have been presented will be related to specific test problems and their solutions. To accomplish this, hardware and software capabilities of an automated test system currently on the market will be described. These capabilities will then be applied to two representative LSI board test problems.

On the sample test system, SA is featured as an at-speed digital functional test technique.[11] The system is utilized in conjunction with bed-of-nails type fixturing and provides a full spectrum of test capability. Shorts/opens, in-circuit component tests, static digital tests, and considerable general purpose analog/hybrid functional test in addition to SA may be performed with this ATE. It is important to note that the objective of thorough test of a board is, in general, best accomplished by taking advantage of all of these capabilities in an integrated test sequence. It is best to resolve passive short/open conditions prior to application of power for functional testing. Discrete component fault and loading errors are best diagnosed using passive state in-circuit testing. Static digital test tools are useful for checking for simple active logic faults and for controlling signals during SA-based dynamic functional testing.

The hardware components of the sample test system typically utilized for SA testing are represented by the simplified block diagram of Figure 5. General system multiplexing is implemented in a row-column matrix configuration in which any individual test point may be switched to none, one, or any combination of signal busses in a seven bus multiplexing scheme These busses may be programmed for use with both in-circuit and functional testing as required. For example, the bus structure can be configured to route any one of up to approximately 500 nodes into the SA data stream processor. In addition, columns or individual relays may be configured to multiplex START, STOP, and CLOCK signals.

The signature analysis hardware option for the system includes additional multiplexing dedicated to SA as represented in the large block of Figure 5. Forty low capacitance test pins and four each START, STOP, and CLOCK

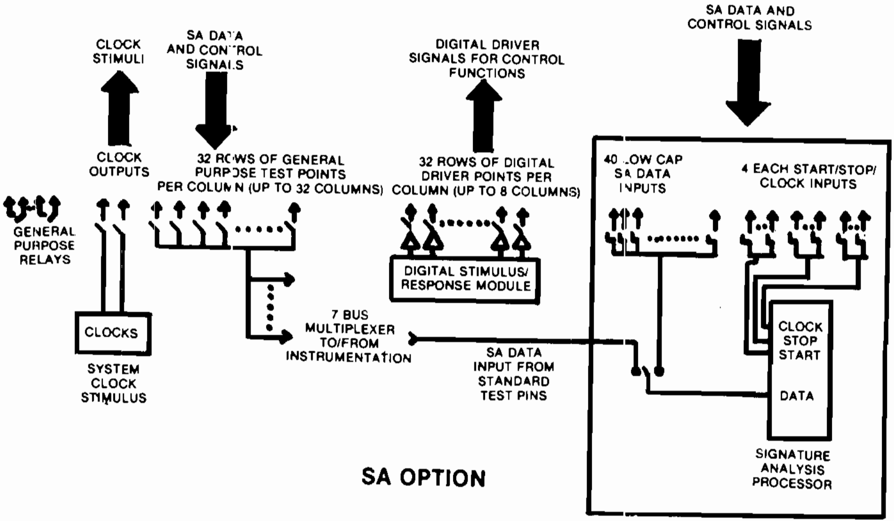

CLOCK STIMULI
SA DATA AND CONTROL SIGNALS
DIGITAL DRIVER SIGNALS FOR CONTROL FUNCTIONS
SA DATA AND CONTROL SIGNALS
CLOCK OUTPUTS
32 ROWS OF GENERAL PURPOSE TEST POINTS PER COLUMN (UP TO 32 COLUMNS)
32 ROWS OF DIGITAL DRIVER POINTS PER COLUMN (UP TO 8 COLUMNS)
40 LOW CAP SA DATA INPUTS
4 EACH START/STOP/ CLOCK INPUTS
GENERAL PURPOSE RELAYS
DIGITAL STIMULUS/ RESPONSE MODULE
CLOCKS
7 BUS MULTIPLEXER TO/FROM INSTRUMENTATION
SA DATA INPUT FROM STANDARD TEST PINS
CLOCK STOP START
DATA
SYSTEM CLOCK STIMULUS
SIGNATURE ANALYSIS PROCESSOR
SA OPTION

Figure 5. One example of the hardware implementation of Signature Analysis test capability in a commercially available ATE system.

inputs are provided. Note that the cost of the SA option is on the order of $100 per pin for the dedicated pins with no additional cost for utilizing standard in-circuit test pins for SA testing.

With respect to software, the sample test system features a BASIC-like interpreter structure enhanced by approximately 40 programming statements designed especially for board test programming. Loops, variable manipulation and branching typical of BASIC and FORTRAN processor-based systems may be applied freely with the special board test language (BTL). For SA testing, six of the BTL statements are commonly used. Of the six, two are statements dedicated to SA processing while the remaining four are useful in several of the available system test modes and are general purpose in nature.

The first of the two SA statements "saset", is used to select input pins for START, STOP, and CLOCK, and to set up processing on the basis of rising or falling edge conditions of these signals. The second SA statement, "sig", is used to measure signatures and its parameters include provision for learning signatures (from the reference assembly) or for measurement of signatures from the board under test with or without automatic fault message print out. Whether a signature is measured on the basis of a START/STOP combination or START and a specified number of clock pulses is also controlled by "sig" statement parameters. The SA processor can be set up to test on the basis of a repetitive stimulus or a single cycle stimulus sequence with "sig".

The remaining four BTL statements which will be utilized to illustrate automated SA testing include a test pin multiplex statement (mcon), a statement for generating a clock signal (clock), a statement for setting digital levels (apply), and a statement for programming the logic "1" and "0" voltage levels for SA signal processing. All of the BTL statements used in following illustrations appear in Figure 6 with a description of their programmable functional parameters.

Testing a Memory Board

Consider the problem of testing the memory board shown in Figure 7. The board consists of both ROM's and RAM's and the data and address busses are accessible via a front panel connector. Each memory element is selected via a chip enable line which is decoded from address bus data.

Statement	Purpose/Description
saset	Select START, STOP, CLOCK input connections and whether signatures are to be taken on rising or falling edges of these control signals. Initializes SA processor. saset 1, "R", 3 ,"F", 4, "F" CLOCK input #1 - rising edge START input #3 - falling edge STOP input #4 - falling edge
sig	Measure a signature. sig "Ul-pin 6", "3961", 32767, S, S$[7] Fault Message / Expected Signature / Number of clocks after START. / Variable to which measurement code is returned. / Alphanumeric variable to which measured signature is returned.
mcon	Close multiplexer test point relay. mcon N, M Close relays N and M where N, M are row/column positions given by X.$R_1R_2C_1C_2$ X = Node; R_1R_2 = Row Number; C_1C_2 = Column Number
apply	Set digital driver patterns. apply "INTRPT" 3, 0, 3 or apply "INTRPT", "11", "00", "11" Set the two-bit digital control signal named "INTRPT" high, low, then high again.
rcv ref	Set SA or static digital reference voltage levels. rcv ref 1, 0.2, 2, 3.2 Set SA reference voltages for: ≤0.2V for low; ≥3.2V for high
clock	Set up digital clock stimulus. clock 1, 1e6, 1e5 Set up clock number 1 to output a 1 MHz signal for 10,000 pulses. clock output / frequency / Number of pulses or free-run

Figure 6. An ATE system with software designed for the particular problem of circuit testing can make the test programming job easier. The statements shown are used to program SA functional testing in one commercially available ATE system.

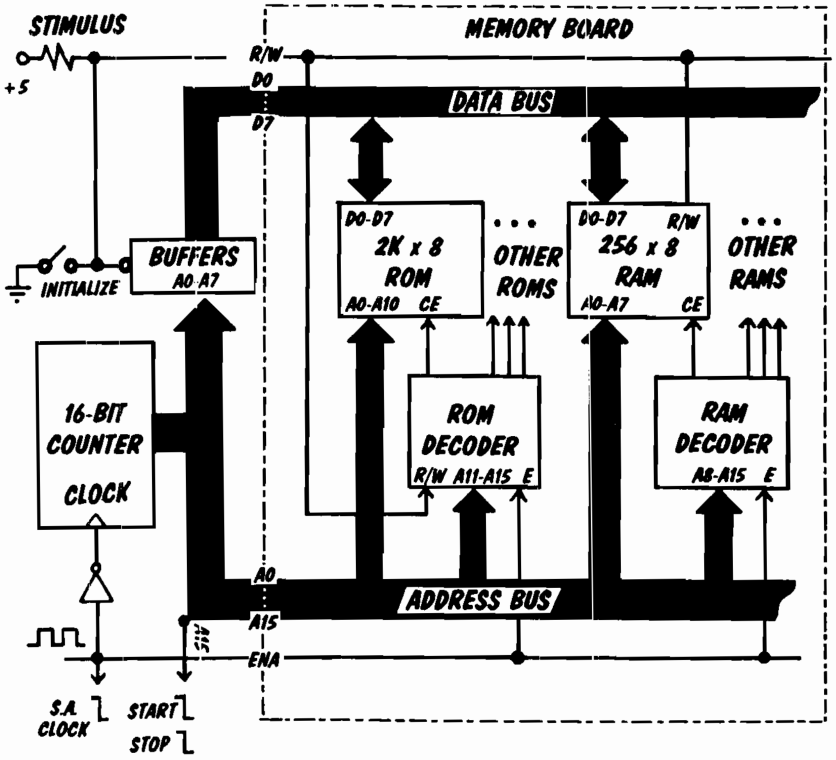
Two control lines are used to direct data flow. Read/write (R/W) determines the direction of data flow while Enable (ENA) controls timing-specifying when each device is to drive the data bus.

In this case, there is no source of internal stimulus for the board. An external stimulus is to be designed into the test fixture. Consider first, test of only the ROM portion of the board. The decoder logic is stimulated by the various combinations of address lines A11 through A15. Particular addresses of the decoded ROM are selected by the address lines AØ through A1Ø. An externally applied 16-bit counter may be conveniently utilized to stimulate the address lines. The counter itself will be driven via a test system clock. This clock signal is also supplied to the CLOCK input of the SA processor. The R/W control line will be controlled via a static digital drive signal from the test system.

The RAM's will also be exercised by the counter. However, before taking meaningful signatures involving RAM output data, the RAM's must be initialized. One way to accomplish this is to fill each location in RAM memory with its own address. Figure 7 illustrates how this may be implemented with the same counter stimulus as described above. The R/W line is set to the write state, the buffers are enabled and address data, bits AØ through A7, is applied to the data bus. As the counter is cycled through all possible addresses, each writable location is initialized with its particular address. With the RAM's so initialized, they are treated like ROM's for signature tests.

A GO/NO-GO test may be performed on the board by collecting just the eight data bus signatures. The counter is set up to cycle continuously and the SA measurement window is set to monitor all 2^{16} possible states. The most significant bit of the counter is connected to both the SA START and STOP inputs. In less than one second all decode and data functions may be checked as reflected in the eight data bus signatures. Figure 8 shows both the hardware connections and software required to perform this test.

Suppose one or more faulty signatures are measured during the GO/NO-GO test described above. Fault isolation may be easily performed using inside-out tracing. First, the 16 address lines driven by the counter are verified over the entire count cycle. If a bad signature is measured, there is a problem with the stimulus circuitry which must be corrected. If the stimulus is verified to be working properly (16 signatures in approximately one second), the decoder logic outputs are checked. In the



MEMORY BOARD EXAMPLE

Figure 7. A memory board consisting of both ROM's and RAM's can be tested using Signature Analysis. An external stimulus is applied using a 16-bit counter to drive the address lines and to initialize RAM.

event of a bad decoder signature, back tracing is performed through the decoder logic until a circuit element with good input signatures but bad output signature(s) is implicated as the faulty device.

Correct signatures at the decoder outputs imply a faulty memory device. Individual memory device functions may be isolated by varying the source of the START signal, broadening the address range for successive measurements of the data bus line where the faulty node was found during GO/NO-GO testing. The faulty device is easily determined on the basis of the START signal being used when the failure occurs. Another possibility is to continue using a STAR"/STOP which encompasses the entire range of addresses but to select the enable pulses of individual memory elements for use as the CLOCK signal. In such a case, only data bus output for the single device will be monitored by the SA processor. A third possibility for isolating the faulty memory element is to take signatures on specified clock counts with or without varying the START signal source. The count parameter may be set to the number of addresses of the device(s) being tested with the START signal derived from the counter bit representing the first address to be monitored.

Figure 9 illustrates sample test system software to isolate a faulty memory device in a 32K memory system consisting of eight 4K devices. Variable arrays are used to tabulate test node locations, expected signatures and to set the order of the fault isolation sequence.

In summary, it has been shown that a typical memory board can be fully tested at speed with SA. Fixturing requirements include a simple counter circuit for SA measurement stimulation and provision for writing RAM data. A GO/NO-GO test can be conducted in less than one second by

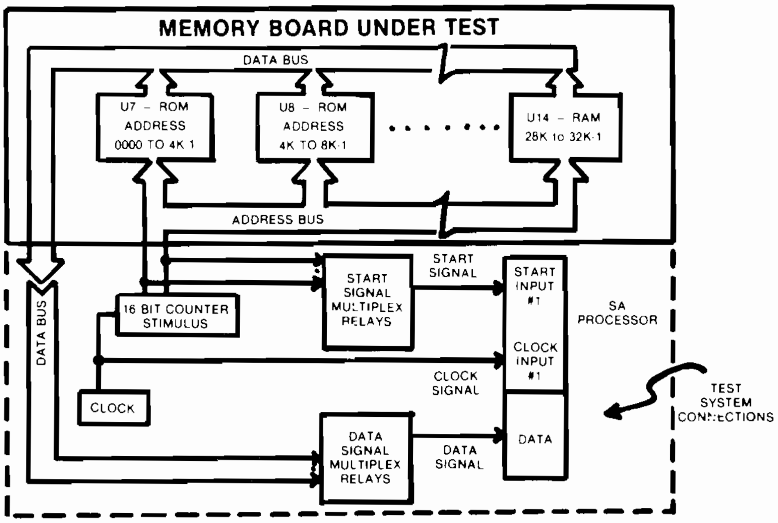
Provisions which have been made within the design requirements for SA testability include:

- Signal lines which may be programmed via switches or ATE system digital drivers to select vectors to run particular programs in ROM. Used in conjunction with a non-maskable interrupt (NMI) to cause the microprocessor to jump to a firmware test sequence.
- Special firmware included in on-board ROM and accessible as described above for exercising board elements during signature measurements.
- A signal line which may be driven by the ATE system to disable the data bus buffers and break the feedback path to the microprocessor for "free-run" testing.
- Pull-up and pull-down resistors which may be applied to the microprocessor input to provide a no-operation (NOP) instruction during "free-run" testing.

The test strategy for the board includes first in-circuit shorts/opens and discrete component testing followed by a functional test sequence including both free-run and internal stimulus driven tests:

- Free-run tests
  - Verify that microprocessor can address memory properly
  - Verify contents of ROM which contains SA stimulus firmware

Figure 9. Sample ATE software for isolating faulty memory elements on a memory board using signatures taken only over the span of memory addresses for each 4K memory device.



- Data relays set to monitor node where data bus failure occurred during go/no-go test encompassing all addresses.
- Array S[ ] holds relay data corresponding to address line which indicates first address of each device.
- Array U$[ ] holds designators for each memory element; S$[ ] holds good signature data from reference device.
- RAM previously initialized.

```
     Program Statement          Description

151: for J = 1 to 8             Set up loop to test 8 devices
152: mcon S[J], N[I]            Close start signal relay and data node relay
153: sig "", S$[J], 4096        Take signatures for 4096 clock pulses
154:                            (total memory of current device), compare
155:                            to reference signature S$[J].
156: if flg11; gsb "PRINT       Test flag to see if signature test passed.
     FAULT"
157:                            If not, go to subroutine to print fault
                                message.
158: next J                     End of fault isolation loop
159: gto "RESTART"              After all tests, start new bcard test.
160:
161: "PRINT FAULT":             Entry point of subroutine to print fault
                                message
162: wrt "PRINTER", "RE-        Write device replacement message.
     PLACE", U$[J]
163: ret                        Return from fault message subroutine.
```

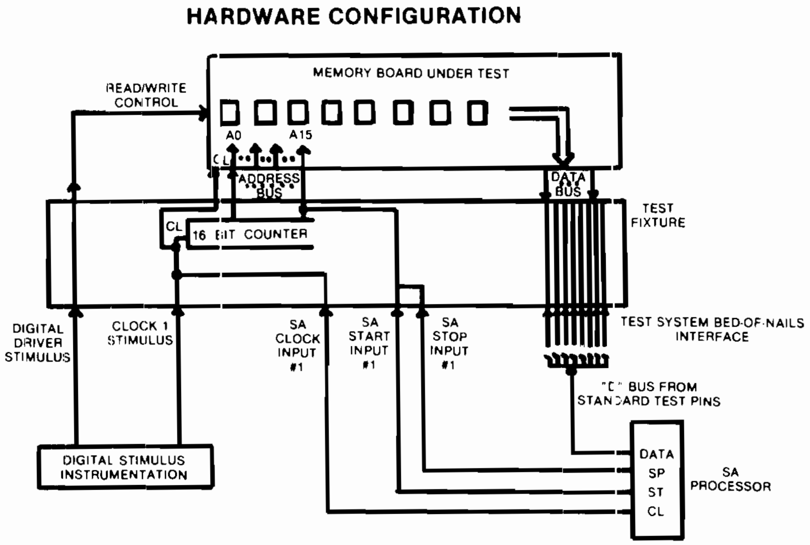


Figure 8. Example of hardware connections and software required to perform a GO/NO-GO test for a memory board using Signature Analysis.

Test Program for Go/No-Go Test

Line #	Program Statement	Description
1:	"Go/No-Go test for memory board":	Comments for Documentation
2:	"Tests signatures of 8 data bus":	
3:	"Lines for all possible addresses":	
4:	"RAM's have been initialized":	
5:	rcv ref 1,0.4,2,3.2	Set SA low, high reference voltages
6:	saset 1,"R",1,"F",1,"F"	Set SA to process clock 1
7:		input, rising edge, start 1,
8:		stop 1, falling edges Initializes SA Processor
9:	clock 1,1e6,1e7	Set clock to 1 MHz, free run
10:	apply "R/W", 1	Set RAM's to Read
11:	for I = 1 to 8	Start of loop for 8 signature measurements
12:	mcon N [I]	Close test node relay specified by variable N[I]
13:	sig "", S$[I]	Measure and check signature against stored reference signature S$[I]
14:	if flg11; gto "DEBUG"	Test pass/fail flag; if fail, branch to debug subroutine
15:	next I	End of measurement loop
16:	dsp "BOARD PASSED"; gto "START"	Display PASS message
17:	"DEBUG":	Start of debug procedure
18:	. . . .	

monitoring data bus signatures and fault isolation can be performed, typically in 5 to 15 seconds to the component level.

Testing a Microprocessor Board

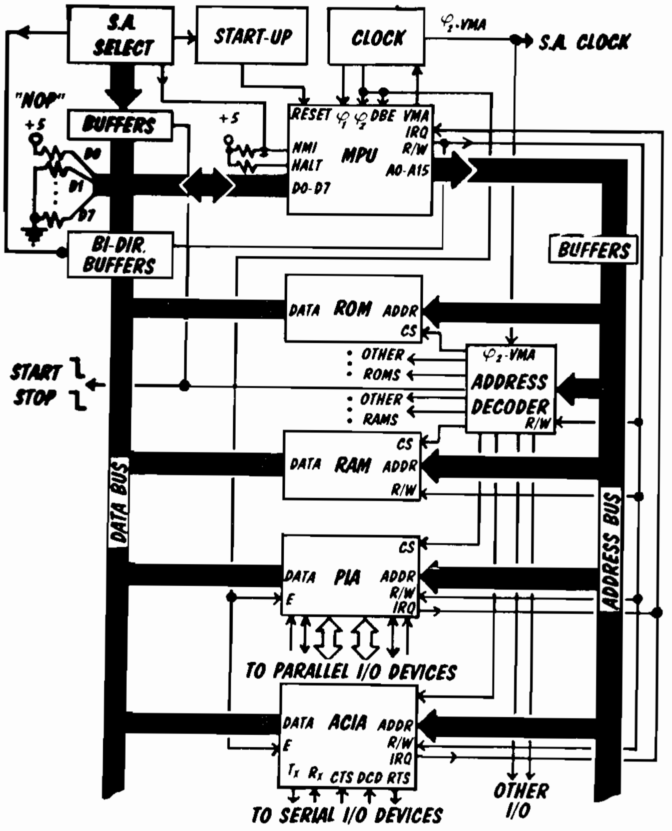
The second SA test example is the board represented by the diagram of Figure 10. It consists of a Motorola 6800 microprocessor, an onboard clock, power-up support circuitry and various I/O circuitry. On board memory elements include both ROM and RAM. There are both hardware and software elements included to enhance Signature Analysis testability.

Architecturally, the board utilizes a 16-bit address bus, an 8-bit data bus. There are buffers on the address bus lines and bi-directional buffers on the data bus lines. Control lines employed by the board include read/write (R/W), valid memory address (VMA), data bus enable (DBE), and interrupt request (IRQ). ROM and RAM elements are driven by combinatorial address decoder logic.

The I/O circuits include a peripheral interface adapter (PIA) for parallel I/O devices such as printers, displays and keyboards, and an asynchronous communication interface adapter (ACIA) for serial I/O devices such as a terminal.

Figure [illegible]. A board with microprocessor and specialized peripheral chips can be effectively tested using SA. Hardware and firmware designed into the board can enhance SA testability and permit fault isolation to the component level.

MICROPROCESSOR BOARD EXAMPLE



- Internal SA stimulus tests
  - Verify contents of all ROM's
  - Test RAM's
  - Test PIA
  - Test ACIA

The free-run tests are initiated by electrically disabling the data bus buffers and applying a NOP instruction to the microprocessor via the pull-up/pull-down resistors. This has the same effect as applying a 16-bit counter to the address bus - the program counter of the processor will cycle through all possible addresses. The valid memory address (VMA) line is multiplexed to serve as SA CLOCK while the address line A15 (most significant bit of address bus) is applied to both SA START and STOP. As in the previous example, all addressing functions and all data storage functions except RAM functions (which must be initialized) may be verified by checking the eight data bus lines during free run. In this case, a different approach will be taken to illustrate alternative SA test procedures.

First, address functions including the address bus, buffers, and decoder logic is verified by taking signatures on the address bus on both sides of the buffer and at the output nodes of the address decoder logic. Then, the contents of ROM's is verified by selecting START and STOP lines from the address bus corresponding to the low and high addresses different paths depending upon whether a PASS or FAIL condition occurs. If the signature of a constant high condition is monitored during this RAM write/read procedure, a different signature will be registered for each pass/fail condition (since the test signature will reflect the number of clock pulses required to execute the firmware-based procedure).

A table of signatures for all pass/fail conditions may be established within the ATE test program. The RAM's can also be checked by monitoring the data bus during the read/write sequence. This would require eight signature measurements over eight identical stimulation cycles. By monitoring the high-level signal and having the microprocessor perform the data check function, the same test is executed on the basis of one signature, one stimulation cycle. In the event of a RAM failure, a troubleshooting procedure may be executed as described for the memory board example to isolate to the faulty device.

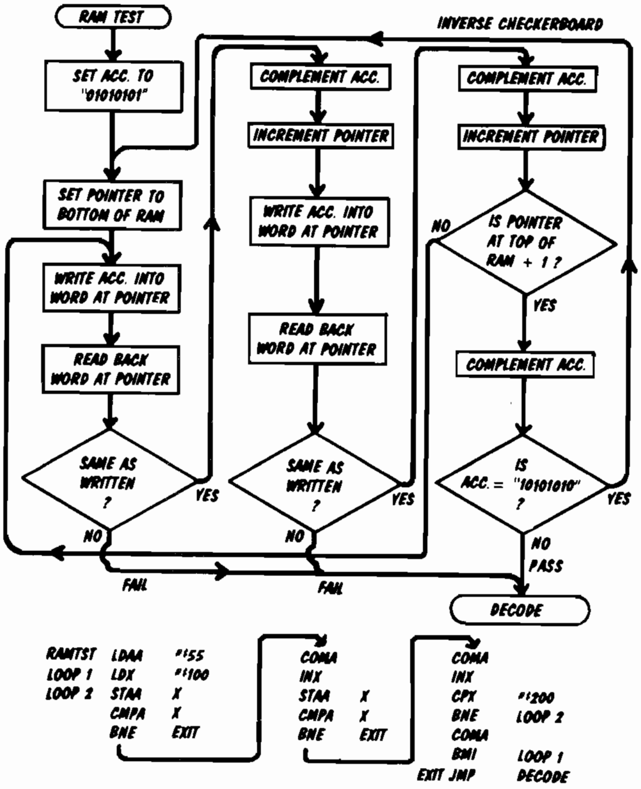
The PIA provides two eight-bit ports which may be programmed for either input or output functions. The two ports act like latches. Control lines define the data direction and handshaking schemes for passing data. The PIA is effectively stimulated via "PIA" routines in the SA stimulus ROM. Test system relays can be used to connect the two I/O ports together at the board edge connector. Then, one port can be programmed to output while the other is configured as input. Signatures may be taken directly on data bus lines to verify proper activity for virtually all possible combinations of the data and control signals, or the processor itself can be programmed to perform data check functions while the high line signature is monitored as for the RAM tests. After tests are completed for one combination, the input/output functions can be reversed and the test procedure repeated. The PIA test firmware can be written to stimulate the device as its used in the final application.

The ACIA is similar in nature to the PIA except that data transfer may be asynchronous to the processor clock and one port is dedicated for the transmit functions and a second port is dedicated to receive functions. Again, if the two ports are connected together, firmware in the SA ROM may be used to exercise the device. However, it will be necessary to make the test synchronous to some reference clock in order to use the normal SA test mode. If this is not possible, then nodal activity may be characterized by taking a high level signal signature while using the test node signal as SA CLOCK. The signature will thus test that the correct number of transistions occur on the test node.

The ATE test program can be written to perform an efficient GO/NO-GO test on the board by using a free-run test to verify the board kernel and ROM's. Then RAM and peripheral firmware may be executed while monitoring the high level signature as described to quickly verify correct or incorrect operation. If faulty signatures occur, some faults will be directly specified by the high line signature. In other cases, tracing algorithms will be required to isolate to the faulty component level.

In summary, SA can be effectively applied at normal operating speeds to test a board with a microprocessor and auxiliary peripheral chips. A few hundred bytes of firmware, in conjunction with the processing capabilities of the microprocessor, can be used to verify normal operating modes without knowledge of the internal logic structure of the devices used in the board design.

Figure 11. Sample algorithm executed from on-board firmware while signatures are collected by ATE for RAM tests. Firmware can be designed so that a single signature in conjunction with microprocessor data processing capability can indicate faulty or correct RAM operation and which, if any, RAM is faulty.



```
RAMTST  LDAA   #$55        COMA               COMA
LOOP 1  LDX    #$100       INX                INX
LOOP 2  STAA   X           STAA   X           CPX    #$200
        CMPA   X           CMPA   X           BNE    LOOP 2
        BNE    EXIT        BNE    EXIT        COMA
                                              BMI    LOOP 1
                                         EXIT JMP    DECODE
```

of the ROM's and monitoring signatures on the eight data bus lines. All of this is accomplished in the free-run mode and would require less than five seconds to run. Any failures which occur are handled as discussed in the previous example. One exception to this might be a test of the board clock in the event of an apparent microprocessor failure as reflected in a faulty signature on the address bus. The clock into the processor could be checked with amplitude/frequency measurements or by measuring a constant high condition signature over the free-run window. Such a signature is a function of the number of clock pulses in the measurement window only.

Given that the ROM containing the SA stimulus tests "good", then that stimulus may be used for further tests on the board. The firmware for a particular test sequence is executed by controlling interrupt and vector address data to the processor as described previously. The firmware may be written to perform individual tests or to string individual tests together in any desirable sequence - all controlled by vectoring to the appropriate location in the ROM.

To perform RAM tests, the firmware-based sequence of Figure 11 may be effectively utilized. First, the processor writes alternating 1's and 0's into each word of RAM. Complementary patterns are used in even/odd locations. These are well established patterns for testing RAM's for adjacent bit and adjacent address shorts. Each word is read back and tested by the processor. After all addresses are checked, the patterns in each address are complemented and the procedure is repeated. If any cell fails, a flag is set to register the failure. At the end of each RAM test, the flag will be checked and the firmware will take one of two

SA FOR LSI BOARD TESTING - IN SUMMARY

Signature Analysis is an effective method for ATE-based testing of printed circuit assemblies which include LSI devices. When using SA, the tester is synchronized to the device under test and the testing is typically performed at MHz rates. A large number of microprocessor instructions can be tested using a short ATE test program and the entire contents of RAM/ROM verified without much sacrifice to the total test time. LSI devices may, thus, be thoroughly exercised to achieve high operational confidence--encompassing at speed, timing related faults which are often more difficult and costly to detect in the production test flow.

SA imposes no pattern length limitations. Since a compression technique is used, the measurement unit performs no comparisons on the input data stream until the end of the test sequence. This obviates the need for RAM-backed receivers and minimizes both tester memory requirements and the processing time in achieving pass/fail decisions. Thus, SA may help minimize both tester and test time costs. In an ATE environment, test stimulation may be effectively generated either internally or externally to the device under test. Test patterns are often directly related to the application software. With the measurement capabilities of ATE, the stimulus may be either a repetitive or single cycle signal.

The measurement technique of SA is not dependent upon the logic structure of the circuit being tested. Although the stimulus may vary somewhat depending, for example, upon the processor type (one processor test may generate START/STOP on the basis of address decoding whereas a test for another processor may use an I/O line for the same purpose), the ATE software, the measurement approach, and fault isolation processes are largely the same for all types of a generic class of circuits.

In general, SA can be most effectively utilized when the designer of the board is knowledgeable of the technique and plans for the fact that SA production testing is to be employed. Considerations for accomplishing this objective and examples of SA designed-in testability may be found in references 11, 12, and 13. Designing SA into a microprosser-based board may require dedicating a small portion of the on-board memory for the "SA ROM". This memory space can be utilized by the processor to exercise itself and other devices on the board.

"Designed-in" SA testability, such as on-board stimulus, can make SA an effective test technique for both production test and field service test applications. With stimulus generated by the unit under test itself or a simply connected external ROM, SA lends itself to use with portable field service instrumentation. The compatability of SA in production and field service can minimize the cost of developing product test approaches. SA can often reduce warranty costs by eliminating the need for board exchange with its inherent service-module inventory and typically high administrative and handling costs.

SA is not, however, without its limitations. Designing in SA testability may add to product cost, although this cost may well be recovered in reduced testing cost. SA is a synchronous technique. Any bit stream that is examined during a measurement period has to be synchronized to a clock. Hazards which exist in the board or logic races generated by the stimulus could cause unstable signatures. Obviously, it is far preferable to eliminate such hazards and races from the board before testing. If this is not possible, the user must choose the measurement window in a way that will exclude these uncertain data from entering the measurement cycle. Don't care conditions on data or address busses may be handled by choosing a clock that is gated by a data valid signal. Asynchronous signals can be measured using SA processor hardware for transition counting.

As in any other technique of testing, Signature Analysis requires a set of stimulus patterns that functionally test various components and propagate any possible faults to a measurement point. For devices that are directly connected or are readily accessible to the stimulus, it is usually quite simple to generate a high confidence test pattern. For devices that are not readily accessible by the stimulus or are a part of a deep sequential circuit, it requires careful effort to generate a high confidence test pattern. However, by using test sequences directly related to the applications environment, a high confidence level can usually be obtained.

SA is being effectively employed as an ATE-based test tool for LSI-based board testing by a variety of users. It offers unique alternatives to other test methods and can provide high test confidence at reasonable cost for both production and field testing.

REFERENCES

1. "5004A Signature Analyzer", Technical Data, Hewlett-Packard Publication Number 02-5952-7464D, April 1977.

2. Bruce LeBoss, "LSI-Board Test Bedevils Users", Electronics, October 26, 1978, pp. 81-82.

3. "Automated Systems for LSI Device Testing", Electronics Test, May 1979, pp. 14-30

4. Art DeSena, "Automated Board Testing", Electronics Test, March 1979, pp. 32-40.

5. Robert Francis, "Real Time Test Methods for MPU Based Products", Electronics Test, March 1979, pp. 42-46.

6. Chris Sheldon, "Understanding and Solving the Problems of LSI-Based Boards", Journal of ATE, Spring 1978, pp. 4-5.

7. Stephen Bisset, "LSI Tester Gets Microprocessors to Generate Their Own Test Patterns", Electronics, May 25, 1978, pp. 141-145.

8. W. W. Peterson and E. J. Weldon Sr., "Error Correcting Codes", The MIT Press, Cambridge, Massachusetts, 1972.

9. Robert Frohwerk, "Signature Analysis: A New Digital Field Service Method", Hewlett-Packard Journal, May 1977, pp. 2-8.

10. Gary Gordon and Hans Nadig, "Hexadecimal Signatures Identify Trouble Spots in Microprocessor Systems", Electronics, March 3, 1977, pp. 89-96.

11. "Designers Guide to Signature Analysis", Application Note 222, Hewlett-Packard Publication Number 02-5952-7465, April 1977.

12. "Application Articles on Signature Analysis", Application Note 222-2, Hewlett-Packard Publication Number 02-5952-7542, May 1979.

13. "Implementing Signature Analysis for Production Testing", Application Note 222-1, Hewlett-Packard Publication Number 5952-8785.

14. Kamran Firooz, "Signature Analysis - A Technique for Board Test and Field Service Applications", presented at Boston ATE Show, June 1979.

10. Basic Design Considerations for Switching Power Supplies

HEWLETT PACKARD

Craig Maier holds a BSES from New Jersey Institute of Technology. He is presently an Applications Engineer for the Modular power supply product group at the New Jersey Division.

BASIC DESIGN CONSIDERATIONS FOR SWITCHING REGULATOR POWER SUPPLIES

Craig Maier
January 1980
HEWLETT-PACKARD COMPANY
New Jersey Division

One of the first applications of high speed switching technology dates back to the advent of television. A 16kHz flyback circuit utilizing a beam power pentode switch and a vacuum diode damper was used to generate both the second anode CRT high voltage and the horizontal scan ramp. The original system operated off the secondary circuit of a 60Hz power supply transformer. Today, this flyback system is still used, with the variation that it is operated "offline" with the cost, weight and size savings due to the elimination of the 60Hz power transformer. Also, other bias voltages are derived from the flyback circuit to supply DC power to all other circuits in the receiver.

Little attention was paid to this method of power conversion until about 20 years ago in commercial power supply design. Looking back, the switching frequency in commercial switching regulators went from 800Hz to 3kHz during the 1960's, to 10-40kHz in the 1970's. The prospects of switching rates from 200kHz to 1MHz exists for the 1980's The driving force behind increased speed results from the development of new components such as high speed, high voltage and high current power transistors. Also, high speed SRC's, high frequency switching rectifier diodes, Schottky rectifiers, and new developments in VFET technology and gate controlled switches are contributing to increased switching rates. Concurrent with semiconductor improvements are the improvements of magnetics and capacitor technology.

During the last several years, acceptance of the switching power supply, especially in the computer industry, has increased significantly. This is due to the reduction in the size and weight of the switcher. Also, increased efficiency and reliability make them an attractive alternative to the traditional linear power supply. As on-going improvements in LSI technology allow the computer industry to perform more functions in a given size box, the advantage of switching regulators mentioned above shall become more obvious in the 1980's. The reasons behind these switching power supply improvements and the design criteria involved in their implementation shall be explored in this paper.

A regulated DC power supply of any architecture has at least three basic building blocks:

1. A source of DC power
2. A current passing device
3. A control system

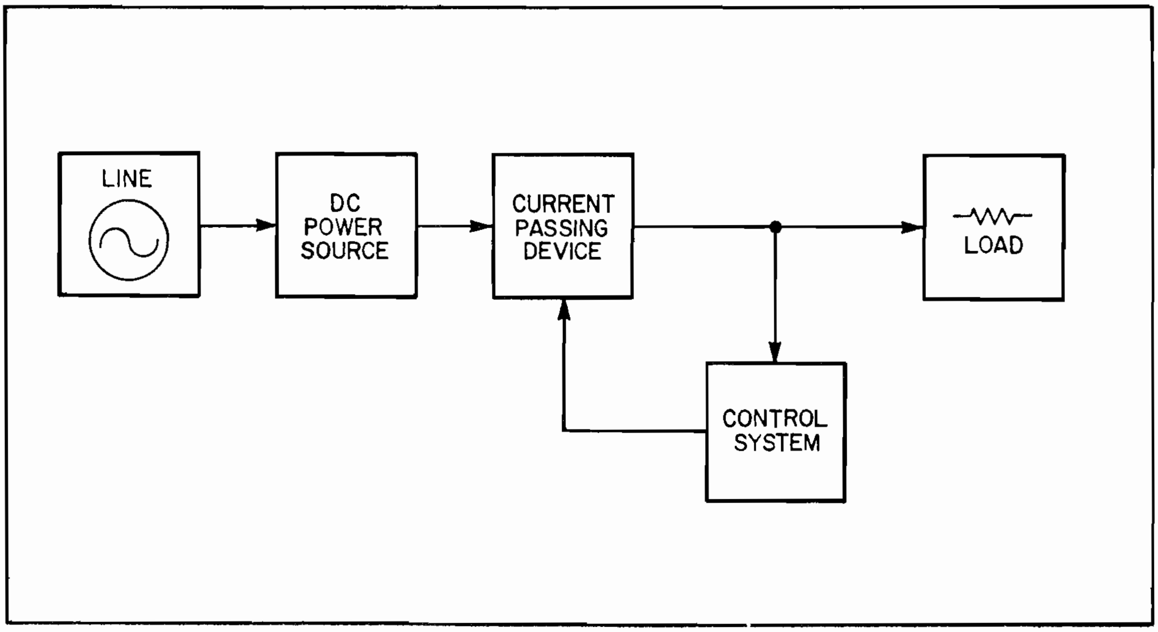


FIG. 1 - BASIC LINEAR POWER SUPPLY BLOCKS

First, we shall take a casual look at the traditional linear power supply, in order to develop similarities and contrasts to switching power supply design. The DC power source in a linear power supply consists of a 60Hz power transformer, a rectifier and an energy storage device, usually a capacitor.

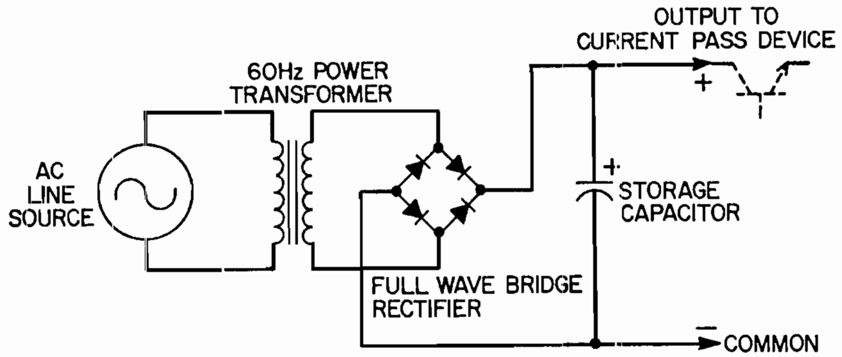


FIG. 2 - BASIC LINEAR POWER SUPPLY INPUT MESH

The load is connected to the DC power source by way of a current passing device which varies the conductive path between the two. The control system samples the output voltage or output current, compares it to a fixed voltage reference, and signals the current passing element to provide the required conductive path to the load. This system has two key features:

1. Isolation is established between line input and DC outuput.

2. The system provides an output voltage or current which is almost independent of input line variations or output load variations.

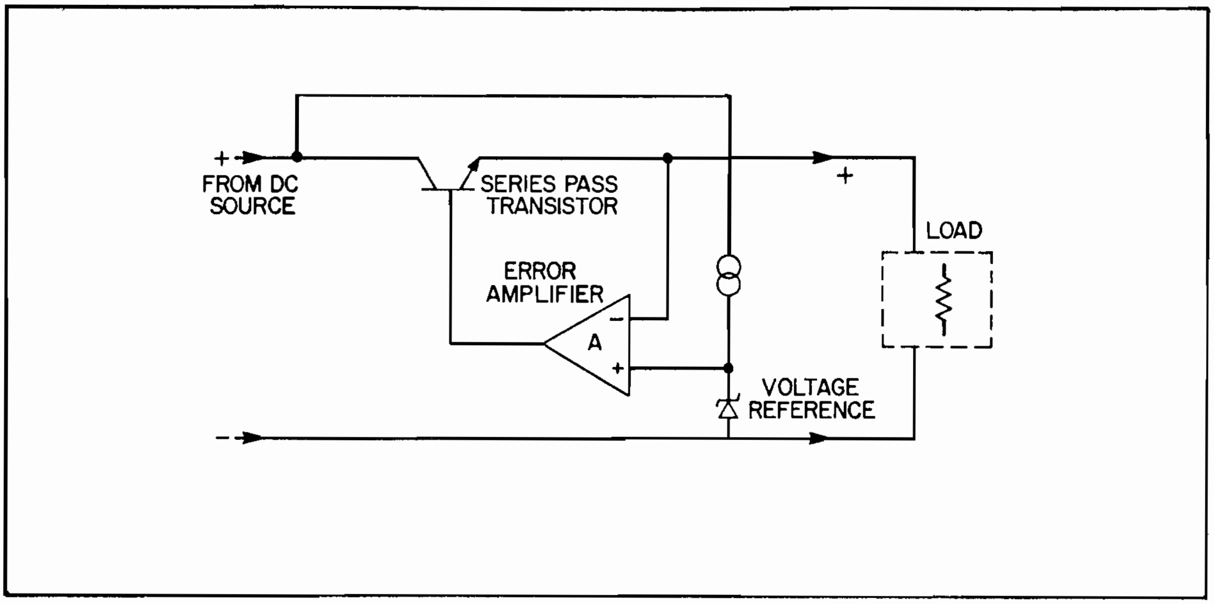


FIG. 3 - BASIC LINEAR POWER SUPPLY CONTROL SYSTEM

The amount of ripple on the output and the degree of regulation is largely a function of the amount of amplication provided by gainblock A. Because of the simplicity of the linear topology, very high gain can be used, with relatively simple compensation networks employed to stabilize the system at high frequencies. The result is that the linear power supply can provide excellent regulation characteristics, with low PARD (Periodic & Random Deviations) and fast load effect transient recovery times. On the negative side of things, the linear supply is large in physical size and heavy for a given output wattage. This is due to the fact that a linear supply achieves isolation with a 60Hz power transformer which must be large and heavy due to the low operating frequency. Also, the linear suffers from poor efficiency due to the necessity of maintaining a voltage drop across the current passing device as current is also passing through it. The product of the Vce drop and Ic produce heat and substantially limit the efficiency of this architecture. Some attempts have been made to improve linear efficiencies with the use of a phase controlled pre-regulator circuit. This circuit is used to maintain a constant and relatively small voltage drop across the series pass device, but still can't approach the efficiency of switching regulator architecture. Also, this technique cannot eliminate the necessity for a 60Hz power transformer, as isolation between line and load must be maintained. Now we shall take a look at some switching techniques.

A simple example of a switching power control cirucit would be that of an electrical light connected to an on/off switch. When the switch is on, the voltage drop across the switch is ideally zero, and when the switch is off the current through the switch is ideally zero. When the state of the switch is changed, the electrical state of the switch is changed instantaneously in the ideal case.

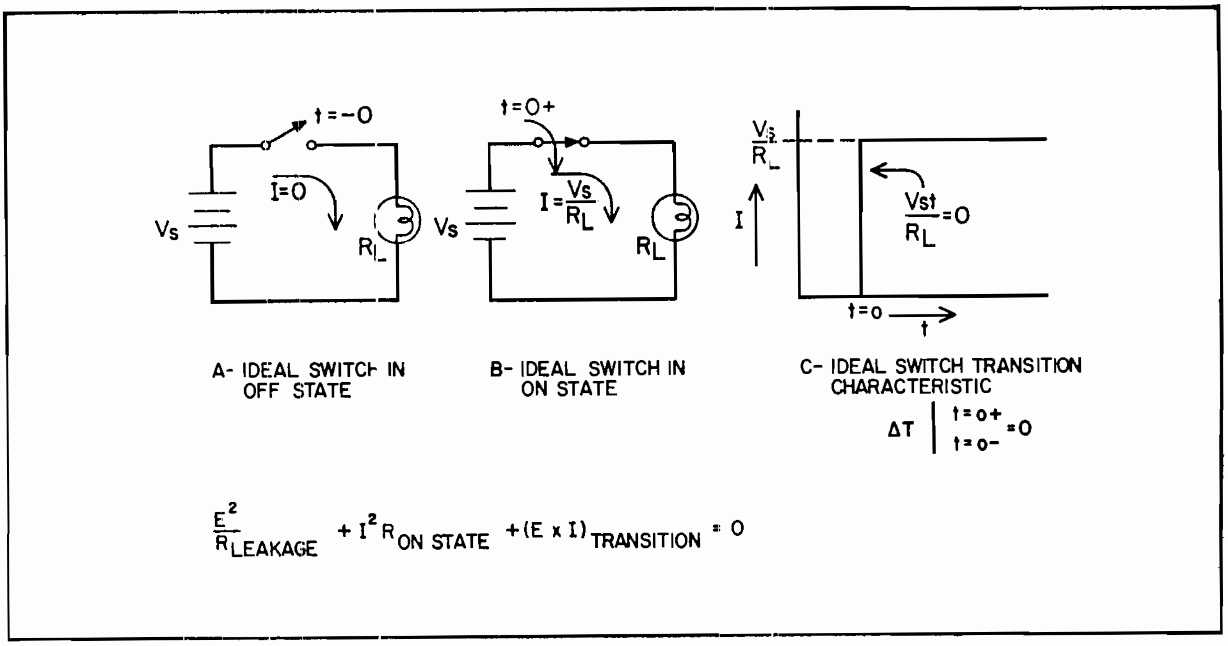

t=-O
I=O
Vs
RL
t=O+
I = Vs/RL
Vs
RL
Vs/RL
I
Vst/RL = O
t=o
t
A- IDEAL SWITCH IN OFF STATE
B- IDEAL SWITCH IN ON STATE
C- IDEAL SWITCH TRANSITION CHARACTERISTIC
ΔT | t=o+ t=o- =O
E²/R LEAKAGE + I²R ON STATE + (E x I) TRANSITION = O

FIG. 4 - IDEAL SWITCH CHARACTERISTICS

One can readily see that in all three states of the ideal switch; on, off and transition, the power dissipated by the switch is zero.

If we now add a third component to the system mentioned, the human eye, and then turn the switch on and off rapidly, the brightness of the bulb shall be proportional to the on state duty cycle of the switch. The eye is, in this case, acting as an integrator to the photonic energy being supplied to it.

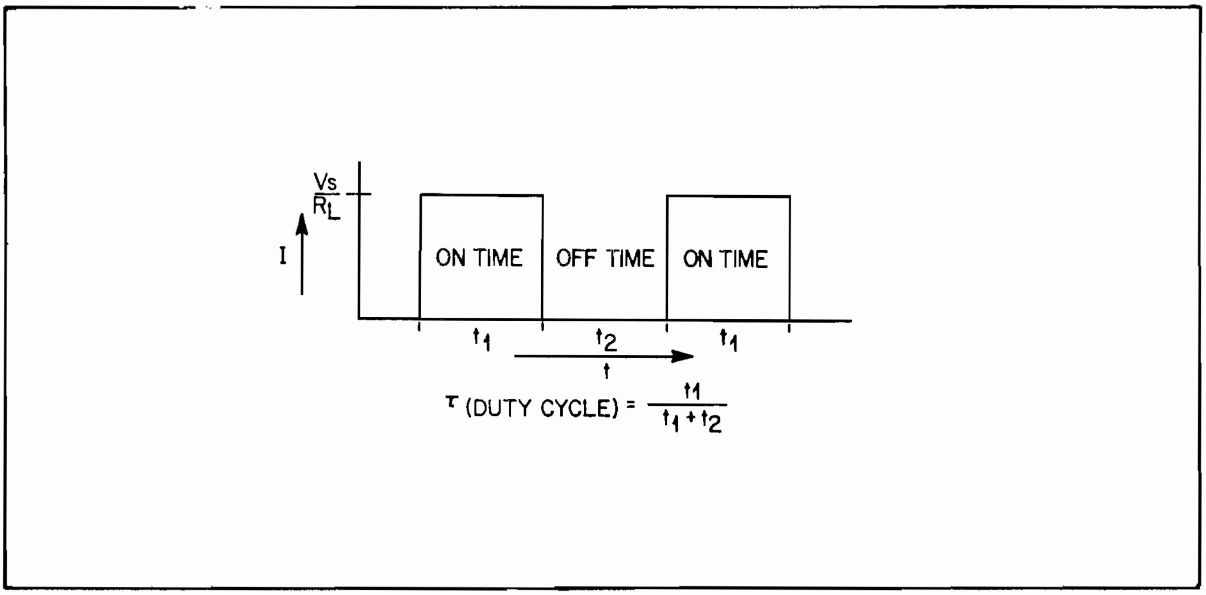


FIG. 5 - SWITCH ON STATE DUTY CYCLE

If we start with a 100 Volt source, and a 100W incandescent bulb (with constant resistance), the on state power should be 100 Watts applied to the bulb. If we set $t_1=t_2$, or we switch at a 50% duty cycle and then integrated through the interval (t_1+t_2), the apparent power seen by the human eye would be the photonic output of the bulb proportioned by only 50W of input power. It can be seen, that varying the duty cycle T, would vary the apparent brightness of the bulb, with no power loss in the control element, the switch. In theory this system is 100% efficient if we neglect the energy required to turn the switch on and off. A real switch has three terms involved with switching loss. They are on state resistance, off state leakage, and transitional delay or storage time. Off state leakage is negligible in todays devices, so attention is paid mostly to the on state resistance of devices and device storage time. These characteristics shall be discussed in more detail later.

In developing a model for a switching regulator, we must expand on our original regulated DC power supply building blocks with the addition of an energy storage system. The basic building blocks of a switching power supply, are therefore:

1. A source of DC power
2. A switching network
3. An energy storage system
4. A control system

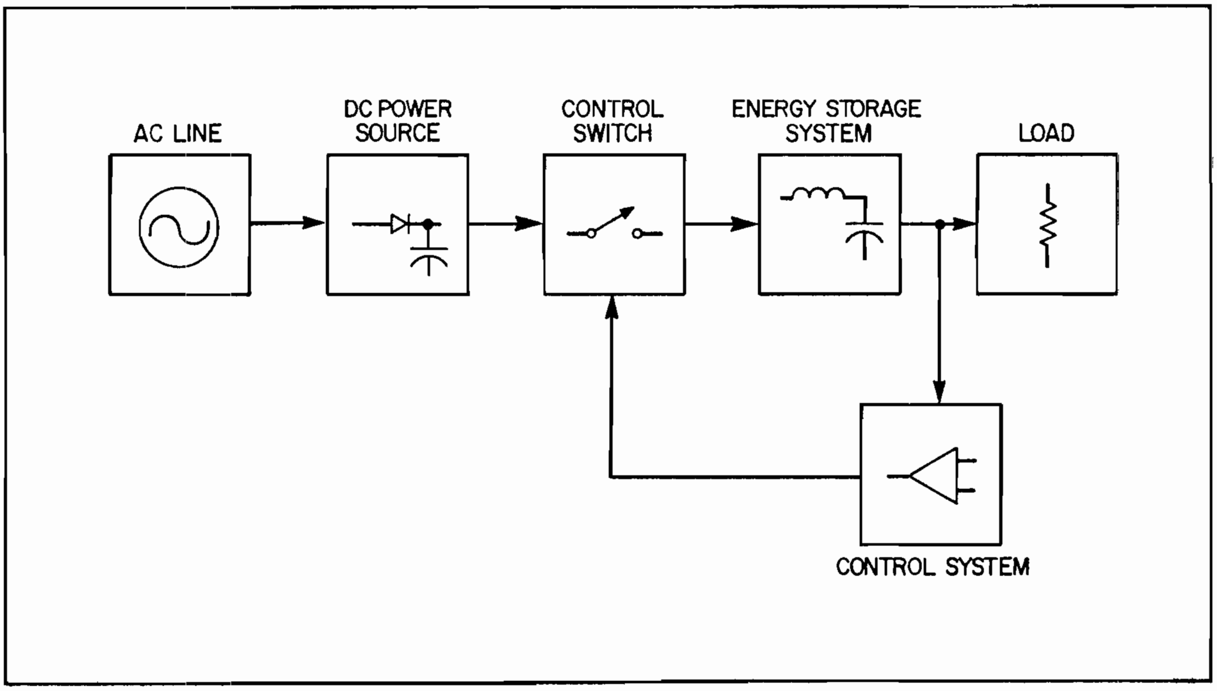


FIG. 6 - BASIC SWITCHING POWER SUPPLY BLOCKS

The energy storage system in most topologies serves as an integrator, such that the output voltage of the sytem under a given load is a function of the switches "on" time duty cycle. The control system, therefore, must sample the output voltage or current, compare them to a fixed reference, and generate an on state duty cycle proportional to the amount of time required to ramp the voltage or current up to the established output demands of the supply.

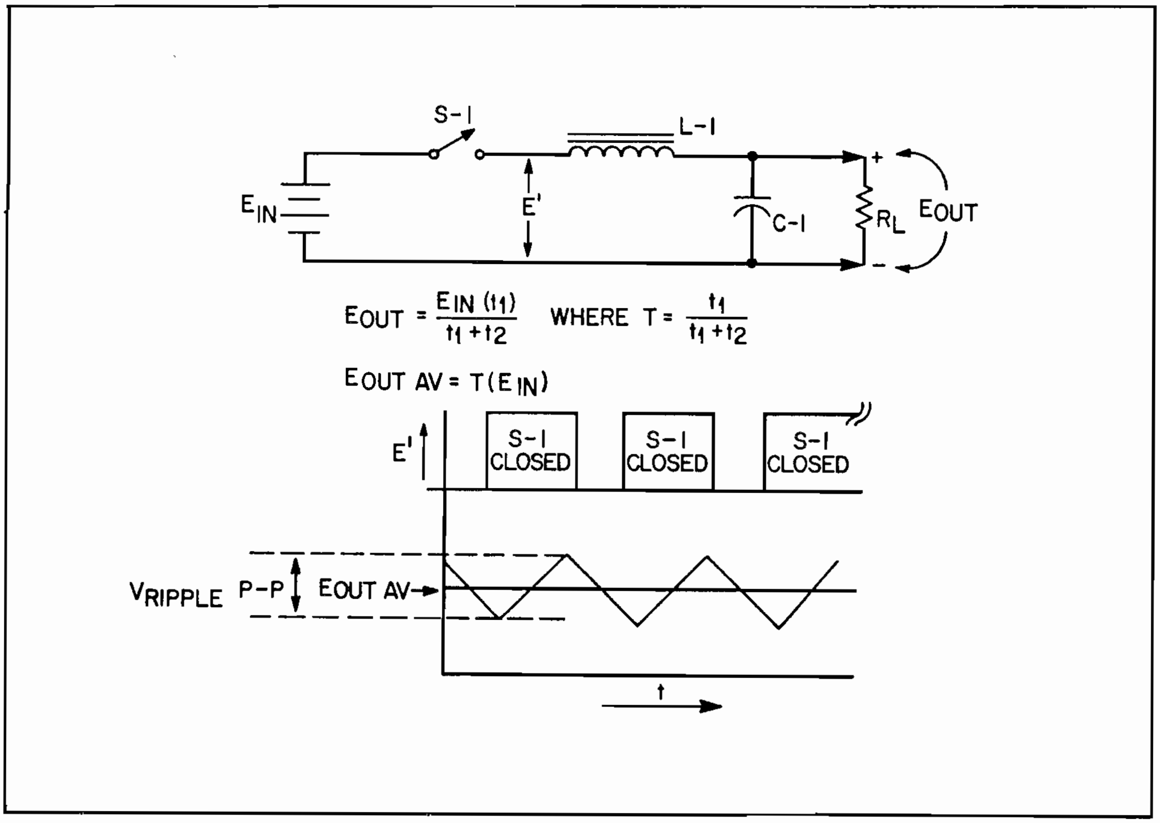


FIG. 7 - SWITCHING POWER SUPPLY STORAGE SYSTEM

Figure 7 shows a representative energy storage system utilizing a two pole L-C filter. Some minimum load must be maintained on the output L-C when this network is enclosed in a control loop in order to prevent the control loop from going unstable under no load conditions. The output voltage ripple in certain control schemes (V ripple) is a constant irrespective of load. This is a result of double ended limit cycle regulation which shall be discussed later. It may be noted that a certain amount of output ripple is required for single loop double limit cycle control.

In a real power supply, as was mentioned earlier, two basic criteria must be satisfied, that of output regulation and line input to load output isolation. Offline switching techniques take advantage of the high speed characteristics of devices by installing the switching network before the power transformer. This requires the addition of a line input rectifier and filter network, but results in substantially smaller power transformer size and weight. The basic block for the power mesh of a switching inverter circuit are shown in Figure 8.

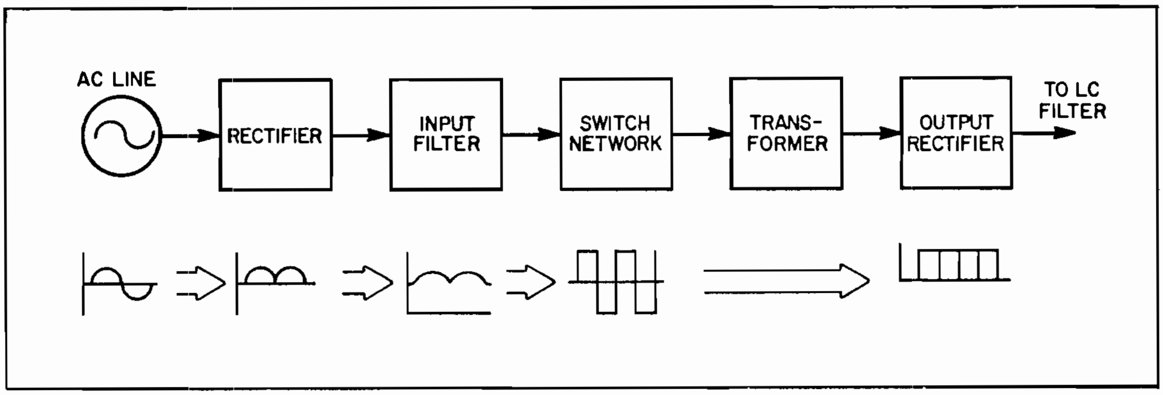


FIG. 8 - SWITCHING POWER SUPPLY POWER MESH

We shall now look at several specific circuit topologies used in switching regulator power supplies. The first one is referred to as a voltage driven push-pull circuit and is shown in Figure 9.

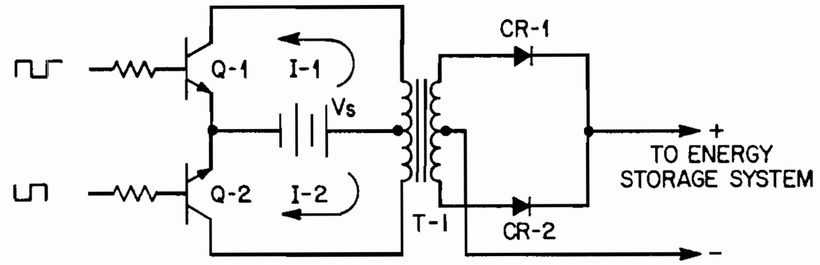


FIG. 9 - SWITCHING POWER SUPPLY PUSH-PULL TOPOLOGY

When Q-1 turns on, T-1's primary magnetizes in one direction as current is pulled through Q-1's collector to emitter. In short turn, Q-1 is turned off while Q-2 is turned on, reversing the flux in T-1 as current flows in Q-2. These commutations are coupled by the mutual inductance of T-1 to T-1's secondary where they are rectified by CR-1 and CR-2. The output of the rectifier is applied to an energy storage network for integration and smoothing purposes. This circuit is considered to be of the voltage fed type, in that the current in the switches is only limited by the inductance of the transformer ($I = \frac{V\ supply}{X_L}$). Because of the voltage fed input to the switches, if both transistors were to be in conduction at the same time, a result of an extraneous transient appearing at both bases at the same time, transformer saturation could occur and collector currents would be limited only by circuit resistance. Because circuit resistance is extremely small, this situation would probably destroy both switches. Another mechanism by which switch destruction could occur would involve too short of a dead zone between switching cycles. Because of storage time in the transistors, they do not turn off immediately after base drive is removed. Even using the method of actively pulling current out of the transistor base does not eliminate the finite switch transistion time. Because of this transistor characteristic, one switch must be allowed to settle down before the other adjacent transistor is turned on, adding to control circuit complexity. This is accomplished with what is known as a dead zone generator, the waveform of which appears in Figure #10.

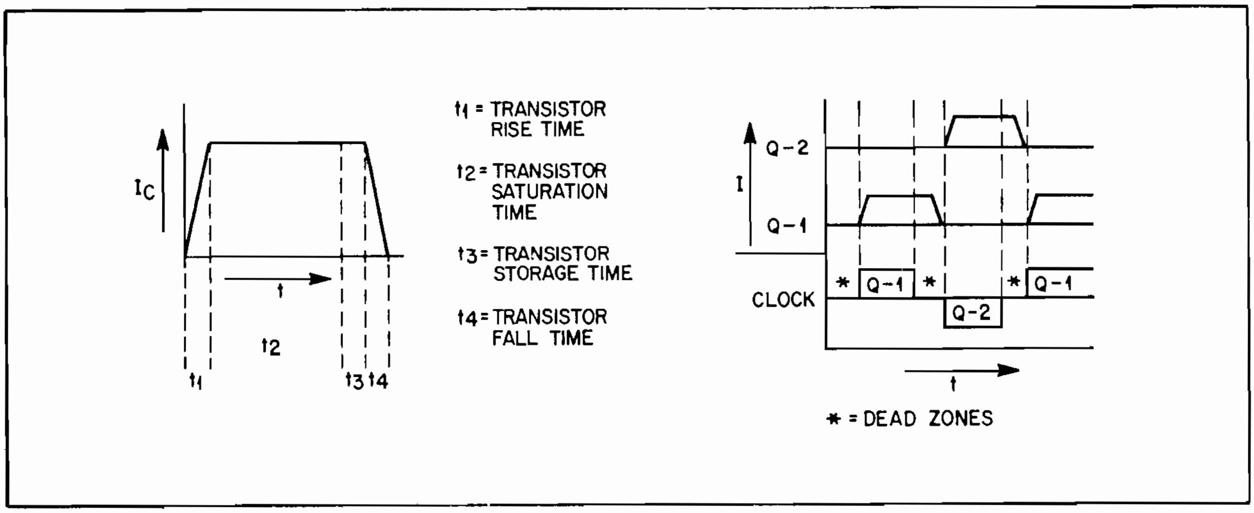

Ic
t
t1
t2
t3 t4
t1 = TRANSISTOR RISE TIME
t2 = TRANSISTOR SATURATION TIME
t3 = TRANSISTOR STORAGE TIME
t4 = TRANSISTOR FALL TIME
I
Q-2
Q-1
CLOCK
* Q-1 * Q-2 * Q-1
t
* = DEAD ZONES

FIG. 10 - BI-POLAR TRANSISTOR SWITCHING CHARACTERISTICS

This dead or transition zone must be guaranteed to remain at a constant time (t-3) irrespective of duty cycle. It also limits the duty cycle to less than 50% for any one of the switches. Another inherent disadvantage of this type of circuit is the extremely high Vceo required of the switching transistors. In the case of an off-line switcher, nominal voltage stress on the switches ignoring switching transient would be $(V_{acrms}) \sqrt{2}$ (2). For a 120 Volt ac input line, this would necessitate the use of a device with a Vceo of 340 Volts minimum. For 240 Volt operation the stress becomes 680 Volts, which is too high when considering the cost of transistors wi:h a breakdown characteristic of that order. When considering the transient condition, the voltage stress can be significantly worse without the use of a voltage snubbing device.

When one of the switching transistors turns off, the inductance of the associated transformer winding generates a voltage opposing the direction of current turn off. This spike appears across the transistor collector to emitter as shown in Figure #11.

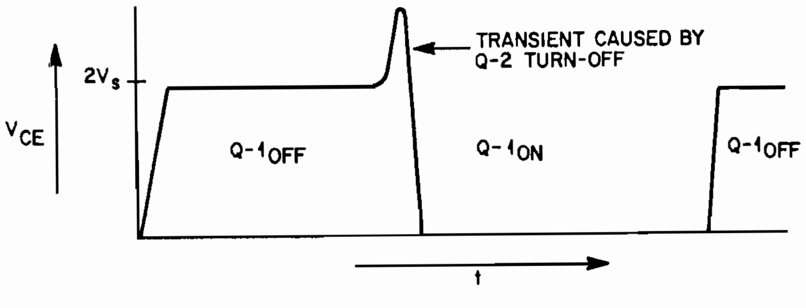


FIG. 11 - PUSH-PULL CIRCUIT TRANSISTOR VOLTAGE STRESSING CHARACTERISTICS

Not only is this a source of possible switching transistor failure, but it is also a source of RFI due to the high dv/dt and associated harmonic content of the spike.

There are two possible solutions to the excessive voltage stressing on push-pull circuit switches. The first circuit solution also reduces the possibility of transistor switching crossover destruction. The first solution requires the addition of a third transistor as shown in Figure 12.

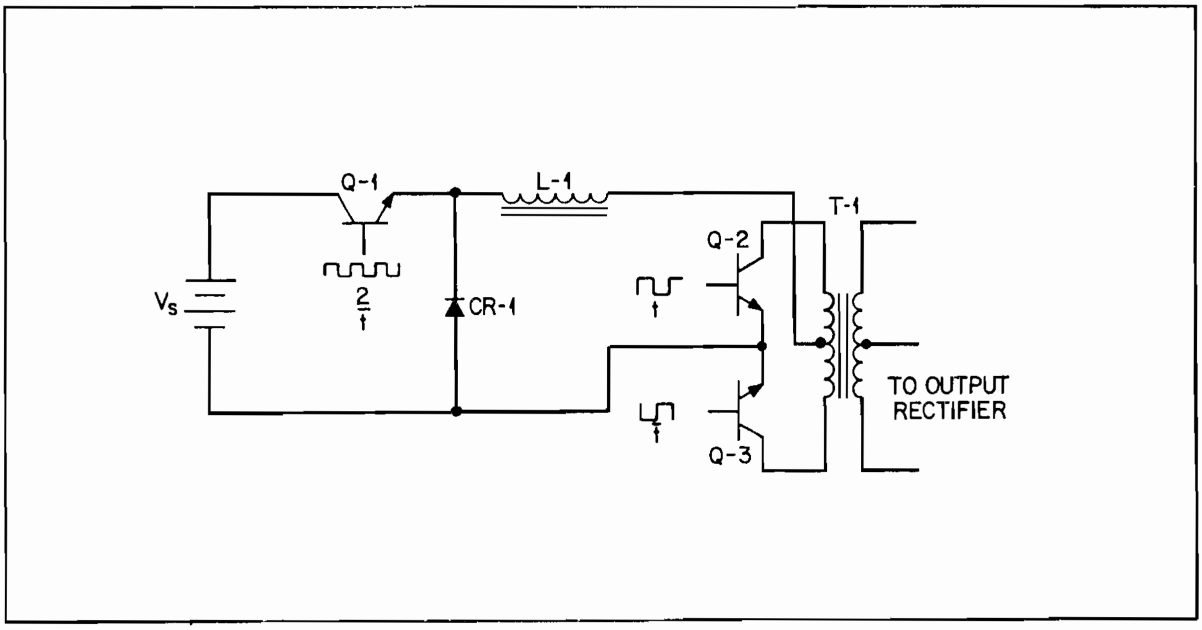


FIG. 12 - CURRENT DRIVEN PUSH-PULL TOPOLOGY

This third transistor is generally switched at twice the inverter switching rate and in sync with it. This acts as a current feed for the push-pull inverter, limiting the maximum current available to the main switches, due to the action of L-1 This topology also reduces the voltage stress on the inverter transistors. Because Q-1 can be connected in the voltage control loop, this topology maintains a comparatively smaller voltage across the push-pull transistors in the inverter circuit.

Another solution to excessive voltage stressing of the main switches employs a pre-regulator circuit of phase controlled architecture. For example, as in Figure 13, a triac could be inserted between the line and the input rectifier to reduce the inverter input voltage to a value less than the line voltage. A sample of the voltage input to the inverter circuit is compared to a fixed voltage reference. The error voltage (the difference between the two mentioned) establishes the appropriate firing angle for the triac to make the correction. This firing angle determines the peak voltage that appears on C-1. It should be noted, however, that this scheme does not solve the problem of output transistor switching crossover destruction, as C-1 appears as a voltage source.

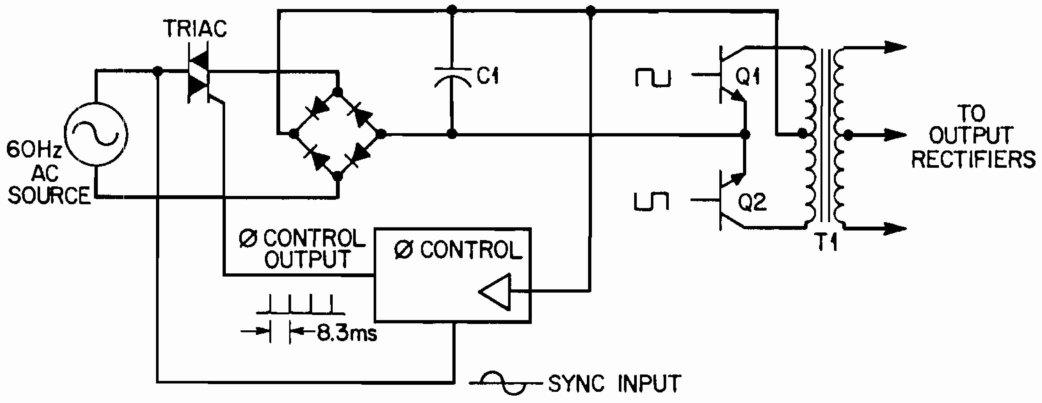


FIG. 13 - PHASE CONTROL PRE-REGULATOR

Another elementary switching regulator power mesh topology is the half bridge configuration shown in Figure 14.

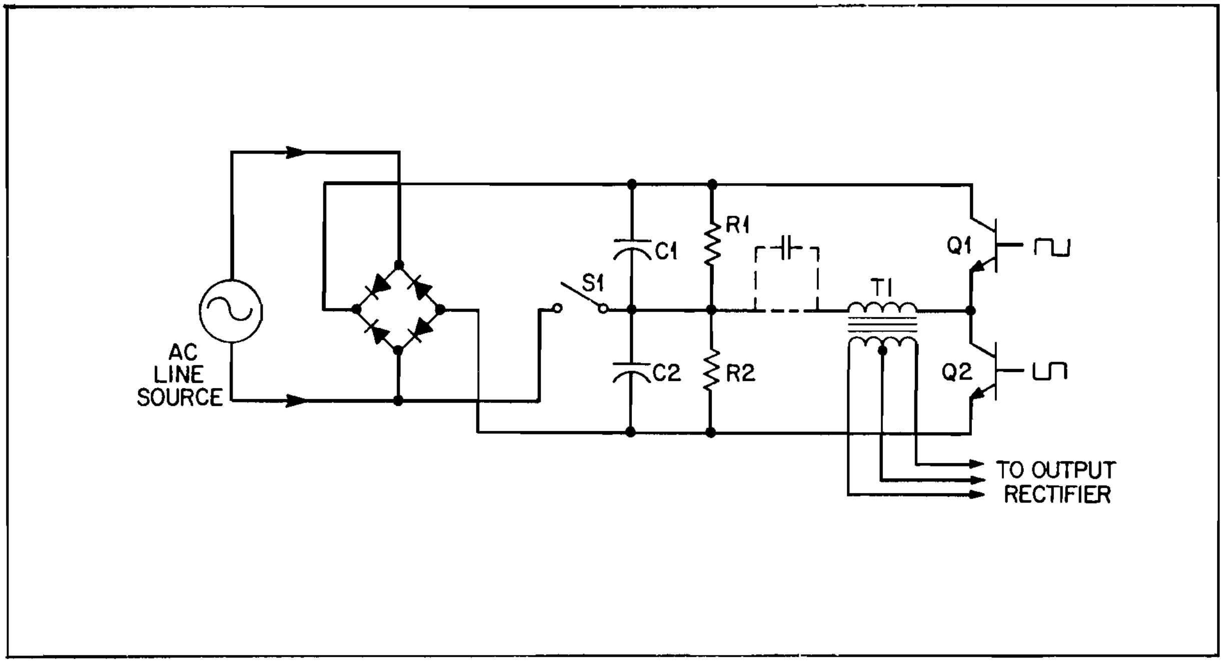


FIG. 14 - SWITCHING POWER SUPPLY HALF BRIDGE TOPOLOGY

It utilizes a split supply consisting of a rectifier, two filter capacitors C-1 and C-2, two voltage balancing resistors R-1 and R-2, two switches Q-1 and Q-2, and a non-center tapped transformer. Because of the use of the non-center tapped transformer, the voltage stress on the switches is half that of the push-pull configuration described earlier. For 240 Volt operation, S-1 is open and the rectifier operates as a standard bridge. For 120 Volt operation S-1 is closed and the rectifier in conjunction with C-1 and C-2 operates as a voltage doubler in a clamp and rectify mode of operation. In either case, 120 Volts or 240 Volt line input, the voltage presented to the switches is $\sqrt{2}$ (240) or 340 Volts, a significant improvement over the push-pull voltage stress. One problem encountered with this circuit results from a possible net voltage x time inbalance between one switch half cycle and the other. If an inbalance in pulse width, within a duty cycle is repeated a number of times, or an inbalance in transistor conduction characteristics exists between the two switching transistors, the core of the transformer could eventually "pump up" flux in the direction of the inbalance. Eventually this can result in transformer core saturation, leaving only intrinsic circuit resistances to limit switch current. This condition will result in overcurrent in the switches causing their destruction. Two solutions to this problem are possible. The first one involves the insertion of a capacitor in series with the transformer primary in order to block the dc offset created by any inbalance. The disadvantage to this solution is that the capacitor must have a large voltage and current rating, must have a small capacitive reactance at the frequency of switching, and must be bi-polar. These capacitors are generally large and costly. Another solution involves the cycle by cycle sampling of current in each switch. When an inbalance is detected in one leg, the adjacent transistors on time can be adjusted to generate a volt•second balance to correct for the transformer flux inbalance. This method can increase the complexity of the control circuit significantly, and may; therefore, become impractical in all but high power applications.

A variation on the theme of the half bridge is the full bridge configuration shown in Figure 15.

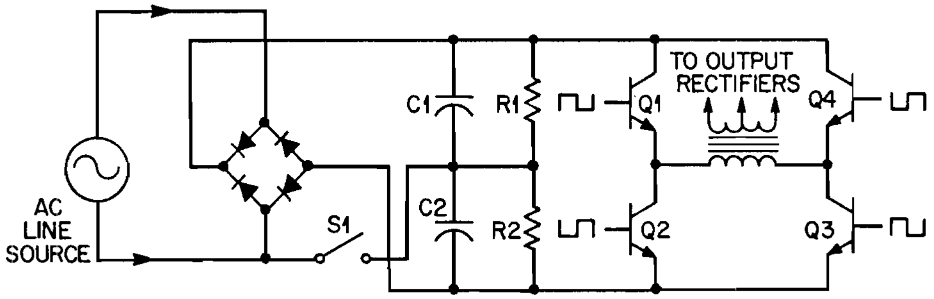


FIG. 15 - SWITCHING POWER SUPPLY FULL BRIDGE TOPOLOGY

The input rectifier and filter network functions identically to that of the half bridge input circuit. Both transistors Q-1 and Q-2 are triggered simultaneously, applying the rail voltage to T-1, with the dot side polarized in the positive direction. During the second half of the switching cycle, Q-2 and Q-4 are triggered, again applying the rail voltage to the transformer only with the dot side polarized in the negative direction. Due to the intrinsic symmetry of the bi-phasic voltage waveform applied to the transformer, transformer core saturation is not a problem. The peak voltage stress on the transistors is V line x $\sqrt{2}$ with S-1 open, or 340 Volts. This configuration, therefore, has many advantages over the others mentioned thus far with the exception that four switching transistors are required. The full bridge is more commonly found in high power switching circuits, ($>$ 1000 Watts), whereas the half bridge is found in low and medium power switching circuits.

Up to now, we have considered all of the multiple switching transistor topologies. We will now consider two more economical, single transistor solutions to switching regulator applications that of the flyback converter, and the forward feed converter.

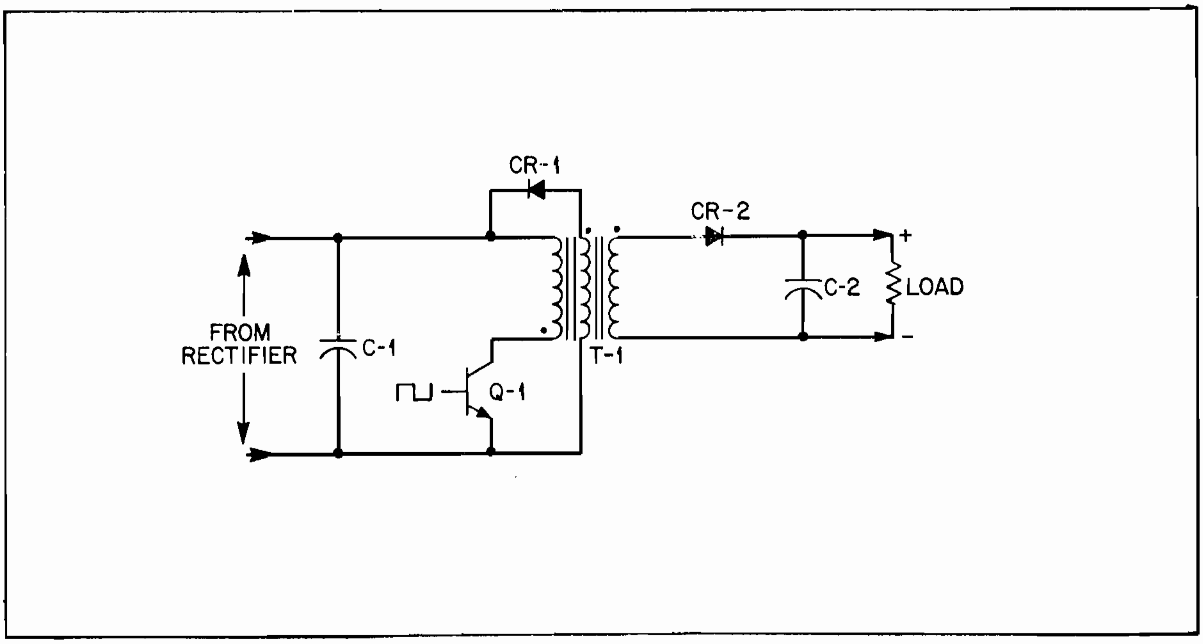


FIG. 16 - BASIC FLYBACK TOPOLOGY

When Q-1 is energized, energy from C-1 is dumped into T-1's primary and stored in its core. During this part of the switching cycle, flyback diode CR-1 and rectifier diode CR-2 are reverse biased. When Q-1 turns off, the tertiary and secondary winding of T-1 builds up a voltage in a direction opposing turn off, putting CR-1 and CR-2 into conduction. The tertiary winding then returns the unused energy which was stored in the transformer inductance to the supply capacitor during the Q-1 off state. In short, energy was stored in the transformer inductance during transistor turn on, and then released during the off state of the switching cycle. The voltage stress on Q-1 in the flyback circuit shown is $2\sqrt{2}$ (V line rms). The principle disadvantages to this realization are the poor RFI characteristics it exhibits as well as the high degree of ripple voltage contained in its output.

In the forward feed through circuit shown in Figure 17 energy is supplied to the load during the conductance state of Q-1.

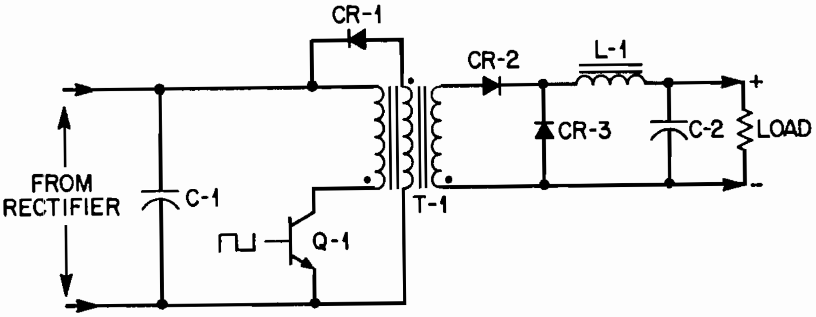


FIG. 17 - BASIC FORWARD FEED TOPOLOGY

When Q-1 is conducting, CR-2 conducts current into the L-C filter comprised of L-1 and C-2. During the off state, flyback diode CR-1 conducts, returning the demagnetizing current of T-1 back into C-1. Flywheel diode CR-3 maintains a path for current flow in the output circuit during the off state of Q-1. As with the flyback circuit, voltage stress on Q-1 is 2 $\sqrt{2}$ (V line rms). The principle advantage of this circuit over the flyback single transistor topology is the reduction of high frequency output ripple content due to the addition of L-1 in the output mesh.

One last topology, that of resonant conversion, deserves mention. Although this technique is not in common usage today, it shall be seen in the future, particularly as switching frequencies increase. Basically, the technique utilizes transistor switches to dump current into a resonant tank circuit which is similar to the topology of a class C RF amplifier circuit. By varying the frequency of the switching a varying envelope voltage will appear across the tank circuit. For example, with a series resonant tank, the voltage across the capacitor shall decrease with increased frequency when operating above the resonant frequency of the tank. In this type of system, energy is stored in the tank and as load demands increase, the switching frequency is brought closer to the resonant frequency of the tank, tending to regulate the output voltage. A chief advantage to this topology is the sinusoidal current waveform present inthe front end mesh. Squarewaves with their associated steep $\frac{di}{dt}$'s generates significant odd order harmonic RFI (H field), as compared to that of a relatively undistorted sinewave. With increased attention being payed to RFI emissions due to FCC and European regulations, more serious attention willbe paid to sine wave power conversion, in the near future.

With the exception of ferroresonance, all switching power supplies employ either limit cycle control, or slow feedback loop control, and some employ a combination of both to achieve output regulation. The simplest system employs limit cycle control, which is capable of high speed operation, and is, therefore, sometimes referred to as fast loop control. Basically a limit cycle system consists of a switch, an energy storage system, and a voltage detector as shown in Figure 18.

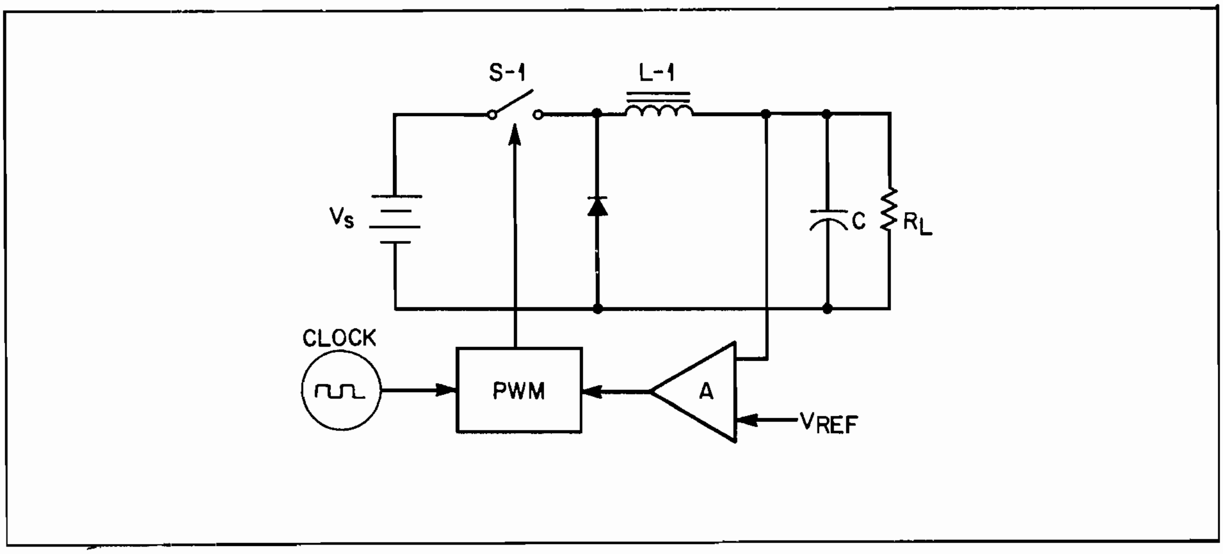


FIG. 18 - BASIC SINGLE ENDED LIMIT CYCLE CONTROL SCHEME

Operation is as follows: Each time a clock pulse occurs, switch S-1 is placed in the on state. Voltage comparator A monitors the output voltage, and when it exceeds V ref. it turns off S-1. Specifically this method is referred to as single ended limit cycle control, because switch turn on is initiated by a free running clock at a fixed frequency. A variation on the single ended limit cycle is the double ended limit cycle control. The only difference is that switch S-1 is not activated by a clock; it is activated when another comparator sees an output voltage below a pre-set reference. In this type of control, switching frequency shall vary as a function of load, whereas the single ended limit cycle operates at a constant frequency. A big advantage to the limit cycle control, is that instability, is a normal function of operation resulting in the fact that the feedback loop does not need compensation. The result is that transient response is excellent with this system because this parameter is primarily a function of the output L-C filter characteristics.

A second system of control involves the use of a slow loop or proportional control. This circuit takes and average value of the output voltage and establishes a pulse width which would provide the necessary voltage output for zero correction in the comparator circuit. This system is very difficult to stabilize, unless the unity gain crossover occurs fairly low in frequecny. This limits the load effect transient recovery response of the supply, but can provide more precise regulation than the limit cycle control.

In order to get the best of both worlds, a combination of the limit cycle and slow loop control may be used. Basically, a second L-C filter is installed after the limit cycle. The output of the second filter is also the power supply output, and is connected to a comparator. This comparator is designed to have a substantially limited upper bandpass and provides the voltage reference for the limit cycle control cycles.

Control system theory in switching power supply design is a rigorous study. Many trade offsare carefully weighed when designing the control loop, and the details of its design are beyond the scope of this paper. As we see advances in semiconductor technology, especially in the area of high speed high power MOSFET's, component selection shall become increasingly more critical especially in the area of transformer specification and capacitor ESR (Effective Series Resistance) characteristics. Adding another design plaque to the switching power supply Engineer, are the increasingly more stringent RFI standards which must be complied with. The net result, is that unlike linear supply design, switching power supply design is anything but trivial.

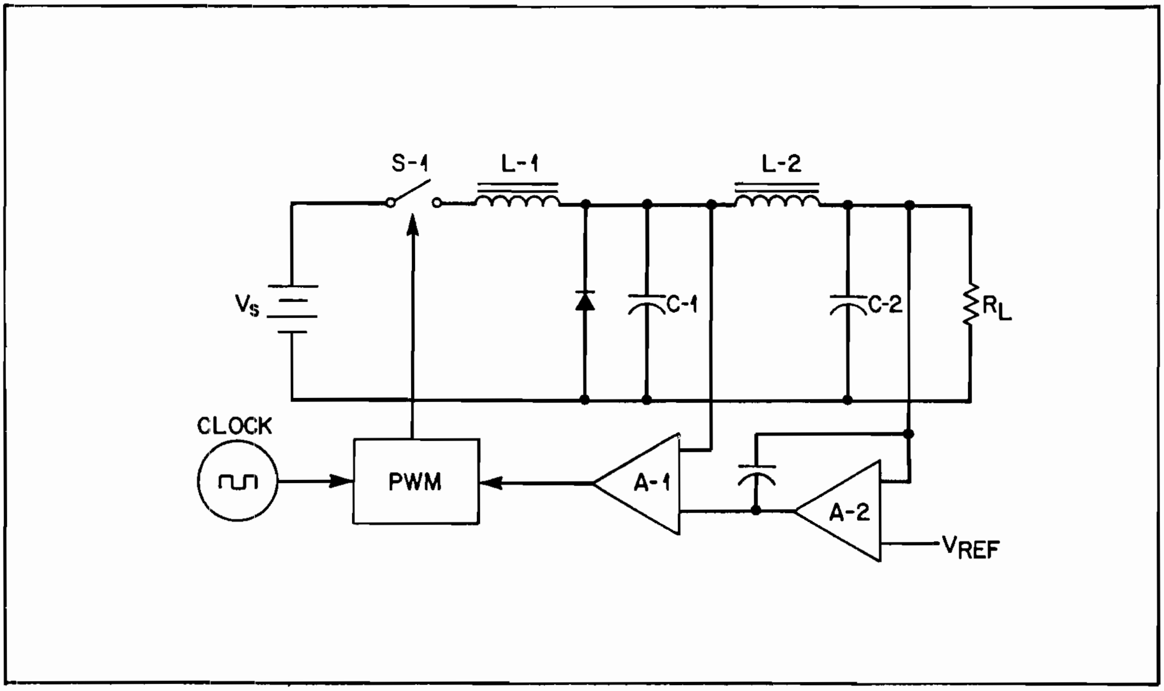


FIG. 19 - LIMIT WITH SLOW LOOP CONTROL SCHEME

11. Digital Test Alternatives - An Economic Model.

HEWLETT PACKARD

Ed White holds B.S.E.E. and MBA degrees from Stanford. Ed is a Product Manager in logic test equipment at the Santa Clara Division.

SECTION A—INTRODUCTION

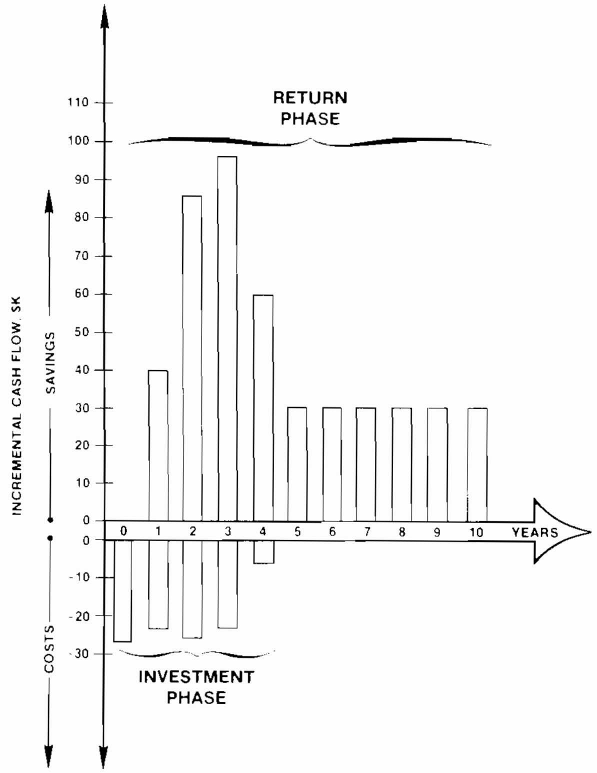
The costs of troubleshooting digital products in final assembly and field service are becoming increasingly visible, and are often perceived to be out of control. Strategies have been advanced which reduce these costs, but which also involve some initial investment in:

- Product Setup
- Test Equipment
- Documentation
- Materials
- Combinations of These Elements

For example, in order to implement Signature Analysis,[1] and take advantage of its savings in labor, processing and test equipment, a product usually needs to be set up, by design or retrofit, to utilize the technique. Therefore, the management decision to adopt or change a test/service strategy for a digital product hinges on the question:

> Will expected cost reductions in final assembly and field service earn sufficient return on the setup investment, and how do the returns for different strategies compare?

1. **Return on Investment.** The comparison of two digital test/service strategies can be considered a return on investment (ROI) exercise. The incremental costs of one strategy over the other are negative cash flows during the *investment* phase of the project. The incremental savings of that strategy are positive cash flows during the *return* phase of the project.



There are several common ROI calculations which allow comparisons of cash flows. This paper utilizes IRR (internal rate of return).

2. **Costs and Savings.** While the ROI calculation is straightforward, the estimation of the cash flows (costs and savings) is not. Existing costs are difficult to measure and proposed savings are difficult to predict. This paper attempts to simplify the exercise by offering some rules of thumb for cost/saving estimation. The rules are very conservative, resulting in higher costs and lower savings than our experience indicates. The effect is a tough comparison, assuring that adoption of a strategy will earn the target ROI.

3. **Model.** The model, then, consists not only of the ROI calculation, but also of a set of simple guidelines for cash flow estimation. It should allow comparison of alternatives in very short study times (one to two days), with minimum research. It is suitable for a wide range of digital products. All calculations are performed on a pocket calculator.

4. **Sample Product.** The paper presents the model via a sample product. The product represents a composite of our experience on hundreds of applications at Hewlett-Packard and other companies.

5. **Organization.** In order to help the reader apply the model to other products, the paper is organized as follows:

Section B — Initial assumptions on the sample product which affect outcome:

- Product Type
- Selling Price
- Cost Structure
- Production Life
- Service Life
- Number of Parts
- Forecast
- Field Failure Rate

Section C — Alternatives to be analyzed and their flow charts:

- Before (or current)
- After (or new)

Section D — Analysis of incremental costs:

- Engineering
- Documentation
- Test Equipment
- Component Stock
- Ongoing Materials

Section E — Analysis of incremental savings.

- Production Labor
- Warranty
- Field Service

SECTION B — INITIAL ASSUMPTIONS

In order to build a realistic economic model, we selected a sample product, and analyzed all of the cost and savings considerations for its digital test and service. So that the process may be applied to other products, this section details the assumptions we made concerning the sample product. It also discusses ways in which these assumptions may affect the results, if varied to accommodate other products. The section concludes with a summary table of the assumptions, with space to state assumptions for another product.

1. **Product Type.** We chose a relatively sophisticated graphic terminal as the sample product, because it incorporates a wide variety of digital troubleshooting challenges: microprocessor, ROM, dynamic RAM, keyboard, CRT controller, character generators, communication ports, etc. The analysis has been applied equally well to both simpler and more complex products.
2. **Selling Price.** The sample product sells for $5,000. The analysis has been used on products ranging from a $300 instrument to a $100,000 ATE system.
3. **Cost Structure.** Any analysis of cost savings depends heavily on the cost breakdown of the product. The larger an existing cost category is, the higher the impact of savings in that category on ROI. For the sample product, we estimated each cost element on the low side, in order to be conservative and lessen the impact of cost savings on ROI.

ELEMENT	COST
Direct Material	$1,000
Direct Labor	$ 250 (25 hours)
Factory Overhead	$ 750
Research and Development	$ 500
Marketing	$ 400
Sales and Service	$ 500
Warranty	$ 100
Other	$1,000

4. **Production Life.** The sample product has an estimated production life of four years. The longer the production life, the greater the impact of cost savings on the ROI model. However, the earlier years have the greatest impact on ROI.
5. **Service Life.** The time during which a product is supported in the field, after production is discontinued, depends on company policy. We assumed a ten-year service life. However, the length of this period has only a minor impact on the ROI calculation.
6. **Number of Parts.** The sample product has 315 IC's, distributed over 5 PC boards. The largest board has 150 IC's, the smallest has 20. There does not appear to be any upper limit on the number of IC's involved. However, it is difficult to show any savings on products of less than 5 IC's.
7. **Sales Forecast.** Assumption of an annual volume forecast over the product life is necessary in order to calculate production, service, and warranty costs and savings. The sample product uses this forecast:

YEAR	QTY SHIPPED	$ VOLUME	SERVICE BASE	WARRANTY BASE
0	0	0	0	0
1	600	$3M	600	600
2	1200	$6M	1800	1200
3	1200	$6M	3000	1200
4	600	$3M	3600	600
5-10	0	0	3600	0

The analysis has been used on products with both higher and lower volumes.

8. **Field Failure Rate.** We assumed a yearly failure rate of 10% of the total installed base for the 10-year service life of the sample product. This is conservatively low, since the higher the failure rate, the greater the savings which can be realized in cutting service costs.

YEAR	FAILURES	IN WARRANTY	OUT OF WARRANTY
0	0	0	0
1	60	60	0
2	180	120	60
3	300	120	180
4	360	60	300
5-10	360	0	360

SUMMARY OF MODEL ASSUMPTIONS

Item	Sample Product	Product Under Study
1. Product Type	Terminal	
2. Selling Price	$5,000	
3. Cost Structure		
Material	$1,000	
Labor	250	
Overhead	750	
R&D	500	
Marketing	400	
Sales/Service	500	
Warranty	100	
Other	1,000	
4. Production Life	4 Years	
5. Service Life	10 Years	
6. Number of Parts		
Boards	5	
IC's	315	
7. Forecast		
Year 1	600 units	
Year 2	1200 units	
Year 3	1200 units	
Year 4	600 units	
8. Field Failure Rate	10% Per Year	

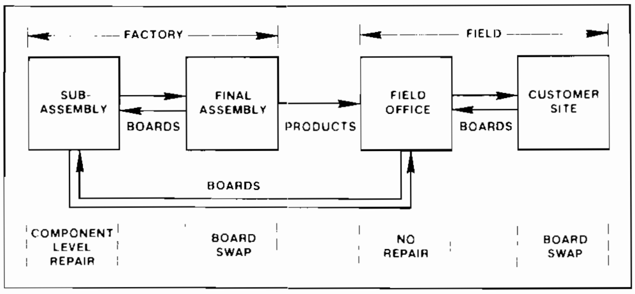
SECTION C — ALTERNATIVES TO BE ANALYZED

The analysis may be used to generate the incremental ROI between any two alternative digital test and service strategies. We chose to compare two strategies which meet these criteria:

1. **Before:** The most common strategy now used by companies with digital products.
2. **After:** The simplest, first-step improvement over the current strategy.

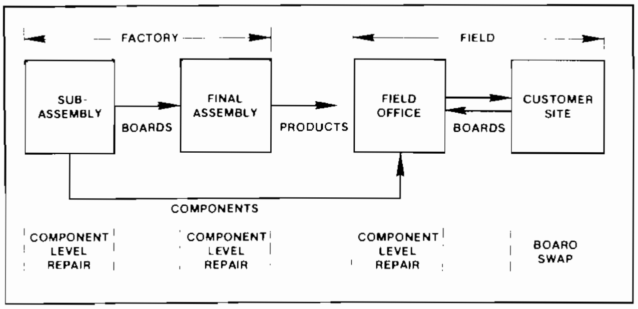
Here is an outline of each of the alternatives studied.

1. **Before:**



This alternative represents the most common strategy for digital test and service. Products which fail in the final assembly area (turn-on, heat-run, final test, QA, etc.) are repaired by swapping PC boards. Bad boards are returned to a subassembly test area for component level troubleshooting and repair. Products which fail in the field are repaired by swapping PC boards at the installation site. Bad boards are returned to the subassembly test area, via the field office, for component level troubleshooting and repair.

2. **After:**

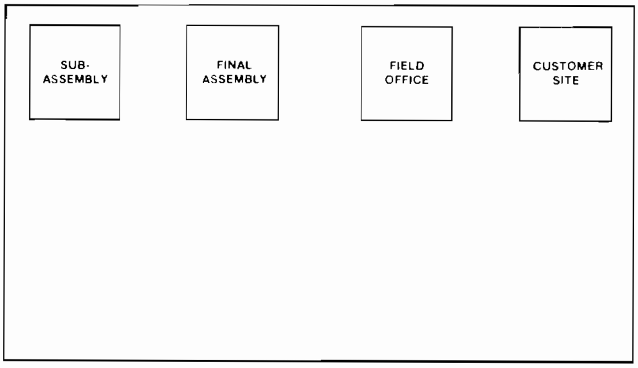


This alternative represents a strategy which could be implemented by setting up the product to be repaired to the component level. Products which fail in the final assembly area are troubleshot and repaired to the component level, without disassembly. Products which fail in the field are repaired by swapping PC boards at the installation site. Bad boards are returned to the field office for component level troubleshooting and repair. No boards return to the factory. We implemented this strategy on the sample product by incorporating the Signature Analysis technique. This required some incremental expenses, over and above what we would have spent to set up the product for straight board-swap repair. However, the savings in repair labor, materials and handling yielded a very attractive ROI. Expenses and savings are detailed in the following sections.

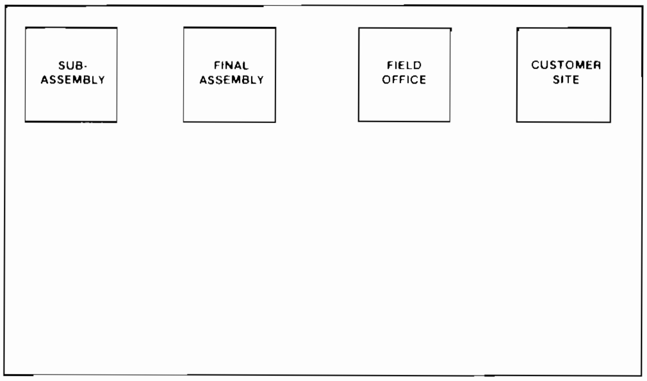
Note: This alternative specified board-swap repair on-site. Additional savings could be generated, with no additional costs, by repairing products on-site, to the component level. There are many examples of this. However, we chose not to take advantage of these savings in the model, since many installation sites are not suitable for replacing components on boards.

3. **Product Under Study.** Here is some space to model comparative test/service strategies for another product.

Before:



After:



Note: Just fill in the appropriate product flows to describe each strategy.

SECTION D — ANALYSIS OF INCREMENTAL COSTS

In order to implement the new strategy for digital test and service in the sample product, we incurred some incremental costs, compared to those for the current strategy:

1. Engineering
2. Documentation
3. Test Equipment
4. Component Stock
5. Ongoing Materials

Here is a detailed analysis of each of the incremental cost areas. The results are summarized at the end of the section, and space is provided to analyze incremental costs for another product.

1. **Engineering**

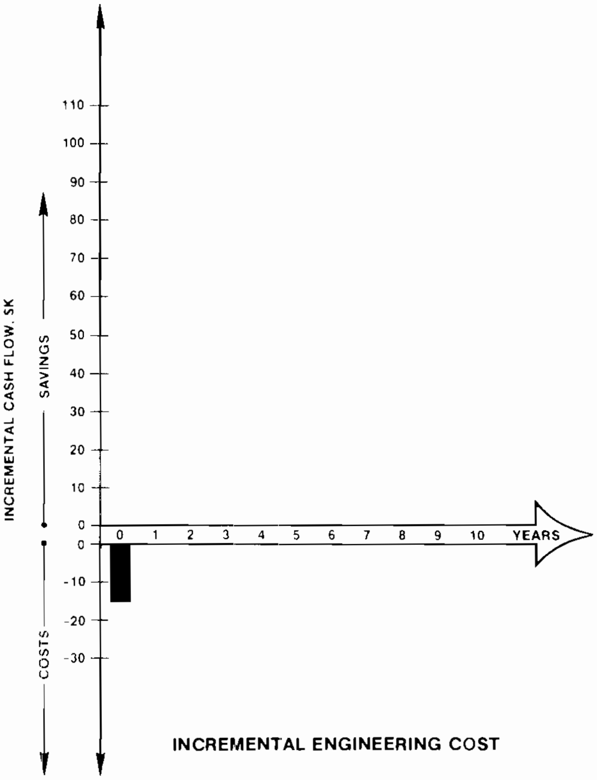
a. **Description.** In order to set up a product to be repaired to the component level with Signature Analysis, some engineering time is required. The time is devoted to minor hardware re-layout and software modification. If implemented in the design phase, the design team handles it. If implemented as a retrofit, the time generally is spent in the manufacturing engineering area.

b. **General Rule.** A good, conservative rule is to use 1% incremental engineering time.

c. **Actual Experience.** We have reported estimates of 1-4 man-weeks of incremental engineering time. This rarely amounts to 1% of the entire design project. The time appears to be the same, whether spent in design or in retrofit. However, there is room for reporting error here, since these are estimates.

d. **Sample Product.** This product required the equivalent of 5 engineers (all types) for 3 years of design. This is 180 man-months, total. Of this, assume that 2 man-months (1%) were devoted to setting up the product for component-level repair. We used a conservatively high figure of $7,500 per month for fully loaded design time, for a total incremental cost of $15,000.

e. **Timing.** The engineering time is spent at the end of the design cycle, just before production. We show it in year 0.



INCREMENTAL ENGINEERING COST

2. **Documentation**

a. **Description.** Troubleshooting procedures are more thorough for component level repair, than for board-swap. Signature Analysis greatly reduces the burden: however, there is some incremental time involved:

- gathering signatures.
- writing procedures.
- verifying both.

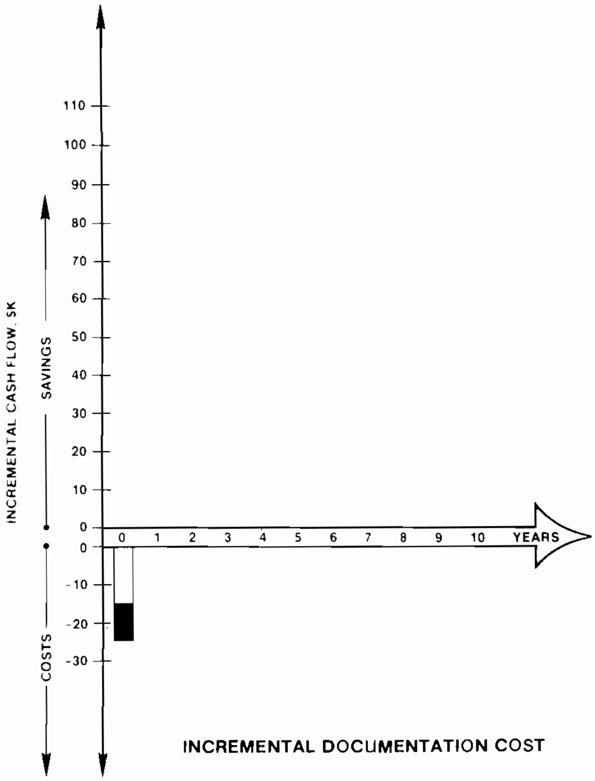
The time is spent in product support engineering, manufacturing test engineering or service engineering.

b. **General Rule.** A good, conservative guideline is to use 2 man-months for incremental documentation time.

c. **Actual Experience.** We have reported estimates of 2-4 man-weeks of incremental documentation time, so the general rule is quite conservative. Remember that this is only the incremental time spent to add Signature Analysis instructions to the procedures which would have been prepared under the current strategy.

d. **Sample Product.** Two man-months of documentation effort were targeted at a loaded cost of $5,000 per month, for a total incremental cost of $10,000.

e. **Timing.** The documentation time is spent in the pre-production and early production life of the product. We show it in year 0.



INCREMENTAL DOCUMENTATION COST

3. **Test Equipment**

a. **Description.** Utilizing the Signature Analysis technique requires purchase of Signature Analyzers, at $1,000 each, in these locations:

(1) Lab — for aiding in the design or retrofit of Signature Analysis into the product.

(2) Service or Test Engineering — for documenting the troubleshooting procedure.

(3) Production — for troubleshooting in the final assembly area.

(4) Field Offices — for field service troubleshooting of returned boards.

b. **General Rule.** Plan on one Signature Analyzer for the lab for design, and one for service/test engineering for documentation. In the final assembly area, plan on one unit per test position or, if the unit can be shared, one per 3 final test technicians. In the field, plan on one unit per office, initially. (The upper limit for field service will vary. One guideline is to put one unit in the field for each field service technician. Another guideline would be to put one unit into each field office for each 50 projected repairs per year.) Each Signature Analyzer, of course, is usable on any future products which utilize the same technique.

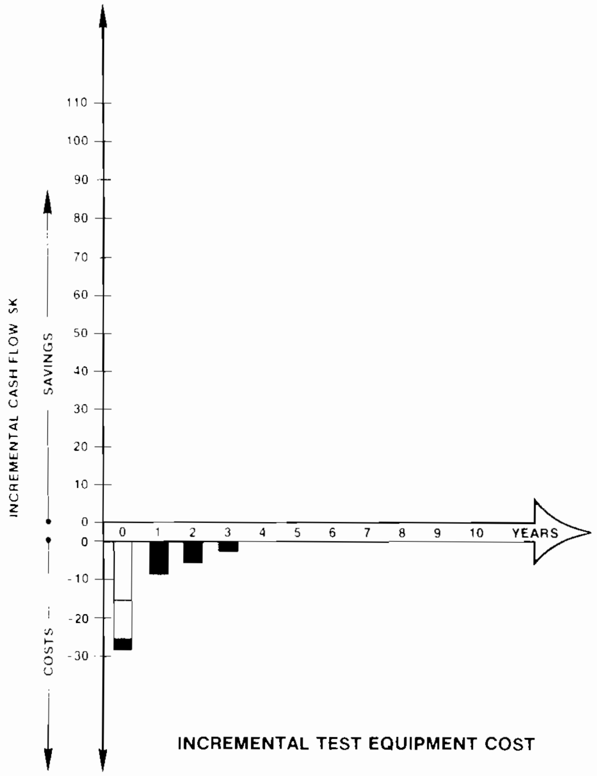
c. **Actual Experience.** Individual companies have implemented component-level repair programs, utilizing Signature Analysis, with from 5 to 400 Signature Analyzers. We find the general rules, above, conservatively high.

d. **Sample Product.** For the sample product, we used one Signature Analyzer each, in design and service/test engineering. We used 5 units to cover 10-12 final test technicians in production, and used 10 units to cover the five offices for field service. The total was 17 units, or $17,000.

e. **Timing.** These 17 Signature Analyzers were acquired over 4 years, as shown.

INCREMENTAL SIGNATURE ANALYZER COST

		YEARS					
LOCATION	**UNITS**	**0**	**1**	**2**	**3**	**4**	**5-10**
Lab	1	1	0	0	0	0	0
Service/Test Engineering	1	1	0	0	0	0	0
Production	5	0	3	2	0	0	0
Field Service	10	0	5	3	2	0	0
Total Units	17	2	8	5	2	0	0
Cost @ $1,000	$17,000	$2,000	$8,000	$5,000	$2,000	0	0



INCREMENTAL TEST EQUIPMENT COST

4. **Component Stock**

a. **Description.** Since the new strategy calls for component-level repair in field offices, we require parts kits, which would not have been stocked currently. These are startup parts kits, which are added during the life of the product, as required by the growing installed base. We do not show factory parts stock here, since it is approximately the same under either strategy.

b. **General Rule.** Itemize one of each IC per kit. Plan on enough kits to handle 2 months' failures, at the predicted mature failure rate. This is very conservative, since successive failures rarely require the same part at the same location.

c. **Actual Experience.** Most service groups have found that they do not require all electronic parts to be in a kit, and that they can understock multiple usage parts. Actual stocking requirements come in well under the generai rule.

d. **Sample Product.** The sample product has 315 IC's (Section B-6), the costs of which are:

300 IC's at $1.00 each = $300.00
15 IC's at $10.00 each = $150.00

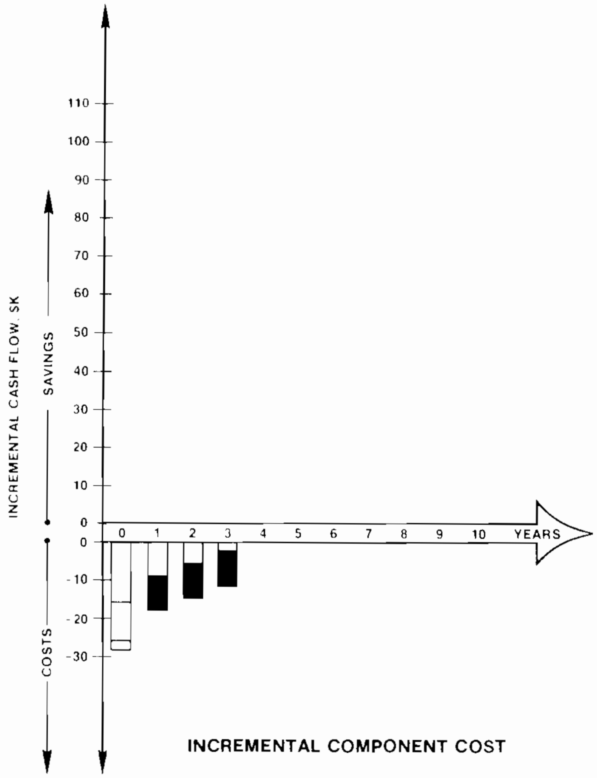
Total cost per kit $450.00

The mature failure rate was projected at 30 per month (Section B-8), so two months' failures are covered by 60 kits, or $27,000.

e. **Timing.** We acquire the 60 kits over a period of 3 years, keeping well ahead of the projected installed base, as follows:

INCREMENTAL COMPONENT COST

		YEARS					
	QTY REQUIRED	**0**	**1**	**2**	**3**	**4**	**5-10**
Field Parts Kits	60	0	20	20	20	0	0
Kit Costs at $450.00	$27,000	0	$9,000	$9,000	$9,000	0	0



INCREMENTAL COMPONENT COST

5. **Ongoing Materials**

a. **Description.** When setting up a product for Signature Analysis troubleshooting, there are generally some additional parts required in the product itself. These usually consist of switches, jumpers, test points, a socket, etc. (Active components are rarely required.) These parts then contribute to a small incremental material cost.

b. **General Rule.** Add 1% of the standard material cost.

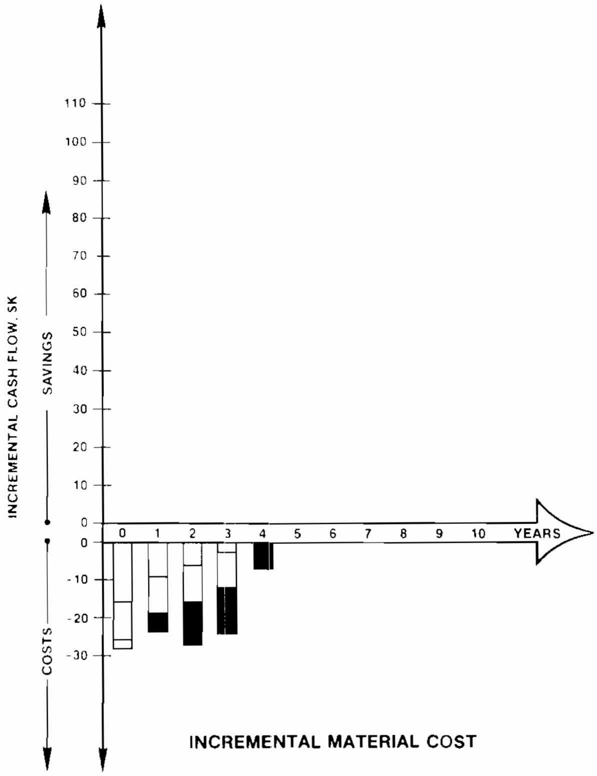
c. **Actual Experience.** Incremental material costs (assuming no additional ROM space is required) range from 0-$5.00 per unit.

d. **Sample Product.** Using the 1% rule on the $1,000 material cost (Section B-3), we have a conservatively high incremental material cost of $10.00 per unit. This is then applied to the forecast (Section B-7) to obtain the annual incremental cost.

e. **Timing.** Material costs are incurred as follows:

INCREMENTAL MATERIAL COST

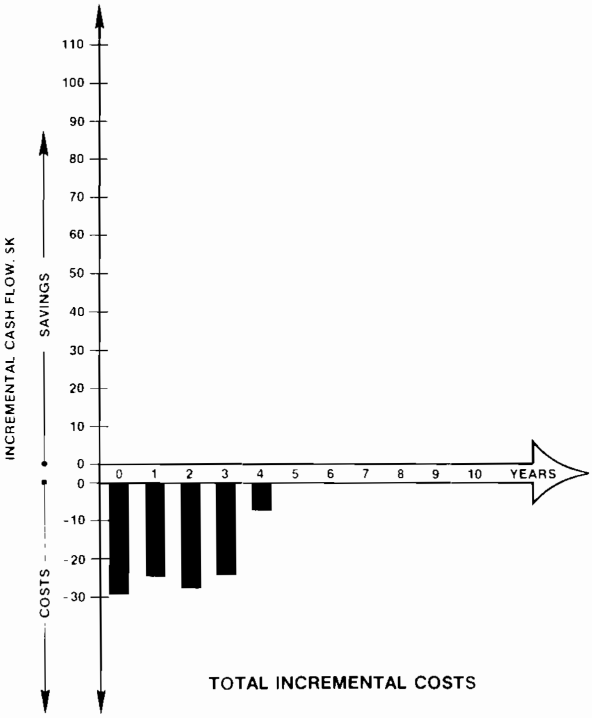
	YEARS					
	0	1	2	3	4	5-10
Forecast, Units	0	600	1,200	1,200	600	0
Annual Cost at $10.00	0	$6,000	$12,000	$12,000	$6,000	0



INCREMENTAL MATERIAL COST

SUMMARY OF INCREMENTAL COSTS

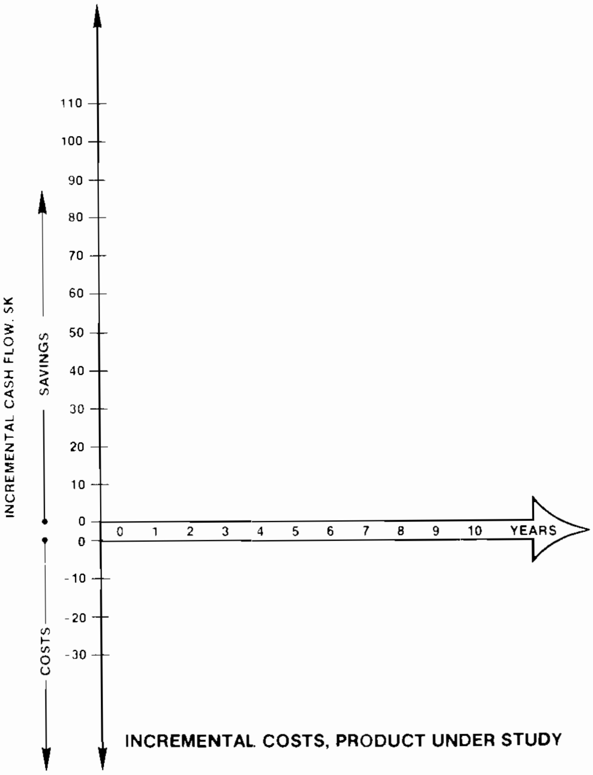
	YEARS					
COST ITEM	0	1	2	3	4	5-10
1. Engineering	15,000	0	0	0	0	0
2. Documentation	10,000	0	0	0	0	0
3. Test Equipment	2,000	8,000	5,000	2,000	0	0
4. Component Stock	0	9,000	9,000	9,000	0	0
5. Ongoing Materials	0	6,000	12,000	12,000	6,000	0
Totals	$27,000	$23,000	$26,000	$23,000	$6,000	0



TOTAL INCREMENTAL COSTS

INCREMENTAL COSTS, PRODUCT UNDER STUDY

	YEARS					
COST ITEM	0	1	2	3	4	5-10
1. Engineering						
2. Documentation						
3. Test Equipment						
4. Component Stock						
5. Material						
Totals						



INCREMENTAL COSTS, PRODUCT UNDER STUDY

SECTION E — ANALYSIS OF INCREMENTAL SAVINGS

By implementing the new digital test/service strategy, we experience significant incremental savings over the current strategy, in these cost areas:

1. Production Labor
2. Warranty
3. Field Service
4. PC Board Stock

Here is a detailed analysis of each of the incremental savings areas. The results are summarized at the end of the section, and space is provided to analyze incremental savings for another product.

1. **Production Labor**

a. **Description.** The new strategy utilizes Signature Analysis to generate substantial savings in the final assembly troubleshooting area. This is because a final test technician can now make component level repairs in about the same time as it formerly took just to locate and verify a bad board.

b. **General Rule.** Isolate the average time per unit normally spent on troubleshooting and repairing an assembled product on the line, and reduce it, 2:1.

c. **Actual Experience.** Most cases do not generate "before/after" data since, when Signature Analysis is employed, the "before" case is never practiced. However, those cases we have studied show time improvements from 4:1 to 8:1. Therefore, the general rule, above, is very conservative.

d. **Sample Product.** Using the 2:1 rule, and a cost breakdown for the sample product (Section B-3), we calculate production labor savings of $50.00 per unit.

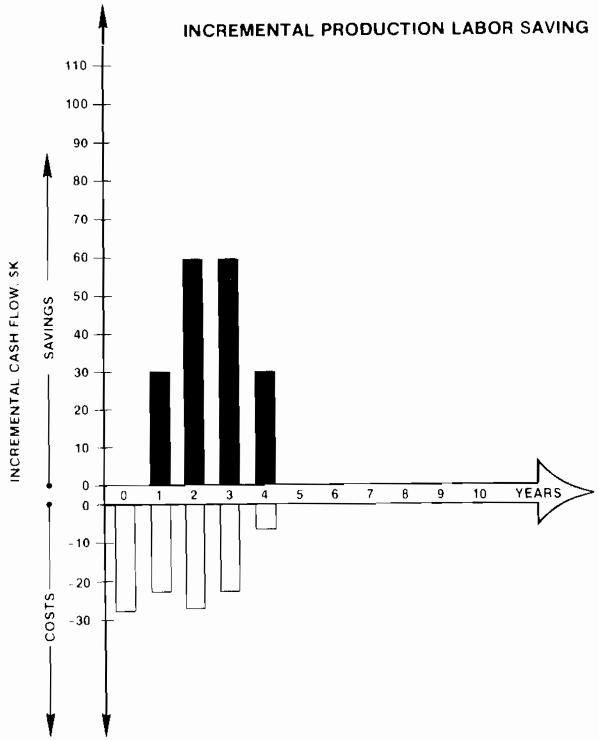
	HOURS PER UNIT		
TYPE OF LABOR	BEFORE	AFTER	SAVED
SUBASSEMBLY	4	4	0
FINAL ASSEMBLY	6	6	0
TEST	5	5	0
TROUBLESHOOTING	10	5	5
TOTAL HOURS	25	20	5
COST AT $10.00 PER HOUR	$250.00	$200.00	$50.00

This unit saving is then applied to the forecast (Section B-7) to obtain the annual incremental saving.

e. **Timing.** Production labor savings are realized as follows:

INCREMENTAL PRODUCTION LABOR SAVING

	YEARS					
	0	1	2	3	4	5-10
FORECAST	0	600	1,200	1,200	600	0
ANNUAL SAVINGS AT $50.00	0	$30,000	$60,000	$60.000	$30,000	0



INCREMENTAL PRODUCTION LABOR SAVING

2. **Warranty**

a. **Description.** By repairing PC boards in the field, instead of at the factory, we generate savings in the operation of a board exchange program in the areas of:

- inventory.
- administration.
- logistics/distribution.
- no-trouble-found boards.

The repair cost reduction impacts both warranty and field service savings.

b. **General Rule.** This can be a complex area in which to make estimates. However, the following formula is conservatively safe:

Savings per repair =
Average board exchange price (before).
Less: Average repair parts price (after).
Less: Incremental field repair labor (after).

Some rules of thumb are:

(1) Average board exchange price - $100.00.
(2) Average repair parts price - $10.00.
(3) Average incremental field repair labor - 1 hour = $10.00.

Therefore, a conservatively low saving figure would be $80.00 per repair. This assumes that the former board exchange fee was set in such a way as to just cover the costs of running the program.

c. **Actual Experience.**

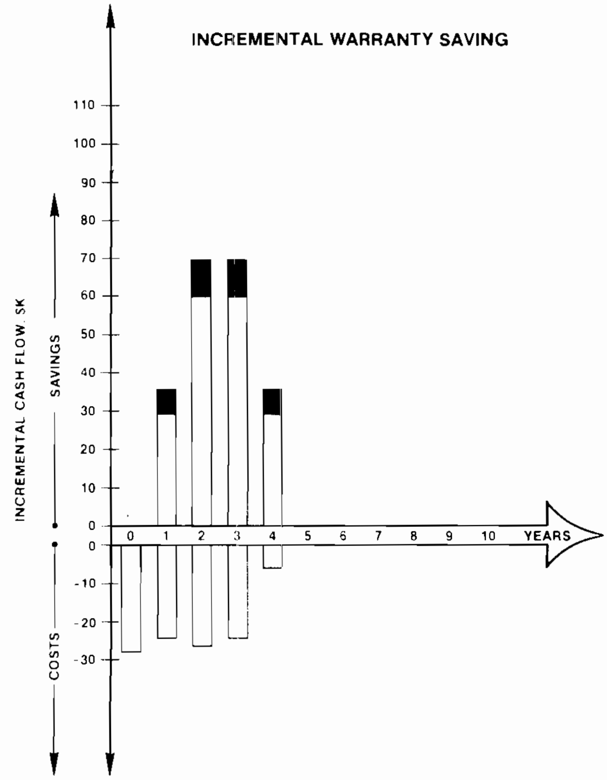
(1) Board exchange prices range from $50.00 to $500.00.
(2) Repair component prices range from $.50 to $50.00.
(3) Incremental labor is often well below the minimum charge, since boards can usually be repaired at the field office in the same time it formerly took to verify the bad board, process paper work and handle shipping.
(4) The $80.00 figure is conservative.

d. **Sample Product.** We applied the $80.00-per-repair figure to the projected warranty failure rate (Section B-8) to obtain the annual incremental saving.

e. **Timing.** Warranty savings are realized as follows:

INCREMENTAL WARRANTY SAVING

	YEARS					
	0	1	2	3	4	5-10
WARRANTY FAILURES	0	60	120	120	60	0
SAVINGS AT $80.00	0	$4,800	$9,600	$9,600	$4,800	0



3. **Field Service**

a. **Description.** Same as 2-a, above.

b. **General Rule.** $80.00 per repair.

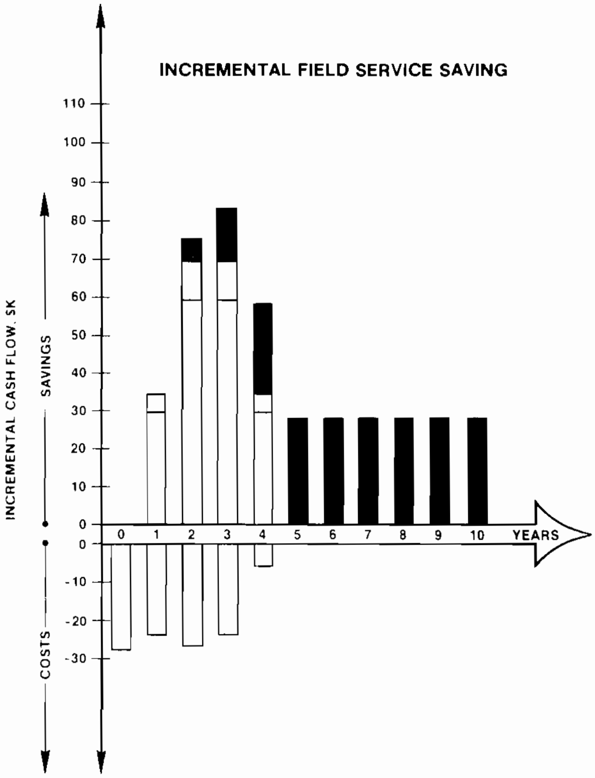
c. **Actual Experience.** Same as 2-c, above.

d. **Sample Product.** We applied the $80.00-per-repair figure to the projected non-warranty failure rate (Section B-8) to obtain the annual incremental saving.

e. **Timing.** Field service savings are realized as follows:

INCREMENTAL FIELD SERVICE SAVING

	YEARS					
	0	1	2	3	4	5-10
NON—WARRANTY FAILURES	0	0	60	180	300	360
SAVINGS AT $80.00	0	0	$4,800	$14,400	$24,000	$28,800



4. **PC Board Stock**

a. **Description.** The new strategy requires parts kits in field offices for component-level repair (Section D-4). Because of that, fewer board kits will be required in field offices. Board kits will be required only for on-site repair, with the bad boards being repaired at the field office. The reduction in startup board kits constitutes an incremental saving for the new strategy.

b. **General Rule.** Calculate the number of board kits required under the current strategy, enough to cover 2 months' failures at the projected mature failure rate. Calculate the number of board kits required under the new strategy, about one-third of the current figure. The difference in numbers of kits is the saving. A good approximation of kit cost is the product's material cost. Usually, with Signature Analysis, a stripped mainframe of the product is required in each office, in order to power the board for troubleshooting. So, from the board kit saving, be sure to deduct the cost of a mainframe for each office. Again, the product material cost should cover this.

c. **Actual Experience.** Reports indicate that the reduction of board kits is usually more than outlined above. The current strategy often requires 3 months' failure coverage, due to pipeline and turnaround problems. The new strategy often requires only 1 board kit per field office.

d. **Sample Product.**

(1) **"Before"** — Projected mature failure rate is 30 per month (Section B-8). Two months' coverage would be 60 kits. Cost per kit (material cost in Section B-3) is $1,000, for a total of $60,000.

(2) **"After"** — One-third of the "before" figure is 20 kits, or $20,000.

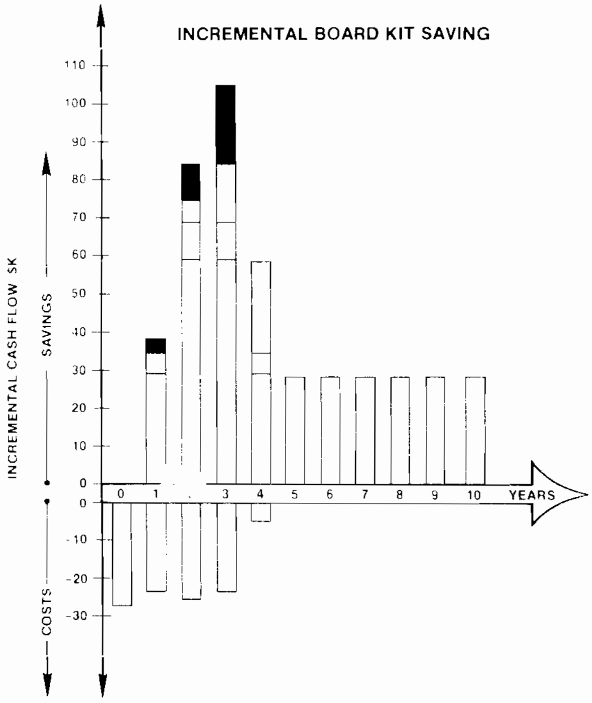
(3) **Mainframes** — One stripped mainframe, at material cost, for each of the five field offices amounts to $5,000.

(4) **Total Saving** — $60,000 less $20,000, less $5,000 = $35,000.

e. **Timing.** The $35,000 saving in board kits is distributed over 3 years, as the kits would have flowed into the field pipeline with the growing service base.

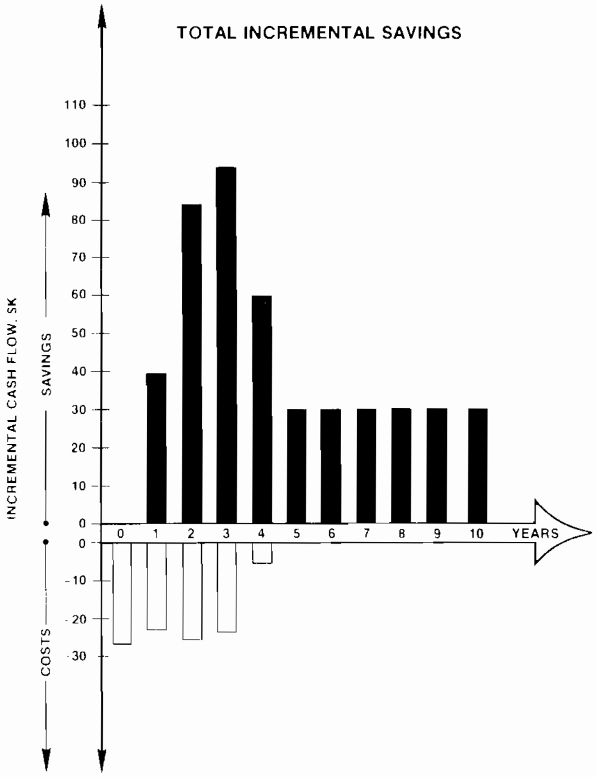
INCREMENTAL BOARD KIT SAVING

	YEARS					
	0	1	2	3	4	5-10
KITS REQUIRED, BEFORE	0	20	20	20	0	0
LESS KITS REQUIRED, AFTER	0	−10	−10	0	0	0
KITS SAVED	0	10	10	20	0	0
COST SAVINGS AT $1,000	0	$10,000	$10,000	$20,000	0	0
LESS SERVICE MAIN-FRAMES, 5 AT $1,000	0	−$ 5,000	0	0	0	0
NET SAVINGS	0	$ 5,000	$10,000	$20,000	0	0



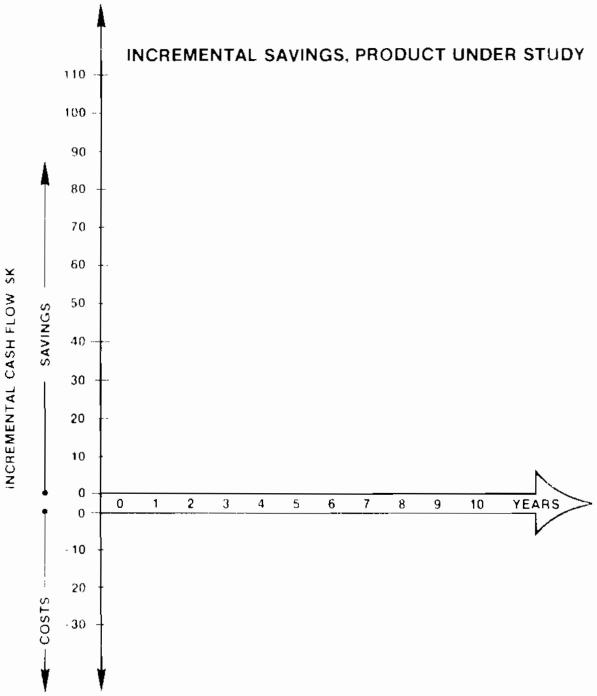
SUMMARY OF INCREMENTAL SAVINGS

	YEARS					
SAVINGS ITEMS	0	1	2	3	4	5-10
1. Production Labor	0	30,000	60,000	60,000	30,000	0
2. Warranty	0	4,800	9,600	9,600	4,800	0
3. Field Service	0	0	4,800	14,400	24,000	28,800
4. PC Board Stock	0	5,000	10,000	20,000	0	0
Totals	0	$39,800	$84,400	$104,000	$58,800	$28,800



INCREMENTAL SAVINGS, PRODUCT UNDER STUDY

	YEARS					
SAVINGS ITEMS	0	1	2	3	4	5-10
1. Production Labor						
2. Warranty						
3. Field Service						
4. PC Board Stock						
Totals						

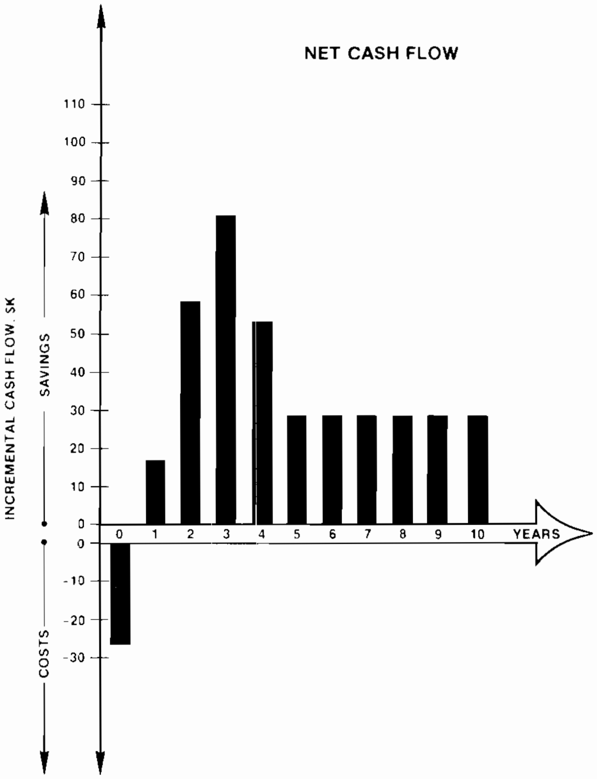


SECTION F — RETURN ON INVESTMENT

Once the incremental costs and savings are determined (Sections D and E), we can use them as data for any of the common return-on-investment calculations. We chose the IRR (internal rate of return) function of the Hewlett-Packard Model 38E Calculator. IRR is the compound interest rate which returns a series of positive and negative cash flows to zero present value. The cash flows for the sample product are:

ANNUAL INCREMENTAL CASH FLOW

YEAR	INCREMENTAL COSTS	INCREMENTAL SAVINGS	NET CASH FLOW
0	$27,000	0	$-27,000
1	23,000	$ 39,800	+16,800
2	26,000	84,400	+58,400
3	23,000	104,000	+81,000
4	6,000	58,800	+52,800
5	0	28,800	+28,800
6	0	28,800	+28,800
7	0	28,800	+28,800
8	0	28,800	+28,800
9	0	28,800	+28,800
10	0	28,800	+28,800

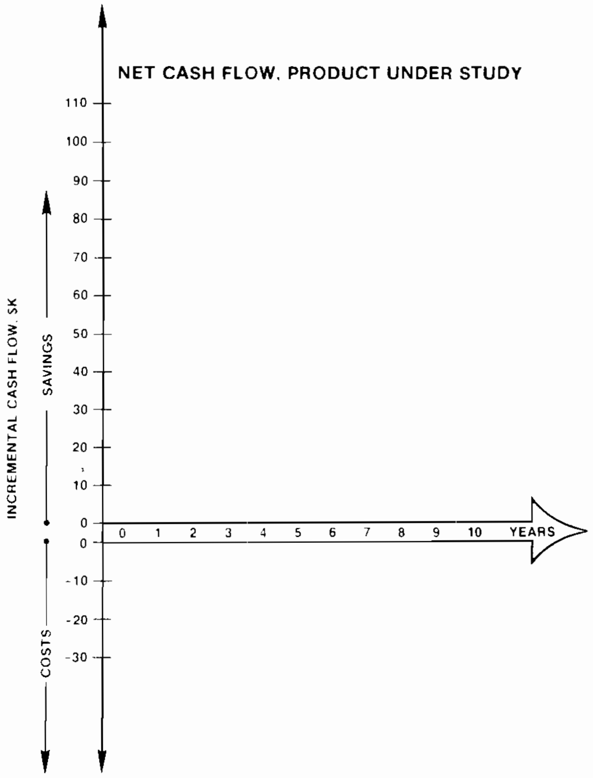


IRR = 132.6%

Note that, even though every effort was made to be conservative in the cost and saving estimates, the IRR is still very high for implementing the new digital test/service strategy. Here is space to calculate the cash flows and IRR for another product:

INCREMENTAL CASH FLOW, PRODUCT UNDER STUDY

YEAR	INCREMENTAL COSTS	INCREMENTAL SAVINGS	NET CASH FLOW
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			



IRR = ______

The IRR calculation, above, was based on implementing a digital test/service strategy which took advantage of all the possible areas of savings offered by Signature Analysis. However, sometimes it is necessary to investigate the return of only a partial implementation of the technique. Here are two cases of interest:

Case 1—Include only production and warranty savings. Do not include field service savings, since the savings here may be passed on to customers via lower service charges. Or, the savings may be realized by a third-party service organization, not by the manufacturer.

Case 2—Include only production savings. This case assumes that Signature Analysis is used only in final assembly test, not in field service at all.

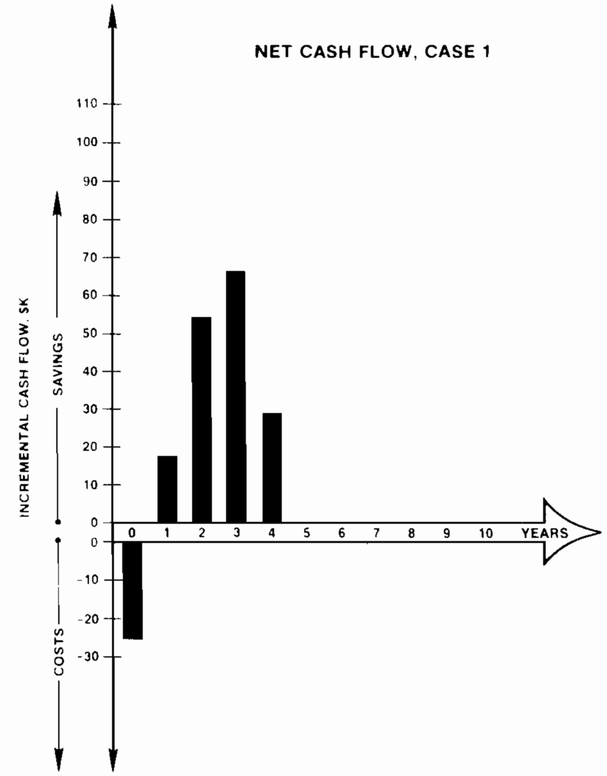
Case 1—All incremental costs are included (Section D) but only the following incremental savings are included:

- Production Labor (Section E-1)
- Warranty (Section E-2)
- PC Board Stock (Section E-4)

Field service savings are not included.

ANNUAL INCREMENTAL CASH FLOW, CASE 1

YEAR	INCREMENTAL COSTS	INCREMENTAL SAVINGS	NET CASH FLOW
0	$27,000	0	$-27,000
1	23,000	$39,800	+16,800
2	26,000	79,600	+53,600
3	23,000	89,600	+66,600
4	6,000	34,800	+28,800
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0



IRR = 116.8%

In this case, eliminating field service savings still results in a very attractive IRR.

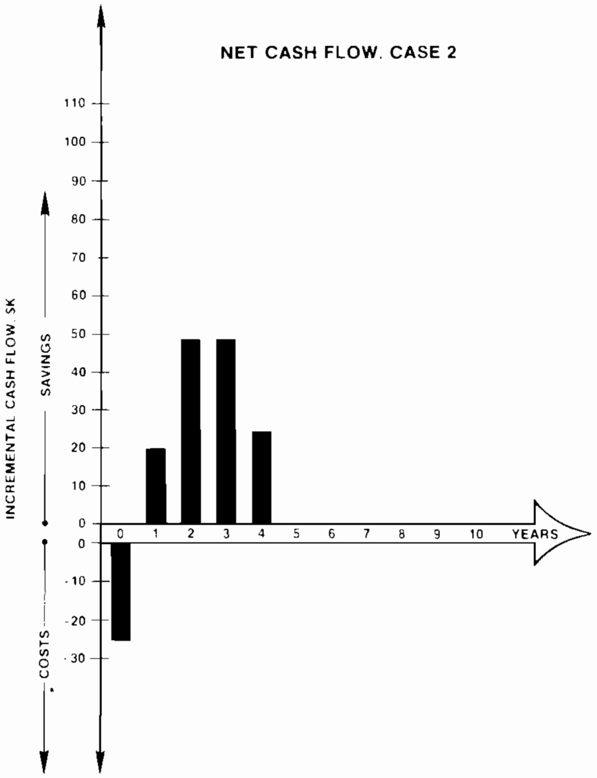
Case 2—Only those incremental costs are included which are required to set up a product for Signature Analysis in final assembly test:
- Engineering (Section D-1)
- Documentation (Section D-2)
- Test Equipment (Section D-3), reduced from 17 units to 7.
- Ongoing Materials (Section D-5).

Only those incremental savings are included which are realized in final assembly test:
- Production Labor (Section E-1).

Warranty, field service and PC board stock savings are not included.

ANNUAL INCREMENTAL CASH FLOW, CASE 2

YEAR	INCREMENTAL COSTS	INCREMENTAL SAVINGS	NET CASH FLOW
0	$27,000	0	$-27,000
1	11,000	$30,000	+19,000
2	12,000	60,000	+48,000
3	12,000	60,000	+48,000
4	6,000	30,000	+24,000
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0



IRR = 107.4%

Even in this case, IRR is over 100%, in a situation where 20-30% is considered attractive.

SECTION G — CONCLUSION

1. **Summary.** The use of the model to compare two digital test/service strategies can also show the IRR derived by switching from one (current) strategy to another (new) one. A significant finding was that, even when the new Signature Analysis strategy was only partially implemented, the IRR was over 100%. This can be important in getting a new strategy started, by reducing the number of departments which must collaborate. In Sample Case 2, the Production Department could have justified the project on production savings alone. The Service Department would have had the option of adopting it later.

2. **Next Step.** The reader may apply the model to another product by utilizing the blank tables and plots in the paper. Here is an index to them:

12. Retrofitting a Z80-Based Personal Computer for Signature Analysis Troubleshooting - A Case Study.

HEWLETT PACKARD

Errol Shanklin is an Application Support Engineer for the 5004A Signature Analyzer at the Santa Clara Division. Errol holds a B.S.E.E. degree from University of Pacific.

David Staugas is a Software Design Engineer with the Exidy Corporation of Sunnyvale, California. Dave earned a BA in Computer Science from University of California, Berkeley.

Retrofitting a Z80 Personal Computer for Signature Analysis Troubleshooting— A Case Study

This article shows how stimulus programs were developed for troubleshooting a personal computer to the component level with Signature Analysis (SA), on circuits such as ROM, RAM, serial and parallel I/O, and a video text generator. It also shows how to retrofit FREERUN, a technique that allows SA diagnosis of circuit faults that prevent the stimulus programs from running.

Errol Shanklin
Hewlett-Packard Company
Santa Clara, California

David Staugas
Exidy Corporation
Sunnyvale, California

SECTION A—INTRODUCTION

The Signature Analysis Technique

By designing or retrofitting the Signature Analysis (SA) technique into a microprocessor-based digital product, a manufacturer can provide simplified field service and production line procedures for component level repair of the product using a Signature Analyzer for troubleshooting. Use of a Signature Analyzer requires that some test features be designed or retrofitted into the product to be tested. This article assumes some familiarity with these features. A series of Hewlett-Packard Application Notes on Signature Analysis[1] provide the basics on how to add these features to a product to be serviced with a Signature Analyzer.

The Application

This application shows how SA was retrofit into a personal computer to test and troubleshoot it on the manufacturer's production line, customer service centers, and eventually at their distributor's service centers. Here is the strategy for testing the computer on the production line where SA is used to troubleshoot a completely unknown board. This means:

- No shorts or opens testing is done on blank boards.
- No incoming inspection testing of RAMs is performed, including 4K and 16K dynamic RAMs.
- Visual inspection is used to find most process faults of an assembled board such as solder splashes and bridges, and incorrectly installed ICs and components. No ATE is used.
- Boards are powered-up in an unknown state immediately after assembly and inspection.
- When a board does not operate correctly when power is applied, diagnostics separate from SA are used in an attempt to isolate failures down to small blocks of the circuit (e.g. it will indicate that the failure seems to be in ROM or RAM or the supporting address decodes and control circuits), but not to the component or process fault level in most cases.
- SA is used to find the components or process faults using SA stimulus routines on those boards that the diagnostic routine can indicate where to begin troubleshooting.
- SA is also used to troubleshoot boards that won't allow the diagnostic to run using a technique called FREERUN.

The distributors still plan on board exchange with the customer's failed unit, then bringing the board back to the shop for repairs with SA. The SA documentation is not yet available to the customer so that he could make his own after-warranty repairs easier with SA.

The Computer

This article assumes a working knowledge of microprocessor-based systems and raster-scan CRT text generators. Hardware and software manuals on the computer are available from the manufacturer[2]. They include a theory of operation and detailed schematics.

Figures 1 and 2 show the block diagram and memory map of the standard computer unit. The standard computing system incorporates the circuits of the block diagram within a single housing around the keyboard. Optional equipment is available which expands the capabilities of the computer but exists as separate items. They are a CRT display, a line printer, a floppy disc, and a S-100 bus expansion module. SA has been retrofit into the standard computer unit. It has not yet been implemented in any of the optional peripherals.

The Article

This article shows the details of implementing SA into the standard computer unit in terms of the hardware, software and test connections. The figures show how SA was retrofit into the computer while the accompanying text discusses the major decisions and tradeoffs associated with retrofitting SA into the product and the effects some of those decisions had on the ease of troubleshooting the computer. The SA stimulus routines for the computer are effective in finding faults with SA. However, the way in which they are implemented is not necessarily the only way nor the most efficient way it could be done.

[1] Application Note 222, A Designer's Guide to Signature Analysis; 222-1, Implementing Signature Analysis for Production Testing; 222-2, Application Articles on Signature Analysis.

[2] The personal computer is called the SORCERER II and is manufactured by Exidy Corporation of Sunnyvale, California.

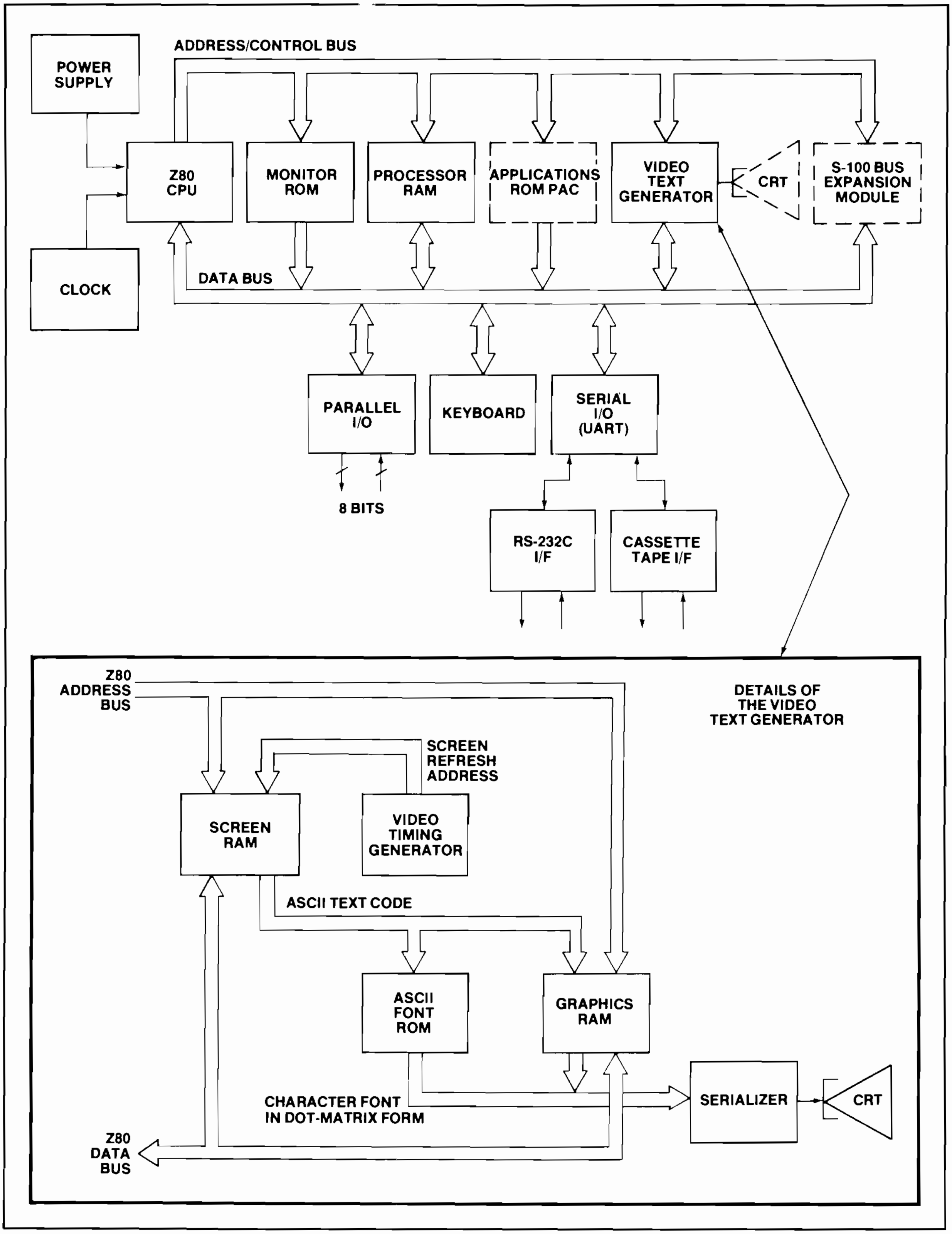


Figure 1. This personal computer contains a Z80 microprocessor, an operating system contained in MONITOR ROM, space for user created programs in PROCESSOR RAM, and a slot on the side of the computer to insert a ROM PAC that contains ROM-based applications programs. Also included is a VIDEO TEXT GENERATOR that outputs to a CRT for display. It includes memory space for ASCII text characters in SCREEN RAM, character font in ASCII FONT ROM, user defined graphics font in GRAPHICS RAM, and discrete VIDEO TIMING GENERATOR circuits. The interfaces to the computer consist of two SERIAL I/O channels for a cassette tape unit and a general purpose RS-232 link, both supported by a UART, an 8-bit PARALLEL I/O channel of discrete logic, a KEYBOARD with associated scan mechanism and a S-100 BUS EXPANSION port that is a buffered extension of the Z80 microprocessor bus. Optional equipment is outlined in dashed lines.

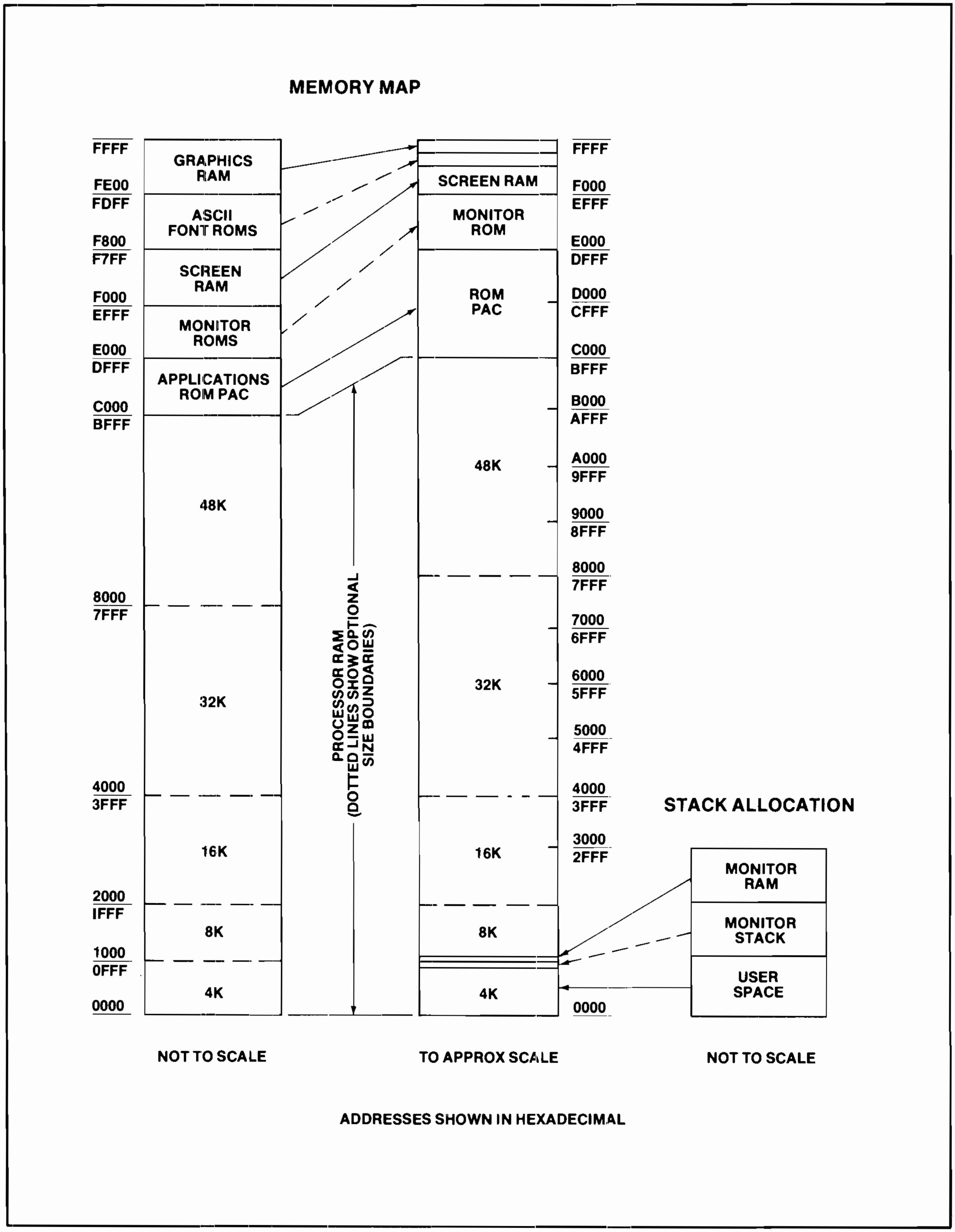


Figure 2. This diagram shows the Z80's 64K byte memory space divided among the circuits of the personal computer. All memory assignments are fixed except for the PROCESSOR RAM which can vary from 4K bytes to 48K bytes depending upon the user's memory requirements. The KEYBOARD, PARALLEL and SERIAL I/O devices reside within the input/output space of the Z80 while the S-100 BUS EXPANSION memory devices automatically map over the PROCESSOR RAM as needed.

SECTION B—FREERUNNING THE Z80

When the Z80 is placed into the FREERUN mode using the FREERUN fixture of the figures below, the Z80's continuous cycling of the address bus stimulates the kernel, or heart, of the computer system. SA is then used to isolate kernel circuit failures. The kernel is defined as those circuits required to be functional so that the microprocessor can execute ROM-based SA stimulus programs for SA troubleshooting of circuits beyond the kernel. The kernel circuits consist of:

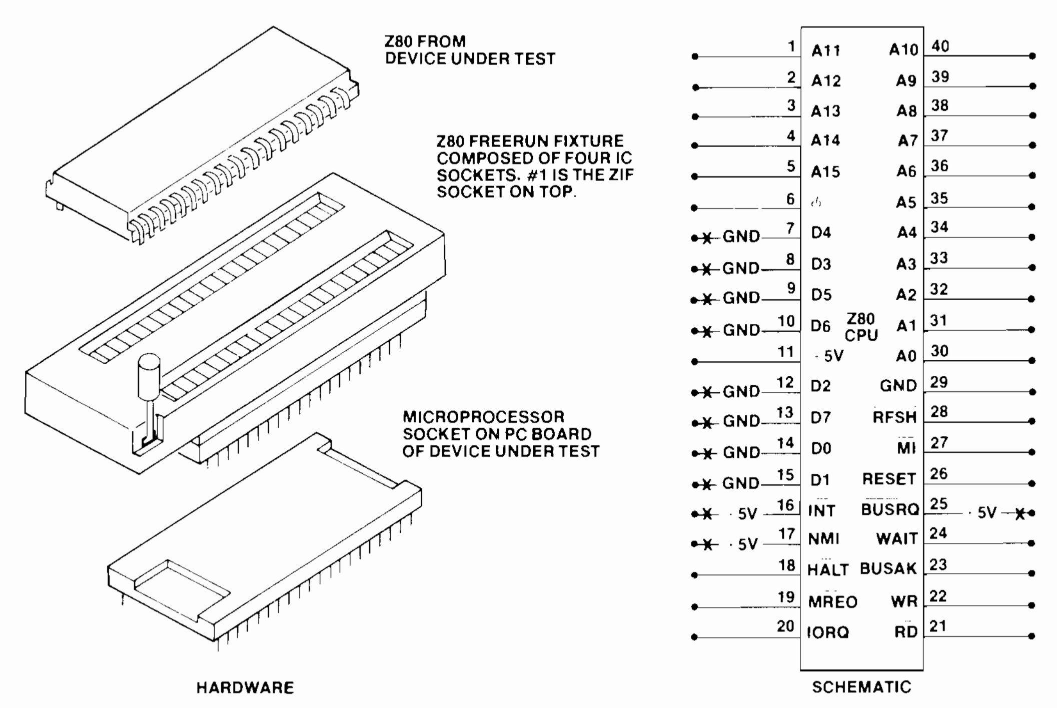
1. The power supply and Z80's clock.
2. The Z80 microprocèssor.
3. The Z80 control lines including gating and buffers.
4. Address and data buses including buffers.
5. Address decode circuits that create the ROM and RAM chip selects.
6. ROMs, including those that contain the SA stimulus programs.

The power supply and Z80's clock are troubleshot with conventional equipment such as voltmeters, frequency counters or oscilloscopes instead of SA.

Although FREERUN uses the Z80 to stimulate other circuits of the kernel, FREERUN does not check the Z80's ability to execute code. Here are several ways to verify the Z80's health with increasinq levels of confidence.

1. Since most failures internal to the Z80 show up as incorrect signatures on the address bus during FREERUN, assume that further testing of the Z80 before it is used to execute the SA stimulus code is not required.
2. Assume that if the Z80 can execute any of the SA stimulus routines, it is operating correctly and doesn't need to be tested further.
3. Add a Z80 instruction set test to the SA stimulus library. However, since a complete test consumes many bytes of code, and it is difficult to write and can't test the Z80's AC parameters, it could be unjustified considering the amount of circuitry it tests.

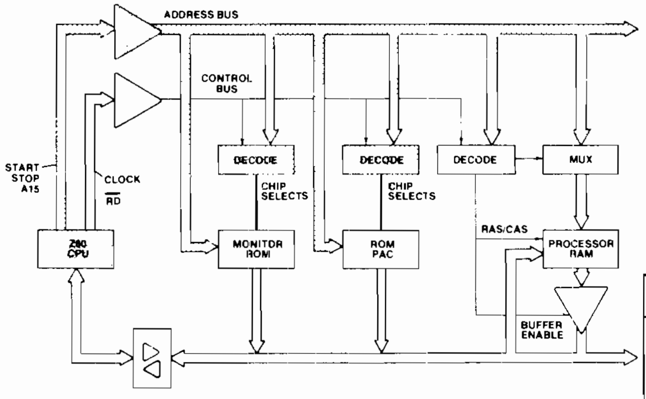
It was assumed here that FREERUN and the Z80's ability to execute the SA code was a sufficient test of the microprocessor.



Z80 FREERUN FIXTURE

Figure 3. This fixture, quickly constructed from available IC sockets, allowed FREERUN to be easily retrofit into the computer. Modifications to the four sockets that make up this fixture break the data bus between the system and the Z80 and apply a NOP instruction to cause it to FREERUN. Socket #1 is a zero-insertion-force socket that allows the Z80 to be removed from the computer and placed into the fixture without damage. Socket #2 is altered by bending pins on the socket to open the data bus as indicated by the 'X' on the schematic. The NOP instruction is applied to the Z80 by wiring the open data bus to ground on socket #2. Similar treatment was done to several of the Z80 control pins by opening them and wiring to +5vdc or ground as required to force them to a known state during FREERUN independent of the computer's response.

Figure 4. FREERUN stimulates the fundamental circuits of the computer system so that SA can be used to isolate failures of the kernel, or heart, of the system. The kernel circuits are checked with four setups of the signature analyzer as shown. a.) The first setup checks the integrity of the address bus and related address decode circuits. b.) The second setup verifies that code within ROM is correct and that the data bus is free and clear so that instructions can pass from the ROM to the Z80. c.) The third setup is used only when there is an apparent failure in the Z80's control signal outputs. d.) The fourth setup checks the dynamic RAM refresh circuits of the PROCESSOR RAM, but not the RAM itself. The RAM cannot be troubleshot with FREERUN because it powers-up in a random state. There is no way to initialize it with the processor because the data bus has been opened up. All other circuits beyond the kernel are troubleshot using SA stimulus programs contained in ROM.

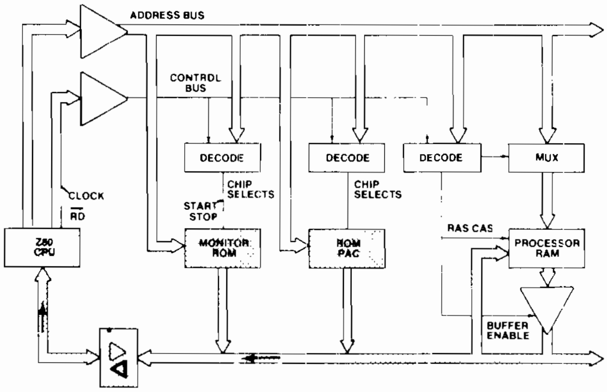


SIGNATURE ANALYZER		Z80	
INPUT	EDGE	SIGNAL	PIN
START	↓	A15	5
STOP	↓	A15	5
CLOCK	↓	$\overline{RD}$	21

ADDRESS BUS
SIGNATURE TABLE

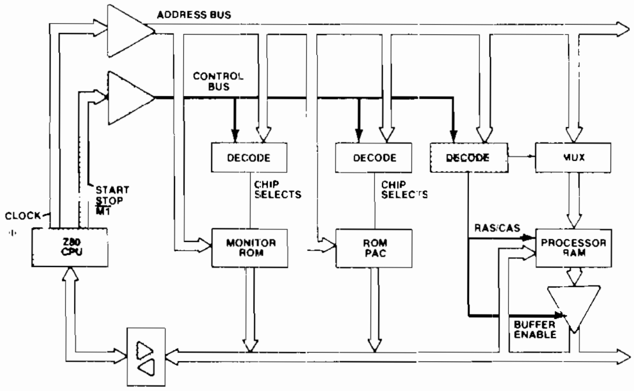
ADDRESS	PIN	SIGNATURE	ADDRESS	PIN	SIGNATURE
A15	5	755P	A7	37	52F8
A14	4	3827	A6	36	UPFH
A13	3	3C96	A5	35	OAFA
A12	2	HAP7	A4	34	5H21
A11	1	1293	A3	33	7F7F
A10	40	HPP0	A2	32	CCCC
A9	39	2H70	A1	31	5555
A8	38	HC89	A0	30	UUUU

a. ADDRESS BUS AND DECODERS



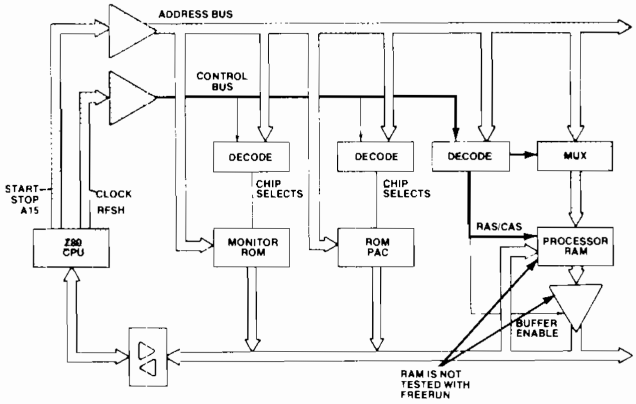
SIGNATURE ANALYZER		MONITOR ROM #1		MONITOR ROM #2	
INPUT	EDGE	SIGNAL	PIN	SIGNAL	PIN
START	↓	CHIP SELECT	—	CHIP SELECT	—
STOP	↑	CHIP SELECT	—	CHIP SELECT	—
CLOCK	↑	Z80 $\overline{RD}$	21	Z80 $\overline{RD}$	21

b. ROM CONTENTS AND DATA BUS



SIGNATURE ANALYZER		Z80	
INPUT	EDGE	SIGNAL	PIN
START	↓	$\overline{MI}$	27
STOP	↓	$\overline{MI}$	27
CLOCK	↑	Φ	6

c. Z80 CONTROL LINES



SIGNATURE ANALYZER		Z80	
INPUT	EDGE	SIGNAL	PIN
START	↓	A15	5
STOP	↓	A15	5
CLOCK	↓	RFSH	28

d. DYNAMIC RAM REFRESH MULTIPLEXER

Figure 4

SECTION C—SA TEST STORAGE, ACCESS AND SELECTION

Storage

The external applications ROM PAC provides the most convenient and quick means to store the SA test stimulus for several reasons.

1. No other ROM needs to be removed as in most retrofit situations in which ROM-substitution is used. The operator simply inserts the SA ROM PAC into a slot on the computer to run them.
2. It's harder to damage a ROM PAC than it is a ROM because the ROM PAC intèrfaces to the computer with a PC board edge connector instead of fragile IC pins.
3. Production technicians, customer service personnel and distributors of the computer already know how to use the ROM PAC. The SA tests simply become one of many applications ROM PACs available for the computer.

The disadvantages of storing the SA tests in the ROM PAC occur because of the method the computer uses to access programs stored there.

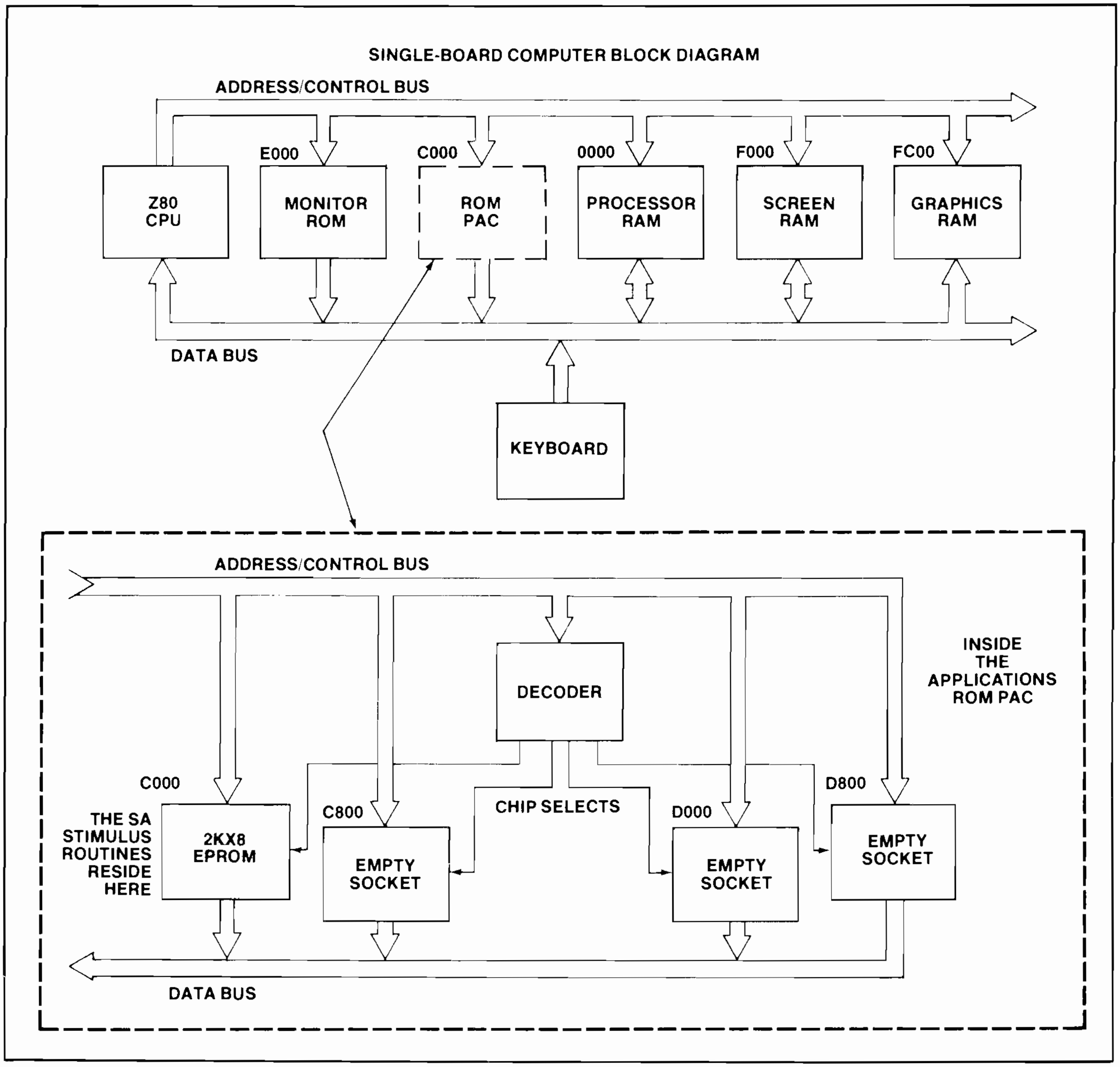


Figure 5. The SA stimulus routines (SA tests) are stored in a 2K × 8 EPROM within an applications ROM PAC that is inserted into the side of the computer. The SA tests are 944 bytes long of which 534 bytes are the ASCII characters of a test selection menu, 110 bytes are a branch table for test selection and 300 bytes form the actual SA tests. The program starts at the first address within the applications ROM PAC memory space, address C000H.

Access

When power is first applied to the computer, the Z80 CPU is reset to address 0000H and begins execution there. Since the ROM PAC resides at address C000H, a means is provided to cause the Z80 to jump there to begin execution of the SA tests. However, a ROM PAC is not always inserted into the computer. The program must also recognize the presence or absence of the ROM PAC and jump to the ROM PAC program if it's there by means of an operating system stored in the monitor ROMs.

The monitor ROMs reside at memory address E000H, not at address 0000H. (The first address executed by the Z80 at power-on.) To compensate, a special circuit that resets to zero at power-on temporarily maps the monitor ROMs to location 0000H. The first three locations of the monitor ROMs contain an unconditional jump instruction to location E003H. When the Z80 addresses E003H for the next instruction, the special circuit remaps the monitor ROMs back to their true address space of E000H to EFFFH so that further operating system code in the monitor ROMs can be executed.

The operating system then tries to establish a stack in any available functional RAM so that monitor subroutines can be accessed by the user. The operating system first checks processor RAM to see if enough locations behave like RAM. If the RAM is functional, the operating system establishes a stack and proceeds to do several housekeeping chores to set up the computer for interaction by the user. If processor RAM is not functioning, the computer will continue to search memory for the next available RAM locations (e.g. SCREEN or GRAPHICS RAM) until a stack can be established. If no RAM is functional, the operating system will continue to search forever and never release control to the user.

If a stack can be established, the operating system then checks for the presence of a ROM PAC to see if program execution should continue there. The process of determining this requires several subroutines within the operating system that uses the stack. Finally, the operating system sees that the SA ROM PAC has been inserted and jumps to the first location (address C000H) to put up the SA test selection menu. Once the SA tests have been selected and are executing, neither the operating system nor the stack are used.

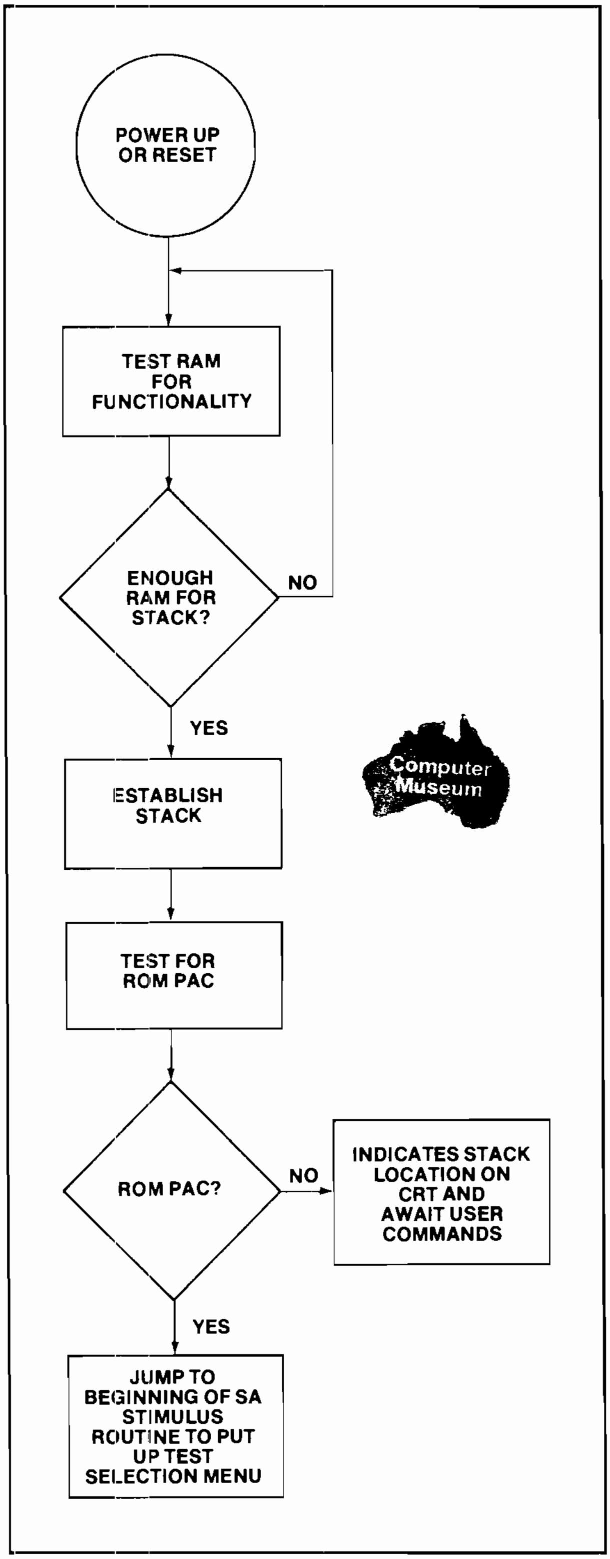

POWER UP OR RESET
TEST RAM FOR FUNCTIONALITY
ENOUGH RAM FOR STACK?
NO
YES
ESTABLISH STACK
TEST FOR ROM PAC
ROM PAC?
NO
INDICATES STACK LOCATION ON CRT AND AWAIT USER COMMANDS
YES
JUMP TO BEGINNING OF SA STIMULUS ROUTINE TO PUT UP TEST SELECTION MENU

Figure 6. The SA tests are automatically accessed by the computer upon power-on when an SA applications ROM PAC is inserted. The system first goes through a power-up test of RAM to find a place for the operating system's stack to reside. The program then checks for the presence of a ROM PAC by checking the first few locations of ROM PAC memory space for non-zero entries. If the ROM PAC is present, the program then jumps to the first ocation of the ROM PAC to begin execution. The first steps of the SA tests put up a test selection menu on the CRT. See the next figure.

With this background of how the SA tests are accessed, we can now discuss the tradeoffs of storing the SA tests in the ROM PAC versus storing them in a ROM which can be substituted in place of the monitor ROMs. (Substitution is the more common approach to retrofitting SA.) By placing the programs in the ROM PAC, all kernel circuits must be functional in order to run any SA tests. FREERUN is used to check them when they aren't. However, this application also contains circuits that cannot be troubleshot with FREERUN, but must also be functional. They are:

1. RAM, so that the operating system can establish a stack. This includes RAM chip selects and support circuits.
2. The special circuit that maps the monitor ROM to location 0000H at power-on.

An assumption was made that one of the three sections of RAM (either PROCESSOR, SCREEN or GRAPHICS

RAM) would be functional so that even bad RAM at one of the locations could be troubleshot with the SA tests once a stack was established in another section of RAM and the programs were accessed. However, it turned out that this assumption was not good in practice because of a high percentage of process faults such as solder splashes, open circuit traces and incorrectly installed components. The RAMs, including the dynamic 4K and 16K parts, were not pretested before loading. Their failure rate, combined with the process faults resulted in the majority of boards having no functional RAM.

Another assumption was made about the special circuit of item #2. Since the circuit consists of one IC, it was assumed that the circuit could be easily troubleshot by other means such as logic probes.

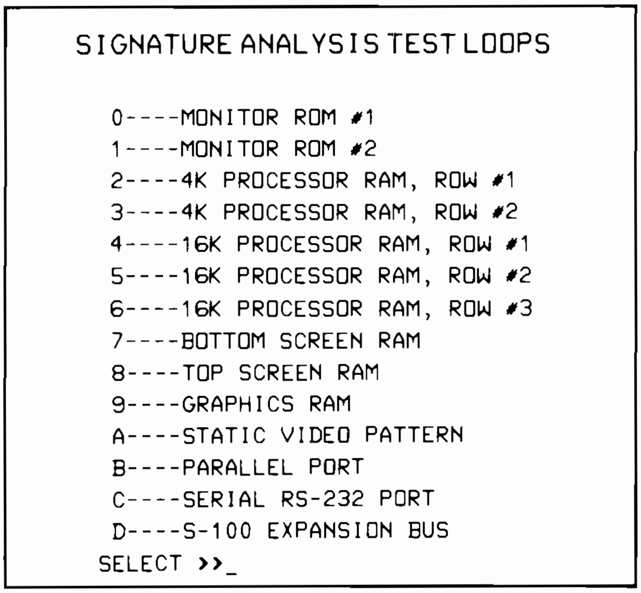
If the program had been stored in ROM to be substituted in place of a monitor ROM instead of placed in the ROM PAC, then RAM would not need to be functional in order to run the SA tests. Remember that the operating system would be replaced with the SA ROM which is then accessed directly at power on (if the mapping circuit is functional). With this substitution technique, it would be possible to troubleshoot RAM even if none was functional. However, there are some problems with the monitor ROM substitution method for this product because of the retrofit situation.

1. The monitor ROM is masked and an EPROM version could not be directly substituted into the product without cutting traces and rewiring a small section of the board. This was not acceptable. However, a masked ROM version of the SA test was equally not acceptable. The volume was not judged to be high enough to justify a mask charge because additions to the test repertoire are still planned. Note that monitor EPROM substitution could have been planned in the design stage by making the circuit easily switched between ROM and EPROM. Or the SA tests could have been designed into the monitor ROM so that it would always be available in the product without substitution or ROM PACs.
2. Since the monitor ROMs contain the keyboard and video CRT driver subroutines that allow easy operator interaction with the test selection menu, the drivers would need to be duplicated in the ROM that would substitute the monitor ROM. It was easier to keep the monitor routines in tact so that the SA tests in the ROM PAC could use the drivers as subroutines to put up the menu and allow the keyboard to be used to select the tests. The drivers could have been avoided if a less elaborate test selection method had been designed into the product such as DIP switches that could be read by the microprocessor.

Selection

The keyboard provides an easy means for the troubleshooter to quickly select a test. It is probably the easiest method when combined with the test selection menu on the CRT. However, should the keyboard fail, the operator cannot run any SA tests, including any that might help him troubleshoot the keyboard itself. Here it was decided that if the keyboard should fail, the operator would unplug the failed unit and exchange it with another one. However, this brings up two problems:

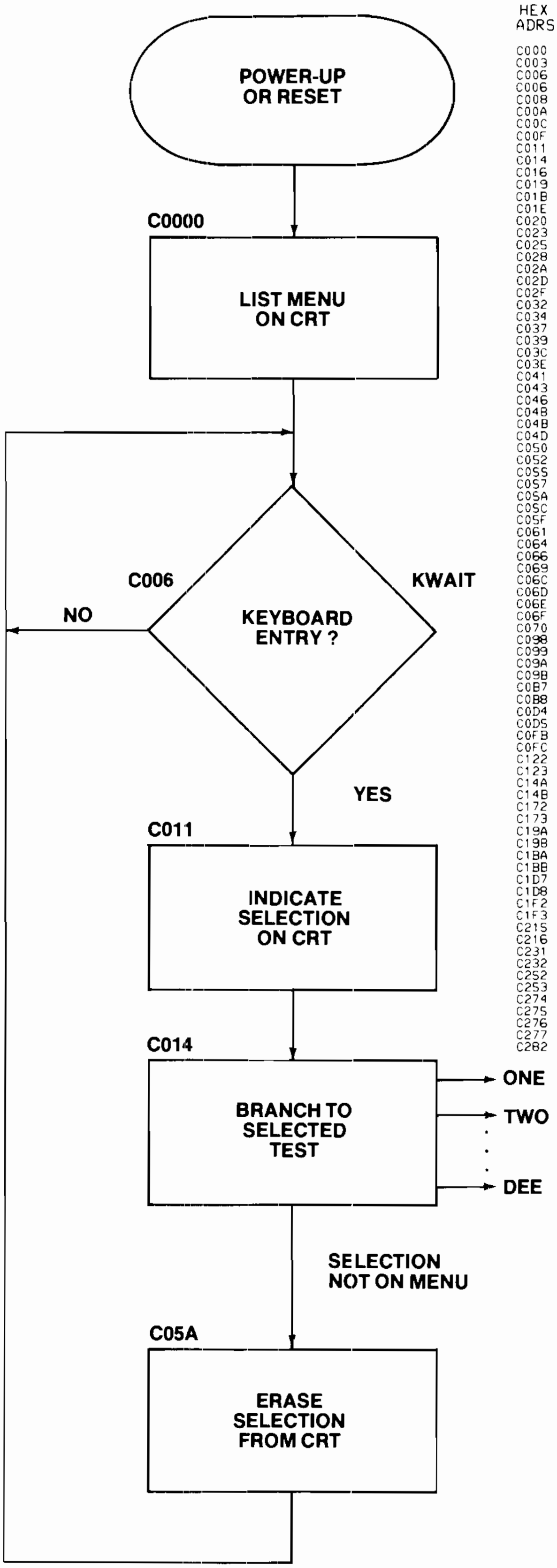
1. It assumes that all logic associated with the keyboard function exists on the keyboard PC board which is not the case here. The keyboard PC board contains only the key switches and one IC. Several other IC's associated with detecting a key closure are on the main board. The result is that exchanging the keyboard may not fix the problem.
2. Someone has to troubleshoot the keyboard when it fails, and SA cannot help if the tests are selected by means of the keyboard. Even FREERUN cannot help because the keyboard is treated as an I/O device that is not stimulated by FREERUN. It was assumed in this case that the keyboard could be fixed by other means.



```
SIGNATURE ANALYSIS TEST LOOPS

   0----MONITOR ROM #1
   1----MONITOR ROM #2
   2----4K PROCESSOR RAM, ROW #1
   3----4K PROCESSOR RAM, ROW #2
   4----16K PROCESSOR RAM, ROW #1
   5----16K PROCESSOR RAM, ROW #2
   6----16K PROCESSOR RAM, ROW #3
   7----BOTTOM SCREEN RAM
   8----TOP SCREEN RAM
   9----GRAPHICS RAM
   A----STATIC VIDEO PATTERN
   B----PARALLEL PORT
   C----SERIAL RS-232 PORT
   D----S-100 EXPANSION BUS
  SELECT >>_
```

Figure 7. This menu appears on the CRT to prompt the user to select the number of the desired SA test with one keystroke. Once a test is selected it runs continuously and further keyboard entries are ignored. A reset from the keyboard, like power-on, stops the test and recalls the menu to the CRT for a new test selection. The details (code, flowcharts, and circuits) of tests 0-9 are shown in the remaining sections. Tests A-C are in separate sections with headlines that match the circuit they test and the menu number. For example, the PARALLEL PORT test is found in the section headlined " PARALLEL PORT , TEST B". TEST D is not implemented at this time.

Figure 8. Here is the flowchart and Z80 assembly language code that allows program selection. These are the mnemonics particular to the computer's assembler. PRTX is a monitor subroutine which transfers the ASCII characters of the menu from SA ROM to the CRT, starting at "MENU" (address C06CH) to "DEFB" (address C282H). Subroutine KEYBRD places a key entry into the accumulator with the zero flag reset. If no key is pressed, the zero flag is set. VIDEO writes the key entry on the CRT by the menu prompt "SELECT >>". If the entry matches a test number, the program branches to the test with the same label. (e.g. 3 branches to "THREE".) Invalid entries are erased from the CRT and the program returns to wait for a new key.



```
HEX
ADRS  CONTENTS  LABEL    INSTRUCTION

C000  216C00             LD    HL,MENU
C003  CDBAE1             CALL  PRTX
C006            KYWAIT
C006  DBFE               IN    A,(0FEH)
C008  CB6F               BIT   5,A
C00A  28FA               JR    Z,KYWAIT-$
C00C  CD18E0             CALL  KEYBRD
C00F  28F5               JR    Z,KYWAIT-$
C011  CD1BE0             CALL  VIDEO
C014  FE30               CP    30H
C016  CA83C2             JP    Z,ZERO
C019  FE31               CP    31H
C01B  CA8CC2             JP    Z,ONE
C01E  FE32               CP    32H
C020  CAABC2             JP    Z,TWO
C023  FE33               CP    33H
C025  CAB4C2             JP    Z,THREE
C028  FE34               CP    34H
C02A  CABDC2             JP    Z,FOUR
C02D  FE35               CP    35H
C02F  CAC6C2             JP    Z,FIVE
C032  FE36               CP    36H
C034  CACFC2             JP    Z,SIX
C037  FE37               CP    37H
C039  CAD8C2             JP    Z,SEVEN
C03C  FE38               CP    38H
C03E  CAE1C2             JP    Z,EIGHT
C041  FE39               CP    39H
C043  CAEAC2             JP    Z,NINE
C046  FE41               CP    41H
C04B  CA4AC3             JP    Z,AY
C04B  FE42               CP    42H
C04D  CA6DC3             JP    Z,BEE
C050  FE43               CP    43H
C052  CA7FC3             JP    Z,SEE
C055  FE44               CP    44H
C057  CAAFC3             JP    Z,DEE
C05A  3E08               LD    A,8
C05C  CD1BE0             CALL  VIDEO
C05F  3E20               LD    A,''
C061  CD1BE0             CALL  VIDEO
C064  3E08               LD    A,8
C066  CD1BE0             CALL  VIDEO
C069  C306C0             JP    KYWAIT
C06C  CC        MENU     DEFB  12
C06D  0D                 DEFB  13
C06E  CD                 DEFB  13
C06F  CD                 DEFB  13
C070  20202020           DEFM   '
C098  CD                 DEFB  13
C099  CD                 DEFB  13
C09A  0D                 DEFB  13
C09B  20202020           DEFM   '
C0B7  CD                 DEFB  13
C0B8  20202020           DEFM   '
C0D4  CD                 DEFB  13
C0D5  20                 DEFM   '
C0FB  CD                 DEFB  13
C0FC  20202020           DEFM   '
C122  CD                 DEFB  13
C123  20                 DEFM   '
C14A  CD                 DEFB  13
C14B  20202020           DEFM   '
C172  CD                 DEFB  13
C173  20202020           DEFM   '
C19A  CD                 DEFB  13
C19B  20202020           DEFM   '
C1BA  0D                 DEFB  13
C1BB  20202020           DEFM   '
C1D7  CD                 DEFB  13
C1D8  20202020           DEFM   '
C1F2  0D                 DEFB  13
C1F3  20202020           DEFM   '
C215  0D                 DEFB  13
C216  20202020           DEFM   '
C231  0D                 DEFB, 13
C232  20202020           DEFM   '
C252  0D                 DEFB  13
C253  20202020           DEFM   '
C274  0D                 DEFB  13
C275  0D                 DEFB  13
C276  0D                 DEFB  13
C277  2053454C           DEFM  ' SELECT>>'
C282  00                 DEFB  0
```

SIGNATURE ANALYSIS TEST LOOPS

0----MONITOR ROM #1

1----MONITOR ROM #2

2----4K PROCESSOR RAM, ROW #1

3----4K PROCESSOR RAM, ROW #2

4----16K PROCESSOR RAM, ROW #1

5----16K PROCESSOR RAM, ROW #2

6----16K PROCESSOR RAM, ROW #3

7----BOTTOM SCREEN RAM

8----TOP SCREEN RAM

9----GRAPHICS RAM

A----STATIC VIDEO PATTERN

B----PARALLEL PORT

C----SERIAL RS-232 PORT

D----S-100 EXPANSION BUS

SECTION D—THE HARDWARE AND SOFTWARE BEHIND START, STOP AND CLOCK

Creating START and STOP

For tests 0-9 and B-C, the START and STOP edges were created by writing code that controlled available hardware. This combination of hardware and software is called program controlled gating. Figures 9-12 show the program controlled gating used in the computer. The START, STOP and CLOCK connections for FREERUN and test #A are shown in the corresponding sections of this article because they are NOT examples of program control gating and do not require any code to be written. The discussion here deals with some of the considerations in choosing an I/O register's output as the START and STOP connections for program control gating in the computer.

Here are some general guidelines for creating START and STOP. Create successive START and STOP connections that build on the kernel. That is, considering the START and STOP connections used in FREERUN as the first and most basic set, then the second set should be able to be troubleshot with the set used in FREERUN. The third set should be able to be troubleshot with the second and so on. Here are some suggestions for building a set of START and STOP connections for the Z80 from FREERUN to more complicated circuits.

1st set: FREERUN connections. Generally the most significant address bit of the Z80. Used to troubleshoot the next set.

2nd set: Chip selects and address decodes within the memory space of the processor that are stimulated during FREERUN to allow troubleshooting of them with START and STOP of the first set. Also controlled by software to create the required START and STOP edges for troubleshooting the next set.

3rd set: Bits in I/O registers that can be troubleshot with an SA stimulus routine that uses the START and STOP connections of the 2nd set. When the 3rd set is used as START and STOP, the software sets and resets the bit to create the required edges.

START and STOP for this computer fall into the 1st and 3rd sets. The 3rd set cannot be troubleshot with FREERUN, nor is there another test that can be run should START and STOP fail. This has caused the troubleshooter to resort to other methods such as shotgunning when the circuits creating START and STOP fail.

Detecting START, STOP and DATA with the CLOCK

Here are some guidelines for choosing a CLOCK.

1. A clock edge must occur both before and after the START and STOP edges to assure detection and correct GATE action. Be sure this is met when gated CLOCKs (defined below) are used.
2. The CLOCK must be synchronous to the START, STOP and DATA inputs it samples. This guideline is generally met if $\overline{RD}$ or $\overline{WR}$ from the Z80 is used as the CLOCK when testing any circuits that are accessed by the Z80.
3. Chose a CLOCK that will avoid sampling a 3-state node when it is in the 3rd state. This is generally done by creating or using a gated CLOCK.

A gated CLOCK is defined as combining or gating a constantly occurring clock such as the Z80's $\overline{RD}$ or $\overline{WR}$ lines with other signals such as address decodes so that the signature analyzer is synchronized to the data of interest during the test. Tests 0-6 and B-C do not require a gated CLOCK. They use $\overline{RD}$ and $\overline{WR}$ directly from the Z80 as their CLOCK as shown in figures 9 and 10. When $\overline{RD}$ was tried as a CLOCK for tests 7-9, the GATE light was flashing indicating that the START and STOP edges were being detected properly, but unstable signatures were also occurring because:

1. The SCREEN and GRAPHICS RAM are both dual-ported and are accessed by two processes that are asynchronous to each other. The Z80 occasionally accesses the RAMs to store characters and graphics font for eventual display on the screen. The VIDEO GENERATOR TIMING circuits also gain access to the RAM so that the characters and font stored there are displayed on the CRT on a continuing basis.
2. $\overline{RD}$ is active during the Z80's op-code fetches from ROM PAC as well as during a read of the RAM during the SA test. When $\overline{RD}$ is used as a CLOCK while probing the circuits of the RAM, data bits will be entered into the signature analyzer that are associated with the CRT refresh process (because of the CLOCK during op-code fetch) when what's really wanted is the data bits associated with the test.

To get stable signatures, a gated CLOCK is required to sample RAM data only when the Z80 has access to the RAMs and not when the screen refresh process takes place nor when op-code is fetched. To "window out" this unwanted data during the signature measurement cycle, $\overline{RD}$ is gated with the address decode of the RAM being tested. As is often the case, the gated CLOCK was already available in the address decode circuits for the RAM. No modifications to the circuit were required.

The resulting gated CLOCK occurs only when the RAM is accessed by the Z80. This should have resulted in stable signatures, but when actually tried, there wasn't even a GATE. Changing to a gated CLOCK also eliminated the CLOCK edges around both the START and STOP edges. To solve the problem, a CLOCK cycle (an access to the RAM being tested) has been added between the generation of the START and STOP edges in tests 7-9 to ensure their detection as shown in figure 12.

One final note of interest. Because the SCREEN RAM and GRAPHICS RAM have different address decodes, two separate CLOCKS are used depending upon which RAM is being tested. It would be convenient for the troubleshooter not to have to move the clock if at all possible. But because of the retrofit situation, it was not possible to do this here.

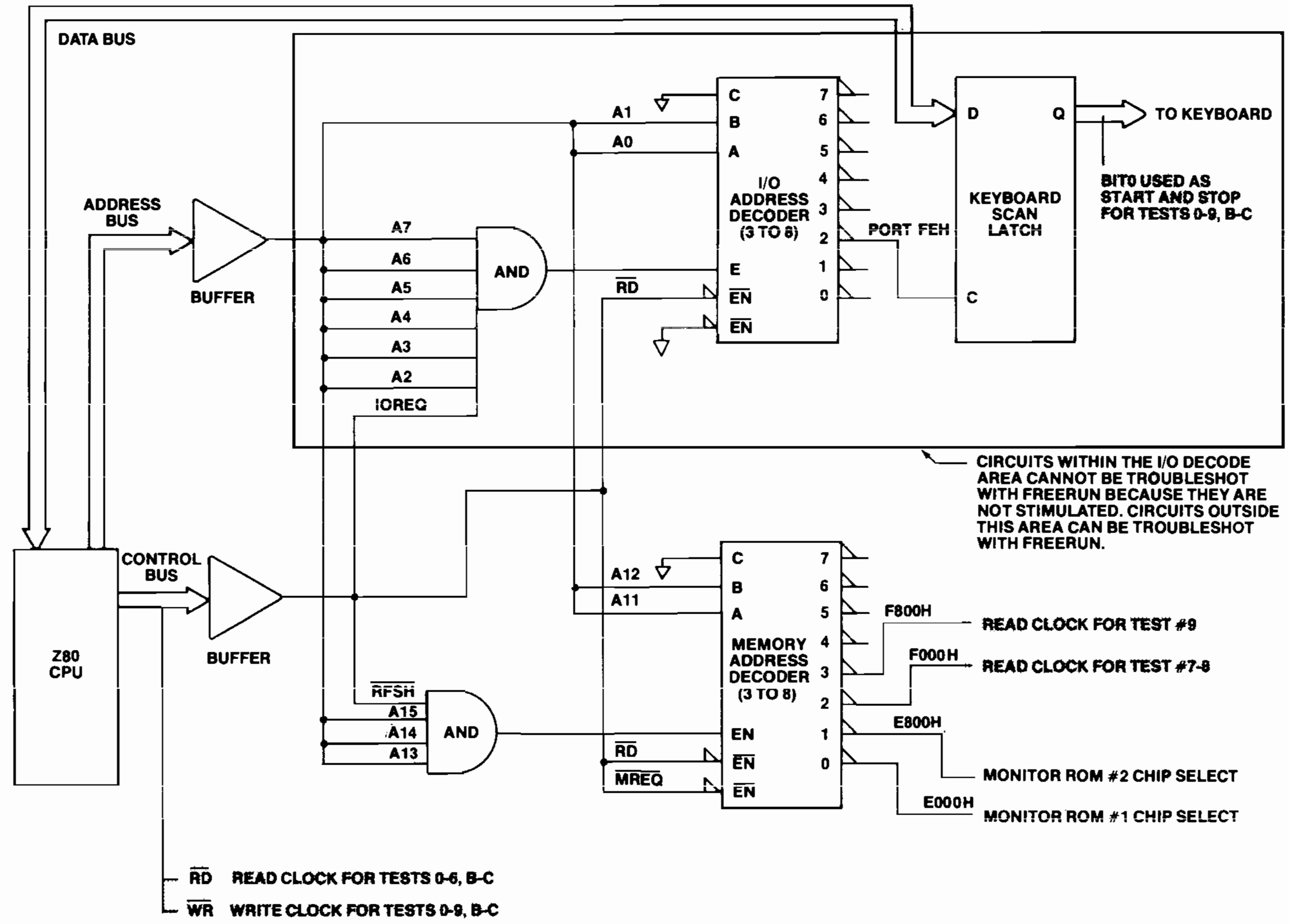
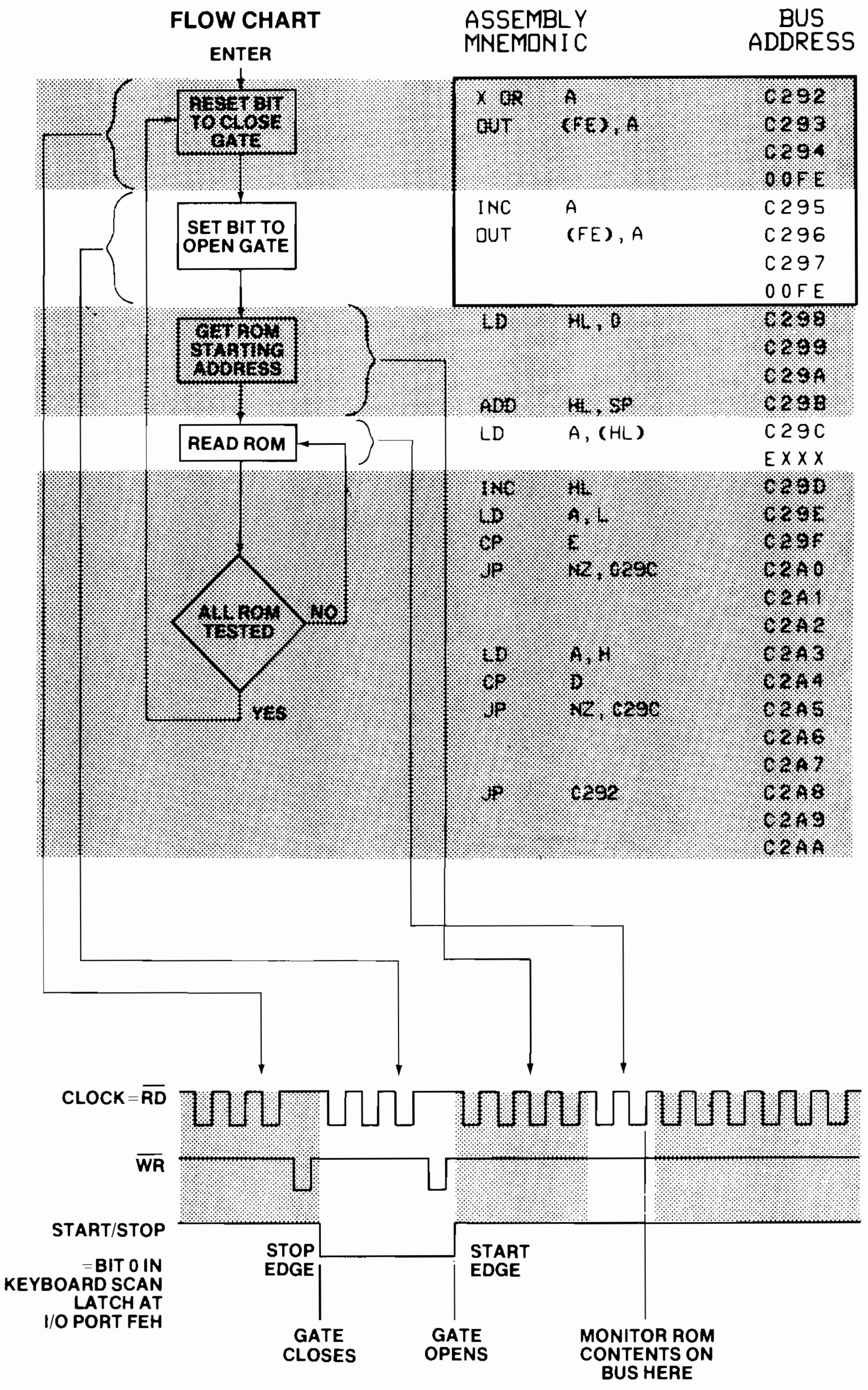

DATA BUS
ADDRESS BUS
BUFFER
CONTROL BUS
BUFFER
Z80 CPU
A7
A6
A5
A4
A3
A2
IOREQ
AND
A1
A0
RD
C
B
A
I/O ADDRESS DECODER (3 TO 8)
E
EN
EN
7
6
5
4
3
2
1
0
PORT FEH
D
Q
KEYBOARD SCAN LATCH
C
TO KEYBOARD
BIT0 USED AS START AND STOP FOR TESTS 0-9, B-C
CIRCUITS WITHIN THE I/O DECODE AREA CANNOT BE TROUBLESHOT WITH FREERUN BECAUSE THEY ARE NOT STIMULATED. CIRCUITS OUTSIDE THIS AREA CAN BE TROUBLESHOT WITH FREERUN.
RFSH
A15
A14
A13
AND
A12
A11
RD
MREQ
MEMORY ADDRESS DECODER (3 TO 8)
EN
EN
EN
F800H
READ CLOCK FOR TEST #9
F000H
READ CLOCK FOR TEST #7-8
E800H
MONITOR ROM #2 CHIP SELECT
E000H
MONITOR ROM #1 CHIP SELECT
RD READ CLOCK FOR TESTS 0-6, B-C
WR WRITE CLOCK FOR TESTS 0-9, B-C

Figure 9. For all SA tests (except #A), START and STOP are both connected to bit 0 in the KEYBOARD SCAN LATCH addressed as I/O port FEH. The programs set and reset the bit to create edges that can be detected by the signature analyzer's START and STOP inputs. START is selected to trigger on a rising edge and STOP triggers on a falling edge. Setting the bit at the beginning of the test opens the GATE while resetting the bit immediately after the last test step closes the GATE. Figures 10 and 12 show the Z80 code that creates the START and STOP edges. The CLOCK connection depends on which test is run as discussed in figures 10-12. The START, STOP and CLOCK connections for test #A are shown in the separate description of that test.



```
ASSEMBLY                      BUS
MNEMONIC                      ADDRESS

X OR   A                      C292
OUT    (FE),A                 C293
                              C294
                              00FE
INC    A                      C295
OUT    (FE),A                 C296
                              C297
                              00FE
LD     HL,0                   C298
                              C299
                              C29A
ADD    HL,SP                  C29B
LD     A,(HL)                 C29C
                              EXXX
INC    HL                     C29D
LD     A,L                    C29E
CP     E                      C29F
JP     NZ,C29C                C2A0
                              C2A1
                              C2A2
LD     A,H                    C2A3
CP     D                      C2A4
JP     NZ,C29C                C2A5
                              C2A6
                              C2A7
JP     C292                   C2A8
                              C2A9
                              C2AA
```

Figure 10. Here is the Z80 code that creates the START and STOP edges by setting and resetting bit 0 in the KEYBOARD SCAN LATCH at I/O port FEH. For tests 0-6 and B-C, two CLOCKS ($\overline{RD}$ and $\overline{WR}$) are used to detect the START and STOP edges. A $\overline{RD}$ CLOCK occurs with every op-code fetch of the Z80 which is more than adequate to detect the START and STOP edges. (A clock edge must occur both before and after both the START and STOP edges to ensure detection.) A $\overline{WR}$ CLOCK occurs when the I/O port FEH and the device being tested are written. The code shown is for MONITOR ROM tests 0-1. A $\overline{WR}$ CLOCK does not occur for these tests. Similar code applies to tests 0-6 and B-C.

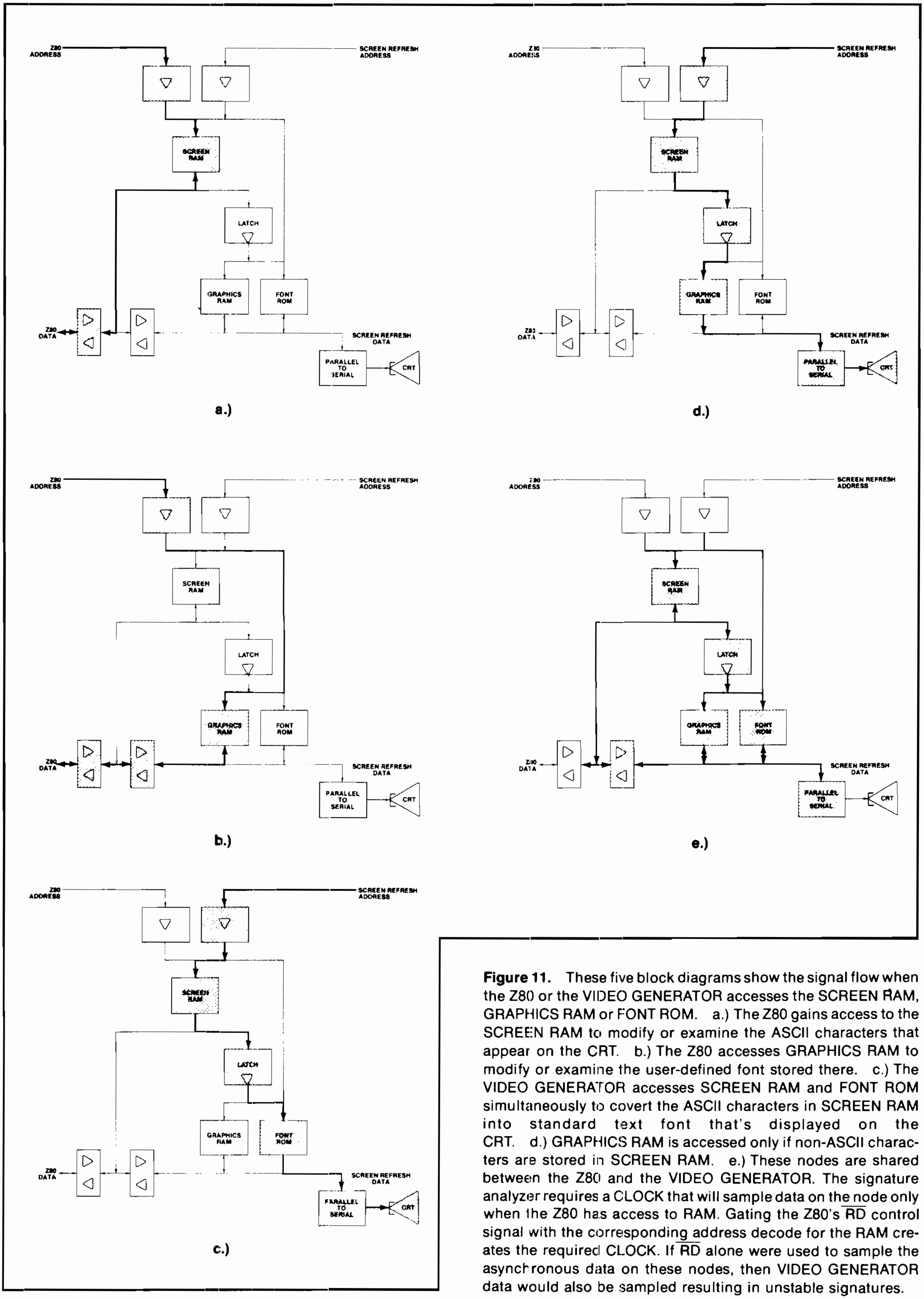
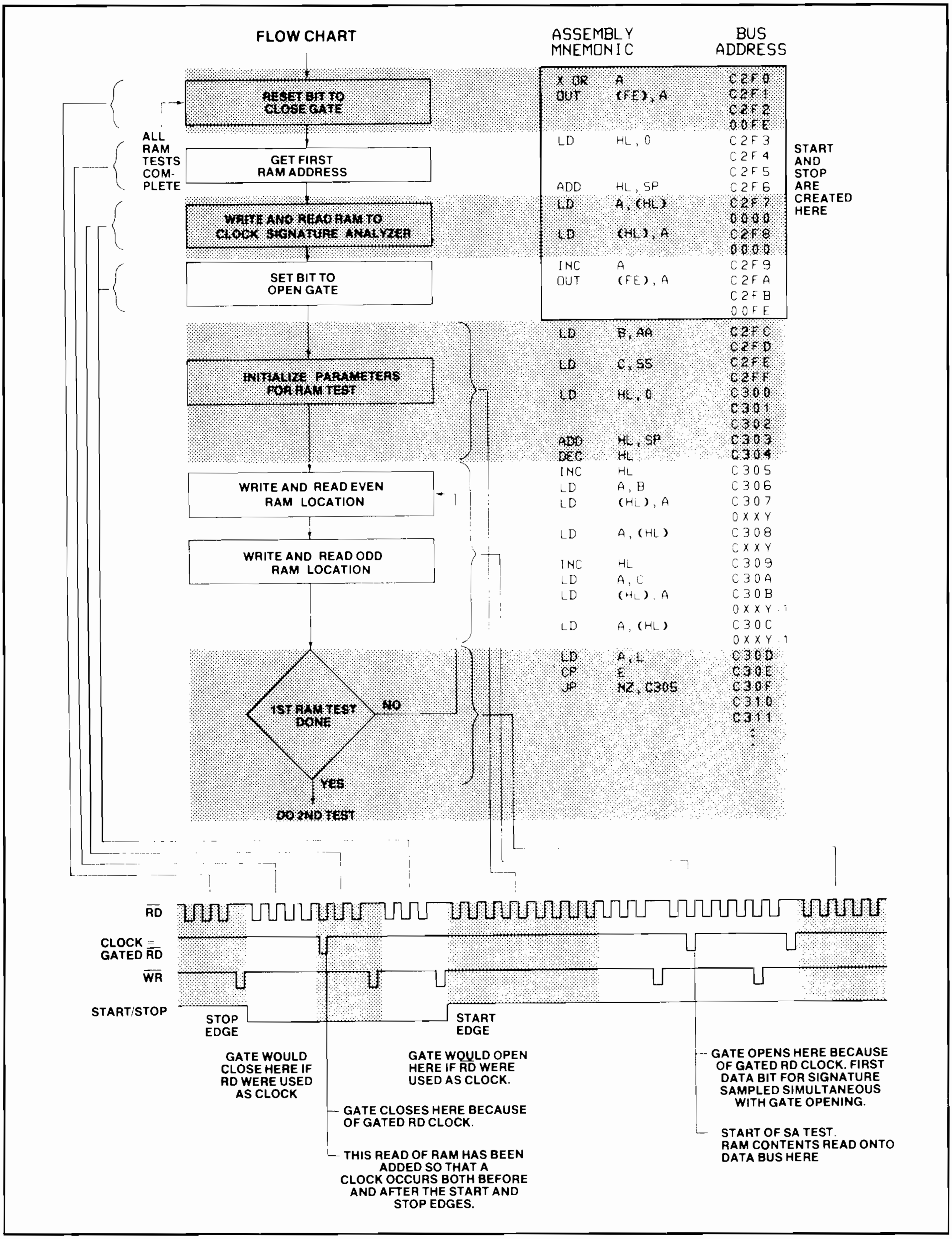


Figure 11. These five block diagrams show the signal flow when the Z80 or the VIDEO GENERATOR accesses the SCREEN RAM, GRAPHICS RAM or FONT ROM. a.) The Z80 gains access to the SCREEN RAM to modify or examine the ASCII characters that appear on the CRT. b.) The Z80 accesses GRAPHICS RAM to modify or examine the user-defined font stored there. c.) The VIDEO GENERATOR accesses SCREEN RAM and FONT ROM simultaneously to covert the ASCII characters in SCREEN RAM into standard text font that's displayed on the CRT. d.) GRAPHICS RAM is accessed only if non-ASCII characters are stored in SCREEN RAM. e.) These nodes are shared between the Z80 and the VIDEO GENERATOR. The signature analyzer requires a CLOCK that will sample data on the node only when the Z80 has access to RAM. Gating the Z80's $\overline{RD}$ control signal with the corresponding address decode for the RAM creates the required CLOCK. If $\overline{RD}$ alone were used to sample the asynchronous data on these nodes, then VIDEO GENERATOR data would also be sampled resulting in unstable signatures.


FLOW CHART
RESET BIT TO CLOSE GATE
GET FIRST RAM ADDRESS
WRITE AND READ RAM TO CLOCK SIGNATURE ANALYZER
SET BIT TO OPEN GATE
INITIALIZE PARAMETERS FOR RAM TEST
WRITE AND READ EVEN RAM LOCATION
WRITE AND READ ODD RAM LOCATION
1ST RAM TEST DONE
NO
YES
DO 2ND TEST
ALL RAM TESTS COM- PLETE
ASSEMBLY MNEMONIC
BUS ADDRESS
START AND STOP ARE CREATED HERE
RD
CLOCK = GATED RD
WR
START/STOP
STOP EDGE
START EDGE
GATE WOULD CLOSE HERE IF RD WERE USED AS CLOCK
GATE WOULD OPEN HERE IF RD WERE USED AS CLOCK.
GATE CLOSES HERE BECAUSE OF GATED RD CLOCK.
THIS READ OF RAM HAS BEEN ADDED SO THAT A CLOCK OCCURS BOTH BEFORE AND AFTER THE START AND STOP EDGES.
GATE OPENS HERE BECAUSE OF GATED RD CLOCK. FIRST DATA BIT FOR SIGNATURE SAMPLED SIMULTANEOUS WITH GATE OPENING.
START OF SA TEST. RAM CONTENTS READ ONTO DATA BUS HERE

```
ASSEMBLY              BUS
MNEMONIC              ADDRESS
XOR   A               C2F0
OUT   (FE),A          C2F1
                      C2F2
                      00FE
LD    HL,0            C2F3
                      C2F4
                      C2F5
ADD   HL,SP           C2F6
LD    A,(HL)          C2F7
                      0000
LD    (HL),A          C2F8
                      0000
INC   A               C2F9
OUT   (FE),A          C2FA
                      C2FB
                      00FE
LD    B,AA            C2FC
                      C2FD
LD    C,55            C2FE
                      C2FF
LD    HL,0            C300
                      C301
                      C302
ADD   HL,SP           C303
DEC   HL              C304
INC   HL              C305
LD    A,B             C306
LD    (HL),A          C307
                      0XXY
LD    A,(HL)          C308
                      CXXY
INC   HL              C309
LD    A,C             C30A
LD    (HL),A          C30B
                      0XXY+1
LD    A,(HL)          C30C
                      0XXY+1
LD    A,L             C30D
CP    E               C30E
JP    NZ,C305         C30F
                      C310
                      C311
                      ⋮
```

Figure 12. Generation of the START and STOP edges for tests 7-9 is similar to tests 0-6 and B-C of figure 10. In this case, extra code has been added between the generation of START and STOP because of the gated CLOCK. The gated RD CLOCK used in these tests occurs only when the device being tested is read. The extra code adds both a read and a write access to the device under test between the STOP and START edges to ensure detection.

SECTION E—FAULT ISOLATION OF BUSED DEVICES

SA Test Organization Makes the Difference

The way the SA tests are organized can make a difference in the ease of isolating a fault in a device that communicates over a data bus. SA tests are generally written two different ways depending upon the troubleshooting environment.

1. Go/No-go indication of all bused devices.

   All bused devices are tested at the same time within one SA test loop so that:

   a. If there is no fault, further testing is not required.

   b. If there is a fault, the failure is indicated to be within a limited area of the PC board.

2. Fault isolation of a specific bused device.

   The troubleshooter knows within which area of the board the fault lies, and now he's trying to locate the device or process fault causing the problem.

For example, consider the MONITOR ROM tests 0-1. They are written to test each MONITOR ROM separately. But imagine for a moment that both ROMs are tested within the same SA Loop. That is, the contents of both ROMs are read back onto the data bus between the same START and STOP. When the test is run, a go/no-go indication can be obtained from eight signatures taken on the data bus. Correct signatures indicate that everything is correct for both ROMs. Incorrect signatures would indicate a failure in one of the ROMs or in the supporting circuitry. But it's not known which ROM has failed until the contents of each ROM is individually examined with signatures. There are several ways to do this:

1. Remove all ROMs from their sockets or disable all chip selects. Then add one ROM at a time to the circuit until incorrect signatures reoccur on the data bus.
2. FREERUN the Z80 and "window" around each ROM's data by moving START and STOP to each chip select until incorrect signatures occur.
3. Create a separate test for each ROM so that only that ROM's data is placed on the bus. START, STOP and CLOCK can remain connected to the same signals if under program control.

Separate tests were written for several reasons:

1. The health of each ROM and supporting circuitry is determined quickly by running each test and taking only eight data bus signatures.
2. Diagnostics other than SA stimulus routines had already limited the failure to ROM. The troubleshooter was now looking for the device or process fault causing the failure.
3. It was simpler to select a new test to run than to remove parts from the board or move START, STOP and CLOCK around from chip select to chip select.
4. No modifications to the board or hardware could be added to allow the ROM chip selects to be disabled because of the retrofit situation.

Separate tests have also been written for the PROCESSOR RAM of tests 2-6. They are also implemented to find the one bused RAM out of several that could be causing the fault. Separating the tests also made easy testing of optional sizes of PROCESSOR RAM. Each socket for a RAM can accept a 4K or 16K dynamic RAM part, or no part at all, allowing PROCESSOR RAM to vary from 4K to 48K bytes total. Each variation of PROCESSOR RAM requires a new signature set. Since the PROCESSOR RAM tests are independent of the configuration, so is the signature documentation.

READ and WRITE as CLOCK Help Find the Faults

The SA test both reads and writes devices such as RAM, so that the Z80 s $\overline{RD}$ and $\overline{WR}$ outputs can be used as the CLOCK to determine if the fault is caused by the RAM being incorrectly read or written. When $\overline{RD}$ is used as the clock, and signatures on the data bus are correct, the RAM is both being read and written correctly. When signatures are incorrect, the CLOCK is moved to $\overline{WR}$ to check the signatures on all inputs to the RAM including control lines. If all signatures are correct, the problem exists with reading RAM. The CLOCK is moved back to $\overline{RD}$ to isolate the problem caused by reading of the RAM. It may be the RAM itself, control line inputs to the RAM and the support circuits that generate them, or a process fault.

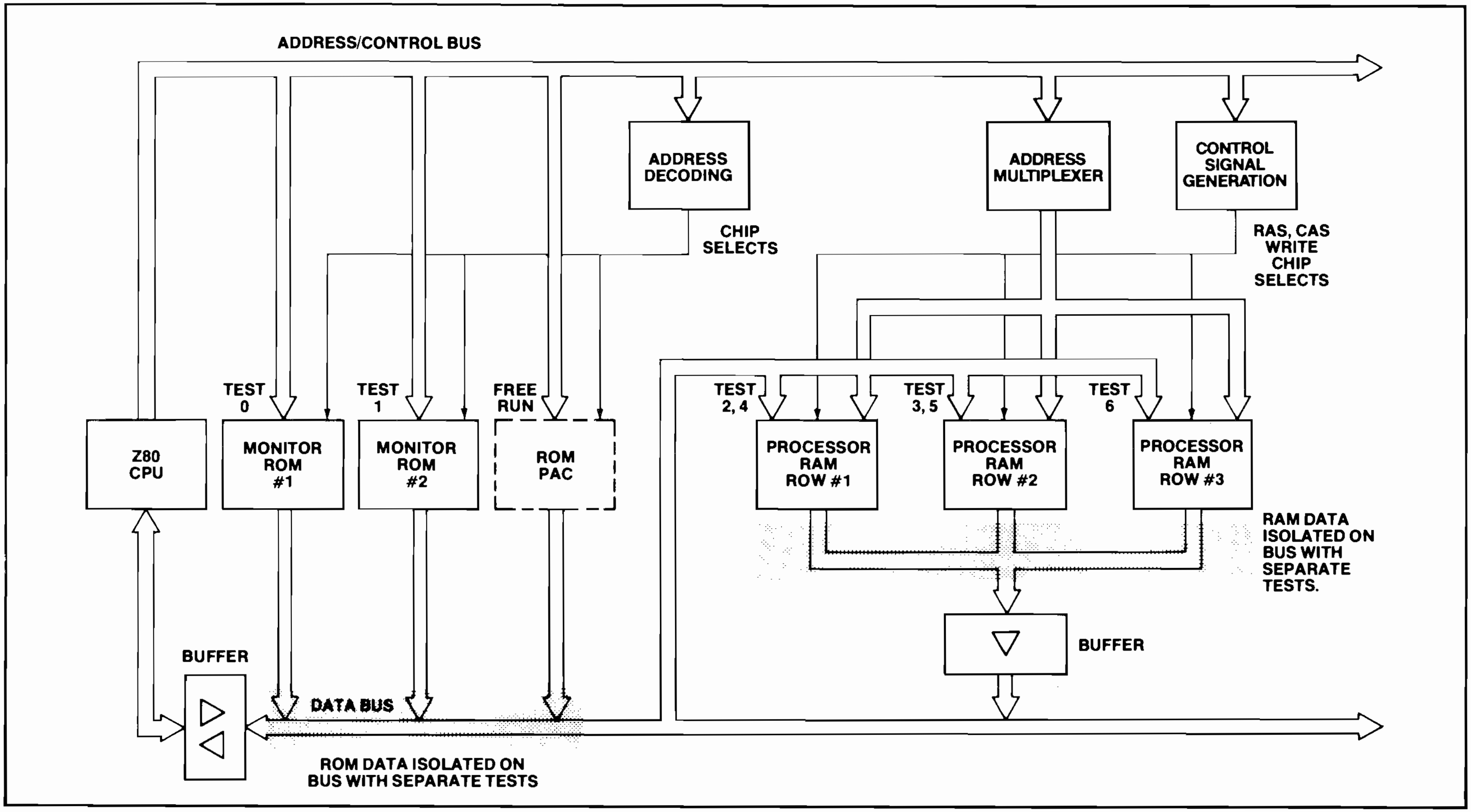


Figure 13. Each MONITOR ROM and each row of PROCESSOR RAM is tested separately to allow easier fault isolation of these bused devices. This ensures that the only data on any node comes from the device being tested and nothing else. (Except for the program that is executing from ROM PAC, which has to be good for the program to run. It is tested with FREERUN when it's bad.) A faulty device being tested is isolated by following incorrect signatures back to the source of the fault. Similarly, when data comes from devices that shouldn't be on the bus, it is detected as an incorrect signature and is also traced to the fault. $\overline{RD}$ and $\overline{WR}$ as CLOCKS also sort out faults. When incorrect signatures appear on the outputs of the RAM during a test using $\overline{RD}$ as a CLOCK, sometimes one or more RAM cell is bad, or control signals into the RAM are faulty (such as RAS, CAS, and CHIP SELECT), or the data was incorrectly written into RAM. To isolate, $\overline{WR}$ is used as a CLOCK to check signatures on the inputs that could affect writing of RAM (e.g. the data bus inputs and control lines such as RAS, CAS, WRITE and CHIP SELECT). If they have the correct signatures, the problem has to do with reading RAM.

SECTION F—STATIC VIDEO PATTERN, TEST A

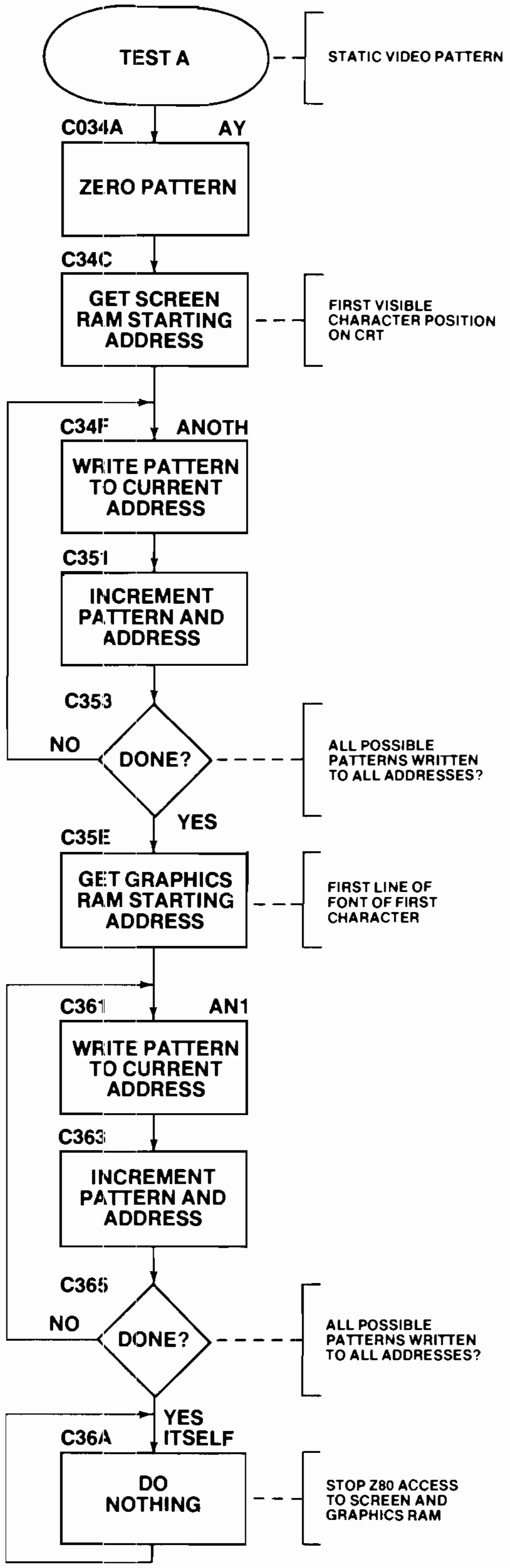

TEST A
STATIC VIDEO PATTERN
C034A
AY
ZERO PATTERN
C34C
GET SCREEN RAM STARTING ADDRESS
FIRST VISIBLE CHARACTER POSITION ON CRT
C34F
ANOTH
WRITE PATTERN TO CURRENT ADDRESS
C351
INCREMENT PATTERN AND ADDRESS
C353
NO
DONE?
ALL POSSIBLE PATTERNS WRITTEN TO ALL ADDRESSES?
YES
C35E
GET GRAPHICS RAM STARTING ADDRESS
FIRST LINE OF FONT OF FIRST CHARACTER
C361
AN1
WRITE PATTERN TO CURRENT ADDRESS
C363
INCREMENT PATTERN AND ADDRESS
C365
NO
DONE?
ALL POSSIBLE PATTERNS WRITTEN TO ALL ADDRESSES?
YES
ITSELF
C36A
DO NOTHING
STOP Z80 ACCESS TO SCREEN AND GRAPHICS RAM

Figure 14. The static video pattern is written into RAM by this program. All possible characters and all possible patterns of user-defined font are written into SCREEN RAM and GRAPHICS RAM respectively. The program then enters a small loop that keeps the Z80 from further interaction with either RAM. This keeps the node activity limited to the CRT refresh process so that signatures are stable.

```
 HEX
ADRS    CONTENTS   LABEL    INSTRUCTION

C34A               AY
C34A    0600                LD     B,0
C34C    2180F0              LD     HL,0F080H
C34F               ANOTH:
C34F    78                  LD     A,B
C350    77                  LD     (HL),A
C351    23                  INC    HL
C352    04                  INC    B
C353    7D                  LD     A,L
C354    B7                  OR     A
C355    C24FC3              JP     NZ,ANOTH
C358    7C                  LD     A,H
C359    FEF8                CP     0F8H
C35B    C24FC3              JP     NZ,ANOTH
C35E    21000FC             LD     HL,0FC00H
C361               AN1:
C361    78                  LD     A,B
C362    77                  LD     (HL),A
C363    23                  INC    HL
C364    04                  INC    B
C365    7D                  LD     A,L
C366    B4                  OR     H
C367    C261C3              JP     NZ,AN1
C36A    C36AC3     ITSELF   JP     ITSELF
```

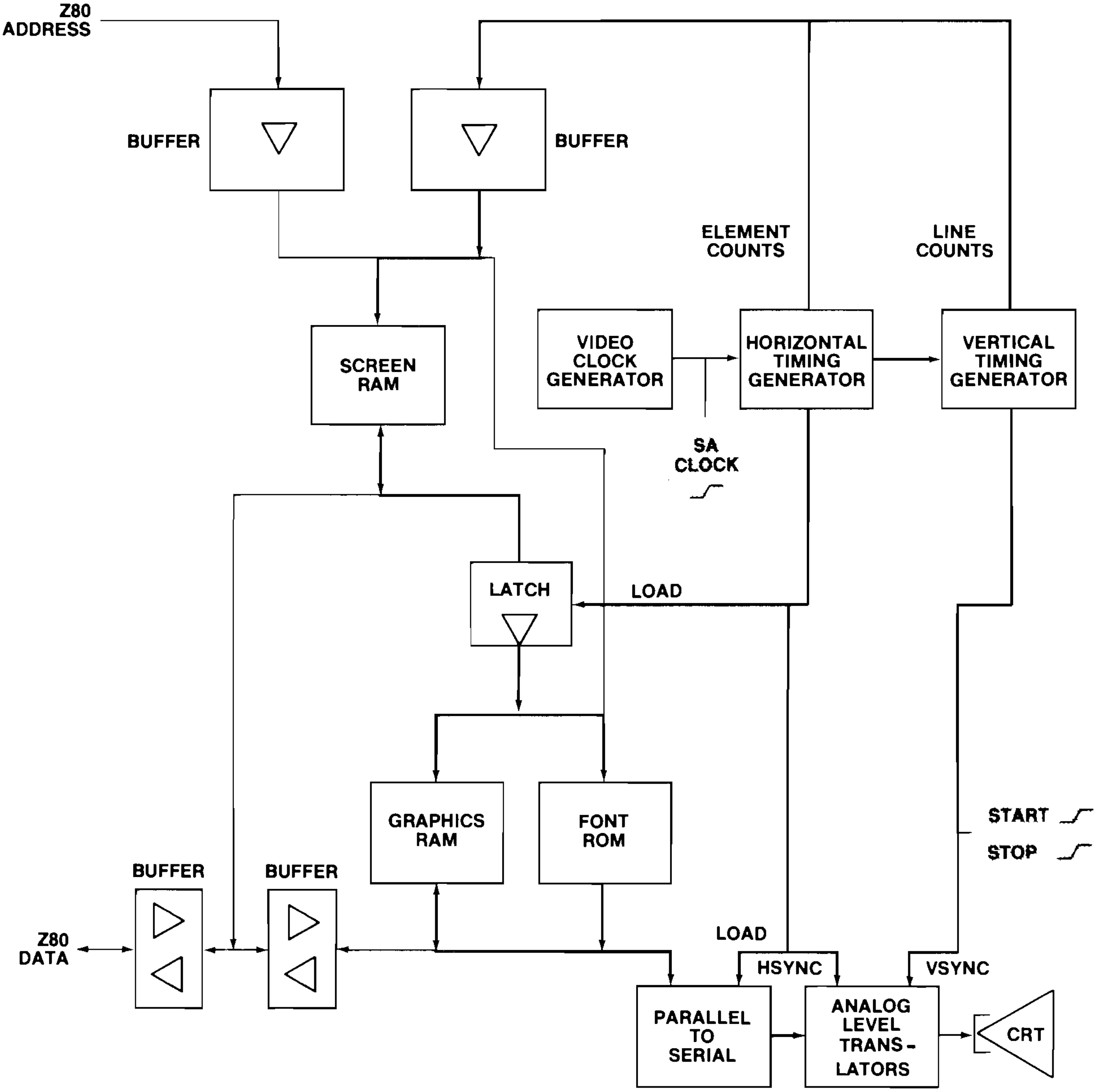


Figure 15. These nodes are stimulated by the CRT refresh process while the static video pattern is displayed on the CRT. These START, STOP and CLOCK connections form a GATE that is open for one complete refresh cycle of the CRT, allowing the signature analyzer to detect errors in any of the circuits including all the font patterns in ASCII FONT ROM. When the TIMING GENERATORS fail, START and STOP are no longer generated. In that case, START and STOP are moved to the connections shown in the next figures, closer to the kernel circuits of the VIDEO TEXT GENERATOR.

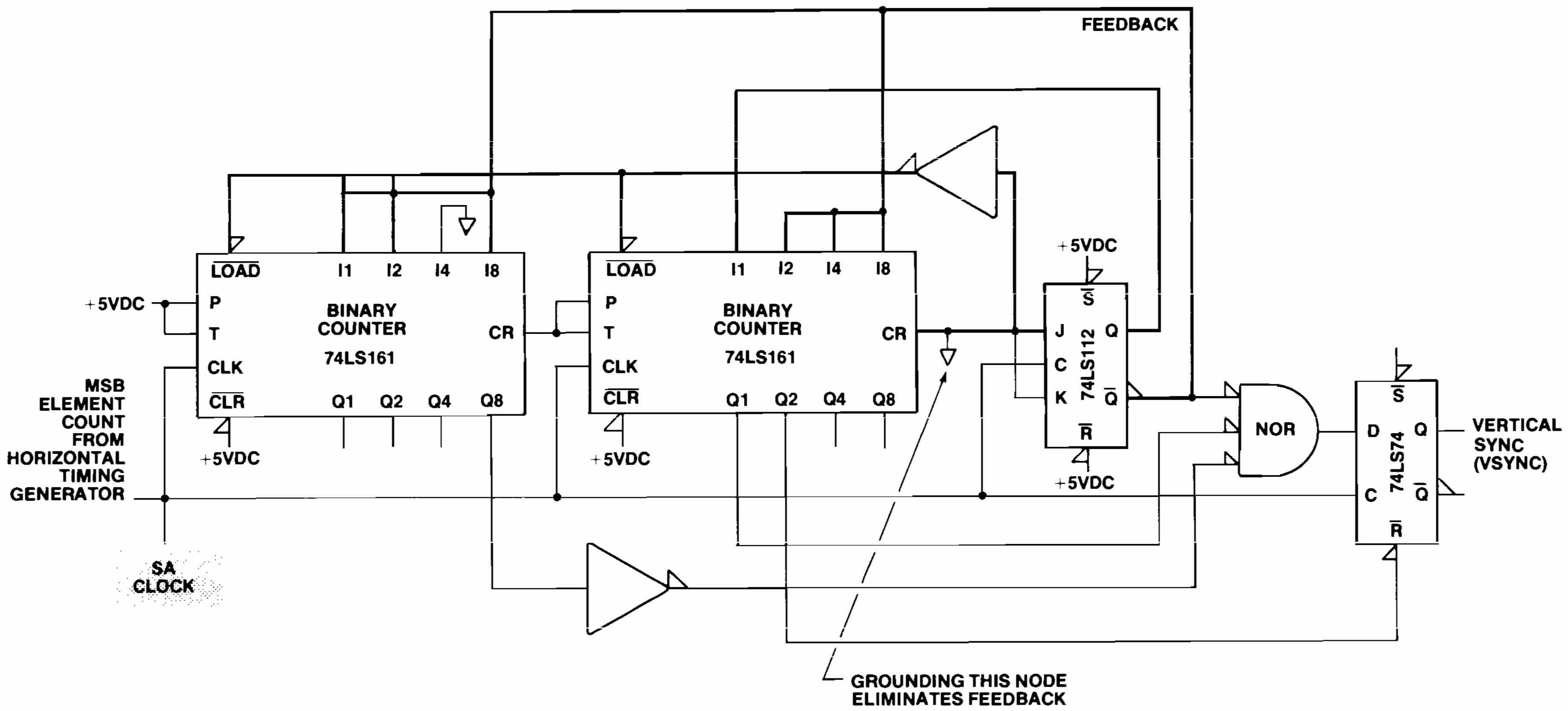


Figure 16. Grounding one point causes these VERTICAL TIMING GENERATOR counters to "FREERUN" through all possible states instead of counting CRT vertical lines. It does this by eliminating feedback loops. While grounding the output of a TTL device is not a recommended procedure, it was not possible to design-in a jumper that would disconnect the output from the circuit and ground the remaining inputs. To troubleshoot the freerunning counters, CLOCK is connected to the counter's clock input (the MSB of the HORIZONTAL TIMING GENERATOR of the next figure). The DATA input is placed on a source of logic high such as +5vdc. START and STOP are both moved to the circuit node under test to take a signature. The signature for any node then represents the number of CLOCK edges between START and STOP. If incorrect signatures occur for all nodes of these counters, then START, STOP and CLOCK are moved as shown in the next figure.

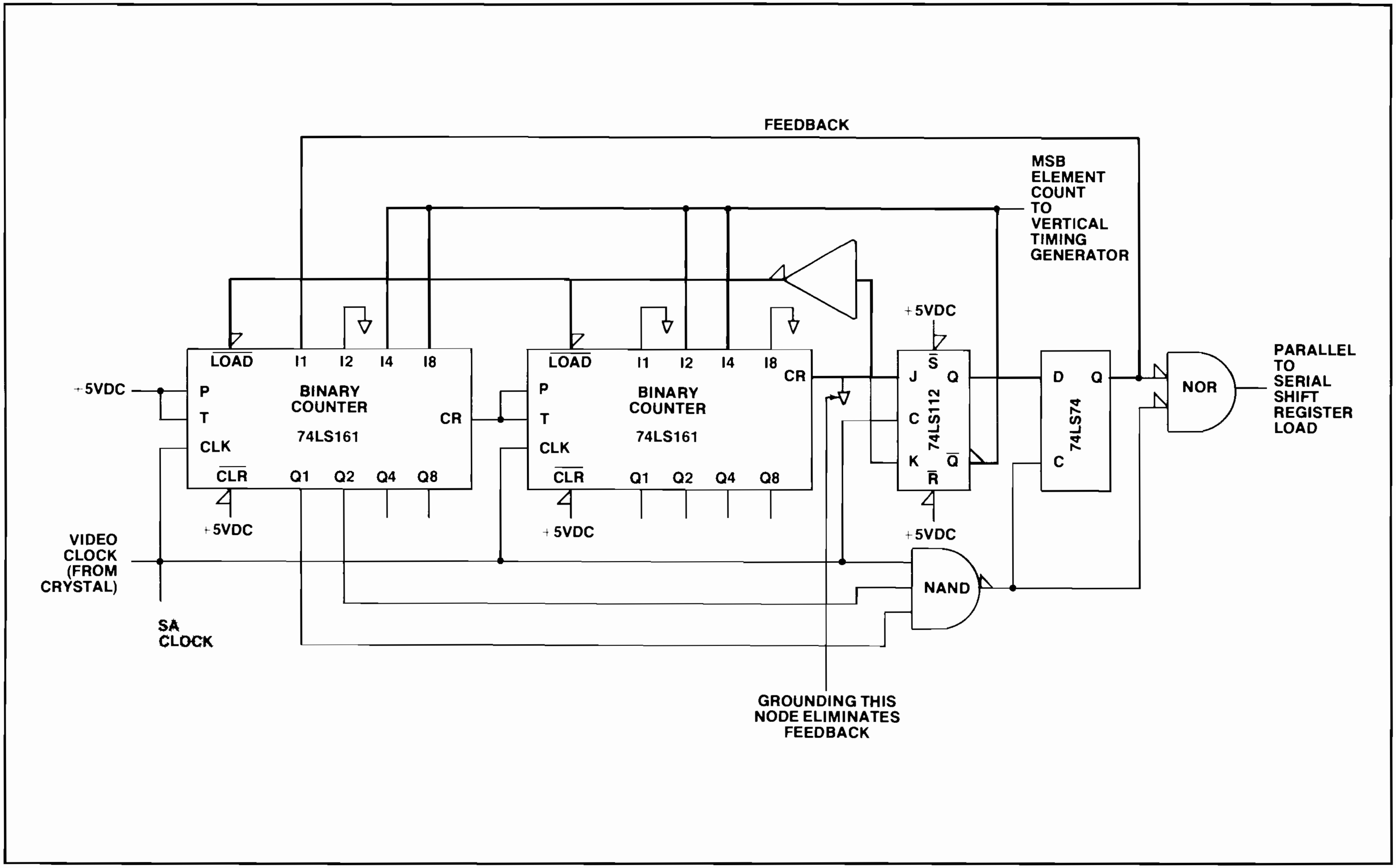


Figure 17. Freerunning the HORIZONTAL TIMING GENERATOR counters is similar to the VERTICAL TIMING GENERATOR counters of the previous figure. Grounding one point FREERUNs the counters. CLOCK is connected to the counter's clock input, the source of the VIDEO CLOCK. The DATA input is placed on +5vdc and START and STOP are moved from node to node to take signatures. These counters only require that the crystal oscillator of the VIDEO CLOCK GENERATOR be operating for FREERUN to occur.

SECTION G—PARALLEL PORT, TEST B

External hardware was required to stimulate the INPUT PORT. Wires on an external connector loop the patterns that are written to the OUTPUT PORT back to the INPUT PORT. Timing requirements of the HANDSHAKE control lines made it impossible to simply loop their outputs back to the inputs with similar wires on the connectors. It was decided not to add the ICs that would be required to fully stimulate the HANDSHAKE circuits. Two things determined this. First, the extra hardware would not be available to the distributors or field service personnel. Second, the HANDSHAKE circuits consist of one IC that can be easily troubleshot with other means such as logic probes.

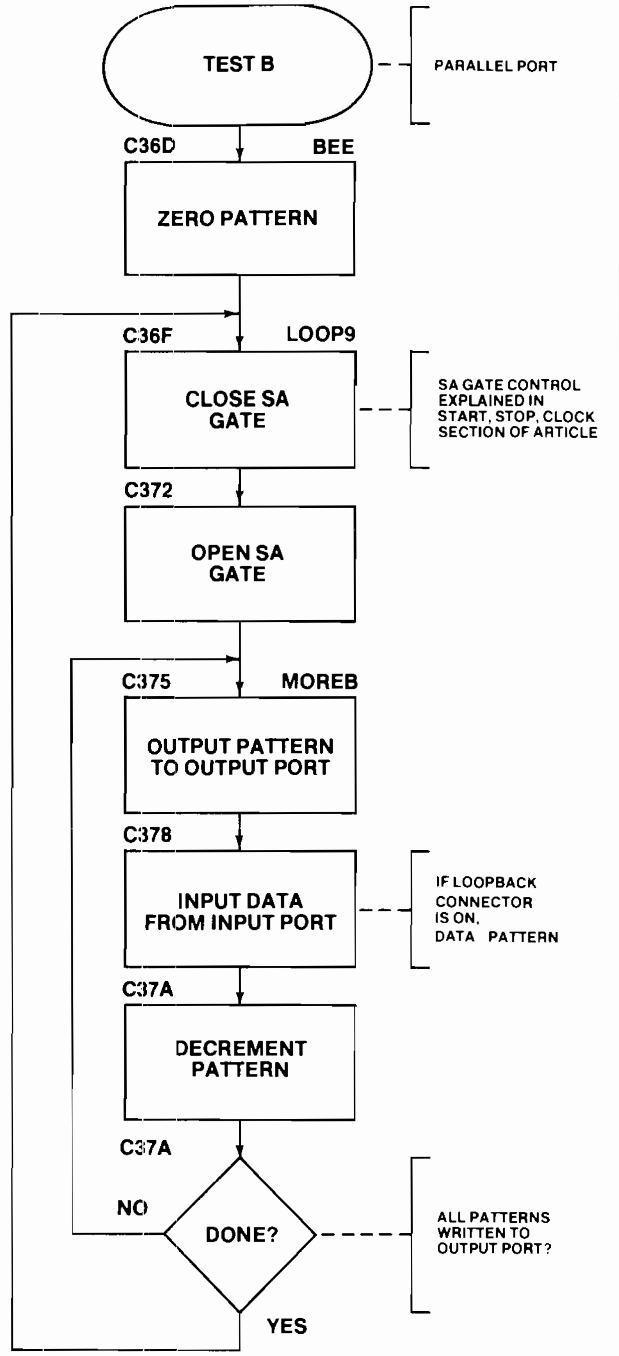

TEST B
PARALLEL PORT
C36D
BEE
ZERO PATTERN
C36F
LOOP9
CLOSE SA GATE
SA GATE CONTROL EXPLAINED IN START, STOP, CLOCK SECTION OF ARTICLE
C372
OPEN SA GATE
C375
MOREB
OUTPUT PATTERN TO OUTPUT PORT
C378
INPUT DATA FROM INPUT PORT
IF LOOPBACK CONNECTOR IS ON, DATA PATTERN
C37A
DECREMENT PATTERN
C37A
NO
DONE?
YES
ALL PATTERNS WRITTEN TO OUTPUT PORT?

Figure 18. This program stimulates both the OUTPUT and INPUT PARALLEL PORTS by continuously writing all possible patterns to the OUTPUT PORT. The program also reads the INPUT PORT whether it is stimulated or not. The INPUT PORT is stimulated by looping the OUTPUT PORT back to the INPUT PORT using the connector shown in the following figure.

```
HEX
ADRS   CONTENTS   LABEL    INSTRUCTION

C36D              BEE:
C36D   0600                LD     B,0
C36F              LOOP9:
C36F   AF                  XOR    A
C370   D3FE                OUT    (0FEH),A
C372   3C                  INC    A
C373   D3FE                OUT    (0FEH),A
C375              MOREB:
C375   78                  LD     A,B
C376   D3FF                OUT    (0FFH),A
C378   DBFF                IN     A,(0FFH)
C37A   10F9                DJNZ   MOREB-$
C37C   D36FC3              JP     LOOP9
```

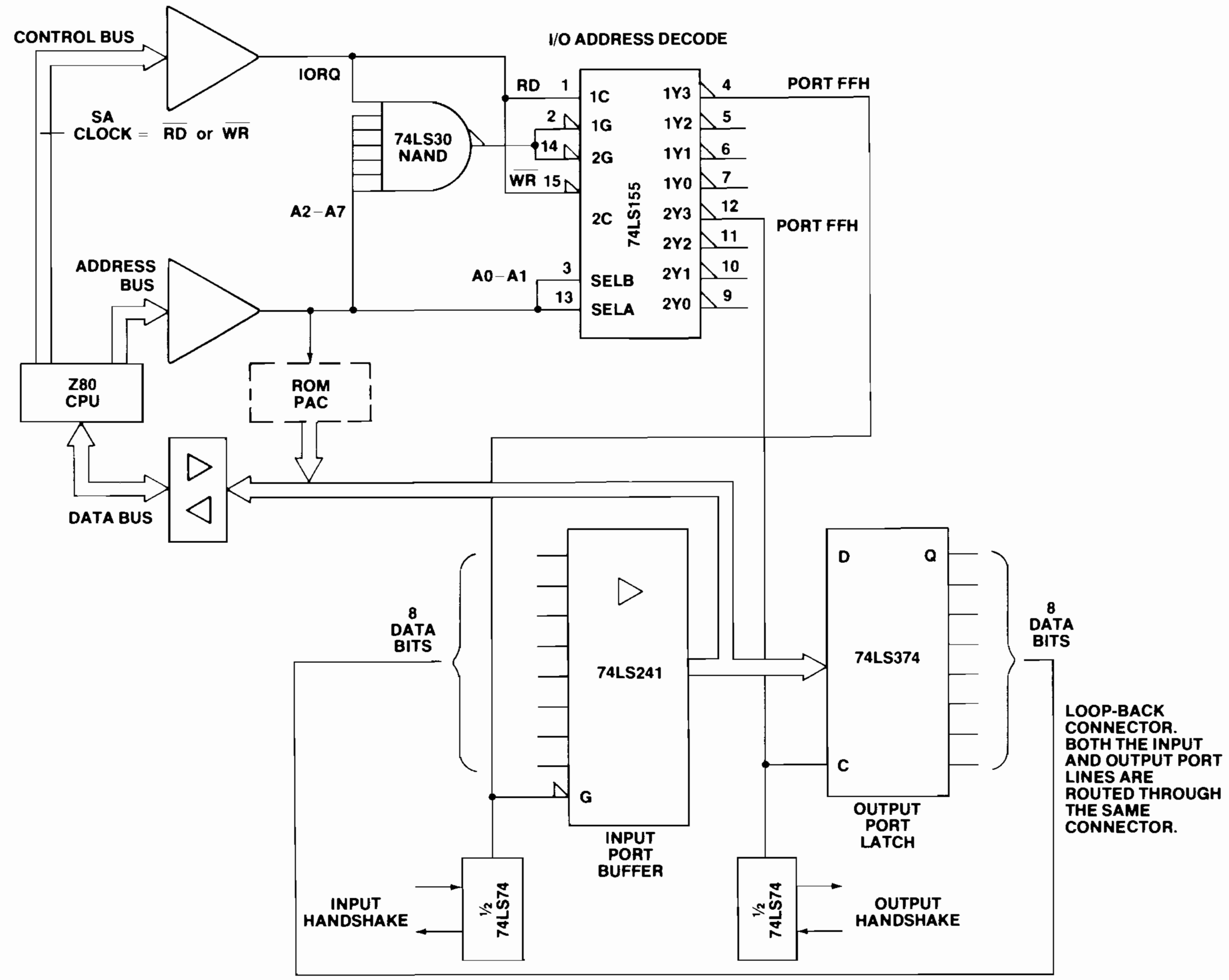


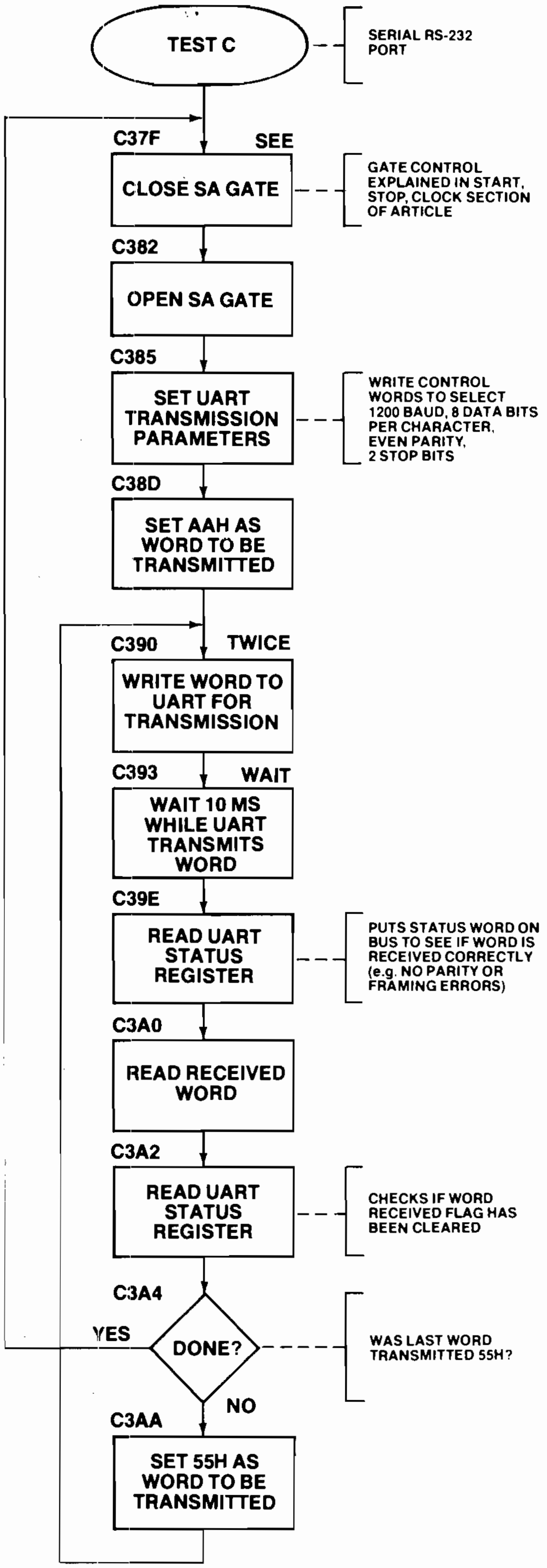
Figure 19. These circuits are stimulated by the PARALLEL PORT test. START and STOP are connected to a bit in the KEYBOARD SCAN LATCH and are controlled by the program as shown in the section called THE HARDWARE AND SOFTWARE BEHIND START, STOP AND CLOCK. With the loop-back connector on, signatures are taken on the data bus using $\overline{RD}$ as a CLOCK. If signatures are correct, then both the OUTPUT and INPUT PORT are operating correctly. If signatures are incorrect, the connector is removed and signatures are taken on the OUTPUT PORT using $\overline{WR}$ as the CLOCK. Repairs are made as required. The loopback connector is then replaced to check the INPUT PORT for problems associated with reading it. The text explains why the HANDSHAKE circuitry is not stimulated by this test.

SECTION H—SERIAL RS-232 PORT, TEST C

This test stimulates the UART so that it can be troubleshot with conventional means such as an oscilloscope. However, unstable signatures on the serial output line make it impossible to troubleshoot it with SA. Unstable signatures result basically because the UART is an asynchronous device with respect to the Z80. SA cannot be used to troubleshoot asynchronous devices in general. Here are the details why SA cannot be used here.

- With START and STOP controlled by the program and RD or WR used as a CLOCK, unstable signatures occur on the UART. This is because the UART's serial output is asynchronous to the RD or WR CLOCK. However, the output **is** synchronous to the serial word transmission clock.
- If the CLOCK is moved to the UART transmitter clock input, then signatures might be stable if a START and STOP connection could be made that was synchronous to the CLOCK. This is not possible without limiting the UART's serial output to a continuous one-word transmission.

The serial transmission of this test consists of two words, 55H and AAH. If 55H were the only word transmitted, START could be connected to the serial output line itself and trigger on the falling edge of the start bit of the serial word. STOP could be connected to an output of the UART that signals the serial transmission is complete when at least one stop bit has been detected. CLOCK could be connected to the UART's serial transmission clock input. This one-word transmission technique should work but has not been tried as of the publication of this article.


TEST C
SERIAL RS-232 PORT
C37F
SEE
CLOSE SA GATE
GATE CONTROL EXPLAINED IN START, STOP, CLOCK SECTION OF ARTICLE
C382
OPEN SA GATE
C385
SET UART TRANSMISSION PARAMETERS
WRITE CONTROL WORDS TO SELECT 1200 BAUD, 8 DATA BITS PER CHARACTER, EVEN PARITY, 2 STOP BITS
C38D
SET AAH AS WORD TO BE TRANSMITTED
C390
TWICE
WRITE WORD TO UART FOR TRANSMISSION
C393
WAIT
WAIT 10 MS WHILE UART TRANSMITS WORD
C39E
READ UART STATUS REGISTER
PUTS STATUS WORD ON BUS TO SEE IF WORD IS RECEIVED CORRECTLY (e.g. NO PARITY OR FRAMING ERRORS)
C3A0
READ RECEIVED WORD
C3A2
READ UART STATUS REGISTER
CHECKS IF WORD RECEIVED FLAG HAS BEEN CLEARED
C3A4
YES
DONE?
NO
WAS LAST WORD TRANSMITTED 55H?
C3AA
SET 55H AS WORD TO BE TRANSMITTED

```
HEX
ADRS   CONTENTS  LABEL   INSTRUCTION

C37F             SEE:                        ;SERIAL RS-232 PORT
C37F   AF                XOR   A
C380   D3FE              OUT   (0FEH),A      ;CLOSE  S.A.GATE
C382   3C                INC   A
C383   D3FE              OUT   (0FEH),A      ;OPEN S.A.GATE
C385   3EC1              LD    A,0C1H        ;1200 BAUD, RS-232 PORT
C387   D3FE              OUT   (0FEH),A
C389   3E0F              LD    A,0FH         ;8 BITS/CHAR, 2 STOP BITS,
C38B   D3FD              OUT   (0FDH),A      ;EVEN PARITY.
C38D   3EAA              LD    A,0AAH
C38F   57                LD    D,A
C390             TWICE:
C390   7A                LD    A,D
C391   D3FC              OUT   (0FCH),A
C393   01DC05            LD    BC,05DCH      ;10 MS DELAY CONSTANT
C396             WAIT:
C396   0D                DEC   C
C397   C296C3            JP    NZ,WAIT       ;21010 T-STATES  LATER
C39A   05                DEC   B
C39B   C296C3            JP    NZ,WAIT
C39E   DBFD              IN    A,(0FDH)
C3A0   DBFC              IN    A,(0FCH)
C3A2   DBFD              IN    A,(0FDH)
C3A4   7A                LD    A,D
C3A5   FE55              CP    55H
C3A7   CA7FC3            JP    Z,SEE
C3AA   1655              LD    D,55H
C3AC   C390C3            JP    TWICE
```

Figure 20. The serial inputs and outputs of a UART are stimulated by this program along with the control and status registers. The UART is first set to send and receive serial words with eight data bits, even parity and two stop bits at 1200 baud. Next the program writes the word AAH into the UART and waits for its transmission to complete. Then the status word is read onto the data bus along with the serial word received, and then the status word again. Finally 55H is loaded for transmission similar to the word AAH. The program jumps back to the beginning of the loop upon completion.

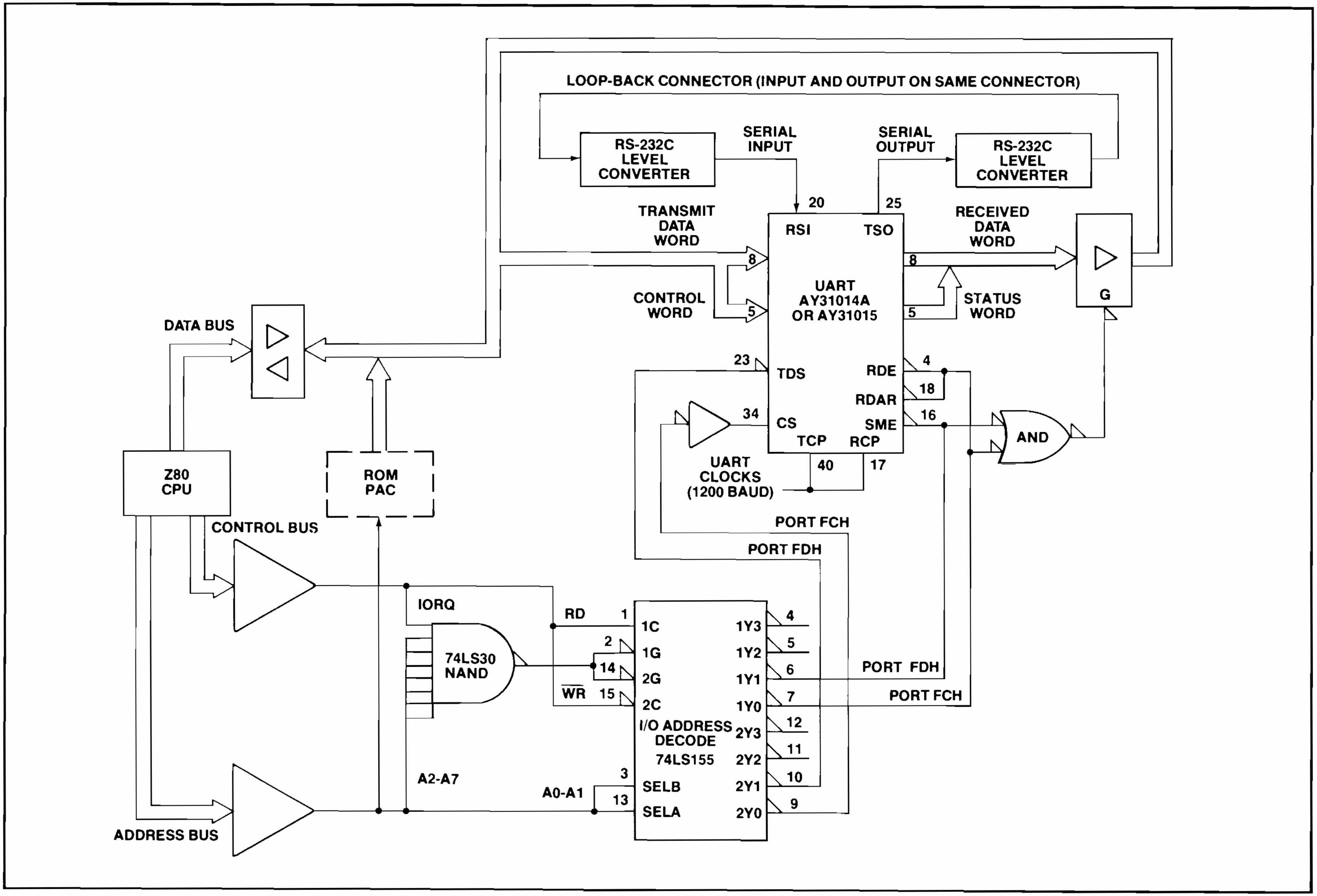


Figure 21. The stimulus provides a go/no-go indication of correct circuit operation but cannot be used to troubleshoot the UART with SA. Unstable signatures result on the outputs as outlined in the text.

SECTION I—A DESIGNER'S CHECKLIST FOR GETTING STARTED WITH SA

Here's a summary of this article that can be used as a checklist when designing or retrofitting SA into a microprocessor based system. It is not limited to Z80-based systems or personal computers.

1. Provide a means to FREERUN the microprocessor by using a FREERUN fixture or by designing in a way to open the data bus and force the NOP or FREERUN instruction into the processor. Find START, STOP and CLOCK connections on the processor.
2. Provide a means to store, access, and select the SA stimulus programs. Try to find a way that depends only upon circuits that can be troubleshot with FREERUN or logic probes in a simple manner.
3. Create START and STOP using software to control hardware that's already available or hardware that is specially designed into the product for SA testing. Use circuits that can be checked by FREERUN, a previous SA test, or other easy means such as logic probes.
4. Choose a CLOCK that is synchronous to START, STOP and DATA on all nodes being tested. Be sure there's a CLOCK edge both before and after the START and STOP edges. Avoid CLOCKing DATA from a node when it's in the 3rd state.
5. Create software test loops that can give both a go/no-go indication of all bused devices (like a diagnostic) and also allow fault isolation of a bad component or process fault even with bused devices. The tests can be separate.
6. Be sure your test can isolate a failure because of either a read or write problem with the device (e.g. RAM).
7. Provide a way to stimulate uncontrolled inputs of I/O devices in a synchronous fashion using loop-back connectors or external stimuli.
8. Provide means to open feedback loops. Usually only a concern in circuits that are independent of the processor.
9. One-shots and UART serial outputs generally cannot be tested with SA, so find a way to bypass them (eliminate their effect on other circuit elements) during the SA test so that all nodes operate synchronous to the CLOCK.
10. Be sure your tests don't depend upon circuits working that are being tested. Usually done by running the SA tests open-loop (i.e. the tests only stimulate the devices but don't check the results to see if it was accomplished. The signature analyzer will check the results.). Sometimes can happen inadvertently when controlling START and STOP with software.

13. Developing Digital Hardware Using Word/Data Generators.

HEWLETT PACKARD

Guenter Riebesell of the Boeblingen Division was Project Leader for the 8084A and 8170A Signal Sources. Guenter earned his Diplom Ingeniur at Technische Universitaet, Braunschweig.

Title: Applying Modern Data Generators to Design and Production Problems of Digital ICs and Printed Circuit Boards.

Abstract: This paper will investigate the testing requirements during the design and production of digital ICs and printed circuits. Solutions using data generators in a stimulus-DUT-response system will be presented. Individual examples for functional and parametric tests will provide further insight into problems, present general solutions and hp contributions.

A brief outlook of trends in this area will conclude this paper.

I INTRODUCTION

Todays digital products include ICs, instrumentation and systems. Reflecting the demands of the application, they range from simple to highly complex devices which offer the user revolutionary capabilities. Engineers responsible for digital products demand economical, efficient test equipment to evaluate and troubleshoot their designs.

As microprocessors - and, indeed, all forms of digital circuitry - become more complex and widespread with their penetration of new applications, the need for versatile, cost-effective equipment is of increasing importance.

This paper will concentrate on instrumentation that fulfills the measurement requirements in digital computation and digital data processing. Particular attention will be paid to performance evaluation of a device rather than to its design.

In development as well as production or maintenance testing, word generators and logic analyzers are used alongside the traditional oscilloscope mainly in bench applications, but also in systems.

II Phases and Activities in Design and Production

Before looking into the various types of data generators and their applications, a brief review of the main departments of electronics companies shall characterise the phases and activities, in which data generators ease the work.

Typically an electronics company is organized in a vertical structure of departments (R&D, Production, Marketing, QA, Services, Sales, etc.) headed by general management.

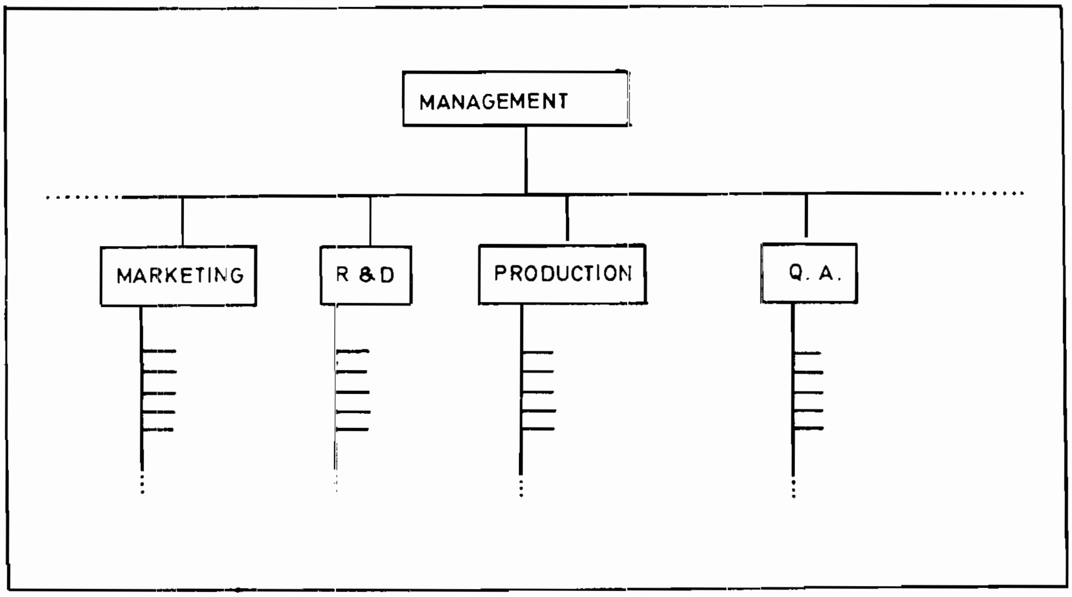


Fig. 1: Vertical Structure of Companies

Although companies producing ICs are visually organized in a similar way to instrument manufacturers, there are significant differences in the test equipment they use.

The activities carried out by an IC manufacturer's R&D and production departments can be represented by the following figure:

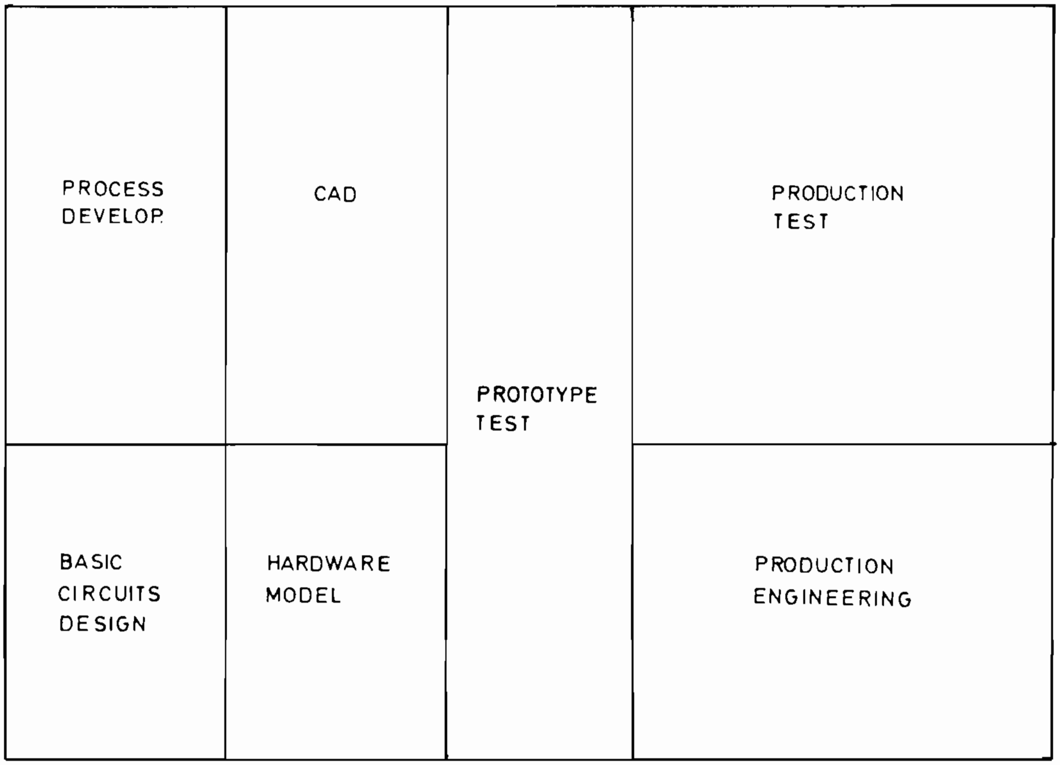


Fig. 2: Model of IC Manufacturers activities in R&D and Production

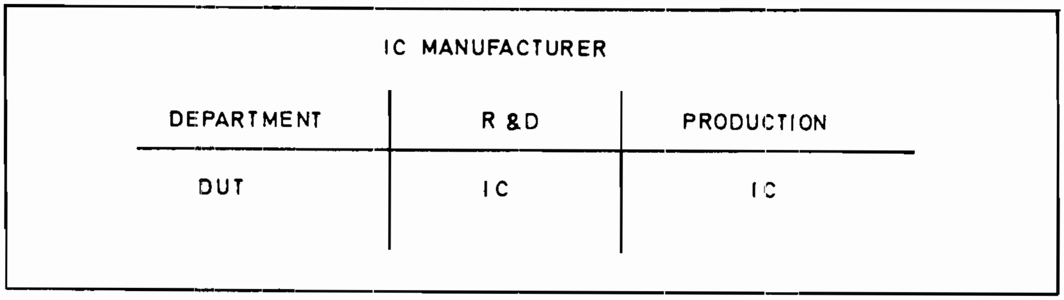
Process Development and Basic Circuit Design are responsible for new semiconductor technologies and for new and better processes. This activity can be part of a single R&D lab or concentrated in a central research facility. Their equipment is mainly of physical and chemical nature plus instrumentation for measurements of capacitance, voltages and low currents.

The second column in the figure represents IC development. Here large computers with powerful software have recently become major design tools for engineers. CAD (Computer Aided Design) programs and simulators do most of the circuit design work, while CAA (Computer Aided Artwork) helps the engineer with the circuit layout. Nevertheless, hardware models will hold places in future where high speed or complex circuit design could be critical. Evaluation of the model is carried out by measuring its performance when stimulated by a pulse or data generator. After design and layout work has been finished prototypes are manufactured and tested. In most cases prototypes are tested on wafer with commercially available IC-test systems. Only in high speed functions are the IC's packaged and tested with self-tailored systems which include high-frequency pulse and data generators.

In the majority of production facilities the high volume of IC's are tested either before packaging, at wafer stage, or after packaging.

Automatic test systems, which fit all the needs required, are used. In some cases these test systems are tailored to a specific task. They are assembled either with available instruments (where again pulse and data generators are found) or with self-developed circuits.

As a summary, an IC manufacturer's R&D and production departments work on the same DUT (the IC). As will be seen, this contrasts with the instrument manufacturer's situation.



IC MANUFACTURER

DEPARTMENT	R &D	PRODUCTION
DUT	IC	IC

Fig. 3a: Device types tested in different departments: IC Manufacturer

CIRCUIT DESIGN / INSTRUMENT MANUFACTURER

DEPARTMENT	I.I.	R & D	PRODUCTION
DUT	IC	IC/PC BOARD	PC BOARD / SUB-ASSEMBLY

Fig. 3b: Device types tested in different departments: Instrument Manufacturer

The instrument manufacturer buys his components from vendors. In many cases he makes an Incoming Inspection in order to find out defective parts and thus minimise failure costs in production cycle. IC-test-systems are found here, only if high volume of few components justify it. If system investment is not justified, bench instrumentation is used. This may include pulse and data generators, logic analyzers and computers/calculators for automatic operation.

In R&D, engineers take IC's from the companies component stock for the development of their circuits. They will also evaluate new IC's which promise a solution - or a better or more cost effective solution. This evaluation is done with existing lab equipment because the task is unique and not repeated.

When a new instrument is to be developed a functional description, with all specifications, is first established. To assure that the functional modules will work together later on, a thorough, (hopefully) complete and correct definition of interface signals and perhaps of interface hardware is done. Each functional module with its related interface definition then becomes a design task for an engineer.

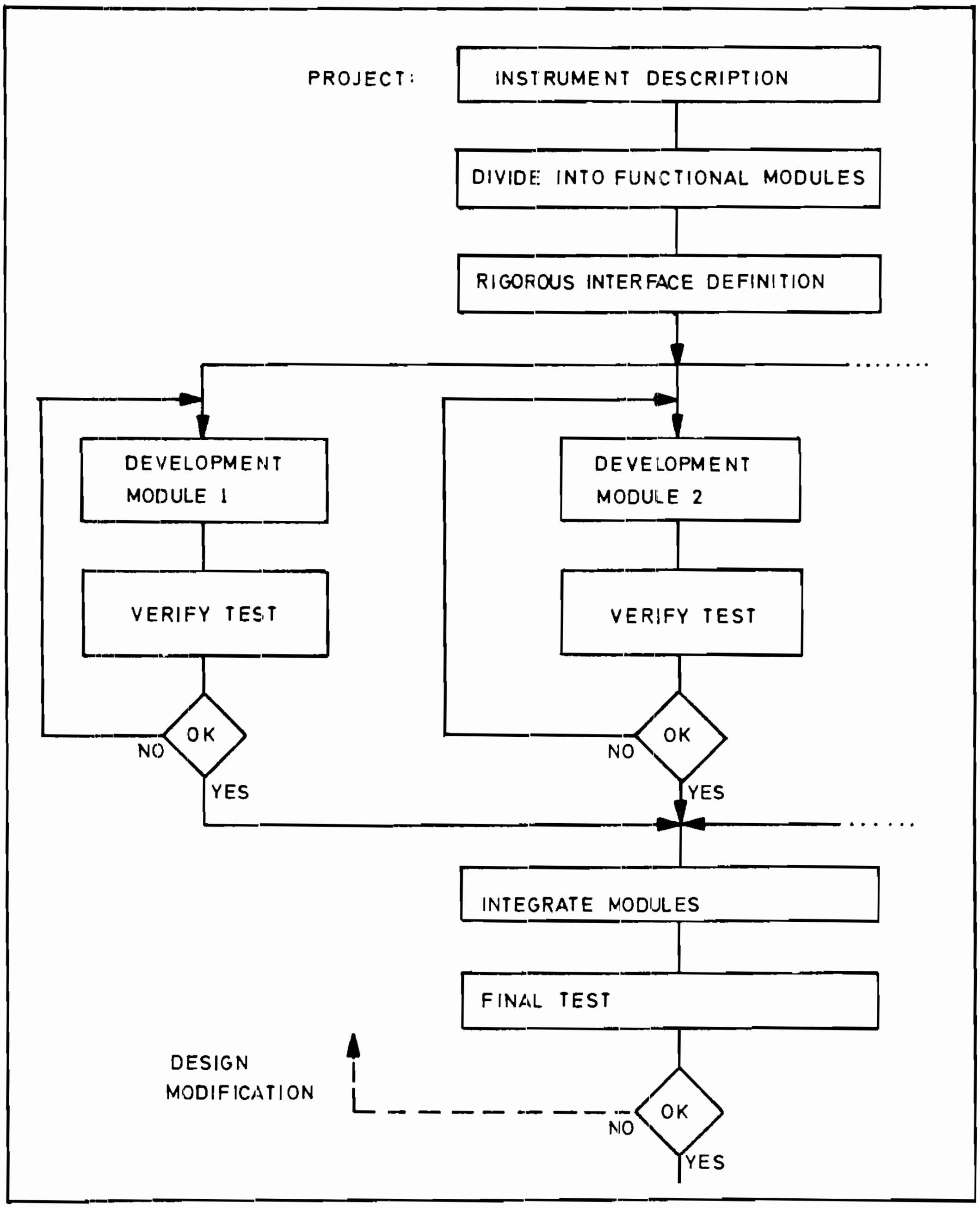


Fig. 4: Simplified R&D flow diagram of an instrument's development

Very soon the engineer will come to a point where he needs the interface signals from the other modules. But these modules are also in development, forcing their engineers to the same problem. Here a data generator can substitute the missing modules by simulating the data required by the module of interest. The simulated data is determined by the bit pattern loaded into the data generator.

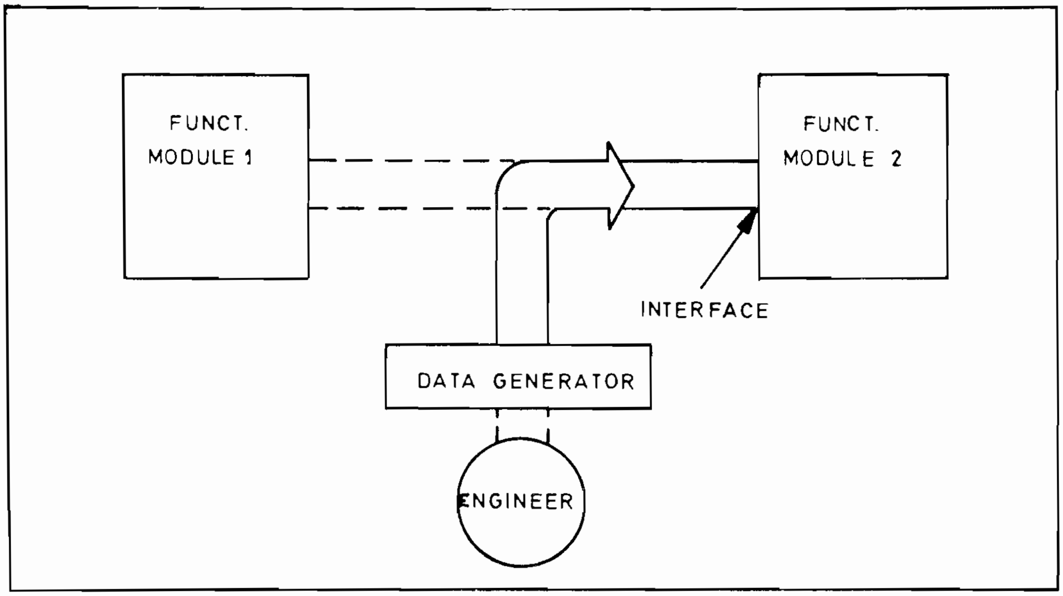


Fig. 5: Data Generator substituting and simulating the accompanying functional module

After final development of each module, the verification of all functions and measurement of all parameters, the modules are integrated to one instrument or system and an overall final test checks all functions where again data generators or pulse generators can simulate the outside world.

In production a similar picture emerges: a production run for an instrument is started, material ordered from component stock. The individual PC-boards are loaded, soldered and pretested. Consequently, this pretest also requires signals to be fed to the PC-edge connector as well as power, which again is a wide application area for pulse and data generators. After pretest of the simple boards the instrument is assembled and tested as an entity.

2.1 Functional and Parametric Testing

Circuit test can be carried out in two fundamentally different ways: functional and parametric.

Functional testing is the logic verification of the DUT's operation: the logic validity of the DUT is verified against a known truthtable or flow chart, with uncritical timing and safe logic levels.

Parametric testing evaluates the DUT's performance under marginal conditions, e.g.: slow or fast pulses, late or early synchronization, duty cycle, logic levels.

The test requirements of these two areas put different demands on data generators, although the basic block diagram of Fig. 8 is still valid for both types. A data generator designed mainly for functional test applications will have fixed output pulse and timing characteristics. As functional testing requires wide and deep data patterns the memory size is generally larger than for data generators in parametric test applications. The characteristic of generators for parametric test is that electrical parameters, like timing edges and logic levels, can be varied to permit worst-case testing and also, for example, to investigate set-up times, hold times and critical levels.

III DUT Stimulation

3.1. Computers and I/O Cards

Now the question may arise "Why not use a computer output card for stimulating circuits?". The computer has a great amount of intelligence for generating data arithmetic or algorithmic (for instance a PRBS-Algorithm can be programmed) or outputting data from memory or any mass storage (paper tape, disc etc). This method is sometimes used when speed and driving requirements arenot critical. The driving capability of an I/O card is typically specified as a fan-out of a number of TTL loads. Output speed will significantly exceed 1 megahertz only if DMA technique (Direct Memory Access) is provided. Today's semiconductor devices and circuits, however, now require faster data rates. Furthermore, circuit stimulation and test need a better DUT interface than that provided (if provided at all) by I/O cards; control over level, timing etc, is needed, and outputs must be protected against shorts to ground and supply.

3.2. Electronic Tools

Lab-built toggle switch boxes present another extreme im comparison to the computer I/O method. Used for setting input lines high or low, the function of a subassembly is verified by comparing its output patterns with the truth table. Unfortunately switch boxes are capable only of static testing. Furthermore, contact bounce may generate an uncontrolled number of clock pulses. Consequently investigations of complex devices over several clock cycles are awkward and time consuming.

3.3. Self-Tailored Equipment

As an alternative, many users build their own data generators. These, however, generally have limited sets of words and are usually restricted in bit rates and output levels, quite apart from the costs of building them and their one time use for a particular task.

3.4. Data Generators

For logic users with a variety of applications, the data generator is convenient for bench and system use in R&D and production. Very likely used by engineers, the data generator's flexibility is invaluable for stimulating a wide range of circuitry. This does not mean that in some cases problems cannot be solved with computer I/O's or switch boxes.

IV Data Generators

4.1. What is a Data Generator?

Before discussing various aspects and applications of data generators, first consider the question "What is a Data Generator?". The answer may look like this: A data generator is an instrument that generates a stream of ones (high level in positive logic) or zeros (low level). The operator has complete control over bit position, bit/word frequency, amplitude etc. (A word being a combination of any number of serial or parallel bits.)

4.2 Data, Word ... etc.

In literature and in conversation many words are used to name the logical output sequences: pattern, word, data, data pattern, (data-) vector. They are all used to denote the same: a combination of logic ones and logic zeros either serial or parallel or as a whole block.

4.3. Generator Families

Some families of data generators are presented so that their use in some applications following later can be more easily understood.

4.3.1. Time Linear versus Status Linear

4.3.1.1. Time Linear

Most data generators output their data as shown in Fig. 8 and Fig. 9, a clock oscillator determines the preset or programmed interval time between consecutive readouts. This is called a Time Linear Data Generation because all clock periods have the same length. In other words, the clock provides a constant time axis. Pulse durations in NRZ format of more than one period are generated by storing a one in consecutive addresses (see Fig. 9, periods 2, 3, 4).

4.3.1.2. Status Linear

In contrast to that, the Status Linear Technique defines the time between transitions as shown in Fig. 6.

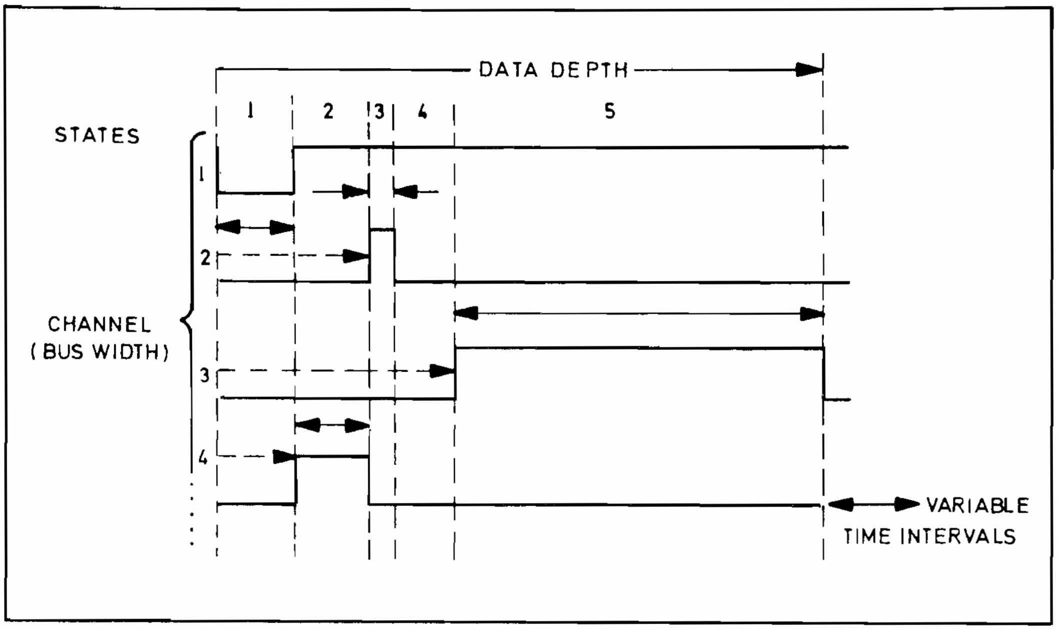


Fig. 6: 'Status Linear' Data Generation

An occurance-related structure is thus established which can give adequate resolution to short events whilst saving memory space during longer events. Furthermore the need to program consecutive ones (or zeros) where no transitions occur can be avoided. This technique is used for multichannel logic activities of extreme duty cycles, for instance in the development of electronics for machine control.

4.3.2 Parallel (Multichannel) Sources versus Serial (Single Channel) Sources

4.3.2.1 Serial Output Generators: Data Generation and Application Area

To satisfy very different application areas, two different types of data generators are found. Test equipment requirements for communication, fiber optic, and other serial data transmission require data generators with a single output. As these data generators generally simulate a data source feeding into a transmission channel or data link, they have to meet a variety of data transmission protocols and standardized levels.

Two ways of generating serial data are found: first a RAM type where the user has full control over each single bit and the memory part for readout. Second there is a shift register type where a data sequence is generated during output time. This type of data sequence is also known as PRBS (Pseudo Random Binary Sequence). The method is shown in Fig. 7.

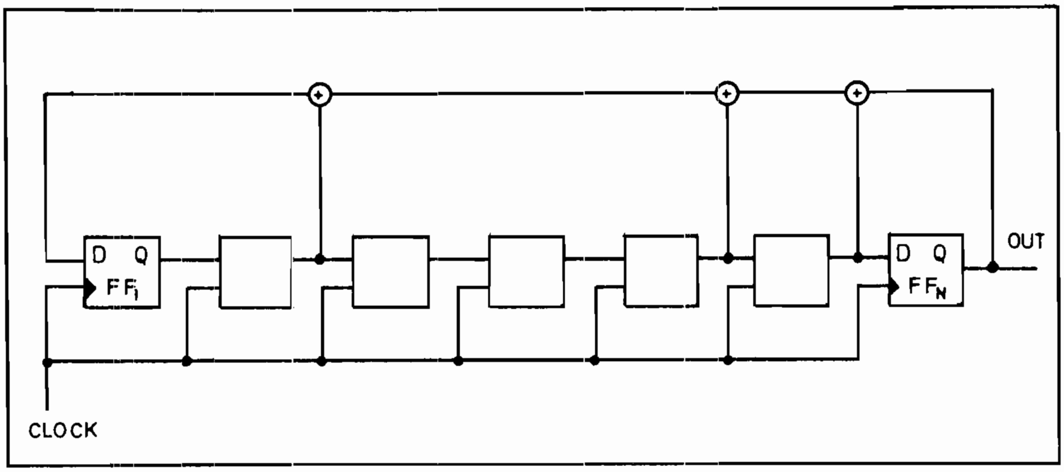


Fig. 7: PRBS Generator. Feedback taps around a shift register generate long pseudo random binary sequence. Length is calculated as 2^n-1 (n = number of FF).

Sometimes RAM and PRBS types are combined in order to generate preamble-data-postamble protocols. The data part - whose actual content is immaterial for DUT purposes - is generated by the PRBS. All combinations of n bits are generated except n consecutive zeros.

4.3.2.2. Parallel Output Generators: Data Generation and Application Area

Parallel data generators serve as data source for data bus application, stimulation of parallel input digital circuits etc. Data generation in nearly all cases is done by readout of previously stored data out of a read-write random access memory as described in the next Chapter. Data generators with several parallel outputs are also used in uni- or bidirectional data transmission if they are tailored to that purpose. Protocols like 3-wire handshake on an 8 bit data bus (GPIB, hp-IB, IEC-Bus, IEEE 488-1975) or 2-wire handshake on an 8 or 16 bit data bus of computer I/O are performed. Parallel output data generators for digital applications are generally most versatile: output levels are settable over a wide range, frequency and other timing parameters are adjustable, bits and sequence length are programmable as needed. Thus the needs of nearly all applications can be met directly. This type of data generator may also be found in very high speed communication applications including fiber optics where the high speed data stream is generated from parallel inputs by time division multiplexing. In serving the needs of most applications, the parallel data generator has a large market in the growing field of complex digital development.

4.3.3. Details on Data Generators

Figure 8 illustrates a typical data generator block diagram:

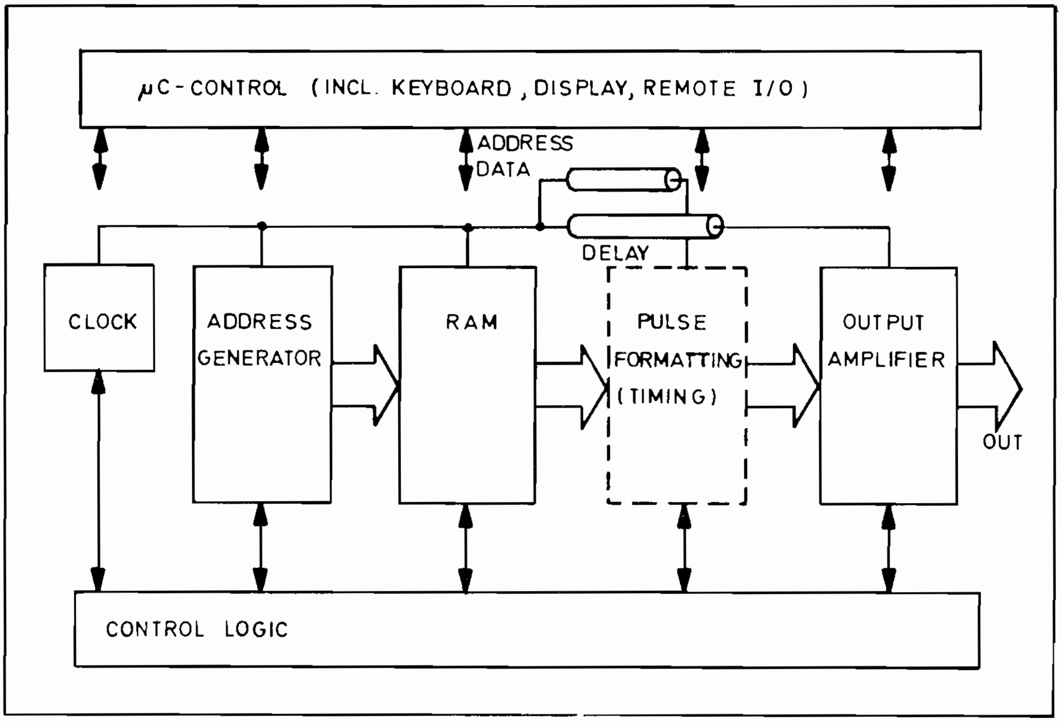

μC-CONTROL (INCL. KEYBOARD, DISPLAY, REMOTE I/O)
ADDRESS
DATA
DELAY
CLOCK
ADDRESS GENERATOR
RAM
PULSE FORMATTING (TIMING)
OUTPUT AMPLIFIER
OUT
CONTROL LOGIC

Fig. 8: Block diagram of a data generator

The heart of a data generator is a read-write random access memory (RAM). The semiconductor memory is chosen from whichever technology provides the requirements of speed, magnitude, non-volatility etc. Apart from some exceptions the organisation of the RAM reflects the data generating capabilities in terms of number of words and bits per word. Readout of the data content is done by an ascending address generator (counter) being (re-) started and stopped at predetermined values (First Address, Last Address). A clock generator gives the interval between consecutive readouts. In most cases the output of the RAM is latched for good skew, amplified to the desired high- and low levels and brought to the output connectors. In some cases the pulses are formatted from the NRZ (NON-Return to Zero) to the RZ (Return to Zero) format by adding width and/or delay as shown in Fig. 9.

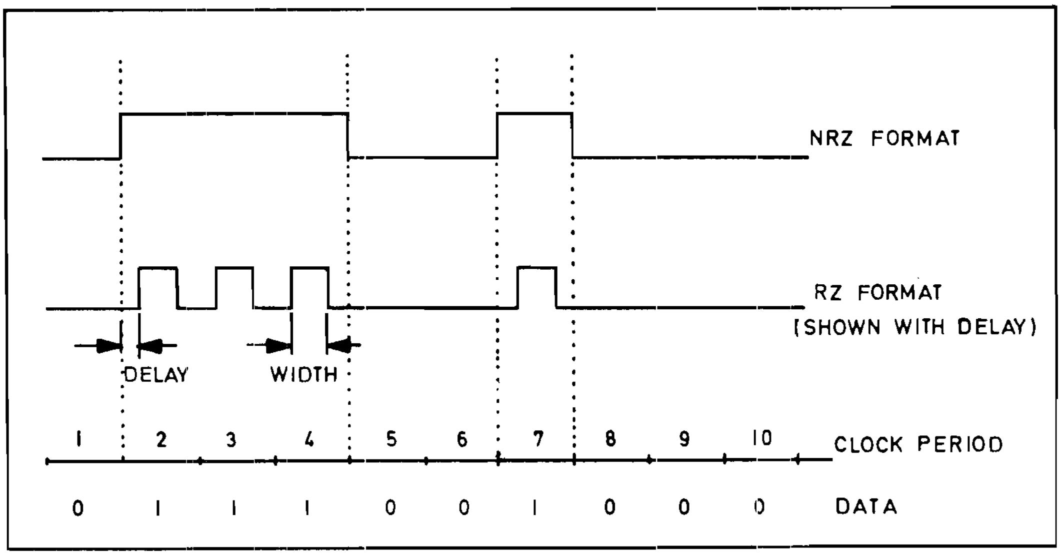


Fig. 9: NRZ and RZ Format. The same data is shown in two formats. In NRZ format, if the output is a 1, the output stays high until a 0 is encountered. In RZ format, if the output is a 1, the high level returns to 0 within the same period. Delay and Width characterize a pulse in RZ format.

Whereas a microcomputer does all sophisticated controls, programming, calculations as well as human and remote interfacing to the user, some function and control switching requires high speed logic.

V Test Pattern Generation

Except for PRBS generators in which the sequence is generated arithmetically during readout the user has to initialize the data generator by storing bits in the memory as required by the test task.

The easiest way to get words, patterns or whatever would be in an application where a DUT is to be tested to parametric worst case margins by replacing the existing data source by a data generator. The user reads the data by means of an oscilloscope or other data collector (logic analyzers etc) and keys them into the data generator. Alternatively he can do it automatically by use of remote I/O's in a calculator/computer controlled environment. Thus, when he outputs from the generator, he has the same data but he now also has full control over each bit and over all parameters.

Data-gathering differs slightly depending on the type of data source. For example, verifying the logical function of simple SSI/MSI IC's merely requires ones and zeroes of the known truth-table to be transferred to the data generator memory.

In the above case, the user did not have to concern himself with data generation because his source is error-free and the only mistake he can make is a wrong reading error resulting in a one converted to a zero or vice versa or skipping a number of words.

Other laboratory and production experience shows that, in most cases, the user has to work out the data by himself. While flow charts or circuit diagrams are a basis for generating the data, the users know-how about a circuit and its inspiration are required. The intelligence of a microcomputer within a data generator can only help the user in keying in data conveniently, editing it in terms of inserting/deleting lines, changing bits and giving him a quick overview. A check out of the data on the DUT for debugging then ends the data generation. Non-volatile RAM's or magnetic mass storage keeps data alive for multiple use.

5.1 Circuit Simulation

Computer aided circuit simulation and data generation may provide solutions which relieve the user. However, results up to now are not very encouraging because only circuits with few nodes can be simulated and data generation is very limited. Simulation of medium size circuits require the largest and fastest computers and very sophisticated software. But development is going on and the future will tell more.

VI Applications

6.1. Application Examples of Serial Data Generators

6.1.1. Serial Interface Design

Serial buses are often used to exchange information in digital systems. In aircraft, copper or fiber optic cables link radar and navigation systems with central computers and cockpit displays. In data processing and transmission systems a central computer is often linked to its peripherals via serial buses. The interface between a disc and its controller is also serial. Design of such serial systems, especially their interfaces can be simplified a lot if a data source is available to the designer which gives him complete control over the data generation. He can then get his interface operable and test its pattern sensitivity. When functional operation of the device has been established, bit rate may be varied to verify operation over a band of frequencies and establish worst case limits. Signal voltage can also be varied to determine amplitude sensitivity.

SDLC Data Transmission

Data interchanges between computers and peripherals are governed by various serial protocols. The SDLC protocol (for Synchronous Data Link Control) is typical of the many protocols currently in use. The SDLC message begins with a 24 bit preamble which includes synchronizing, address, and control bits. The message continues with a free-form information field of user selectable length and ends with a 24 bit postamble. The postamble contains error detecting bits and an end-of-message delineator.

Fig. 10 illustrates the combination of the three fields in the transmitted signal.

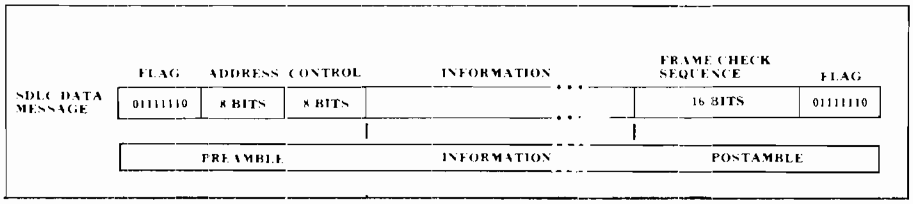


Fig. 10: SDLC Data Transmission

A generator designed for the serial data needed to test transmission lines and systems, sending and receiving hardware of serial interfaces, and telecommunication equipment is the hp 8018A Serial Data Generator.

It meets all the requirements for serial stimulus up to 50 Mbit/s. 2048 individually programmable memory bits combined with Pseudo Random Binary Sequences (PRBS) allows generation of the most complex data patterns to be generated. DC to 50 MHz clocking rates and a well defined 15 V output pulse provide speed and signal levels to work directly with logic families from ECL to CMOS.

Adding to the versatility of this generator are variable word and pattern lengths and dual data output channels. PRBS and programmable data words can be combined into a single data stream, perfect for simulating preamble-data-postamble patterns. A full complement of cycling modes and trigger signals provides easy synchronization to the circuit under test. For production and other systems environments, an optional HP-IB programming interface provides remote control of data generating functions.

Coming back to the SDLC Transmission protocol with its 24 bit preamble, free-form field of user selectable length and 24 bit postamble the 8018a's MIXED mode neatly simulates this message. Two 24 bit words are programmed and data is entered as preamble and postamble. In MIXED mode, a PRBS sequence is automatically inserted between the two programmed words and represents the information field as shown in Fig. 11. The pseudo random signal is a good simulation of actual traffic that the network is likely to encounter and helps to isolate pattern sensitive errors.

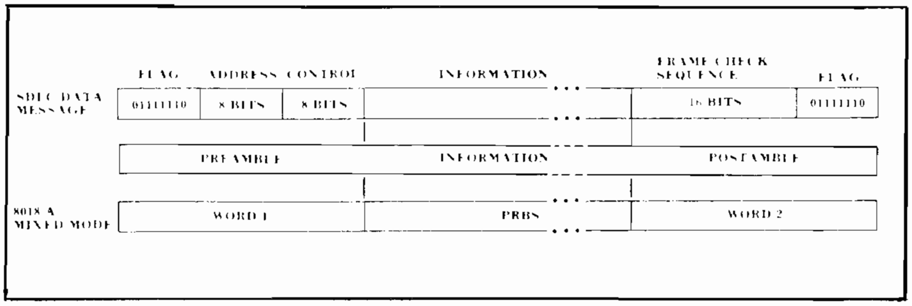


Fig. 11: Generation of SDLC Protocol by hp 8018A SERIAL DATA GENERATOR in MIXED MODE: generation of two 24 bit Data Words with a PRBS Sequence inserted.

The pseudo random signal also contains long, predictable strings of consecutive logic ones and thus can be used to check for proper zero insertion and deletion as prescribed by the SDLC protocol. Simply by varying bits in the preamble and postamble, data can be routed to other address points in the network, and system errors can be simulated.

6.1.2. Telecommunications

In PCM (pulse code modulation) telephone networks and electronic switching systems, voice, dialing, billing and related information are encoded as blocks of serial data. A typical PCM testing example might look like the following.

In a 30 channel system, a basic data frame consists of 32 eight bit words. Words 1 and 17 are synchronizing characters. The remaining 30 words are eight bit, analog-to-digital converted samples of the 30 voice channels. The frame is transmitted at a 2 MHz bit rate.

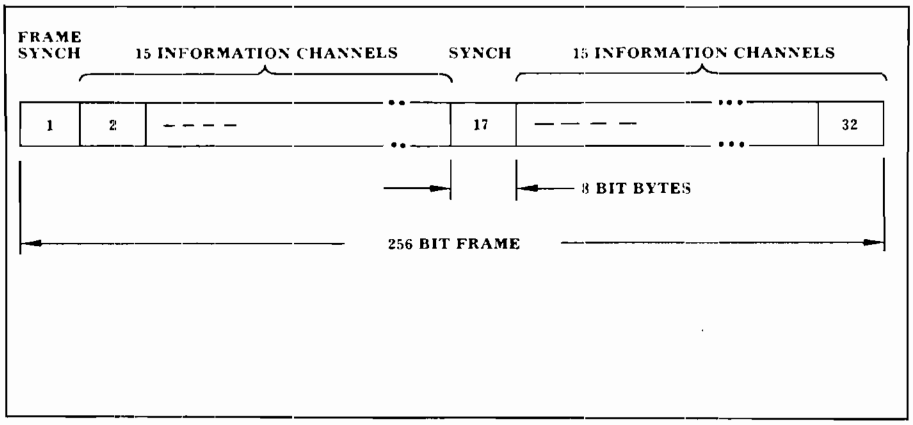


Fig. 12: PCM Data Frame

The 8018A's capability of generating large amounts of serial data makes it a useful laboratory test tool in the design of these systems.

Using DATA mode, up to 8 such frames can be stored and output by the 8018A. Incorrect synchronizing characters may be entered in some of the frames to test susceptibility of the system to this error. The words representing voice information can be randomly programmed to simulate actual voice and data traffic in the network. If a crystal related bit rate is required, it can be entered via the clock input.

6.1.3. Coding

Specific testing applications very often require data formats other than RZ and NRZ. In most applications the 8018A's very large reserves of both speed and memory can be used to directly simulate these required formats. A typical example is described below.

Manchester code always has a transition in the middle of the bit period. A logic zero starts low and ends high, and a logic one is the reverse, as shown in Fig. 13:

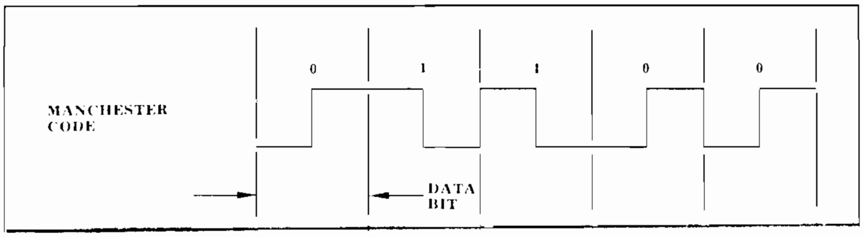


Fig. 13: Manchester Code. Timing graph of the pattern 01100.

Simulating one Manchester bit with two 8018A bits, data entry is as follows. A Manchester logic zero is programmed as 01 and a logic one as 10. The 8018A's clock is set to twice the frequency of the desired bit stream. The DATA B output, when programmed to an alternating ones and zeros pattern, simulates the Manchester clock signal.

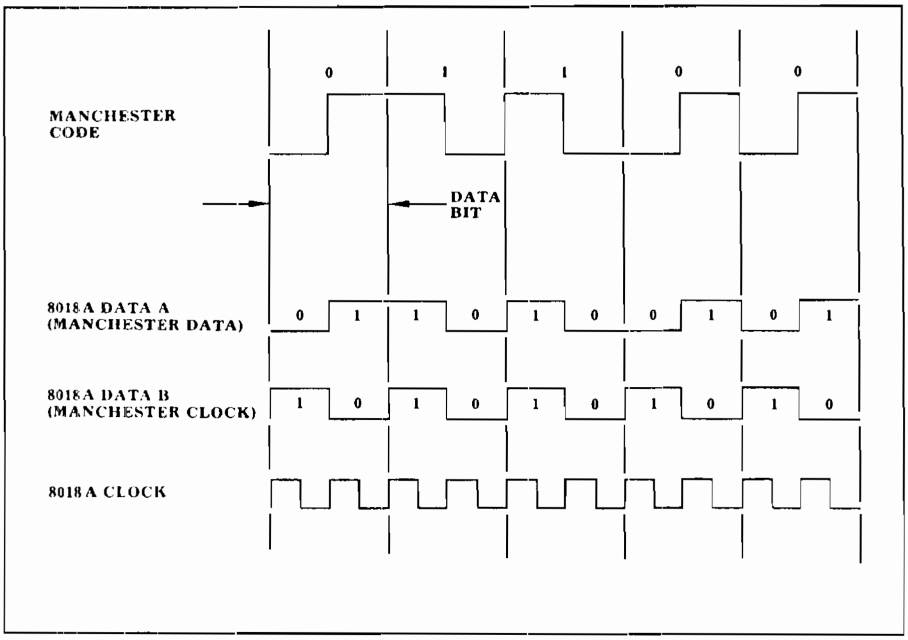


Fig. 14: Manchester Code Simulation by use of 2 channels A and B of 8018A SERIAL DATA GENERATOR and proper programming

Using this technique, Manchester coded data streams can be generated at rates up to 25 MHz and with sequence lengths to 512 bits. The same code generating technique applies to a wide variety of other coding formats. In all cases, several 8018A bits simulate one bit of the required data format, and the 8018A's second channel simulates the clock. The 8018A's large reserves of speed and memory are responsible for the viability of this valuable technique.

6.2. Functional Testing and Parametric Evaluation of Components: A high speed RAM Test

Often users of monolithic and hybrid IC's have to test these components functionally and parametrically. This requirement can arise either in:

R&D - in order to investigate a component for a special use

materials engineering department - where component evaluation tests are to be made, or

receiving department - where an incoming inspection by full screening or by sampling must be made, or where a component must be selected to a tighter parameter but the quantities don't justify an IC test system.

Bench type measurement set-ups assembled for a specific purpose can provide the customer with a low cost, quick solution because of the use of general purpose instruments which can be used elsewhere after completion of a particular task.

The following application deals with a typical LSI device, the Random Access Memory (RAM). Employed where data needs to be stored, read out or overwritten, data is properly accessed or entered only when appropriate delays exist between its data, address, and write enable signals. More specifically defined in Figure 15 as set-up time, hold time, and access time (propagation delay), these delays are significant for evaluating typical RAM performance, in that their specified minimum values provide the check-list for dynamic testing.

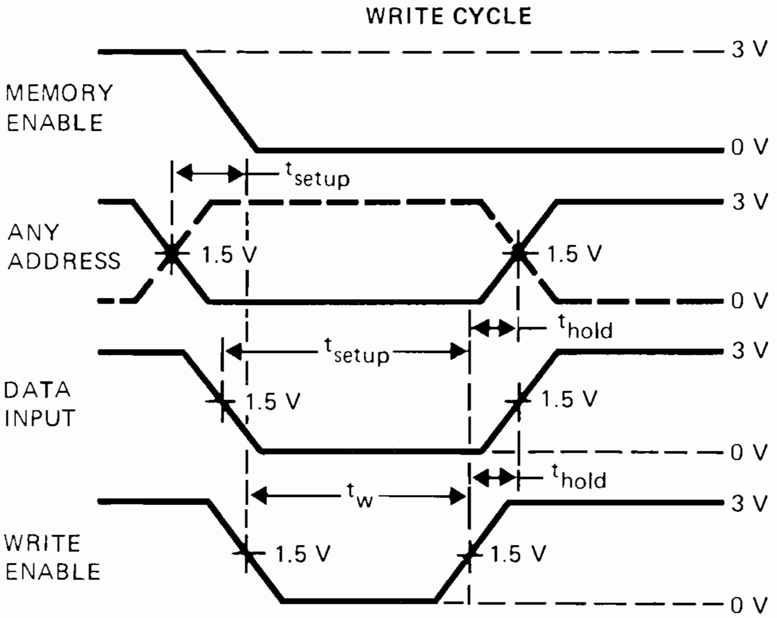


Fig. 15a: RAM Timing Parameters: Write Cycle

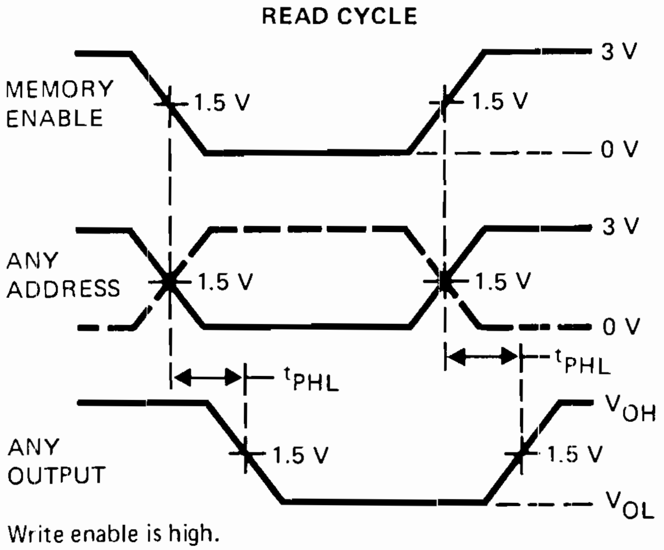


Fig. 15b: RAM Timing Parameters: Read Cycle

Whether a RAM needs to be tested functionally or parametrically, the 8016A Word Generator is capable of delivering the required signal configuration for precise evaluation.

The 8016A Word Generator has eight data output channels, each supplying an independently selectable 32-bit word at clock rates from dc to 50 MHz. Where longer data streams are called for, a serializer cascades the 32-bit outputs up to a maximum 256-bit sequence length. A ninth output is designated STROBE. In addition to the STROBE, the 8016A generates three other auxiliary outputs. The CLOCK output provides pulses at the selected repetition rate, and the FIRST BIT/LAST BIT outputs provide framing pulses as trigger aids for viewing the data sequence. A significant feature of the 8016A is its delay capability, where the clock can be delayed with respect to the data channels, and data channels can be delayed with respect to each other. Output levels can be selected either TTL or ECL compatible.

By selective assignment of 8016A outputs to RAM inputs, the 8016A memory can be programmed to generate the necessary data, address and control signals for functional testing; variable delay and variable RZ-width then supply the dynamic test capability. Where the storage capacity of the RAM precludes a thorough functional test with the 8016A's 256-bit memory, the word generator can still make a valuable contribution in dynamic analysis. Two 8016A's in parallel may even be used to fulfill extra channel needs.

A typical test set-up with a 64-bit RAM as DUT is shown in the following Figure 16.

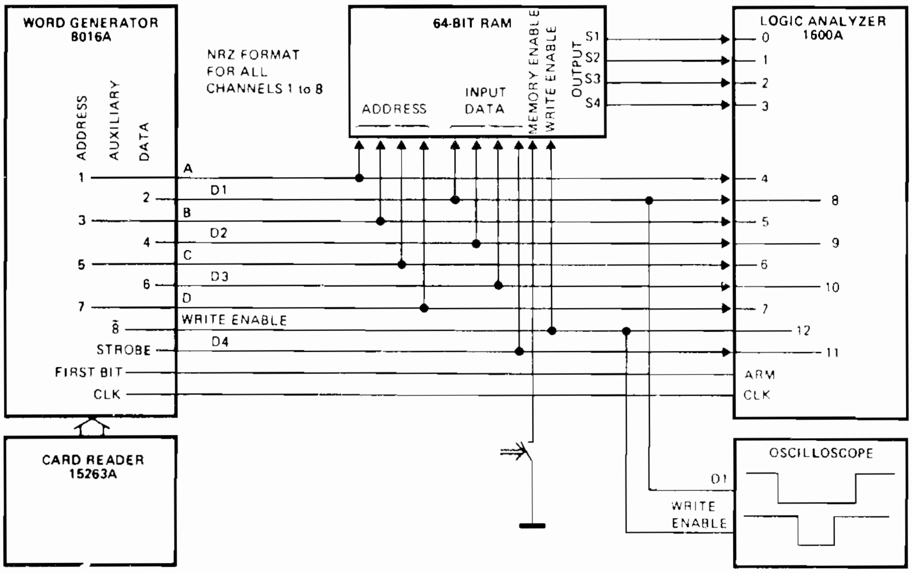


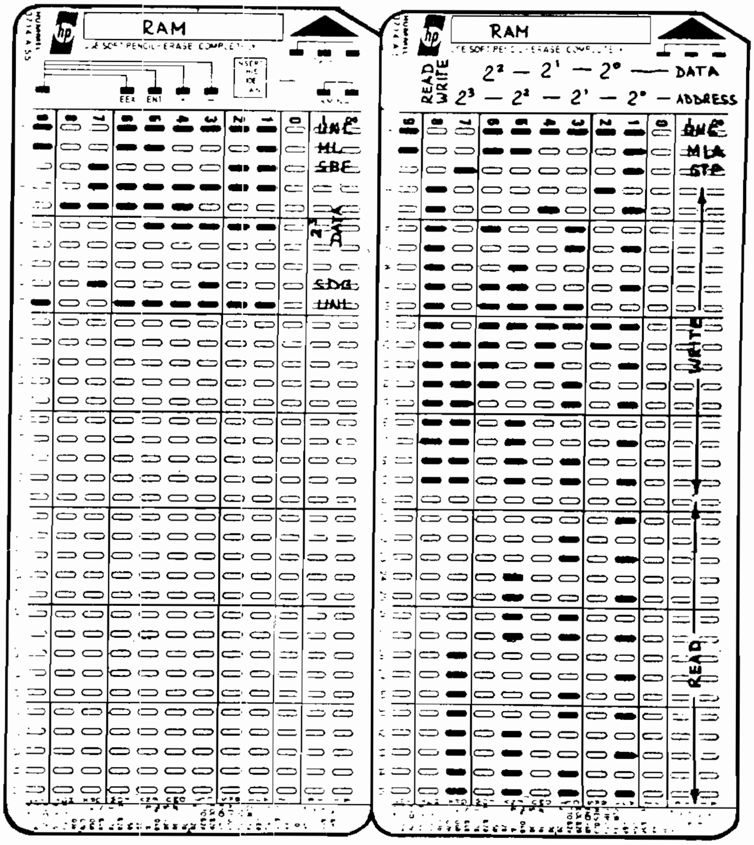
Fig. 16: RAM Test Set-Up

Here again a Word Generator assumes the role of signal source, programmed via a Card Reader; RAM response is verified using a Logic State Analyzer and an oscilloscope. Particularly important to this test set-up is the connective sequence from the 8016A to the RAM. Non-delayable channels 1, 3, 5 and 7 are used to supply RAM address signals. Delayable channels 2, 4, 6 and STROBE supply RAM data signals. The WRITE ENABLE input requires a low-going pulse, and this command is then delivered by the complement output of channel 8.

An extract from the RAM truth table (Figure 17(a)) is transferred onto the marked card as shown in Figure 17(b). Two cards are required here to completely load the 8016A's data memory and strobe channel.

(CLK)	MEMORY ENABLE	WRITE ENABLE W/R	ADDRESS				DATA				DATA			
			2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
1	1									1				
2	1					1			1					
3	1				1			1						
4	1				1	1	1							
5	1			1			1							
6	1			1		1	1	1						
7	1			1	1		1	1	1					
8	1			1	1	1	1	1	1	1				
9	1		1				1	1	1	1				
10	1		1			1	1	1	1					
11	1		1		1		1	1						
12	1		1		1	1	1							
13	1		1	1			1							
14	1		1	1		1		1						
15	1		1	1	1				1					
16	1		1	1	1	1				1				
17	1	1									1	1	1	
18	1	1				1					1	1		1
19	1	1			1						1		1	1
20	1	1			1	1						1	1	1
21	1	1		1								1	1	1
22	1	1		1		1							1	1
23	1	1		1	1									1
24	1	1		1	1	1								
25	1	1	1											
26	1	1	1			1								1
27	1	1	1		1								1	1
28	1	1	1		1	1						1	1	1
29	1	1	1	1								1	1	1
30	1	1	1	1	1	1					1		1	1
31	1	1	1	1	1						1	1		1
32	1	1	1	1	1	1					1	1	1	

a.


RAM
RAM
READ WRITE
2^2 — 2^1 — 2^0 — DATA
2^3 — 2^2 — 2^1 — 2^0 — ADDRESS
ONE
ML
SBE
SDG
WRITE
READ

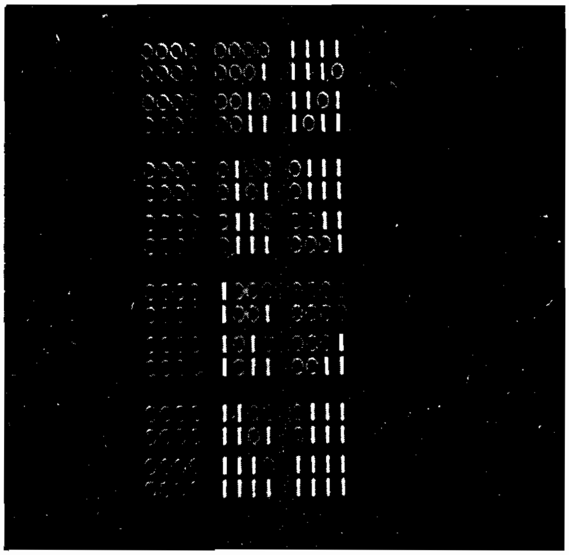
b.

Fig. 17: (a) RAM Truth-Table
(b) Marked Card Program

Note: The GTP (Go To Parallel) for parallel-loading of the data channels; the SBF (Strobe Byte Follows) command for loading the strobe channel; and the SDG (Start Data Generation) command for starting 8016A memory read-out.

After programming the 8016A memory, time positioning of the channel outputs must be correctly adjusted before proceeding with any functional test of the RAM. To simplify this, the 8016A is set to generate RAM data/address information in NRZ format and the WRITE ENABLE command in RZ format, as indicated in Figure 15. By operating the channel 8 delay control, the appropriate set-up times for normal RAM operation can then be set.

With the 8016A suitably adjusted, functional testing of the RAM proceeds as indicated in the Figure 17(a) truth-table. The first 16 clock markers comprise a write-cycle (WRITE ENABLE command pulsed low), with 4-bit word cells 0 through 15 being sequentially addressed and data entered. During the next 16 clock cycles, WRITE ENABLE is held high, and word cells 0 through 15 are again sequentially addressed. As each cell is addressed, the 4-bit complement of its data content should be read out. (This readout can be true or complement of the stored data. It varies according to the tested RAM.) RAM response is then easily verified by comparing the 1600A Logic State Analyzer display (Figure 17(c)) to the Figure 17 (a) truth-table. Should a worst-case pattern exist for a particular RAM, the 8016A memory can be correspondingly programmed via the marked card reader, and the response of the RAM thoroughly checked out.



```
0000 0000 1111
0000 0001 1110
0000 0010 1101
0000 0011 1011

0000 0100 0111
0000 0101 0111
0000 0110 0011
0000 0111 0001

0000 1000 0000
0000 1001 0000
0000 1010 0001
0000 1011 0011

0000 1100 0111
0000 1101 0111
0000 1110 1111
0000 1111 1111
```

Fig. 17: (c) Logic Analyzer Display of RAM Response

Having verified RAM operation, its parametric limits can be determined by adjusting the appropriate 8016A controls until malfunction is observed on the Logic Analyzer Display. Which controls should be adjusted can be deduced from Figure 15. Address set-up time, for example, is analogous to a delay between 8016A channel 8 and the address channels (1, 3, 5 and 7). By adjusting the channel 8 delay potentiometer until RAM operation fails. The minimum set-up time can be measured on a scope.

Similarly, data hold time can be considered analogous to a delay between channel 8 and one of the data channels (2, 4, 6 and STROBE). Adjust the channel 8 delay potentiometer for RAM failure, and read the minimum value from the scope display. The third important parameter, access time, is easily determined by using the scope to measure the delay between a RAM data output and its corresponding address input from the 8016A. As a concluding example, if RZ format is selected for channel 8, the WRITE ENABLE pulse width can be controlled by the 8016A's RZ width potentiometer. With the RZ-width decreased until the RAM fails to respond correctly, the absolute value measured on the scope display is the actual operating minimum of the WRITE ENABLE command width.

6.3 Functional Testing of Circuit Blocks, Boards and Subassemblies

This task opens a wide variety of applications for data generators. Numerous examples of applications could be given which are either typical for a certain task and found nearly everywhere or which are specific for a single user. Two examples will be presented here showing the application of modern data generators in state-of-the-art circuits.

6.3.1. Data loading into a Scanning Display Hardware during Development Phase

Alpha-numeric or graphic CRT displays are examples of equipment where parallel development of microcomputer and control circuits can cut R&D time drastically, eliminate lengthy reiteration to correct hardware and firmware. A typical case is represented by experience gained on the development of a logic analyzer display hardware.

32 lines of information, each 64 characters long are displayed on a CRT. The 2048 character-locations are represented by RAM addresses, each RAM location having 8 bits for defining the character to be displayed. Six of these bits address a ROM which contains digitized pattern information for 64 different characters; the seventh and eighth bits permit blanking and luminance inversion. The ROM is also addressed by 4 additional bits from an address counter. Thus, just the digitized character information for the current CRT scan is accessed (10 scans equal the height of one character including vertical spacing, see Fig. 18). This information parallel loads a shift register which performs a serial conversion for CRT beam modulation.

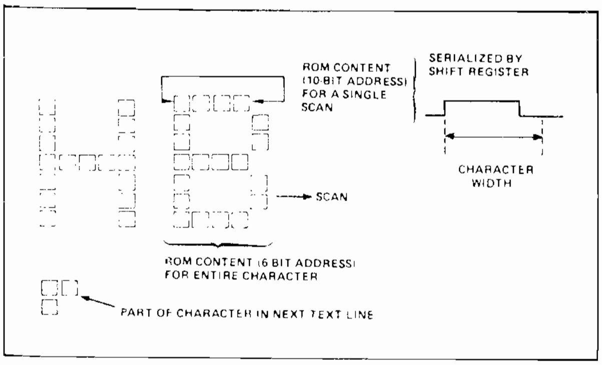

ROM CONTENT (10-BIT ADDRESS) FOR A SINGLE SCAN
SERIALIZED BY SHIFT REGISTER
CHARACTER WIDTH
SCAN
ROM CONTENT (6 BIT ADDRESS) FOR ENTIRE CHARACTER
PART OF CHARACTER IN NEXT TEXT LINE

Fig. 18: Character Synthesis

The address counter (Figure 18) is responsible for synchronizing CRT deflection with RAM readout. In the fully-assembled system, the micro-computer will load the RAM at intervals with new information. When this occurs, RAM read-out is inhibited and each RAM location is addressed in sequence by the micro-computer.

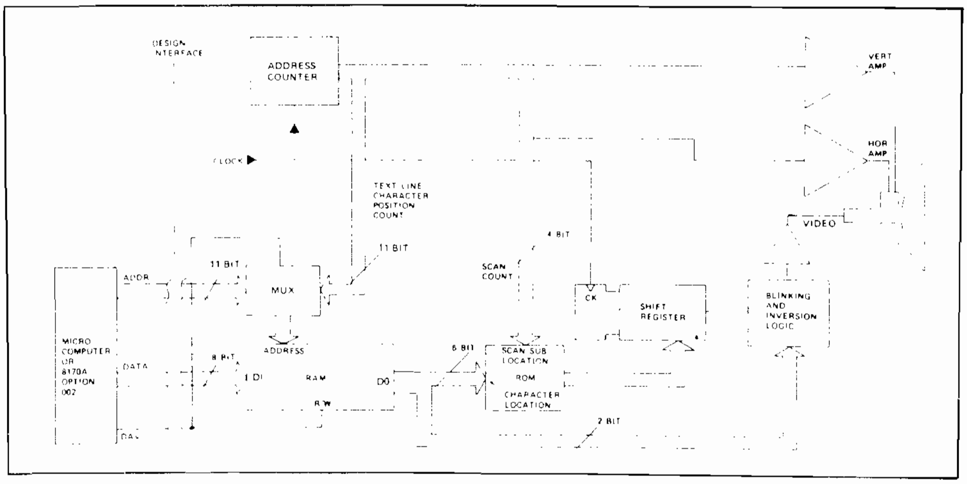


Fig. 19: Display logic

To thoroughly test the display, the RAM requires 256 different bit combinations for each address. While a formidable if not impossible task manually, the hp 8170A Logic Pattern Generator not only generates the necessary data depth for a complete display but can be easily reprogrammed for a different order of symbols. Furthermore, the 8170A also generates address and write enable (DAV) signals. Thus, everything necessary for loading the RAM is available. Moreover, the interface functions in exactly the same manner as that envisaged for the microcomputer.

The Hewlett-Packard Model 8170A is a programmable word generator designed for reliable functional testing of multi-channel hardware.

It generates parallel 8 bit or 16 bit patterns at a memory depth of 1024 or 512 words, optionally extendable to four times that capacity. The clock rate of up to 2 MHz makes the 8170A suitable for real time bus stimulation. Its output levels are well-defined for dependable stimulation of TTL or CMOS circuitry. Output pods are specially designed for efficient bus connection to the test device, making testing faster and easier.

A logic device or subassembly might be part of a complete digital system, for example. During all development stages it serves as an easy-to-use and reliable stimulus that supplies logic patterns for thorough functional check-out of this multi-channel hardware.

In the event of display malfunction, the 8170A data pattern loaded into the RAM provides an ideal basis for logic analyzer measurements at the RAM, ROM and shift register outputs.

Additional advantages using the 8170A are the automatic coding and data programmability. Automatic coding allows data and addresses to be entered directly in the same codes to be used by the microcomputer. Data programmability allows every possible bit pattern in every single RAM location to be checked out. In addition to easy HP-IB programming, a choice of several fixed data patterns are available for quick verification, and manual programming is useful for close examination of suspect areas.

This test method is generally applicable to any output device where parallel data is handled. In many cases, as with the alpha-numeric/graphic CRT, the read-out device itself will be serial in nature but with built-in parallel/serial conversion. Thus teleprinters with RAM backing (text storage) and line printer output systems as well as logic analyzer displays can be rapidly and thoroughly tested using the 8170A as bus stimulus.

6.3.2. PROM Elimination speeds development of heart rate correlator

A new fetal heart rate monitor uses a correlator to extract instantaneous heart rate accurately from complex signals containing large maternal heart beat and noise components. The complex signal is first digitized, then stored in a RAM. From here the data is accessed for correlation. To meet near-realtime requirements, a hard-wired controller is used to supervise RAM and correlator.

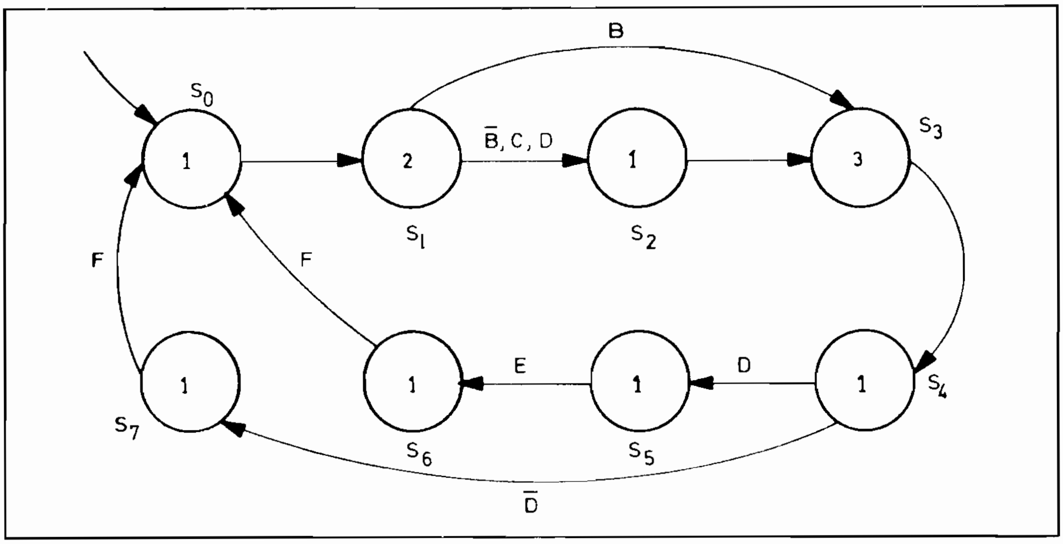


Fig. 20: States. This state diagram illustrates an algorithm that sequences through eight states. The five control variables (B, C, D, E, and F and their complements) instruct the controller where and when to branch. The numbers in the circles indicate the number of micro commands generated in each state.(see also → 5.)

Normally, the design cycle of the controller would have included the use of a PROM for storing the control algorithm so that hardware development on controller and correlator could go ahead. With design maturity, the PROM would provide the data for the ROM used in production. However, the complexity of reprogramming PROMs, and the uncertainties after several erasures, could have led to delays in the main project.

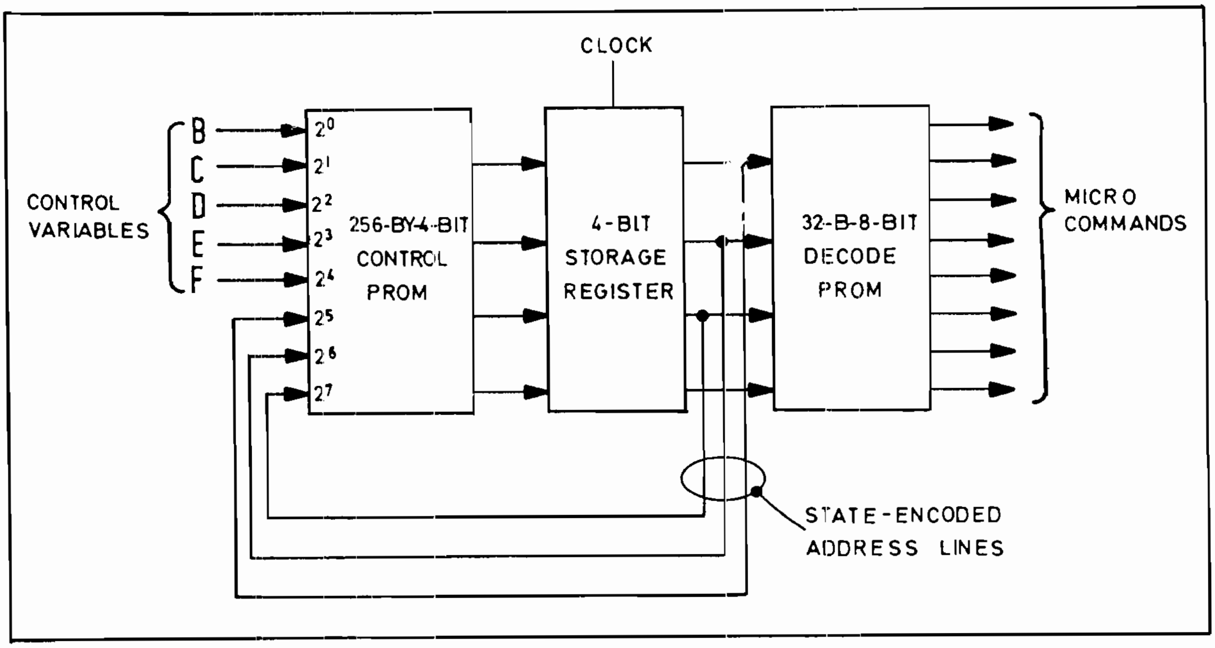


Fig. 21: With a PROM microcontroller, any complex algorithm can be realized with two programmable read-only memories and a storage register. (see also - 5.)

To eliminate these problems, an alternative to the PROM was sought. As the main factor in streamlining this activity would be convenient, fast programming, the 8170A Logic Pattern Generator was chosen. Its external address mode provides PROM-like capability with data rates up to 2 Mbit/s. This is adequate for the functional testing of synchronous devices - even if this means slowing the system clock. Manual, HP-IB or RS232C (CCITT V.24) programming gives keyboard data loading in a selectable code and - important for developing algorithm and hardware - unlimited editing capability.

The mature algorithm, secure in the 8170A's non-volatile memory, can be accessed by internal addressing, HP-IB or RS232C. Thus the most suitable data transfer method for the ROM masking process can be chosen, and possible manual errors avoided.

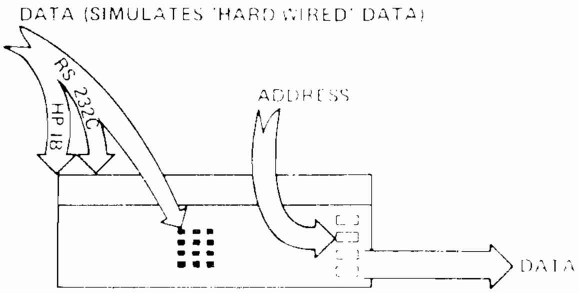
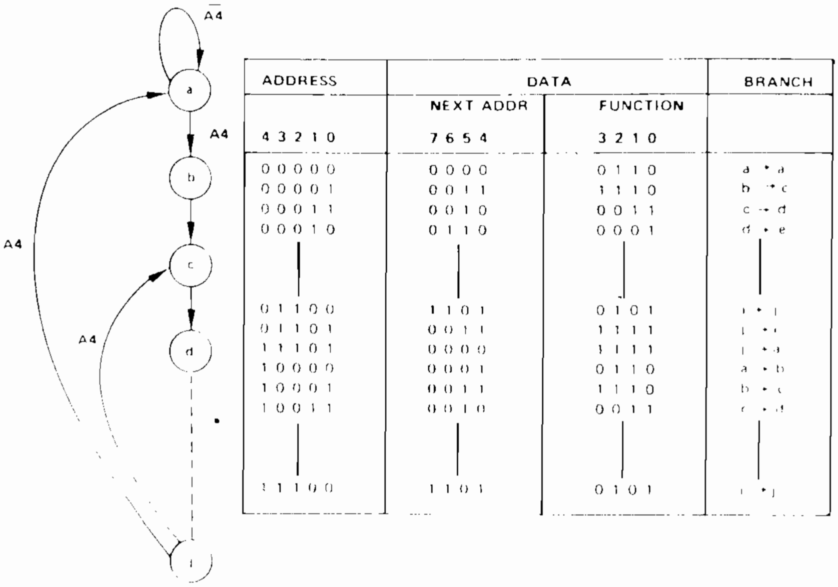


Fig. 22: 8170A operating as a very easily reprogrammable ROM (external address mode).



ADDRESS	DATA		BRANCH
	NEXT ADDR	FUNCTION	
4 3 2 1 0	7 6 5 4	3 2 1 0	
0 0 0 0 0	0 0 0 0	0 1 1 0	a → a
0 0 0 0 1	0 0 1 1	1 1 1 0	b → c
0 0 0 1 1	0 0 1 0	0 0 1 1	c → d
0 0 0 1 0	0 1 1 0	0 0 0 1	d → e
│	│	│	│
0 1 1 0 0	1 1 0 1	0 1 0 1	i → j
0 1 1 0 1	0 0 1 1	1 1 1 1	j → i
1 1 1 0 1	0 0 0 0	1 1 1 1	j → a
1 0 0 0 0	0 0 0 1	0 1 1 0	a → b
1 0 0 0 1	0 0 1 1	1 1 1 0	b → c
1 0 0 1 1	0 0 1 0	0 0 1 1	c → d
│	│	│	│
1 1 1 0 0	1 1 0 1	0 1 0 1	i → j

Fig. 23: State diagram and memory contents

Hard-Wired Controller for Correlator

The RAM and correlator are controlled by a serial algorithm which has 10 different states and three branches. This algorithm is executed by a controller whose major components are a 32-byte ROM and a latch. When the latch is clocked, the ROM is forced to the address which was defined by its data output during the previous clock interval. Thus, by suitable programming the last 4 bits of each byte, any desired progression through the memory can be made. Branching is controlled by Address Bit 4 which depends on the status of the decoder, the RAM address counter, and an external start signal.

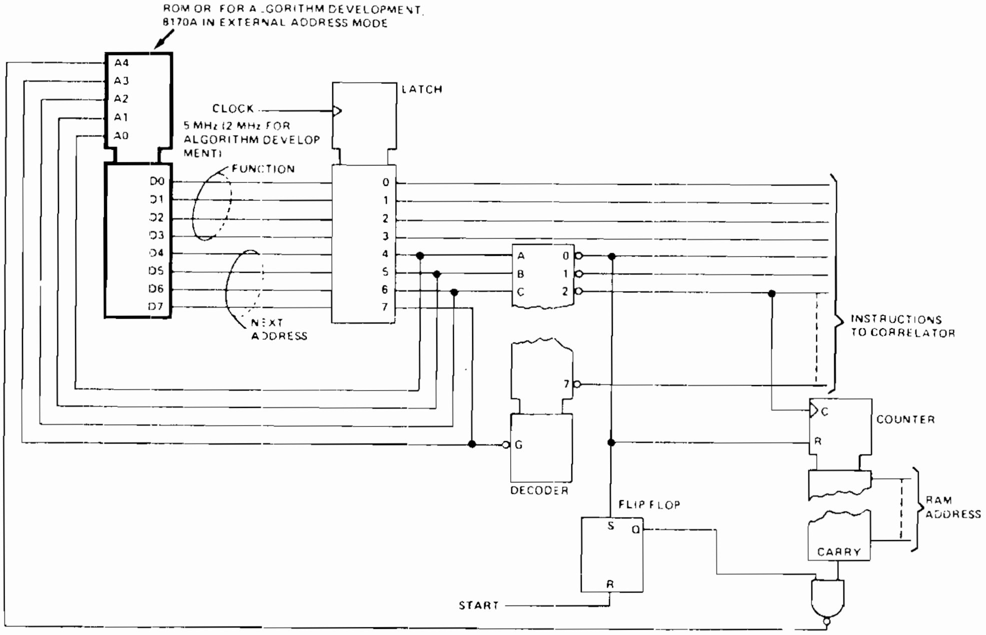

ROM OR FOR ALGORITHM DEVELOPMENT, 8170A IN EXTERNAL ADDRESS MODE
A4
A3
A2
A1
A0
D0
D1
D2
D3
D4
D5
D6
D7
CLOCK
5 MHz (2 MHz FOR ALGORITHM DEVELOPMENT)
FUNCTION
NEXT ADDRESS
LATCH
0
1
2
3
4
5
6
7
A
B
C
0
1
2
7
G
DECODER
INSTRUCTIONS TO CORRELATOR
C
R
COUNTER
RAM ADDRESS
CARRY
FLIP FLOP
S
Q
R
START

Fig. 24: Simplified circuit diagram of the controller

In addition to reducing controller and correlator development time, the use of the 8170A gave complete independence from PROMs, and was a significant contribution to the prompt delivery of the heart rate monitor.

VII Outlook and Trends

Recent developments and trends in data communications, data processing and technology show that the amount of data increases more and more. Thus data handling capabilities at any point - especially here for data stimulating instrumentation - must be extended and more flexible. Data storage capacities must be expanded, and stored data has to be available for processing etc. in less time. Consequently tomorrow's data generators will have larger memories so that more data can be generated in a single block. As the cost per bit of semiconductor memories drops, together with an increase in size and speed, more memory capacity can be easily implemented. This then will make it possible to output much longer data streams on wider busses. 16, 32 or more parallel outputs will no longer be impeded by memory restrictions.

Additionally, any kind of mass storage (cassette, floppy disc, bubble, high density semiconductor memories) will be necessary so that quick access to different data blocks can be provided. Mass memories also allow several users within a department to share one data generator and store individual data.

Larger amounts of data require much more intelligence and data manipulation capabilities. The implementation of microcomputers made it possible to have user friendly displays and keyboards and programmable instruments.

Indeed, a more widespread readiness to accept the microprocessor when first introduced could have led to a greater exploitation of the advantages today. Data manipulation - bit editing, line insertion/deletion, block move/copy, label assignment, coding, etc. - as well as menu concepts for setting up parameters and functions, are features which will make future data generators not only more powerful but easier to operate, parallel to extended hardware features.

Increasing memory capacity is one aspect of meeting tomorrow's requirements. More intelligence in generating data will be a further characteristic of the next generation's data generators. Arithmetic data generation (address sequencing) and real-time algorithmic data generation (data generation with microcontroller by program) are techniques known from IC-test-systems. With them, extremely long data streams can be generated; the memory is more effectively used because one data word needed several times has only to be stored once. These techniques could be adopted for data generators thus providing subroutining, looping, jumping capabilities within data memory. With external qualifier inputs, the data generator can be made to branch, i.e. to react on commands, or data processing results.

The development of technology, especially in the speed of transistors and ICs, makes it possible to increase generator rate. Thus, it will be possible to meet the high speeds required for parallel data transmission on busses as well as for serial communication lines where bit rates exceed 1 GHz. Among the new techniques under discussion for inclusion in high-speed systems are Gallium Arsenide (Ga As) and Josephson Junctions. At such speeds, it is difficult to bring clean signals to the interface port of the DUT.

Interfacing (or Probing) will become a general point of concern because of speed and the increasing number of connections. The near future will show what customer needs must be satisfied and the form the solutions will take.

VIII Literature

1. Arndt Pannach, Wolfgang Kappler

   "A Multichannel Word Generator for Testing Digital Components and Systems"

   hp Journal 8, 1975, pp. 17-24

2. Günter Riebesell, Ulrich Hübner, Bernd Moravek

   "A Digital Pattern Generator for Functional Testing of Bus-Oriented Digital Systems"

   hp Journal 8, 1979, pp. 20-25

3. Application Note 227

   "Word Generator Techniques in Multi-Channel Applications"

   hp-Nr. 5952-9503

4. Applications Information

   a) "No Interface Problems in Parallel Building - Block Development for CRT Symbolic Displays"

   hp-Nr. 5952-9537

   b) "PROM Elimination Speeds Development of Heart Rate Generator"

   hp-Nr. 5952-9538

5. John J. Petrale

   "PROM Controller Makes Fast Work of Serial Jobs"

   Electronics, April 12, 1979, p. 134

6. Trent Cave

   "Compressing Test Patterns to Fit Auto LSI Tests"

   Electronics, October 12, 1978, pp. 136-140

7. Theodore C. Gams

   "Instrumentation For Test & Design of Digital Computers and Data Systems"

   Electronic Instrument Digest, January 1971, pp. 21-26

8. Jerry Heyer
"All word generators are not women"
and
Stephen A. Thompson
"Commercially Available Word Generators"
The Electronic Engineer, January 1970, pp. 42-45
pp. 47-48

additional information

9. Data Sheet "hp 8018A Serial Data Generator"
"Powerful Digital Stimulus For all Your Serial Testing Requirements"
hp-Nr. 5952-9512

10. Data Sheet "hp 8016A Word Generator"
hp-Nr. 5952-6288

11. Data Sheet "hp 8170A Logic Pattern Generator"
hp-Nr. 5952-9518

12. Data Sheet "hp 8080 System"
hp-Nr. 5952-6293

13. Training Manual "Model 8080A: High Frequency Pulse/Word Generator System"
hp-Nr. 5952-6298

14. Christian Hentschel, Günter Riebesell, Joel Zellmer, Volker Ebele
"An Individualized Pulse/Word Generator System for Subnanosecond Testing"
hp Journal 8, 1977, pp. 16-24

HEWLETT
PACKARD