17

Logic Analyzers: What are They, and Where are They Going?

1976 Wescon Professional Program

Los Angeles, Sept. 14-17, 1976

©Copyright 1976 Western Electronic Show and Convention

LOGIC ANALYZERS: WHAT ARE THEY, AND WHERE ARE THEY GOING?

SESSION ORGANIZER AND CHAIRMAN

Stanley Runyon
Electronic Design
Rochelle Park, NJ

17/1 An Advanced Logic State Analyzer for Microprocessors.

Thomas A. Saponas, Jeffrey H. Smith, Hewlett-Packard Company, Colorado Springs, Co.

17/2 Uses and Performance Levels of Current Logic Analyzers and Their Future Trends.

Edward S. Jacklitch, Biomation Corp., Cupertino, Ca.

17/3 The Logic Analyzer--The Compleat System Instrument.

J. Carver Hill, E-H Research Laboratories Inc., Oakland, Ca.

17/4 Pick a Logic Analyzer for Your Job.

Murlan Kaufman, Tektronix Inc., Beaverton, Or.

AN ADVANCED LOGIC STATE ANALYZER FOR MICROPROCESSORS

Thomas A. Saponas
Jeffrey H. Smith
The Hewlett-Packard Company
1900 Garden of the Gods Road
Colorado Springs, Colorado 80907

INTRODUCTION

The wave of new microprocessor applications has brought with it a wave of new measurement problems not easily solvable by traditional time domain analysis. The primary signal path of an 8-bit microprocessor is 24 lines wide compared to 3 lines for a differential amplifier. There is no specific test to be performed on those 24 lines analogous to the rise time measurement, for example, on the amplifier. In fact, the important measurement is the flow of information on those 24 lines. In addition, the status of these lines is only of interest at a specific instant in time related to the system clock.

The logic state analyzer is ideally suited to this type of system. Although the microprocessor buses are only part of a system, they are the best starting point for fault isolation. Once a fault is found at the bus level, either the solution is obvious by the nature of the fault or another form of analysis tool is needed to further narrow down the problem. If it is a timing problem, often resolution of less than 5ns is necessary, so an oscilloscope is used. The logic state analyzer is usually still needed to provide triggering for the scope.

Logic state analyzers have become one of the primary tools used for troubleshooting microprocessors, so a dedicated LSA can certainly augment the useful features of previous LSA's. The Hewlett-Packard 1611A offers just such an LSA, which by virtue of being microprocessor controlled is simple to use.

Figure 1. The Hewlett-Packard 1611A Logic State Analyzer.

TRIGGERING CAPABILITY IS TAILORED FOR MICROPROCESSOR USE

The 32-bit wide TRIGGER of the new logic state analyzer is divided into 3 distinct fields: a 16-bit wide ADDRESS field, an 8-bit DATA BUS field (expandable to 16 bits) and 8 lines of information which are monitored by the separate EXTERNAL probe. Any field may be set to don't care (off), if desired.

Since microprocessor code is usually generated by assemblers which assign addresses using either an octal or hexadecimal number base, the new instrument was given the capability of accepting addresses in either of these formats. DATA BUS trigger words may also be entered in either octal or hexadecimal notation from the keyboard. An EXTERNAL trigger word is entered in binary notation with a don't care (off) option on each bit so that individual inputs may be turned off if the desired trigger word is less than 8 bits wide. Multiple passes through loops or subroutines are examined by entering a TRIGGER OCCURRENCE up to 256.

Windowed triggering permits the 1611A to search for a trigger word in a limited portion of a microprocessor's program execution. Address, Data Bus, and External words may be entered to define a TRIGGER ENABLE event. The arming of the TRIGGER feature is now inhibited until this ENABLE event is encountered. A similar 3-field TRIGGER DISABLE event may be selected to disarm the search for a TRIGGER. This causes the TRIGGER comparator to look for a trigger only after the enable point has been encountered but before the disable point occurs.

```
                ADDRESS   DATA   EXTERNAL
TRIGGER          01E2

ADRS     OPCODE/DATA          EXTERNAL
01E2     CPI  FC              0000 0000
01E4     JZ   01CB            0000 0000
01E7     CPI  C0              0000 0000
01E9     JNC  00B9            0000 0000
01EC     LDA  2C07            0000 0000
2C07      82   READ           0000 0000
01EF     ANI  0F              0000 0000
01F1     LXI  B,0862          0000 0000
01F4     CPI  02              0000 0000
01F6     RET                  0000 0000
37FC      33   READ           0000 0000
37FD      01   READ           0000 0000
0133     JZ   013E            0000 0000
013E     LDA  2C00            0000 0000
2C00      97   READ           0000 0000
0141     LXI  H,3767          0000 0000
```

```
                ADDRESS   DATA   EXTERNAL
ENABLE           0981     11     XXXX0000
TRIGGER   <=    37FF      09     0000XXXX
          >=    37FB
DISABLE          2158     2B     111XX000
TRIGR OCCUR=200   PRE-TRIGR=10

ADRS     OPCODE/DATA          EXTERNAL
077D     INX  B               0000 0000
077E     JMP  076E            0000 0000
076E     LDAX B               0000 0000
0874      FE   READ           0000 0000
076F     CPI  FE              0000 0000
0771     RZ                   0000 0000
37FA      87   READ           0000 0000
37FB      09   READ           0000 0000
0987     XCHG                 0000 0000
0988     JMP  0978            0000 0000
0978     MVI  A,1F            0000 0000
097A     MOV  M,A             0000 0000
34D2      1F   WRITE          0000 0000
097B     INR  L               0000 0000
097C     CMP  M               0000 0000
34D3      1F   READ           0000 0000
```

Figure 2a and 2b. To reduce operator confusion, only the triggering options selected by the user appear in the Trace Specification area of 1611A's display.

DISPLAY TRANSLATES DATA FOR EASE OF INTERPRETATION

Experience has shown that some form of data compaction or interpretation is almost essential in a 32-bit wide logic state analyzer. A display which contains so much data is difficult to read rapidly, especially if the operator must make even a simple number base conversion. The new analyzer alleviates this problem by allowing the operator to convert the displayed address and data bus information to either octal or hexadecimal number bases by means of a panel switch.

```
                ADDRESS   DATA   EXTERNAL
TRIGGER          0130

ADRS     OPCODE/DATA        EXTERNAL
0130      CD    OPCODE      0000 0000
0131      DF    READ        0000 0000
0132      01    READ        0000 0000
37FD      01    WRITE       0000 0000
37FC      33    WRITE       0000 0000
01DF      3A    OPCODE      0000 0000
01E0      C0    READ        0000 0000
01E1      FB    READ        0000 0000
FBC0      3C    READ        0000 0000
01E2      FE    OPCODE      0000 0000
01E3      FC    READ        0000 0000
01E4      CA    OPCODE      0000 0000
01E5      CB    READ        0000 0000
01E6      01    READ        0000 0000
01E7      FE    OPCODE      0000 0000
01E8      C0    READ        0000 0000
```

Figure 3a. Sixteen consecutive states displayed in hexadecimal format.

The sixteen states shown in Figure 3a actually represent a flow of information. An engineer troubleshooting hardware at the gate level will prefer the display of the raw numeric data, but a person who is interested in the overall system operation will prefer to interpret the numeric data as a sequence of instructions. This job, while tedious for a human operator, is a simple conversion from one format to another - a task easily performed by the microprocessor in the 1611A.

```
                ADDRESS   DATA   EXTERNAL
TRIGGER          0130

ADRS     OPCODE/DATA        EXTERNAL
0130     CALL  01DF          0000 0000
37FD      01    WRITE       0000 0000
37FC      33    WRITE       0000 0000
01DF     LDA   FBC0          0000 0000
FBC0      3C    READ        0000 0000
01E2     CPI   FC            0000 0000
01E4     JZ    01CB          0000 0000
01E7     CPI   C0            0000 0000
01E9     JNC   00B9          0000 0000
01EC     LDA   2C07          0000 0000
2C07      82    READ        0000 0000
01EF     ANI   0F            0000 0000
01F1     LXI   B,0862        0000 0000
01F4     CPI   02            0000 0000
01F6     RET                 0000 0000
37FC      33    READ        0000 0000
```

Figure 3b. The data in Figure 3a converted to mnemonic form by use of the ABSOLUTE-MNEMONIC key. Note that more than 16 lines of raw data may be displayed in 16 lines of code since instructions often call for more than one memory fetch.

NEW ANALYZER HAS 4 RUN MODES

Figures 3a and 3b show the result of a TRACE by the 1611A. The machine displays program steps sequentially, starting at the selected Trigger point. It is possible to select a PRE-TRIGGER up to 63 which will cause states prior to the trigger point to be displayed. A DELAY of up to 65.472 can be selected if post-trigger data is of interest.

A TRACE TRIGGERS run will result in the new analyzer storing and displaying only the events which meet the TRIGGER specifications. In addition, two counting modes have been incorporated in the 1611A. TIME INTERVAL gives the time between the ENABLE TRIGGER and DISABLE TRIGGER points with a resolution of 1 μs. COUNT TRIGGERS gives the number of times that the TRIGGER specification is found within this range.

ANALYZER IS RECONFIGURABLE

The convenience of a dedicated probe and mnemonic instruction decoding is made possible only by configuring the instrument to monitor one specific type of microprocessor. Fortunately, all of this specialization is contained within two easily partitioned sections of the instrument: the dedicated microprocessor probe with its associated "front end" circuitry and the program ROM for the instrument's internal microprocessor. These parts, which determine the "personality" of the 1611A, can be removed and exchanged in about 10 minutes. Presently, two personality modules are available, one each for the 6800 and 8080 microprocessors. Additional modules are planned for the near future.

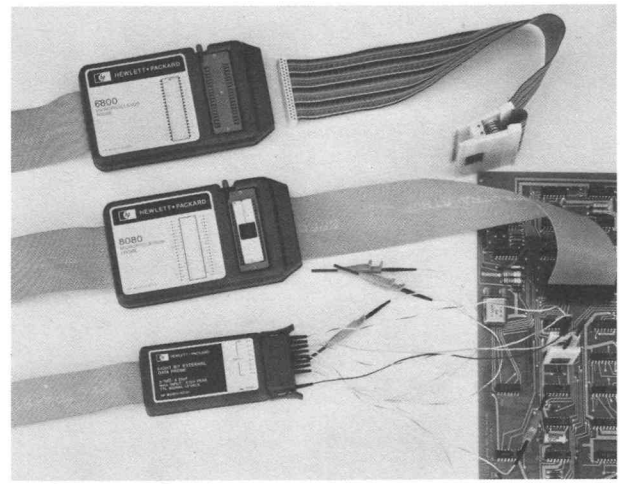


Figure 4. The 1611A connects to the address, data, and clock lines of the test microprocessor by means of a 40-pin dedicated probe. An 8-input "external" probe allows the instrument to monitor additional signals within the system being tested which may not be accessible from the microprocessor address and data buses.

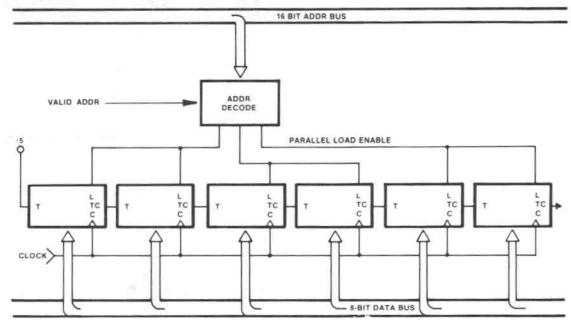


Figure 5. A 24-bit high speed counter.

As an example of the application of this new LSA, consider the troubleshooting of a microprocessor controlled instrument which shows the symptoms of an internal counter being miss-loaded. This counter circuit is shown in block diagram form in Figure 5.

The count is initialized when a microprocessor addresses the appropriate counter and parallel data is loaded from the data bus. The address bus and data bus contents can be monitored at the microprocessor using the 1611A's μPR probe. The contents of the counters are not directly accessible at the microprocessor, so they are monitored at the output pins of the counters with the 1611A External Data probe. The 1611A Trigger is set to $2C02_{16}$ (the loading address of the most significant counter pair) which generates the display shown in Figure 6. The 2 other pairs of counters are examined in a similar manner. From Figure 7, it is obvious that the least significant pair of counters at address $2C00_{16}$ have a problem.

```
                ADDRESS   DATA   EXTERNAL

TRIGGER         2C02

                          PRE-TRIGR=5
-----------------------------------------
ADRS     OPCODE/DATA          EXTERNAL
0123      00   READ          0000 0000
0124     CMA                 0000 0000
0125     STA 2C02            0000 0000
2C02      5F   WRITE         0101 1111
0128     STA 377C            0101 1111
377C      5F   WRITE         0101 1111
012B     MVI A,02            0101 1111
012D     STA 3C60            0101 1111
3C60      02   WRITE         0101 1111
0130     CALL 01DF           0101 1111
37FD      01   WRITE         0101 1111
37FC      33   WRITE         0101 1111
01DF     LDA FBC0            0101 1111
FBC0      7A   READ          0101 1111
01E2     CPI FC              0101 1111
01E4     JZ  01CB            0101 1111
```

Figure 6. A pair of counters being correctly loaded with $5F_{16}$ at address $2C02_{16}$

```
                ADDRESS   DATA   EXTERNAL

TRIGGER         2C00

                          PRE-TRIGR=5
-----------------------------------------
ADRS     OPCODE/DATA          EXTERNAL
010E     DCR A               0000 0001
010F     CMA                 0000 0001
0110     STA 2C00            0000 0001
2C00      97   WRITE         0000 0010
0113     LDA 3766            0000 0010
3766      00   READ          0000 0010
0116     ADI 3F              0000 0010
0118     CMA                 0000 0010
0119     STA 2C01            0000 0010
2C01      C0   WRITE         0000 0011
011C     STA 377D            0000 0011
377D      C0   WRITE         0000 0011
011F     LDA 3765            0000 0011
3765      00   READ          0000 0011
0122     ACI 00              0000 0011
0124     CMA                 0000 0011
```

Figure 7. An unsuccessful attempt to load the 3rd pair of counters with 97_{16} at address $2C00_{16}$.

The counters appear to increment, rather than load new data, suggesting that their LOAD line is never pulled to its active state. This may be verified by using the 1611A in its SINGLE STEP mode of operation to cause the program to single step one instruction at a time through the area of code which loads the counters. It is now possible to monitor the counter LOAD input with a logic probe or a scope.

PERFORMANCE EVALUATION

A problem which often faces real time microprocessor applications is interrupt service time. A microprocessor sometimes finds itself in a position analagous to having someone at the front door and back door as well as having the phone ringing. Even though a system appears to be working, to insure reliability, worst case analysis is necessary to insure that it is possible to answer the doorbell before the phone stops ringing. The execution time for the software modules of interest can be computed by adding up their individual instruction times. If the code in question becomes the least bit long with many nested loops and subroutines, the task becomes so arduous that it is often abandoned.

Instead of analytically determing these times, the 1611A can be used in the TIME INTERVAL mode to actually measure software execution time. This measurement is set up by specifying start and end conditions (ENABLE TRIGGER and DISABLE TRIGGER) which can consist of combinations of address, data and external logic states. For example, to determine the total interrupt service time, one of the external lines is connected to the device causing the interrupt. The start condition for the timing measurement is then set as the occurrence of that external signal, and the end of the measurement is set as the termination address of the specific routine which services this interrupt. In Figure 8 the result of the measurement can be seen as approximately 27 milliseconds.

```
                ADDRESS   DATA   EXTERNAL
ENABLE                         XXXXXXX0
TRIGGER         XXXX

DISABLE         672

-----------------------------------------
ADRS     OPCODE/DATA          EXTERNAL



TIME =          27 038 MICROSECONDS
```

Figure 8. A measurement of the time from an external stimulus to the occurrence of an address.

To examine in detail why this routine takes the amount of time it does, a further analysis can be performed on specific code segments. Perhaps the number of times the BCD add subroutine is called is having a significant effect on the total interrupt service time. If the address of the entry to that subroutine is entered as a Trigger, the COUNT TRIGGERS measurement will give the number of times that this subroutine is called. The result of this measurement (see Figure 9) indicates the add is performed 24 times. By measuring the time it takes for one subroutine execution, it is then possible to compute the percentage of time spent in that subroutine. With this type of analysis it is easy to determine the area of code where optimization is most important.

```
                ADDRESS   DATA   EXTERNAL
ENABLE                           XXXXXXX0
TRIGGER          3755
DISABLE          672
-----------------------------------------
ADRS     OPCODE/DATA          EXTERNAL


COUNT =                24 EVENTS
```

Figure 9. The number of occurrences of the TRIGGER condition are counted between the external event and the access to an address.

WOM ANALYSIS

The write only memory, a fictitious device, which has gained much notoriety in recent years in fact has application in microprocessor systems. WOM's are not actually used, but it is not unusual for a RAM to be connected as a WOM with respect to the processor bus and used as a control memory elsewhere in the system. Since the memory is write only, its contents cannot be listed by the processor or any other instrument on the bus (see Figure 10).

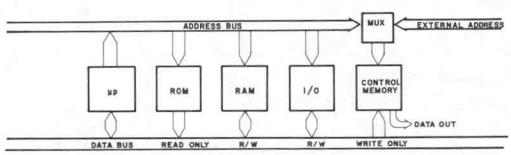


Figure 10. Microprocessor system used in loading a control memory.

```
                ADDRESS   DATA   EXTERNAL

TRIGGER <= 37100
        >= 37000

TRIGGER STORE_____________________
ADRS     OPCODE/DATA          EXTERNAL
037077    321 WRITE          00 000 000
037076    323 WRITE          00 000 000
037075    323 WRITE          00 000 000
037074    323 WRITE          00 000 000
037073    323 WRITE          00 000 000
037072    323 WRITE          00 000 000
037071    323 WRITE          00 000 000
037070    323 WRITE          00 000 000
037067    363 WRITE          00 000 000
037066    363 WRITE          00 000 000
037065    363 WRITE          00 000 000
037064    363 WRITE          00 000 000
037063    363 WRITE          00 000 000
037062    363 WRITE          00 000 000
037061    363 WRITE          00 000 000
037060    163 WRITE          00 000 000
```

Figure 11. The 1611A provides for specification of the TRIGGER ADDRESS as a range of addresses. When this feature is combined with the TRACE TRIGGERS mode a listing of only memory accesses in a specific area is made.

To verify if a problem associated with such a control memory is caused by software, it is necessary to determine what is loaded into memory. Since the memory cannot be dumped easily, the contents must be determined by recording what has been stored. Each stored byte might be the end result of many thousand instructions, so a simple trace through the loading software is not reasonable. The TRACE TRIGGERS mode of the 1611A solves this problem by recording and displaying only the actual writes to the control memory. This measurement is shown in Figure 11.

```
                ADDRESS   DATA   EXTERNAL

TRIGGER <= 37100
        >= 37000

TRIGR OCCUR=48    PRE-TRIGR=20
-----------------------------------------
ADRS     OPCODE/DATA          EXTERNAL
003260   MOV  D,A           00 000 000
003261   MOV  A,L           00 000 000
003262   ADI  005           00 000 000
003264   CPI  122           00 000 000
003266   MOV  L,A           00 000 000
003267   JC   003216        00 000 000
003272   MVI  A,057         00 000 000
003274   CMP  E             00 000 000
003275   MOV  A,D           00 000 000
003276   JNC  003323        00 000 000
003323   MVI  D,076         00 000 000
003325   STAX D             00 000 000
037020    263 WRITE         00 000 000
003326   MOV  A,E           00 000 000
003327   DCR  E             00 000 000
003330   ANI  017           00 000 000
```

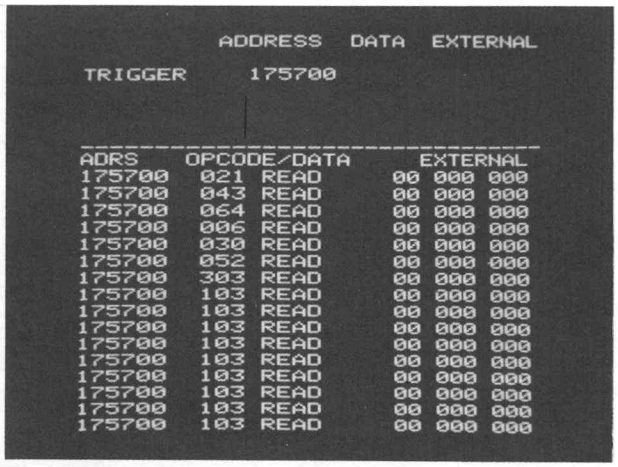
Figure 12. TRACE of the microprocessor execution leading up to the 48th occurrence of the TRIGGER condition.

If an error is found, when analyzing the data, it is probably desirable to view the actual code causing the error. If, for example, the 48th entry was incorrect it would be desirable to list the code leading up to the error. This can be done by specifying a TRIGGER OCCURRENCE of 48 and a PRE-TRIGGER of 20. This will cause the 1611A to store the 20 memory cycles leading up to the 48th occurrence of the TRIGGER (see Figure 12). The same result could have been obtained by using the 47th occurrence of the trigger and delaying to the point of interest.

This last example demonstrates that it is possible to use windowed triggering to generate a trigger only when an expected event does **not** occur. A keyboard is found to randomly ignore a very small percentage of its keystrokes. It is initially assumed that the keys are mechanically intermittent, so they are disassembled and thoroughly cleaned. This does not appear to improve the problem so the design of the keyboard scanner becomes suspect.

This keyboard is addressed and read as memory in a microprocessor driven system. The six least significant bits of its 8-bit output give the keycode of a depressed key. The remaining two bits are used as status bits. Bit 7 (KEY DOWN) goes high when a key is held down and bit 8 (KEY UNREAD) is clocked high when a key is first depressed and then reset low by hardware after a keycode is read. The keyboard output is read repetitively by software until an unread key is detected.

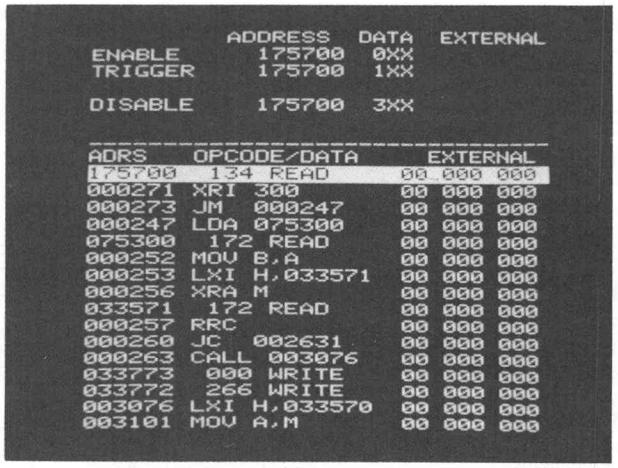
The operation of the keyboard may be examined by triggering the 1611A on the keyboard read address and observing data bus activity. TRACE TRIGGERS is used since only activity at the trigger point is of interest.



```
                ADDRESS   DATA   EXTERNAL
TRIGGER         175700

ADRS    OPCODE/DATA       EXTERNAL
175700   021 READ         00 000 000
175700   043 READ         00 000 000
175700   064 READ         00 000 000
175700   006 READ         00 000 000
175700   030 READ         00 000 000
175700   052 READ         00 000 000
175700   303 READ         00 000 000
175700   103 READ         00 000 000
175700   103 READ         00 000 000
175700   103 READ         00 000 000
175700   103 READ         00 000 000
175700   103 READ         00 000 000
175700   103 READ         00 000 000
175700   103 READ         00 000 000
175700   103 READ         00 000 000
175700   103 READ         00 000 000
175700   103 READ         00 000 000
```

Figure 13. Octal display of the output of the keyboard scanner. When a key is depressed, the most significant two bits are set high. The most significant bit (KEY UNREAD) is cleared following the first read after depression.



```
                ADDRESS   DATA   EXTERNAL
ENABLE          175700    0XX
TRIGGER         175700    1XX

DISABLE         175700    3XX

ADRS    OPCODE/DATA       EXTERNAL
175700   134 READ         00 000 000
000271  XRI 300           00 000 000
000273  JM  000247        00 000 000
000247  LDA 075300        00 000 000
075300   172 READ         00 000 000
000252  MOV B,A           00 000 000
000253  LXI H,033571      00 000 000
000256  XRA M             00 000 000
033571   172 READ         00 000 000
000257  RRC               00 000 000
000260  JC  002631        00 000 000
000263  CALL 003076       00 000 000
033773   000 WRITE        00 000 000
033772   266 WRITE        00 000 000
003076  LXI H,033570      00 000 000
003101  MOV A,M           00 000 000
```

Figure 14. The keyboard read address is 175700_8. The 1611A will generate a trigger if, following a $0XX_8$ at this address, it finds a $1XX_8$ before it finds a $3XX_8$.

Subsequent analysis indicates that the KEY DOWN flag (bit 7) is always set after a keystroke, but that KEY UNREAD flag (bit 8) is sometimes missed. The output is going directly from OXX_8 (all keys up) to $1XX_8$ (key down and previously read) without passing through $3XX_8$ (key down and not yet read). It is desirable to generate an oscilloscope trigger when this event occurs so that the timing of signals in the KEY UNREAD latch circuitry can be examined in detail. The measurement used to generate this trigger is shown in Figure 14.

The 1611A TRACE POINT OUTPUT can now be used to trigger a storage oscilloscope. The trigger is stable with respect to the microprocessor clocks, so alternate single-shot sweeps may be used to examine the critical timing.

CONCLUSION

This new instrument simplifies digital analysis by performing real time measurements of a microprocessor's operation as well as its interaction with the other parts of a system. The use of an internal microprocessor in this logic state analyzer permits ease of data entry and convenience of data display. This improvement allows the operator to easily select from among the extensive triggering options which are necessary because of the enormous amount and complexity of data generated by the system being tested.

USES AND PERFORMANCE LEVELS OF CURRENT LOGIC ANALYZERS AND THEIR FUTURE TRENDS

Edward S. Jacklitch
Biomation Corporation
10411 Bubb Rd., Cupertino, CA 95014

SUMMARY

The Logic Analyzer is a new, but basic type of test equipment for the logic design engineer and the digital troubleshooter. The development of a generic line of equipment in this category is proceeding at a very fast pace. There are time domain analyzers, data domain analyzers, and several instruments have appeared on the scene which are oriented specifically to the analysis of microprocessor-based systems.

Users and manufacturers working in concert, along with available technology, will define the shape and function of future Logic Analyzers.

INTRODUCTION

Up until now, Logic Analyzers have been separated into two distinct categories. One kind of machine was capable of acquiring the electrical signals in a digital system, and displaying these signals as a timing diagram. This instrument is considered to be in the class of time domain analyzers. It allows the user to examine timing relationships of several digital signals.

The other basic category of logic analyzer is the instrument which captures signals and displays them as though they were data signals; i.e., a display field of ONES and ZEROS. The user then examines the information on the display from a software point of view. This instrument is typically known as a data domain analyzer.

Very recently, a third kind of instrument has emerged, which is capable of displaying the digital signals in both domains. The

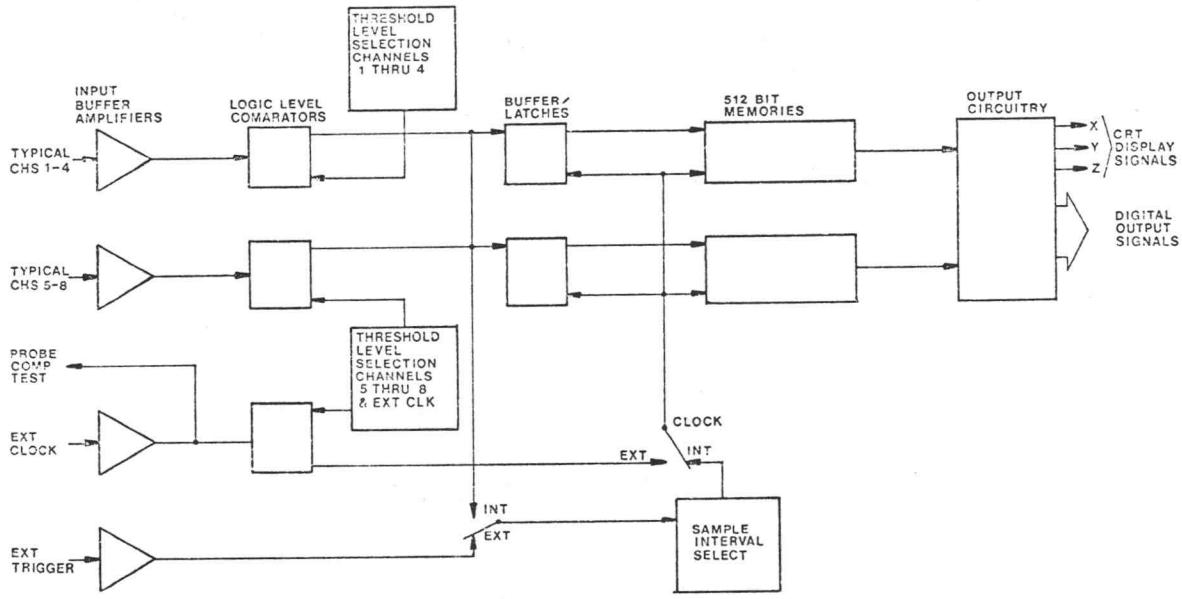


Figure 1. Logic Analyzer block diagram. Data from the system under test is stored in memory, and then read out onto a scope or CRT display.

instrument can either display information in the timing diagram format, or in the data domain.

CURRENT PERFORMANCE LEVELS

Time Domain Instruments

Let us review briefly the operation of a Logic Analyzer by looking at a block diagram of a typical instrument as in Figure 1. The multiple channel inputs have comparators, latches, and a memory for each channel. The output circuitry generates a timing diagram for display on a scope or CRT. The instrument has threshold, and time base and control circuitry. Conceptually, it is quite a simple instrument. As can be seen, there is a good deal of redundancy in the circuit blocks.

In operation, the Logic Analyzer detects the logic states of the digital input signals with respect to preset threshold voltages in the comparators. This is done every cycle of an internal, or external, clock. During a given "record cycle," the instrument fills each memory cell sequentially with ONE/ZERO decisions. When the memory is filled, the record cycle is ended. Then the instrument goes into a display cycle, where it sequentially outputs simultaneously captured data onto a CRT. In the case of the instrument shown, this display takes the form of the familiar timing diagram.

There are advantages to using a timing diagram display format. When operating the Logic Analyzer at record clock rates that are considerably higher than the maximum data rate in the system under analysis, the user wishes to look at timing relationships between various signals. The timing diagram presentation gives the user that flexibility. It is much easier for him to look at timing relationships in this fashion. The display of extraneous signals, or glitches, are much more easily understood within the context of the timing diagram presentation.

In the field of Time Domain Analyzers, there are a range of instruments available that include those costing a few hundred dollars, and can operate at clock rates in the hundreds of kilohertz region, as well as those instruments costing several thousand dollars, which can operate at clock rates up to 200 Megahertz. Figure 2 illustrates those instruments available from Biomation.

To be a useful tool, the time domain analyzer has been endowed with the ability to record data with its own internal clock, or alternately, from the clock of a system under test. It is capable of capturing narrow

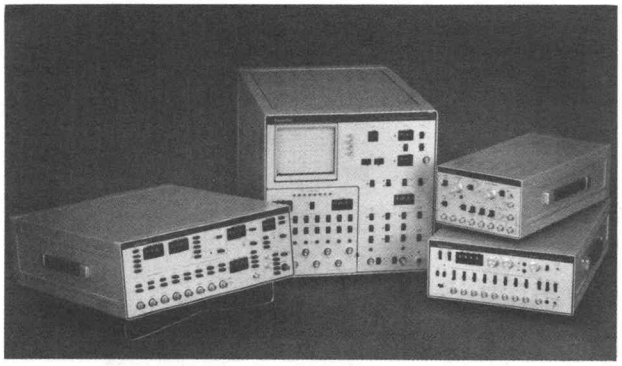


Figure 2. Biomation Logic Analyzer family, spanning a range of maximum clock rates from 10 MHz to 200 MHz.

spikes, or glitches that have widths in the low nanosecond range, since these extraneous signals can be very troublesome to the designer.

Available memory length varies a good bit. Current units have memories that range in size from 16 words to 2048 words. If the Logic Analyzer is to be used for asynchronous recording (recording with its own internal clock, typically much faster than system data flow) it is desirable to have as long a memory as possible. The long memory allows the user to take better advantage of asynchronous recording. He can simply collect more data at faster rates with a longer memory.

Data Domain Instruments

There are occasions when the user wishes to approach his analysis from the data domain standpoint. In this case, the display format of ONES and ZEROS is much more meaningful to him. On these occasions, the user will probably operate the Logic Analyzer synchronously. In other words, he will use the clock that is moving the data around the system to control the record rate of the Logic Analyzer. He can then compare the data presentation with some printed or handwritten source code that he has previously generated. In this manner, he analyzes the system.

Recognizing these differing requirements for different analysis approaches, two instruments are currently available with a display converter. This gives the user the ability to select a timing diagram presentation, or a data domain display by mere pushbutton selection. Both instruments can record with their internal clock for time domain analysis, and use the clock of the system under analysis to record signals for data domain presentation.

The most appropriate display presentation for the so-called microprocessor analyzers, from a user point of view, appears to be the data domain. This is currently implemented by different manufacturers in hexadecimal and binary codes (ONE/ZERO CRT or LED display). There are instruments in this category that are capable of recording data at rates up to 10 MHz, synchronous with the system clock.

Circuit Connections

With the ability to monitor up to 16 signals in a given system, the method of probing the circuit points becomes a serious consideration. The Logic Analyzers currently available tend to divide into two groups: those for which standard oscilloscope probes may be used, and those which require the use of special probes available as part of the Logic Analyzer. There are very definite advantages to both approaches.

In the first case, the user has freedom to select from various different scope probes, according to his specific analysis tasks. If, at a remote site during testing, he damages or loses one or more probes, he can usually commandeer another probe to keep him going. He has much freedom over the physical span of testpoints that he wishes to look at. Consider the application where the user wishes to look at the wire wrapped backplane of a large computer system. Using a standard 3.5 foot scope probe, he can span a distance between testpoints of virtually six feet. On the other hand, if the user is troubleshooting a system with very high density PC boards, the oscilloscope probe tends to be quite unwieldy. It also has a weight disadvantage.

Consider next the use of special probe pods, which each contain multiple channel inputs. These pods provide versatility when troubleshooting equipment with high density PC boards. They are typically light, and have very small signal clips with which to connect to the circuits. However, the probe pod is a special accessory to the Logic Analyzer, and, if one is damaged, it generally means that you're out of commission until you can repair or replace it. There are no substitutes. Also, the probe pod does restrict the physical span of testpoints that are available within its grasp.

In addition, the standard X10 scope probe can generally be selected to provide a rather high input impedance, along with a very small parallel capacitance. This is of great importance when troubleshooting CMOS and ECL circuitry, since the user desires to minimize circuit loading in his system. It is true, however, that added cost is incurred in outfitting the Logic Analyzer with scope probes.

For the microprocessor analyzer, the method of connection to the system is via a "personality module." The user plugs his processor chip into the module, and plugs the module cable into his system.

APPLICATIONS

When troubleshooting various kinds of digital control systems or computer interfaces, we are, many times, faced with the problem of malfunctions that occur in a nonrepetitive fashion. In order to narrow down on these problems, we would usually like to look at several different signal lines at the same time. This allows us to develop cause-and-effect relationships among the various signals, and ultimately diagnose the problem at hand.

Let's consider why the oscilloscope has been supplanted in many cases by the Logic Analyzer. The typical dual trace portable oscilloscope is generally used for troubleshooting digital systems. We face a very basic problem when trying to look at simultaneous signals in either the alternating sweep mode, or in the chopped display mode, on an oscilloscope. Figure 3 illustrates the problem when using the scope in the alternate sweep mode.

The scope first shows signal activity at one of its inputs, then retriggers and shows signal activity on another input. Obviously, one cannot observe signal simultaneity.

If the scope is operated in the chopped display mode, simultaneity in the two signals of interest is achieved, but at the expense of time resolution. See Figure 4. The rate at which the scope is chopping the inputs determines the minimum time resolution available in the displayed image. If glitches occur in Channel 1 during the time that Channel 2 is chopped onto the screen, the user will be unable to see those glitches.

If we relied strictly on a scope for analysis, solving many system problems would be virtually impossible.

The Logic Analyzer, because it has multiple inputs, can achieve simultaneous recording and display of many signals, limited only by the number of input channels provided by the analyzer. The maximum clock rate of the Logic Analyzer still limits the time

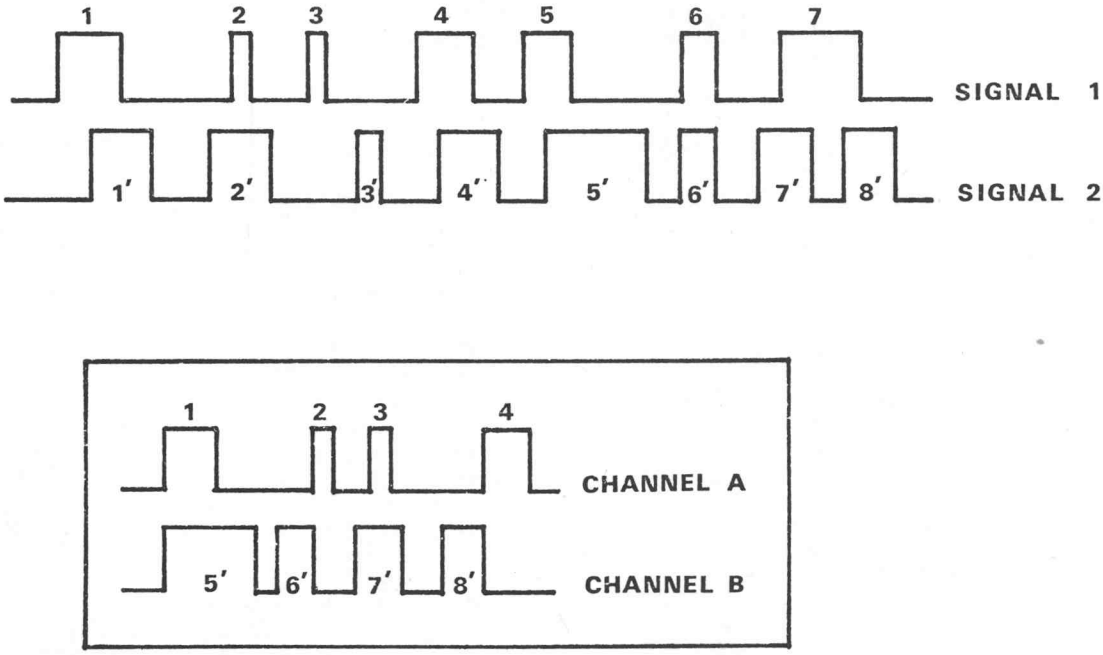


Figure 3. Using a scope in the alternate sweep mode to view two logic signals. Note that the signals are not displayed simultaneously, but sequentially.

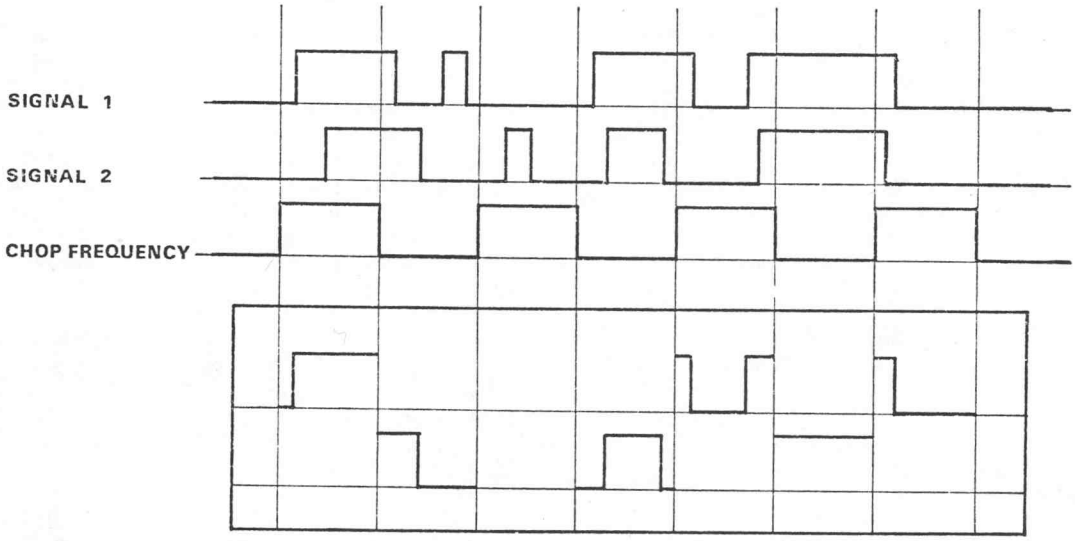


Figure 4. Using a scope in the chopped sweep mode to view two logic signals. Note that the signals are now being displayed simultaneously, but parts of each signal are being lost. Time resolution is limited by the chop rate of the scope.

resolution, but analyzers are available whose clock rates are orders of magnitude faster than the typical chop rate of the average oscilloscope. Also, with the addition of the so-called Latch mode on the Logic Analyzer, the user is assured of being able to capture and display random or periodic glitches.

Since the Logic Analyzer is a capture-and-store type of instrument, it allows the user to view non-periodic signals. This is especially important when malfunctions of a given system are random in nature, or occur once every two weeks. The Logic Analyzer can be set up to capture these random malfunctions without operator assistance, and record and display signals that occurred prior to a trigger event.

Even for those signals that are repetitive, but have a very low repetition rate, the Logic Analyzer provides a much better solution to the problem of viewing such signals. Remember that the brightness of the scope display depends upon the repetition rate of the signal. This is not true for the display output of a Logic Analyzer. The intensity of the display output is independent of the repetition rate of the signal of interest.

Designers have found themselves in the middle of a period of great change in the electronics industry: the advent of the computer on a chip. There are many process control and computational functions that can be more readily implemented by the use of a microcomputer. Once a microcomputer becomes part of a given system, the analysis approach becomes software oriented. Rather than defining the system by means of a timing diagram, the engineer now defines the system operation by means of program statements. He would now like to analyze his system by examining the data flow patterns within the system.

This implies the need for many more channels than have been available up to this time. For instance, the typical 8-bit microprocessor requires a recording capability of 16 lines for address information, and 8 lines for data. It is also desirable to qualify the data by signifying whether a given instruction or piece of information was a read, or a write, operation.

The microprocessor analyzer, new this year, provides these special capabilities. The unit actually records transactions between the microprocessor and its memories. Units are available that really analyze recorded data, and present that data in several different switch selected display modes. The display modes show the recorded data from different aspects. The user can first look at overall system activity during the record period; then he can move in on the data to look at it in more detail. There are also two display modes which relate the data as it occurred in time; i.e., in sequence.

THE FUTURE

As both users and manufacturers become more sophisticated in their approach to Logic Analyzers, we will see models introduced that perform more specific functions. Already, the marketplace has evolved in such a manner that we not only have instruments that can be generally applied to any kind of time domain analysis of circuitry, but also instruments that can be applied to any general problem of data domain analysis. There are instruments available that allow the user to input a stimulus to, for instance, a serial data system, and record its response to that stimulus. Other instruments are becoming available for the express purpose of debugging the software of microprocessor-based systems.

In the future, we will no doubt see Logic Analyzers that can operate at higher clock rates, but the limitations will be the available technology; i.e., how fast can you go with the available integrated circuits? Even more important than speed, user features will start becoming more well developed. The design engineer needs a whole different set of features than the production test technician. More emphasis will be placed on system integration, so that, during production test of various high speed digital circuitry, the test procedure can be automated. This approach requires that the Logic Analyzer be capable of remote control by the computer, and also that the data that the Logic Analyzer acquires can be read by the computer. There is a very basic advantage to using a Logic Analyzer for computer controlled production testing of digital circuitry. The Logic Analyzer acts as a high-speed buffer storage element in the system. That is, the typical computer cannot operate at input speeds much above 1 MHz. The Logic Analyzer, being able to capture and store information at rates that are not dependent upon the computer, and typically much faster, provides a valuable system function. Hence, by interfacing the Logic Analyzer with a computer, the latter gains the ability to look at data with a time resolution as good as 5 nsec.

For a large segment of the electronics community, the production testing of digital hardware is still done pretty much manually. The test technician uses a dual trace oscilloscope at his test station. By providing him

with enhanced capabilities inherent in the Logic Analyzer, the efficiency of his time can be greatly magnified. His special needs will be identified, and incorporated into new instruments.

Logic Analyzers, as they exist today, do not really analyze the recorded data. The instruments do little more than capture and display the digital information. In the future, users will define those tasks that they feel will be of value to them. These tasks will then become a part of the Logic Analyzer. When the Logic Analyzer starts performing real analysis tasks, a point will be reached where a microprocessor control will be desirable. When the marketplace reaches that point, we will probably see a bewildering array of choices in Logic Analyzers.

CONCLUSIONS

Whatever you want to call them, digital logic recorders, logic state analyzers, logic analyzers, or microprocessor analyzers, they are here to stay. They have helped a great deal to further the state of the art in digital techniques. The Logic Analyzer, in the years to come, will become as pervasive as the ordinary bench oscilloscope.

References:

Farnbach, William, "Troubleshooting in the data domain is simplified by logic analyzers," Electronics, May 15, 1975, pp. 103-108.

Hill, J. Carver, "The state of logic analyzers," IEEE Spectrum, December, 1974, pp. 63-70.

House, Charles H., "Engineering in the data domain calls for a new kind of digital instrument," Electronics, May 1, 1975, pp. 75-82.

Schuette, Herman," Logikzustandsanalyse-eine neue Messmethods", electronikpraxis, April 1976.

Tottingham, Roy, "Logic analyzers open a direct path from digital theory to practice," Electronic Products, July 15, 1974.

Trifari, John, "Digital system problems? Use a logic analyzer," Electronic Products, March, 1976, pp. 33-37.

THE LOGIC ANALYZER
THE COMPLEAT SYSTEM INSTRUMENT

J. Carver Hill
E-H Research Laboratories, Inc.
515 11th Street
Oakland, California 94604

SUMMARY

Logic Analyzers to date have used data compression and data recording to effect measurements that are both extremely useful in the context of digital systems and beyond the capability of preceeding instruments such as oscilloscopes.

Slight improvement in their real-time data processing capability, as exemplified by the new E-H 1330 DIGISCOPE, further enlarges the realm of measurements that can only be made with this type of instrument. Incorporating microprocessor control of the data-display function allows recorded data to be presented in either hardware or software oriented formats, thus making the logic analyzer the "compleat" instrument for digital systems.

INTRODUCTION

Why The Logic Analyzer?

The great purpose and advantage of a logic analyzer is that it allows developers (maintainers) of logic systems, i.e. digital systems, to analyze their charges from real-time *observations* of system operation, whether correct or incorrect, rather than from post mortem *deductions* about system malfunctions. To fulfill this purpose logic analyzers must acquire, store, and process a great deal of information in order to make the observations accurate and understandable.

Although there are many different logic analyzers available they all have four salient components:

1. The data acquisition section consisting of probes, cables, and analog to binary conversion circuits (one complete data acquisition path per input channel),
2. The real-time data storage section; frequently used as a display refresh memory as well,
3. Some data interpretation capability, both real-time (combinatorial triggering, greatest address, least address etc.) and display-time (emphasis, doesn't-match-intensification, etc.), and
4. The display section which drives an internal or external CRT to construct a symbolic or diagramatic representation of the data (and some-times the instrument settings as well) on the screen.

Complete or Complement?

The logic analyzer in any of its current implementations really is complete in that it has the three basic attributes that are essential to effective analysis of logic, i.e. digital, systems:

*Simultaneous processing of multiple input channels
*Pre-trigger data acquisition ("negative" trigger delay)
*Transient recording

However, the logic analyzer is able to perform these magnificent feats only because it is a filter—an amplitude and time filter that compresses continuous voltage-versus-time phenomena into descrete voltage-comparison (logic state) versus clock-edge data. Consequently, the logic analyzer is incomplete in that is necessarily rejects some information.

The trick then, to making the logic analyzer a complete instrument is to design the "filter", i.e. the data compression algorithm, in such a way that no "essential" information is rejected. What is "essential" depends very much upon the application.

If the logic analyzer is assumed to be primarily a software (firmware) debugging tool then it is not necessary to record logic-level excursions persisting for less than one complete clock period (glitches); furthermore, the recorded data need not be related to absolute time but onlv to correct time sequence. Under these assumptions the logic analyzer can be implemented without glitch capture circuitry and without a time-base. The resulting instrument is complementary to the oscilloscope and greatly enhances its capability.

On the other hand, if the logic analyzer is assumed to be a hardware debugging tool, as well, then the ability to capture glitches and relate data to real-time can not be discarded. Consequently, under these assumptions the analyzer is implemented with a time-base, glitch-capture circuits and possibly two threshold voltages (to record low, high and in-between logic levels). The resulting instrument is nearly as effective as the oscilloscope in detecting wave-form anomalies of logic signals, yet far more effective in total diagnostic capability because it processes multiple input channels with true simultaneity and records pre-trigger data—two feats impossible for oscilloscopes.

Thus, while particular logic analyzers may be optimized for particular applications in terms of their acquisition "filter" or their display format, as a class they are capable of making nearly all the measurements required to: develope, produce, integrate, and maintain digital systems.

So Far So Good

In very general terms there are only four functions that can be expected of an instrument:

Data Acquisition
Data Recording
Data Interpretation (or processing), and
Data Display.

Logic analyzers have become more effective instruments than oscilloscopes for dealing with measurements typical of digital systems merely bv separating the almagamated data acquisition and data display processes of oscilloscopes into independent processes. This division of function plus some data recording and a smattering of real-time data interpretation ("trigger" word recognition, least address,

greatest address, qualifier inputs etc., etc.) has created an instrument type that makes entirely new kinds of measurements. When considered relative to the enlarged realm of measurement possibilities—

*multi channel simultaniety
*pretrigger data acquisition, and
*transient recording—

tradition instrumentation is incomplete.

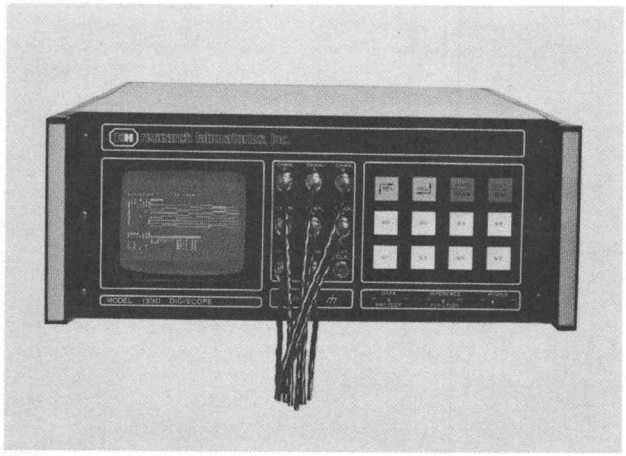


Figure 1. Model 1330 Digiscope

NEW DIRECTIONS

The first advantages of the logic analyzer came principally from a slight rearrangement of the four instrument functions. The new approach has potential for improvements that are only beginning to be realized. Slight improvements in data processing (real-time and display-time) and data display can be particularly rewarding.

A new logic analyzer from E-H RESEARCH, the 1330 DIGISCOPE (Figure 1), has considerably extended the real-time data processing and data display capability of logic analyzers. Even these new features are far from any "upper limit" on what can be done, but they illustrate the nature and extent of *some* ways in which the basic logic analyzer can be improved.

REAL-TIME DATA PROCESSING

Dual Combinatorial Triggering

Most logic analyzers since their inception have had the capability of recognizing any logical combination of their input signals as a "trigger", or stop-recording signal. Call this parallel combination a vector (or "word" or "state"). Vector recognition is a rudimentary form of real-time data processing. The 1330 has extended this capability and can trigger on any combination of *two* successive vectors. One vector is called "T" and the other vector is called "S" (because "S" comes before "T"). The S vector and the T vector are individually like the familiar combinatorial trigger feature. That is each input channel can be programmed as a HIGH, LOW or DON'T CARE participant in the vector event.

The power of the feature comes from the fact that the trigger event can be set as a combination of the two vector events. I.e. the 1330 can be set to trigger on:

(1) vector S immediately followed by vector T
TRIGGER = ST

(2) vector S immediately followed by any vector except T
TRIGGER = S#T, and

(3) Any vector except S followed by vector T
TRIGGER = #ST.

The combination #S#T is not included for obvious reasons. Note that if the S vector is a null vector (all channels programmed as DON'T CARD) and the combination
TRIGGER = ST
is selected, then the T vector corresponds exactly to the familiar single combinatorial-trigger feature available on most logic analyzers.

The combination, trigger on S followed by T allows data acquisition windows to be chosen far more selectively when studying the operation of control logic, interfaces, or data buses.

The combination, trigger on NOT S followed by T, allows the instrument to be triggered when ever the valid state T is entered from any state other than S. Thus while S, T, and the predecessor of T, "P" say, may all be valid states of the system—the state transition P to T may be an illegal transition. The 1330 can trigger on such a transition. (Suppose an interface repeatedly hangs-up in a valid state, "F",—this is possible with pulse-mode logic without DC interlocks—it does no good to trigger on the state "F" and look back in time because "F" is a valid state that occurs millions of times more frequently than the failures. On the other hand, triggering on NOT S followed by T, where S is set equal to the only legal predecessor of F and T is set equal to F, may provide precisely the measurement needed to understand the problem.)

The combination trigger on S followed by NOT T is extremely powerful in tracking down subtle problems. If the function associated with a particular sequence of logic states seems not to occur everytime it should, the instrument can be programmed to trigger on the proper precipitating state followed by any state other than the one which should occur next—it is not necessary to know, *or guess,* what state is being entered erroneously. Basically, the instrument triggers if T doesn't happen after S. This is one way in which the 1330 extends the fundamental capability of logic analyzers—*it can trigger on something that wasn't there when it should have been.*

Delayed Data Acquisition

Another elementary form of real-time data processing is the simple digital delay feature which allows the data acquisition window to be moved down-stream (later in time) relative to the trigger event.

The 1330 has an option, called Mark Delay, that, with a little more real-time processing capability, provides much more measurement capability than a simple digital delay. The Mark Delay option has provision for defining still another combination of the inputs (each of the inputs may be set as a HIGH, LOW, or DON'T CARE participant in the vector)

called the Mark event. The programmed delay value is measured from the Mark event. The instrument can be set to either Load data or Arm for trigger upon expiration of the delay.

Note that if a single combinatorial trigger vector were specified (S or null vector, say) and the Mark vector was set identical to the T vector, and Load-at-expiration was set; then the Mark Delay option would function in a manner identical to the Delay-Only option.

If the Arm at expiration option is elected then the instrument will wait for a trigger event before completing the measurement. Moreover, the ARMed condition will persist for only a settable number of clock-tics. If no trigger event occurs within the ARMed window the instrument will reset the delay counter and return to the Mark-search state (where it looks for a Mark vector). When the Mark vector reoccurs it will precipitate the delay count-down again. Upon expiration of the delay, the 1330 will Arm again for the duration of the specified Arm window etc., etc. A measurement will be completed and presented on the screen whenever a Trigger event occurs within the limits MARK EVENT + DELAY to MARK EVENT + DELAY + ARM DURATION.

Consequently a particular event can be chosen as the trigger event from a number of identical (combinatorially) events based on its time position or sequential order with respect to some other identifiable event—the Mark. There are many possible applications for this feature.

Suppose for example that it is suspected that a one-shot sometimes times-out too early. Assume the one-shot is supposed to have a minimum duration of 11μs. The time scale could be set to 5μs/division to yield a 50ns period, the delay could be set to 12 periods, and the Arm window could be set to 204 periods. If the leading edge of the one-shot is selected as the Mark event and the trailing edge of the one-shot is selected as the Trigger event the instrument will trigger if the trailing edge of the one-shot ever occurs less than 10.90μs after the leading edge of the one-shot (204+12+2 (jitter) = 218 periods equals = 10.90μs). As long as the one-shot exceeds or meets its "minimum" duration of 11μs the instrument will never trigger.

But there is more.....
The user can choose to decrement the delay count by either clock periods or by Trigger events. Thus setting the delay value to 41013, the decrement mode to events (rather than periods) and the expiration choice to Arm will allow the instrument to trigger on the 41014th Trigger event after the Mark event EVEN IF THE TRIGGER EVENTS OCCUR APERIODICALLY. Positioning a variable width, trigger-armed window with respect to an arbitrary Mark state by decrementing a delay value by trigger events or by clock periods (internal or external) is a powerful extension of real-time processing in logic analyzers.

But there is still more.....
Up to now we have taictly assumed that measurement completion involves expiration of the delay but that is not always the case. The Mark-Delay counter can be programmed to be reloadable (like a retriggerable one-shot) in this mode the Mark-Delay counter is reloaded to its initial value every time a Mark event occurs, as long as certain conditions prevail.

In the "GREATER-THAN" mode the counter is reloaded to its initial value upon each occurance of the Mark provided the delay has not expired (the delay may be decremented by tics, or trigger events). If the delay expired before the occurrence of a Mark event, subsequent to the Mark event which last reloaded the counter, then the instrument will exit the delay loop and either Arm or Load as selected.

Consequently, exit from the delay loop is dependent upon the period between two successive occurrences of the Mark event being *greater than* the programmed delay value. This feature can therefore be used to look for long cycles or missing pulses (events). This is one more way that the instrument can be used to *trigger on something that should have been there but wasn't.*

In the "LESS-THAN" mode the Mark-Delay counter is reloaded to its initial value upon each occurrence of the Mark provided the delay *has* expired (again the delay may be decremented by clock tics or trigger events). If the Delay has not expired when a Mark event (other than the first-detected Mark event) occurs then the instrument will exit the delay-loop and Arm or Load as selected.

In this case, exit from the delay loop is dependent upon the period between two successive occurrences of the Mark event being shorter then the programmed delay value. This feature can therefore be used to look for short cycles or extra pulses (events). This is one more way in which real-time data processing allows the instrument to trigger on an event that is combinatorially correct but out of place in time or sequence.

In summary, the DUAL-COMBINATORIAL TRIGGERING and MARK DELAY features allow:

1A. Triggering on a state with an incorrect predecessor
1B. Triggering on a state with an incorrect successor
1C. Triggering on the transition between two specified states
2. Positioning a trigger armed window of selectable width, a selectable distance (measured in clock periods or trigger event) from an independently defined Mark event
3. Triggering on a *missing* event (pulse), or a long cycle, in a regular (in terms of time, clock periods or trigger events) sequence of those events
4. Triggering on a *extra* event (pulse), or a short cycle, in a regular sequence of those events.

These four features greatly enhance the logic analyzer's usefullness in tracking down the cause of subtle system failures, yet the represent only a slight increase in the logic analyzers real-time data processing capability.

DATA DISPLAY

Control Setting Display

The 1330 uses a microprocessor to interface the front panel controls to the instrument's display and functional modules. This approach has several advantages.

First data can be presented in one format to the display and in another format to the functional modules. The threshold voltages, for example appear on the screen in a

decimal format convenient to the operator (Figure 2). The DAC module which actually generates the threshold voltages, however, requires the information in a binary format. The microprocessor handles the decimal-to-binary conversion with about 200 bytes of code, so the operator convenience of decimal readout of an internally binary parameter is provided for very little cost.

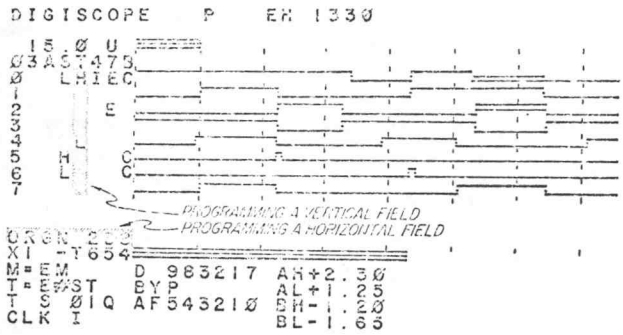


Figure 2. The 1330 Display of Control Settings and Visual Queues

A second advantage of the microprocessor approach is versatility of control interpretation. When the programming mode is entered (by depressing the "RUN/MEASURE" switch) the field to be programmed, the one marked by the cursor, will be video reversed. If this field corresponds to one of the functions programmable by channel, it will be a vertical field. If it is the trigger field for example the entire trigger vector can be programmed "at once" without having to move the cursor to each of the eight rows corresponding to channels. The switch labeled "S0" controls the function for channel 0 and holding S0 down will cause the function to cycle through its possible setting. In the case of the trigger field these settings would be "H" (high), "L" (low) and "Blank" (don't care); when the desired setting appears the switch is released. Switches S1 through S7 control the function for channels 1-7 in an identical manner. If the field to be programmed is a horizontal field the operation of the switches is similar, with S0-S1 usually controlling the leftmost programmable item or digit and S6-S7 usually controlling the rightmost programmable item or digit. When a change of origin is desired, for example, the origin field will be selected and programming will be enabled, then depressing S6 will increment the least significant digit while depressing S7 will decrement the same digit; S4-S5 control the "tens" digit and S2-S3 control the "hundreds" digit in an identical way. Every time the units digit is incremented from 9 to 0 the tens digit is automatically incremented. Decrementing from 0 to 9 cascades in a similar fashion.

As a result, the origin may be "scrolled" slowly by holding down S6 or S7, or rapidly (100 times faster) changed with S2-S3. The interpretation of the front panel switches, the parameter incrementing, decrementing, scrolling, etc. is all done by the microprocessor. The degree of versatility and visual feedback obtained with the microprocessor simply would not be affordable with any other technique.

A third advantage of the microprocessor is that it can use simple program checks to proscribe certain erroneous or misleading control settings that would otherwise have to be precluded by additional hardware or perhaps allowed, despite their potential for operator frustration and confusion. For example: When a times 10 expansion of the display is programmed only 81 bits can be displayed across the screen consequently the microprocessor will not set the display origin to a value greater than 910. Similarly, when the external clock is selected the characters "100 TICS" appear instead of the digital readout of the time value corresponding to 100 periods of the internal clock.

Recorded-Data Display

The display of the recorded data itself is certainly an area ripe for improvement. So far we have had only numerical interpretations or pseudo wave-form interpretations (the state-map display has not yet been implemented for *recorded* data). Yet once the data is recorded there should be no problem in giving the user a choice of data interpretation. The microprocessor display-control of the 1330 makes at least a little flexibility available.

Whenever the X24 time-base expansion is chosen, the data during each bit-time on the screen is interpreted as a column of 3 octal digits directly below the data. (At this expansion, the time allowed to display one bit of data equals the time required to read one row of data from the character generator.) At X24 expansion there are 34 bit-times across the screen so there will be 34 columns of octal digits across the screen. Since scrolling is allowed in the expanded mode the microprocessor is responsible for keeping the correct sequence of 34 octal words on the screen as the operator moves the data viewing window back and forth in the block of 1000 recorded words.

CLOSURE

The logic analyzer is still a new instrument being applied in the newest and most vigorous areas of electronics. This stimulation will keep it evolving for the foreseeable future—particularly in the areas of data interpretation and data display.

PICK A LOGIC ANALYZER FOR YOUR JOB

Murlan Kaufman
Project Manager, Tektronix, Inc.

A number of articles on logic analyzers have been written recently arguing the pros and cons of state versus timing displays, and enumerating the advantages of mapping and other special features. Rumors that the oscilloscope is dead, done in by the logic analyzer, have hopefully been dispelled by now. As logic analyzers come of age, we are beginning to recognize that both oscilloscopes and logic analyzers complement one another.

There is a growing number of LAs on the market, and although they don't yet offer the bewildering variety of features that oscilloscopes do, the need to develop some systematic approach to evaluating them is also increasing. In other words, people need to know how to go about deciding which logic analyzer is right for their job.

The LA user may be involved in a variety of jobs. He may be a hardware or firmware designer, or he may be primarily concerned with debugging existing hardware or software. His title may be production staff engineer, design engineer, or field engineer, and his area of operation may range from incoming inspection to service and repair. The digital circuitry with which the LA user is concerned might be microprocessor-based, random logic, electro-mechanical, or, more likely, a combination of these.

As we consider ways of classifying logic analyzers, it is well to remember that the LA, like the oscilloscope, is a general-purpose tool. An engineer doesn't usually buy an LA with only one application in mind, any more than he would buy a scope because he was having problems with a single circuit. Today he may need only a state analyzer, or only a timing analyzer, but tomorrow's needs may be different. Just as most analog engineers require an oscilloscope with a range of triggering modes and capabilities, the digital engineer would like to switch from state tables to timing displays to an oscilloscope in one instrument.

Of the various special features available in logic analyzers, some are of greater importance for specific applications. For example, an LA with hexadecimal display capabilities is important to anyone who works with microprocessors, since microprocessor programs are listed in hex in order to present instructions and addresses in compact format. If he is creative, the μP designer may be able to make do with sixteen channels at his disposal, unlike the designer of random logic circuits, for whom eight channels may be sufficient.

On the other hand, the engineer in the field may be primarily interested in a portable logic analyzer combined with an oscilloscope.

Another special feature essential to some applications is data comparison sometimes called "Exclusive OR." The people involved with incoming inspection, manufacturing, or testing find this feature invaluable. But some design engineers may seldom use the comparison capability because their circuitry has no existing parallel.

First and Second Order Characteristics

Logic analyzer features may be broadly divided into first and second order characteristics. First-order characteristics are those which define the jobs the machine can do and limit the breadth of application. First order characteristics include memory width and length, speed, synchronous or asynchronous operation, word recognition capabilities, and performance and configuration of signal acquisition (probes).

Second-order characteristics are designed to increase the logic analyzer's efficiency. These include modes of display (waveform, binary or hex), built-in displays, digital delay, latch modes, comparison capabilities, dual and/or separate thresholds, and portability. Sometimes second-order characteristics are added to compensate for practical limitations on first-order parameters. For example, the latch mode, sometimes called a glitch latch or glitch detector, was developed to capture events that occur faster than the sample rate. This function may be implemented in different ways, but its basic purpose is to compensate for limitations in resolution. Other second-order characteristics such as the comparison function are tailored to specific applications.

First-order characteristics are generally the prime consideration of an LA buyer. The answer to how much memory is needed, and how fast a logic analyzer is needed, must be tailored to the budget and the application. Compromises must be made between memory, speed, and dollars. A multimode instrument may cost less and do more than multiple, single-mode instruments. The following summarizes the useful application areas of presently available logic analyzer memory.

TABLE

NUMBER OF CHANNELS		
32	-	total processor operations; addresses, data and control
16	-	μP; mini-program trace; bus transactions
8	-	complex control signals; peripherals
4	-	control signal timing
1-2	-	serial data
MEMORY LENGTH PER CHANNEL		
1k and up	-	high resolution timing; serial data
256	-	timing; long state sequences; one-pass fault analysis
16	-	state sequences
1	-	status latch

Limitations on speed and memory impose special constraints in certain application areas. In general, more memory is required for timing analysis than for state analysis. A digital delay can be used to extend the usefulness of the memory window by moving the window to the area of significant interest. A long memory is especially important in troubleshooting a system whose fault causes the program to crash. Here the troubleshooter will want to use the LA's pre-triggering capabilities to view as large a window as possible of the events before the crash.

Test instruments will always be too slow in one sense, because the fastest technologies available require an even faster instrument to test them adequately. Speed is most important when one is working with technologies having short propagative times. A major limitation of logic analyzers compared to oscilloscopes is that the fastest logic analyzer available today is limited to a sample interval of 5 ns. (Compare this with the oscilloscope's resolution of 0.5 ns or better.)

The LA's signal acquisition capability is another important first-order parameter. The probe must gather data with minimal disturbance of the circuitry; therefore, probe impedance must be high enough to avoid significant ac or dc loading of the circuit. The probe's response must be fast enough not to distort the signal. Convenience of use is another consideration: the probe tips should be designed to attach conveniently to the leads of multilead devices.

Classifying Logic Analyzers: Levels of Abstraction

One can think of any analysis as a process of abstracting or idealizing information. Without some degree of abstraction, it would be impossible to form any judgment at all. One would be inundated by a mountain of meaningless data. Anyone who has ever faced a 50-page computer printout when all he needed was a brief summary can appreciate the value of an appropriate degree of abstraction.

One good way to classify LA systems is by the degree to which they abstract information. The oscilloscope with its voltage versus time display gives the most specific parametric information about what's going on inside the electronic circuit. The timing analyzer, or asynchronous display which establishes a voltage threshold and displays pseudo logic levels versus internal timing intervals, represents the next stage of abstraction. As we move to state or synchronous analysis, we leave the time domain to display logic levels versus an external "clock" or event which may be the system's clock or some other signal or combination of signals. These "clocks" or events may come at irregular intervals; they may be very infrequent, but the engineer is only interested in the logic states in his circuit at those times. He doesn't care about all the information in between. He is "abstracting" the information he needs.

In general, increasing the level of abstraction parallels movement from the realm of hardware into software. The best way to determine what level of abstraction is useful for a particular job is to determine how much you need to know. No one wants to bother unnecessarily with specific information. When trying to analyze complex patterns of data, the searcher doesn't want to be buried under waveform details. State tables in octal or hex will tell him what he needs to know without wasting his time. On the other hand, once he's begun to suspect that the problem is, for example, the propagation delay of a particular component, then he is ready to use the timing analyzer or scope and get down to specifics.

Often the degrees of abstraction represent the natural stages of homing in on a problem, especially when a system such as a microprocessor-based computer goes down. A glance at a map display may reveal the presence of a gross problem in some area of the program. The LA is switched to a synchronous state table mode in order to trace the software program. One advantage of using the state table mode for preliminary search is that it allows you to see a good-sized bite of the program at once. When the point where the program is going awry is found, timing analysis allows the engineer to pinpoint some kinds of faults, by revealing what

is going on between the system clock edges.

In summary, maximum flexibility is obtained when the functions of state analysis, timing analysis and oscilloscope are combined in one instrument. The application stories below illustrate the usefulness of these "levels of abstraction."

OSCILLOSCOPE: THE PHANTOM GLITCH

Martin Marshall, in an article in EDN (June 20, 1975), gave a humorous description of the advantages of the logic analyzer over the oscilloscope, when he compared the LA to Charlie Chan:

Now enters a new master detective, a specialist in digital work, the logic analyzer. While the oscilloscope must wait for a trip of a trigger to begin work, that same trigger signals the end of a logic analyzer's chores. While the oscilloscope is at a loss to cope with a conspiracy between many digital signals, the logic analyzer can tune its ear to listen only for that conspiracy. And, while an oscilloscope requires a villain to repeat his crime enough times to establish a modus operandi, the logic analyzer can haul in the first-time offender.

This effectively sums up the major limitations of oscilloscopes in logic analysis: lack of pre-triggering, limited channels, and difficulty in handling one-shot events. On the other hand, the scope has several advantages over present-day logic analyzers, and in some situations the scope is a necessity. Only the scope can give detailed waveform information. Sometimes this information gives the engineer insight into a problem by a process that can only be described as intuitive (e.g., the "character" of the output signals of an ungrounded TTL package immediately identify the problem to an experienced eye).

The following example illustrates another of the scope's advantages: its ability to look at very fast events.

When the TEKTRONIX 7D01 Logic Analyzer itself was being developed, a glitch problem appeared. A clock in the 7D01 is divided in frequency by four at one point to make a lower frequency clock. Occasionally when the high-frequency clock was being switched to a different frequency during the normal operation of the analyzer, an error would appear on screen.

Another 7D01 was connected to the ECL circuitry of the 7D01 under scrutiny, and showed that one of the dividers was dividing by three instead of by four at the frequency transition. When triggered at the fastest sample interval, the 7D01 would occasionally reveal the presence of an extra-level transition of the high-frequency clock when its frequency was being changed.

In order to take a closer look, the engineer brought in a high-writing-rate oscilloscope (TEKTRONIX 485). He triggered the scope on the signal that switched the fast clock rate, and found a glitch about 3 ns wide in the clock signal, which appeared when the clock signal was high on both sides of the switching time. This tiny glitch was the culprit, causing the divide-by-four circuitry to appear to be dividing by three.

The glitch was the result of a race problem caused by having two different paths for a switching signal to the ECL NOR gates which controlled the frequency transition. The problem was solved by eliminating one path.

The solution to this problem is a good illustration of the homing-in process. The timing analyzer revealed the existence of the problem (divide-by-four circuit occasionally appears to divide by three), and the oscilloscope showed the exact cause (3 ns glitch).

TIMING ANALYSIS: THE CORRUGATED BOX NIGHTMARE

A graphic example of an application of low-speed timing analyzers in maintenance of production-line machinery occurs in the corrugated box industry. A single machine prints the design on the corrugated board, folds the board double, and glues the edges together.

Since the machines are usually run 24 hours a day, downtime for maintenance is costly. Therefore, intervals between maintenance are made as long as possible. Sometimes they are too long, and then disaster may strike. For example, the gluing portion of the machine may jam on a box, which is then treated to repeated applications of a marvelously strong, fast-drying glue. But the printer and folder operations continue to function properly, jamming three boards per second into the gluer, which dutifully pours glue over the whole mess! The result: lots of downtime to clean it up, and missed shipment schedules.

The answer would seem to be more frequent preventative maintenance, but adjustment of these machines is a nightmare. A first try is made and a few boxes run off. They are examined, and a discussion ensues about which tweaks to tweak in which direction. Eventually the machine is in approximate adjustment, and is given the acid test with the throttle open all the way. Sometimes it works . . . and sometimes it doesn't.

Here is where the timing analyzer steps in. The probes are connected to the machine's micro-switches, which report the position of the various machine parts. A short-duration operation of the empty machine allows the timing waveforms of many channels to be captured, and optimum adjustment to be quickly performed. The $200,000 machine is soon back in line, and, since the adjustments were more accurately made, the MTBJ (Mean Time Between Jams) is lengthened.

Preventative maintenance is now possible with no unnecessary downtime, since the timing analyzer can be used periodically to monitor waveforms while the machine is in operation. If adjustments are necessary, they can be performed quickly since the direction and magnitude of the adjustments is known.

STATE ANALYSIS: A KEY PROBLEM

The following problem illustrates the creative use of the pre-trigger capabilities and the Word Recognizer function to determine the cause of an intermittent problem by doing a software program trace.

In the development of a new µP-based product, an engineer discovered that his program was sometimes going out of control when certain keys were pressed. At first he tried stepping through the program to see if things were going as he had planned. No problem was found. Next, by setting break points at several locations, he tried to duplicate the problem, but the program always worked perfectly. Breaking out at almost any point "fixed" it.

He was puzzled: how could he fix a program that was good every time he looked at it, but sometimes went bad when it ran normally? Finally he obtained a 7D01 Logic Analyzer, and hooking all 16 lines to the address bus, ran the program while manually triggering the 7D01, when the processor went wild. He then set the trigger for one of the recurring addresses and restarted the system. When he pressed a key several times, the system went down and the 7D01 triggered. Now, however, he had the instruction addresses that came before the crash because of the pre-trigger capabilities of the 7D01. Tracing back, he found that the first loop to decode the column of the key was "falling" through sometimes when a key was released. It saw that a key was down, but couldn't find it by the time it got to look for it. Some keys were more prone to the failure because they took longer to locate.

The engineer's solution to the problem was to add a "branch always" after the loop, causing the program to continue on to the next sub-routine if "no key" was found.