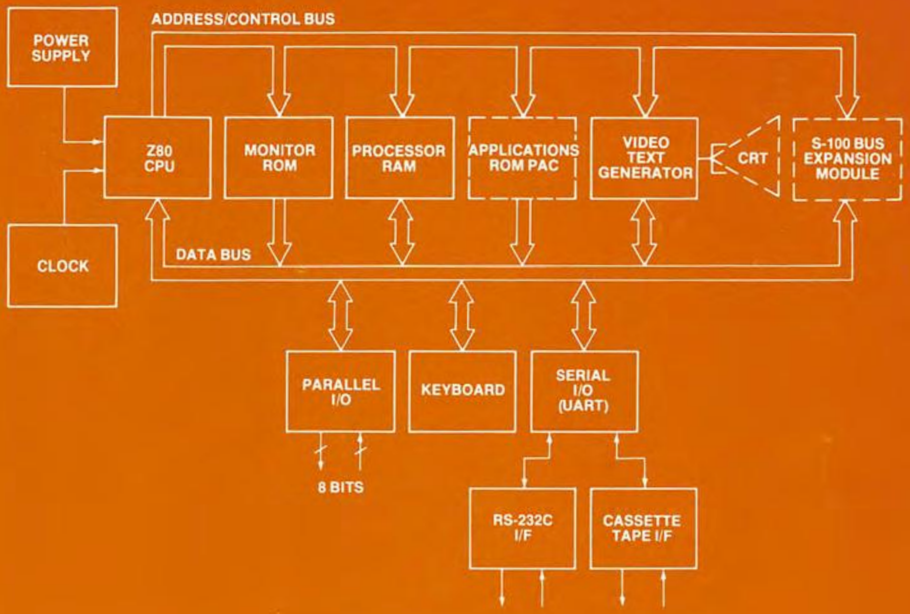
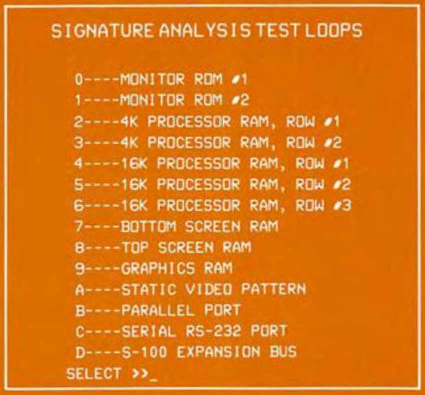
Application Note 222-10
A SIGNATURE ANALYSIS CASE STUDY
of a Z80-Based Personal Computer





```
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 >>_
```

HEWLETT PACKARD

APPLICATION NOTE 222-10

A SIGNATURE ANALYSIS CASE STUDY of a Z80-Based Personal Computer

This case study shows how Signature Analysis was retrofit into a personal computer to allow troubleshooting to the component level with a Signature Analyzer.

FORWARD

ABOUT DIGITAL TROUBLESHOOTING

Microprocessors have revolutionized your product line. Your products are smarter, faster, friendlier and more competitive because they take advantage of μP-based control and computation. They are also harder to build, harder to test and harder to fix when they fail. Complex bus structures and timing relationships have practically obsoleted the scope/voltmeter signal tracing techniques so effective on analog products. The need to enhance the testability and serviceability of your digital products is acute. So is the need for specialized digital troubleshooting equipment.

ABOUT SIGNATURE ANALYSIS

To address these needs, Hewlett-Packard has developed the Signature Analysis technique, as well as a Signature Analyzer product line, for component-level troubleshooting of microprocessor-based products. A Signature Analyzer detects and displays the unique digital signatures associated with the data nodes in a circuit under test. By comparing these actual signatures to the correct ones, a troubleshooter can back-trace to a faulty node. By designing or retrofitting S.A. into digital products, a manufacturer can provide manufacturing test and field service procedures for component-level repair, without dependence on expensive board-exchange programs.

ABOUT THIS CASE STUDY SERIES

Use of a Signature Analyzer requires that some test features be designed or retrofit into the product to be tested. This application note is one in a series of case studies aimed at assisting designers, test engineers, and others in understanding these features so that they can easily add Signature Analysis to their product. These case studies show detailed examples of these features in various digital systems based on specific microprocessors.

ABOUT THIS PUBLICATION

This note shows how SA was retrofit into a Z80-based personal computer to allow testing and troubleshooting on the manufacturer's production line, customer service center, and eventually at their distributor's service centers. It shows the details of circuit stimulus features added for Signature Analysis testing of ROM, RAM and I/O in terms of hardware, software and test connections. It includes stimulus test loop flow charts and Z80 assembly code listings; START, STOP and CLOCK connections and waveforms; and a discussion of the tradeoffs associated with retrofitting SA into this product.

ABOUT OTHER PUBLICATIONS

Application Note 222-0, HP publication 02-5952-7593, is a complete index to the latest Signature Analysis Publications.

It lists all other application notes currently available in the AN 222 series about Signature Analysis. They cover a wide range of interests from how to design or retrofit Signature Analysis into digital systems to the cost reductions that can be expected in production test and field service by doing so.

It also lists all data sheets for the complete line of Hewlett-Packard Signature Analysis products, plus other related publications about digital troubleshooting.

CONTENTS

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. Additional 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.

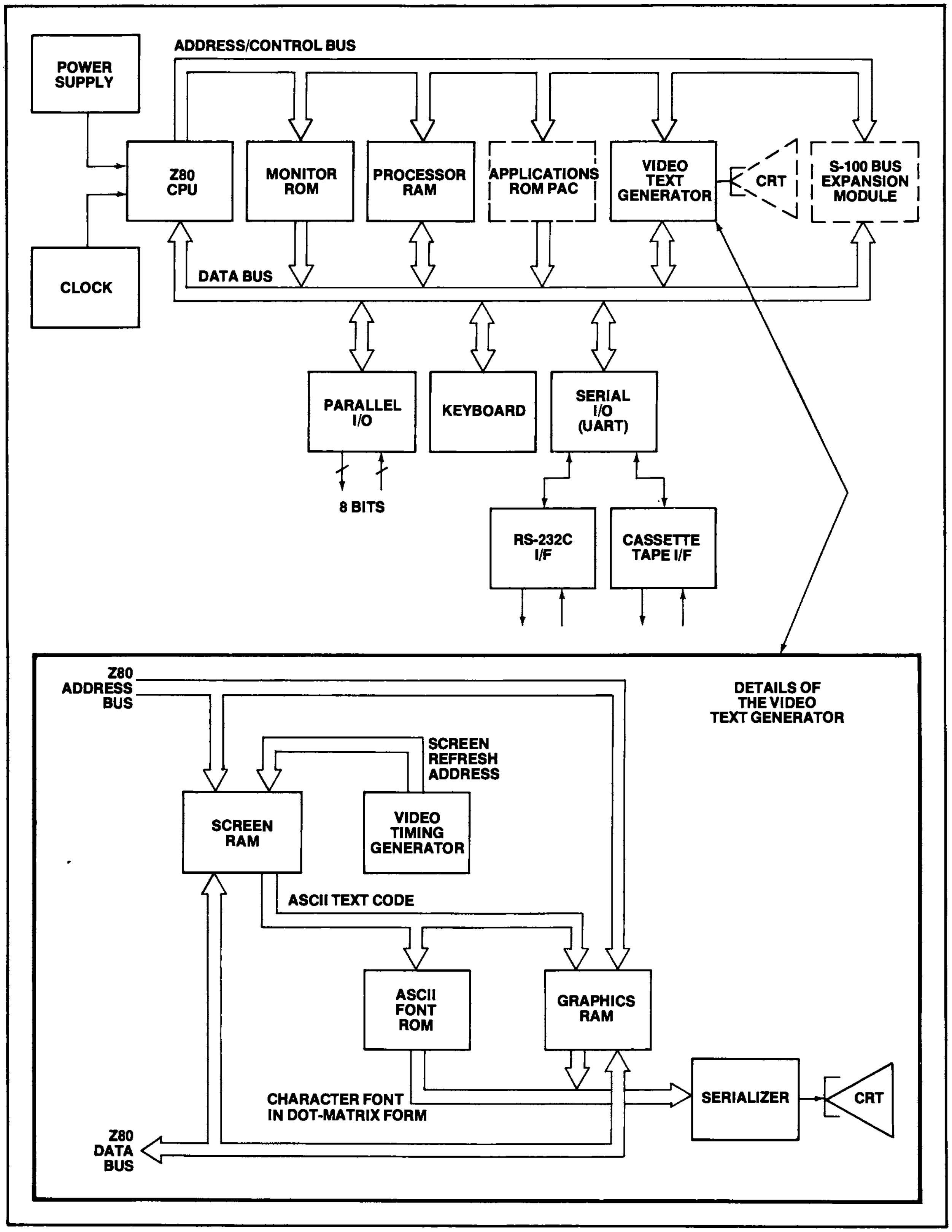


Figure 1. This personal computer contains a Z80 microprocessor, an operating system contained in MONITOR ROM, space for user created programs in PROCESSOR RAM, and a slot on the side of the computer to insert a ROM PAC that contains ROM-based applications programs. Also included is a VIDEO TEXT GENERATOR 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.

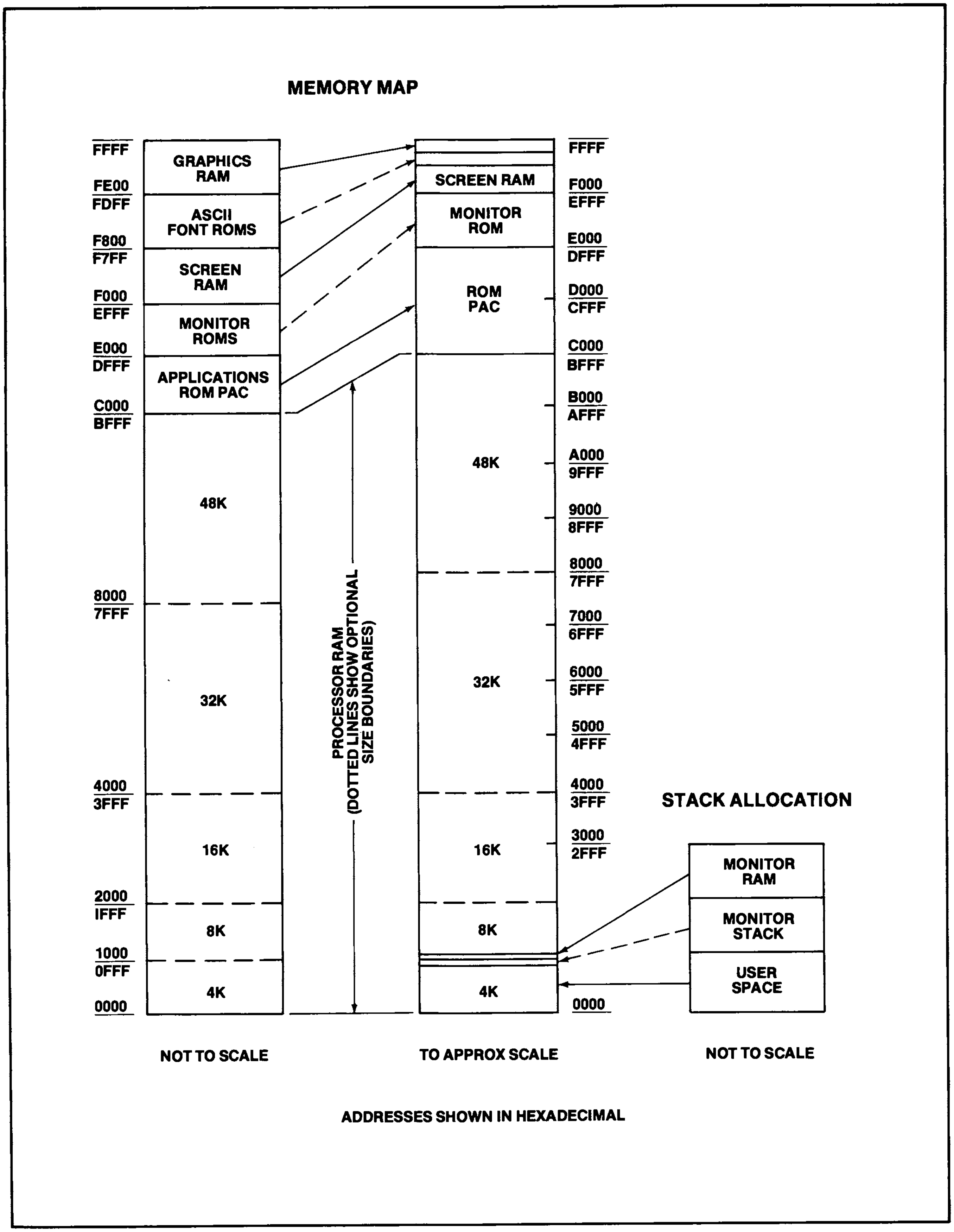


Figure 2. This diagram shows the Z80's 64K byte memory space divided among the circuits of the personal computer. All memory assignments are fixed except for the PROCESSOR RAM which can vary from 4K bytes to 48K bytes depending upon the user's memory requirements. The KEYBOARD, PARALLEL and SERIAL I/O 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 microprocessor.
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 increasing 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.

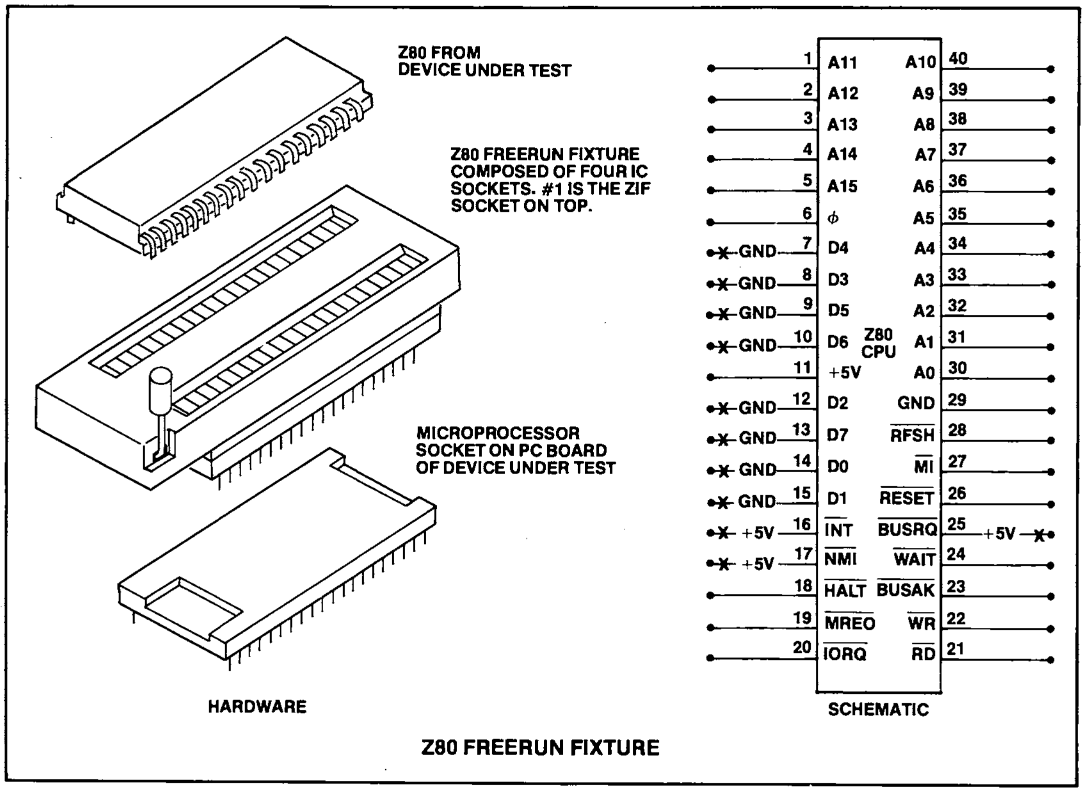
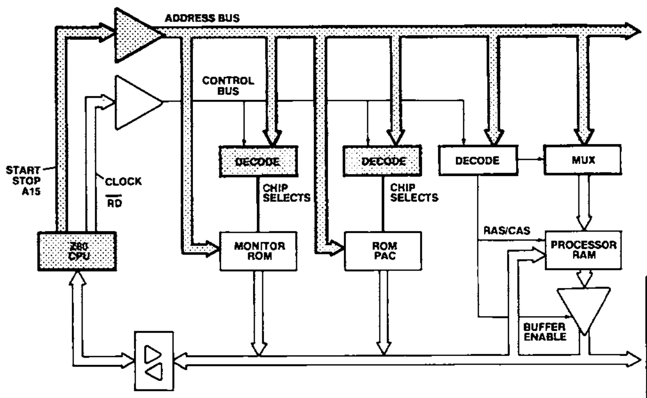


Figure 3. This fixture, quickly constructed from available IC sockets, allowed FREERUN to be easily retrofit into the computer. Modifications to the four sockets that make up this fixture break the data bus between the system and the Z80 and apply a NOP instruction to cause it to FREERUN. Socket #1 is 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.

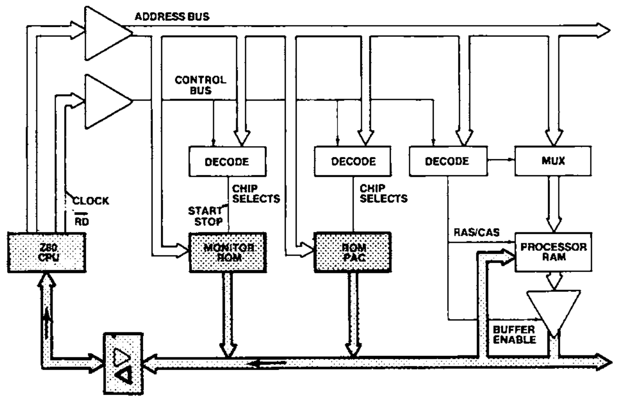


SIGNATURE ANALYZER		Z80	
INPUT	EDGE	SIGNAL	PIN
START	↓	A15	5
STOP	↓	A15	5
CLOCK	↓	$\overline{RD}$	21

ADDRESS BUS
SIGNATURE TABLE

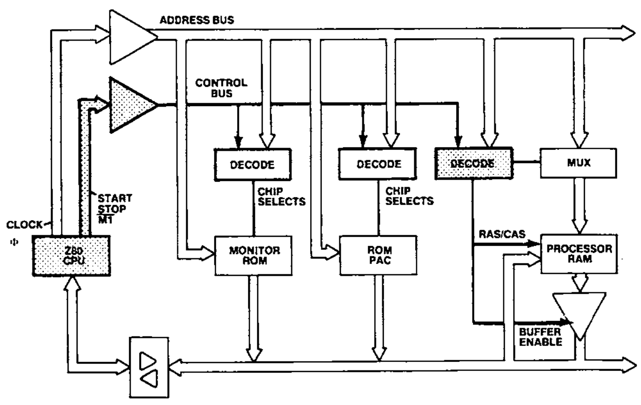
ADDRESS	PIN	SIGNATURE	ADDRESS	PIN	SIGNATURE
A15	5	755U	A7	37	52F8
A14	4	3827	A6	36	UPFH
A13	3	3C96	A5	35	OAFA
A12	2	HAP7	A4	34	5H21
A11	1	1293	A3	33	7F7F
A10	40	HPP0	A2	32	CCCC
A9	39	2H70	A1	31	5555
A8	38	HC89	A0	30	UUUU

a. ADDRESS BUS AND DECODERS



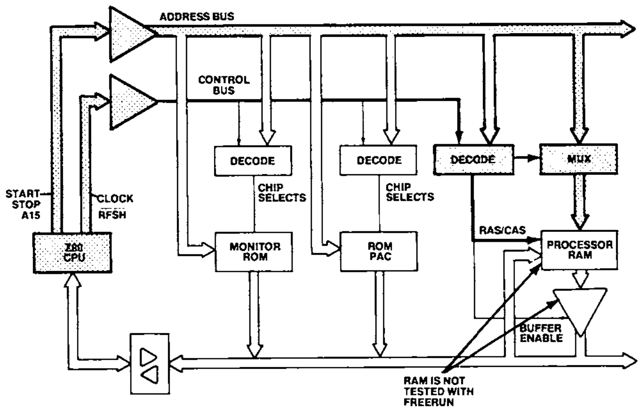
SIGNATURE ANALYZER		MONITOR ROM #1		MONITOR ROM #2	
INPUT	EDGE	SIGNAL	PIN	SIGNAL	PIN
START	↓	CHIP SELECT	—	CHIP SELECT	—
STOP	↑	CHIP SELECT	—	CHIP SELECT	—
CLOCK	↑	Z80 $\overline{RD}$	21	Z80 $\overline{RD}$	21

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	↓	$\overline{RFSH}$	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 interfaces 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.

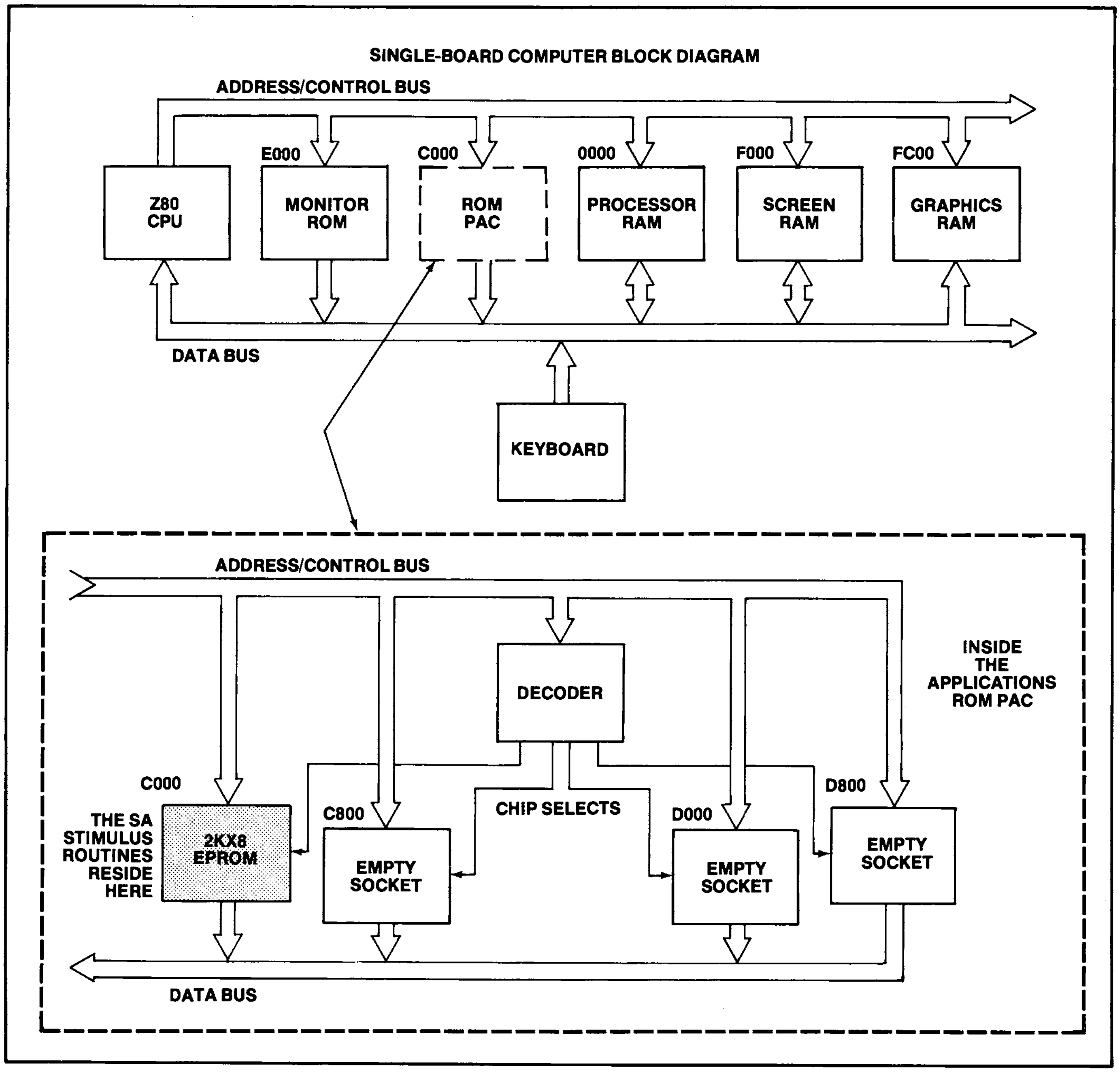


Figure 5. The SA stimulus routines (SA tests) are stored in a 2K × 8 EPROM within an applications ROM PAC that is inserted into the side of the computer. The SA tests are 944 bytes long of which 534 bytes are the ASCII characters of a test selection menu, 110 bytes are a branch table for test selection 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.

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

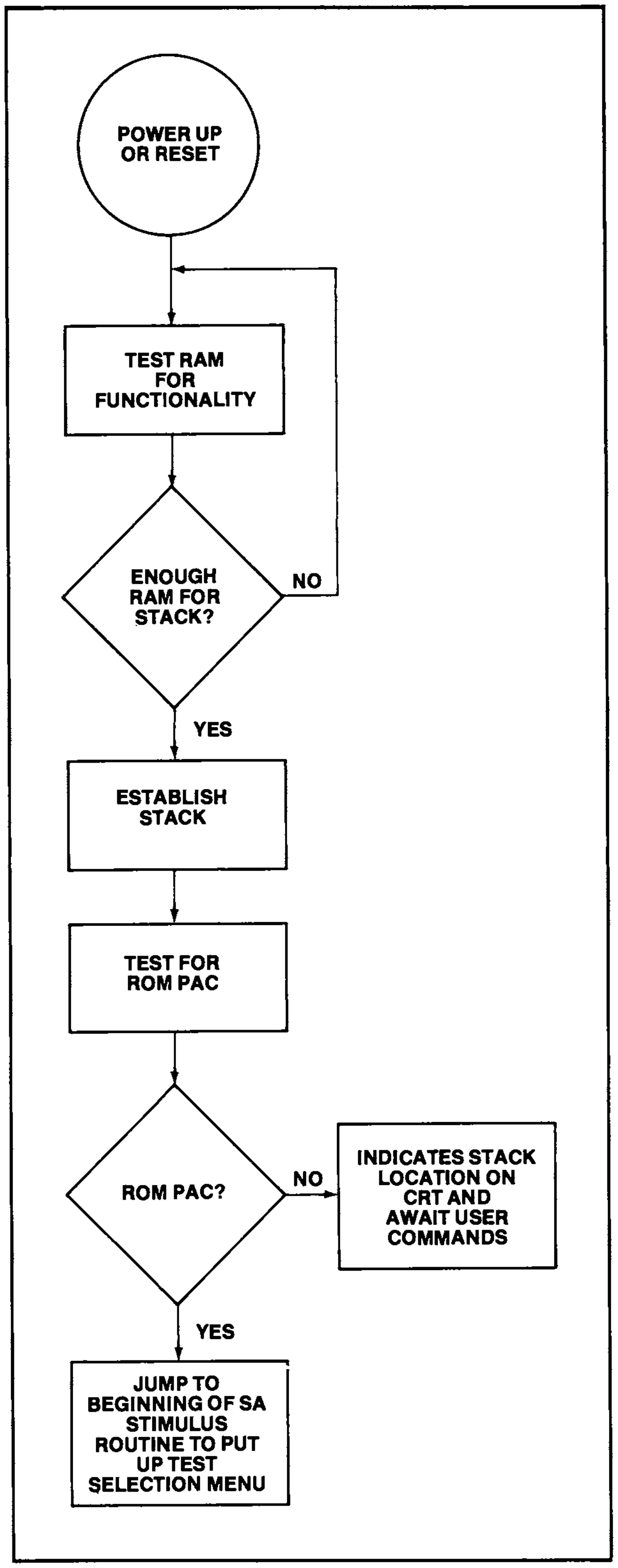

POWER UP OR RESET
TEST RAM FOR FUNCTIONALITY
ENOUGH RAM FOR STACK?
NO
YES
ESTABLISH STACK
TEST FOR ROM PAC
ROM PAC?
NO
INDICATES STACK LOCATION ON CRT AND AWAIT USER COMMANDS
YES
JUMP TO BEGINNING OF SA STIMULUS ROUTINE TO PUT UP TEST SELECTION MENU

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 location 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.

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:

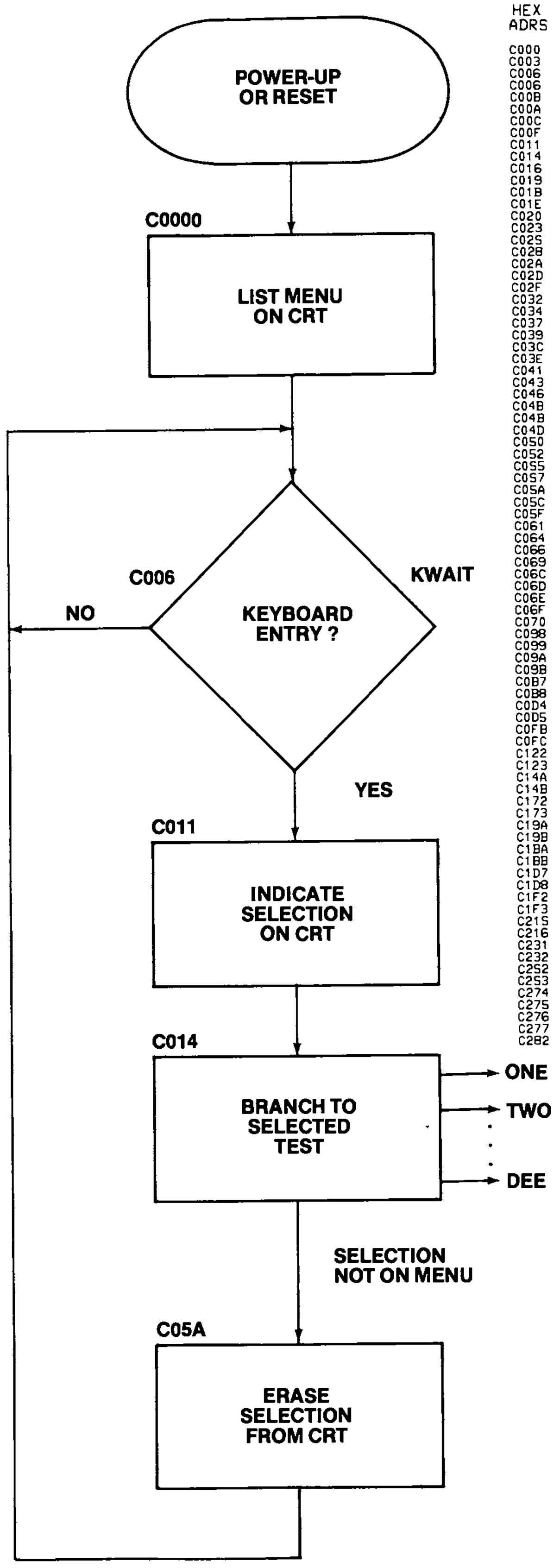
1. It assumes that all logic associated with the keyboard function exists on the keyboard PC board which is not the case here. The keyboard PC board contains only the key switches and one IC. Several other IC's associated with detecting a key closure are on the main board. The result is that exchanging 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)
C00B  CB6F               BIT   S,A
C00A  28FA               JR    Z,KYWAIT-$
C00C  CD1BE0             CALL  KEYBRD
C00F  28FS               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
C02S  CAB4C2             JP    Z,THREE
C02B  FE34               CP    34H
C02A  CABDC2             JP    Z,FOUR
C02D  FE3S               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
C0S5  FE44               CP    44H
C0S7  CAAFC3             JP    Z,DEE
C05A  3E08               LD    A,B
C05C  CD1BE0             CALL  VIDEO
C05F  3E20               LD    A,''
C061  CD18E0             CALL  VIDEO
C064  3E08               LD    A,8
C066  CD1BE0             CALL  VIDEO
C069  C306C0             JP    KYWAIT
C06C  0C        MENU     DEFB  12
C06D  0D                 DEFB  13
C06E  0D                 DEFB  13
C06F  0D                 DEFB  13
C070  20202020           DEFM  '
C098  0D                 DEFB  13
C099  0D                 DEFB  13
C09A  0D                 DEFB  13
C09B  20202020           DEFM  '
C0B7  0D                 DEFB  13
C0B8  20202020           DEFM  '
C0D4  0D                 DEFB  13
C0D5  20                 DEFM  '
C0FB  0D                 DEFB  13
C0FC  20202020           DEFM  '
C122  0D                 DEFB  13
C123  20                 DEFM  '
C14A  0D                 DEFB  13
C14B  20202020           DEFM  '
C172  0D                 DEFB  13
C173  20202020           DEFM  '
C19A  0D                 DEFB  13
C19B  20202020           DEFM  '
C1BA  0D                 DEFB  13
C1BB  20202020           DEFM  '
C1D7  0D                 DEFB  13
C1D8  20202020           DEFM  '
C1F2  0D                 DEFB  13
C1F3  20202020           DEFM  '
C21S  0D                 DEFB  13
C216  20202020           DEFM  '
C231  0D                 DEFB  13
C232  20202020           DEFM  '
C2S2  0D                 DEFB  13
C2S3  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.

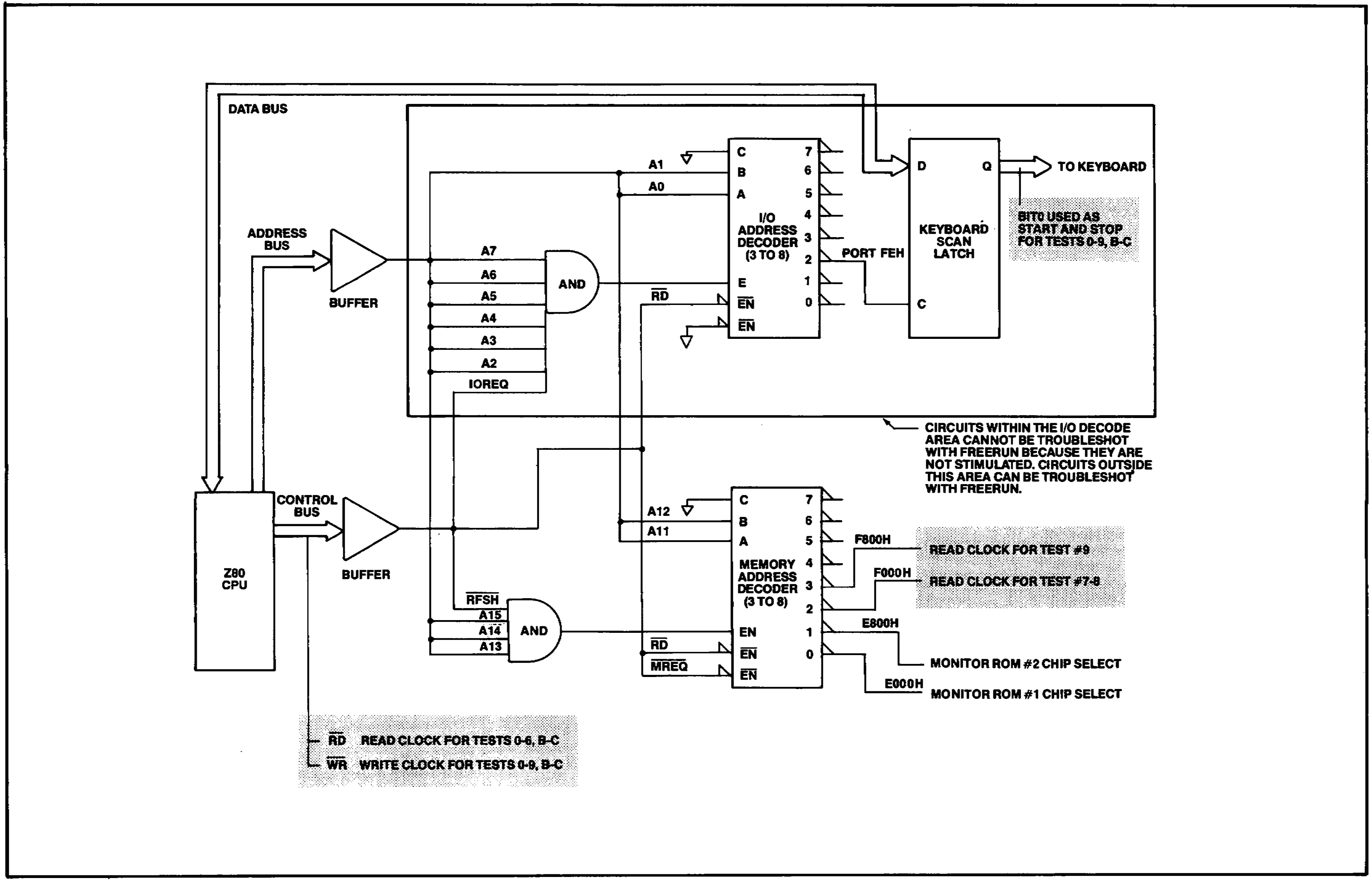

DATA BUS
ADDRESS BUS
CONTROL BUS
Z80 CPU
BUFFER
BUFFER
A7
A6
A5
A4
A3
A2
IOREQ
AND
A1
A0
$\overline{RD}$
I/O ADDRESS DECODER (3 TO 8)
C
B
A
E
$\overline{EN}$
$\overline{EN}$
7
6
5
4
3
2
1
0
PORT FEH
KEYBOARD SCAN LATCH
D
Q
C
TO KEYBOARD
BIT0 USED AS START AND STOP FOR TESTS 0-9, B-C
CIRCUITS WITHIN THE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

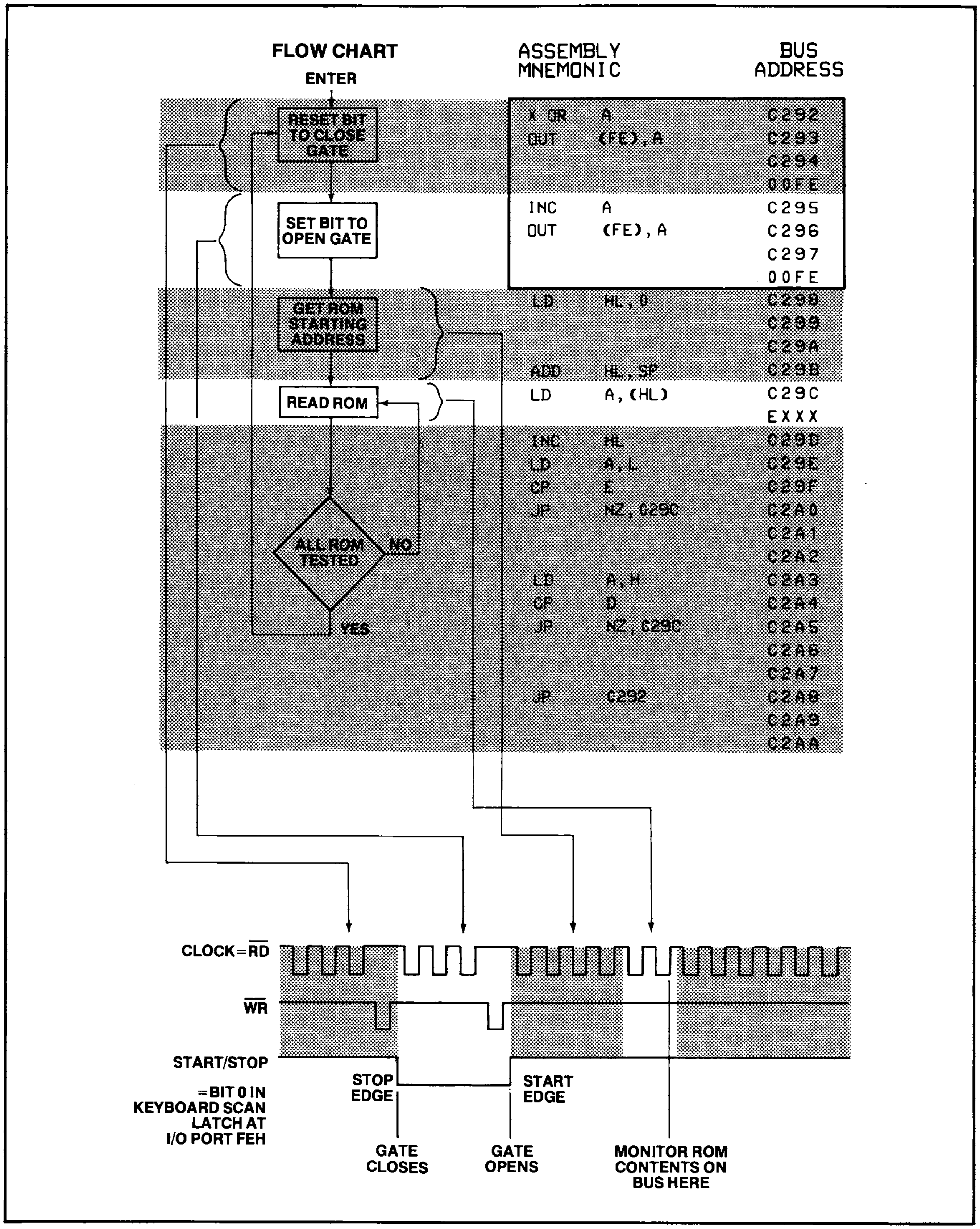
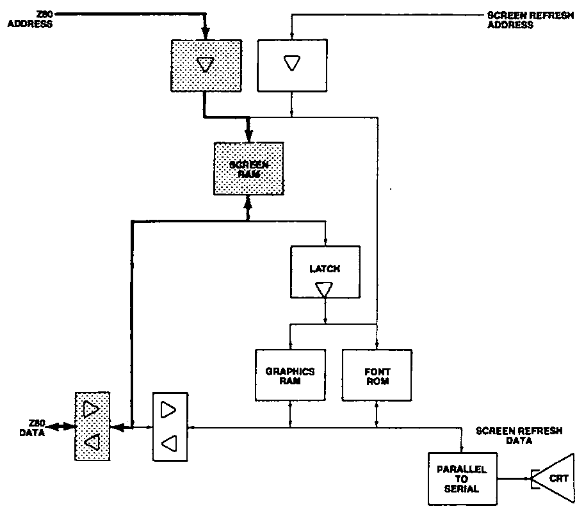
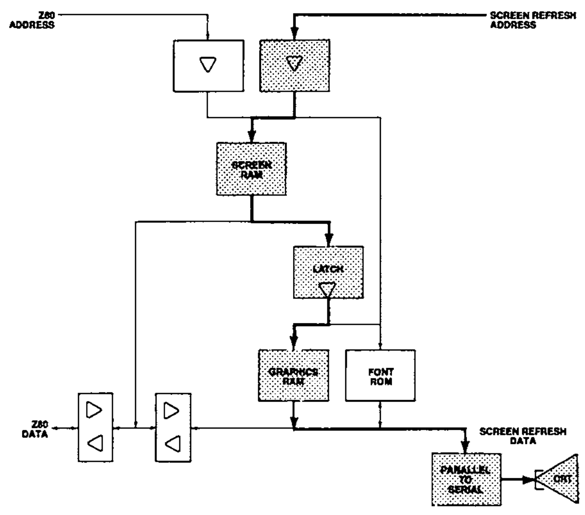


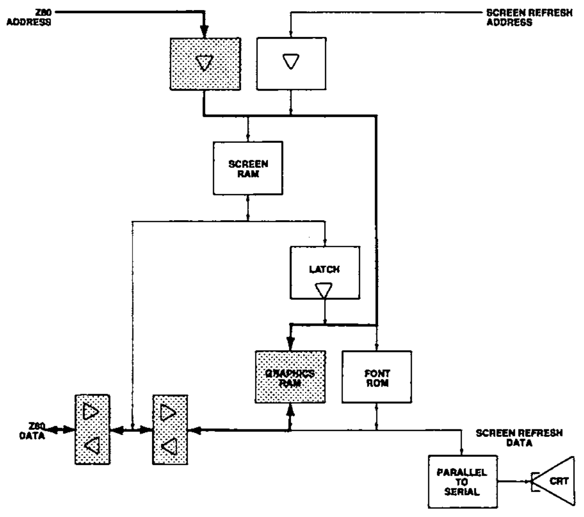
Figure 10. Here is the Z80 code that creates the START and STOP edges by setting and resetting bit 0 in the KEYBOARD SCAN LATCH at I/O port FEH. For tests 0-6 and B-C, two CLOCKS ($\overline{RD}$ and $\overline{WR}$) are used to detect the START and STOP edges. A $\overline{RD}$ CLOCK occurs with every 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.



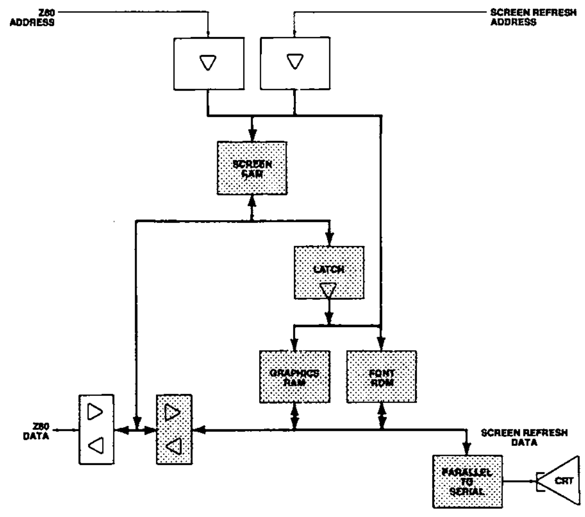
a.)



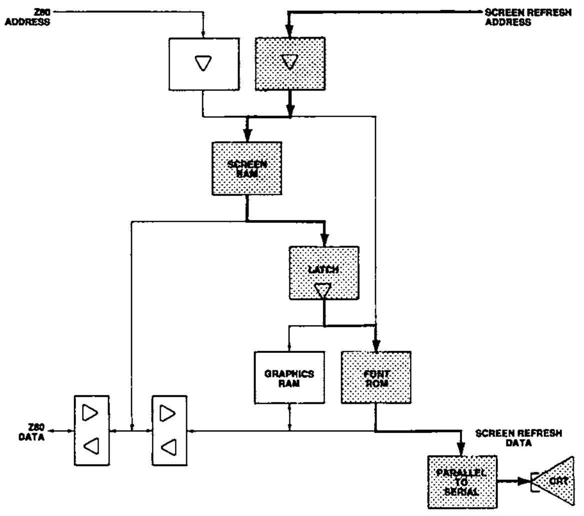
d.)



b.)

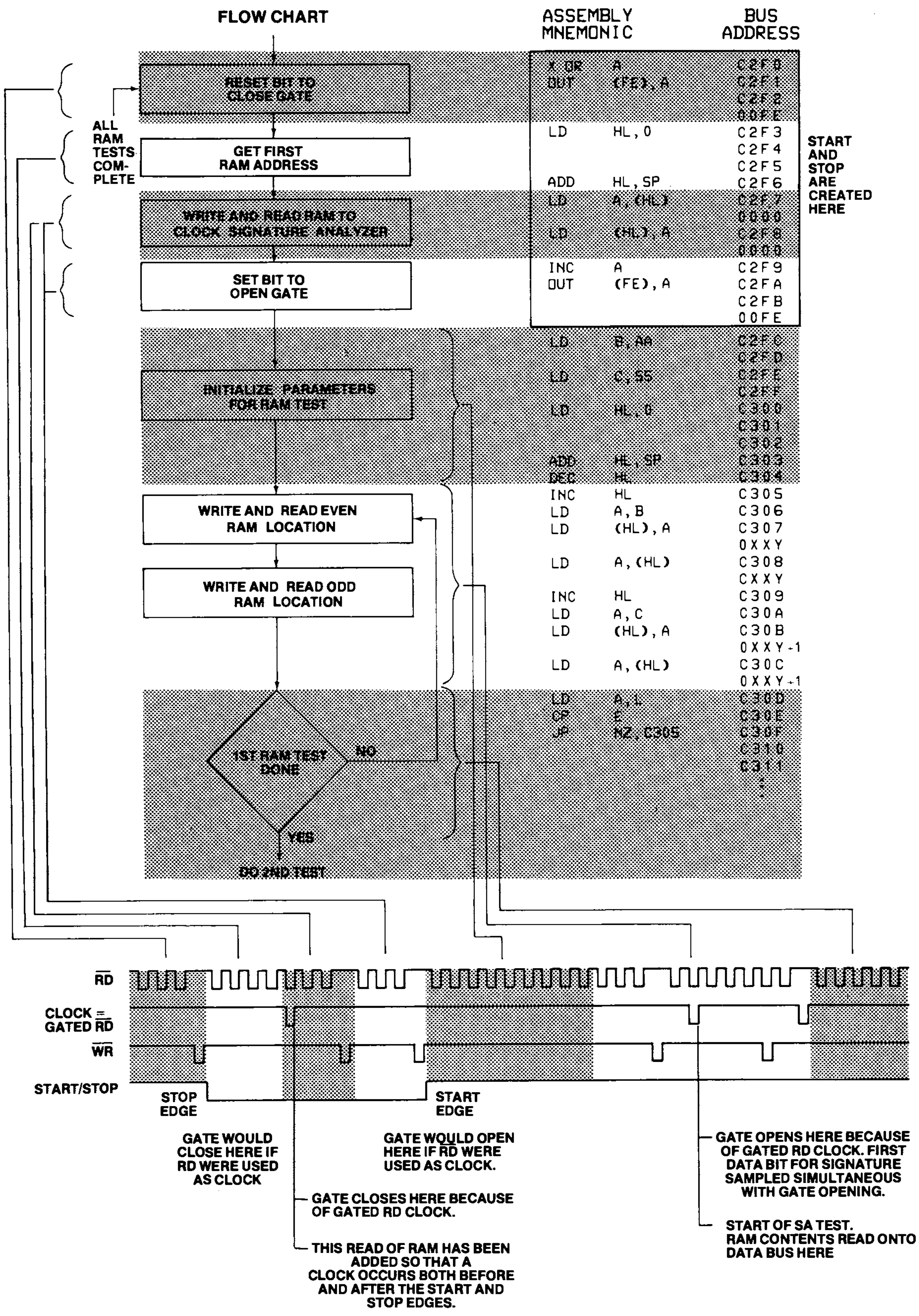


e.)



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
ALL RAM TESTS COMPLETE
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
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 $\overline{\text{RD}}$ 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.

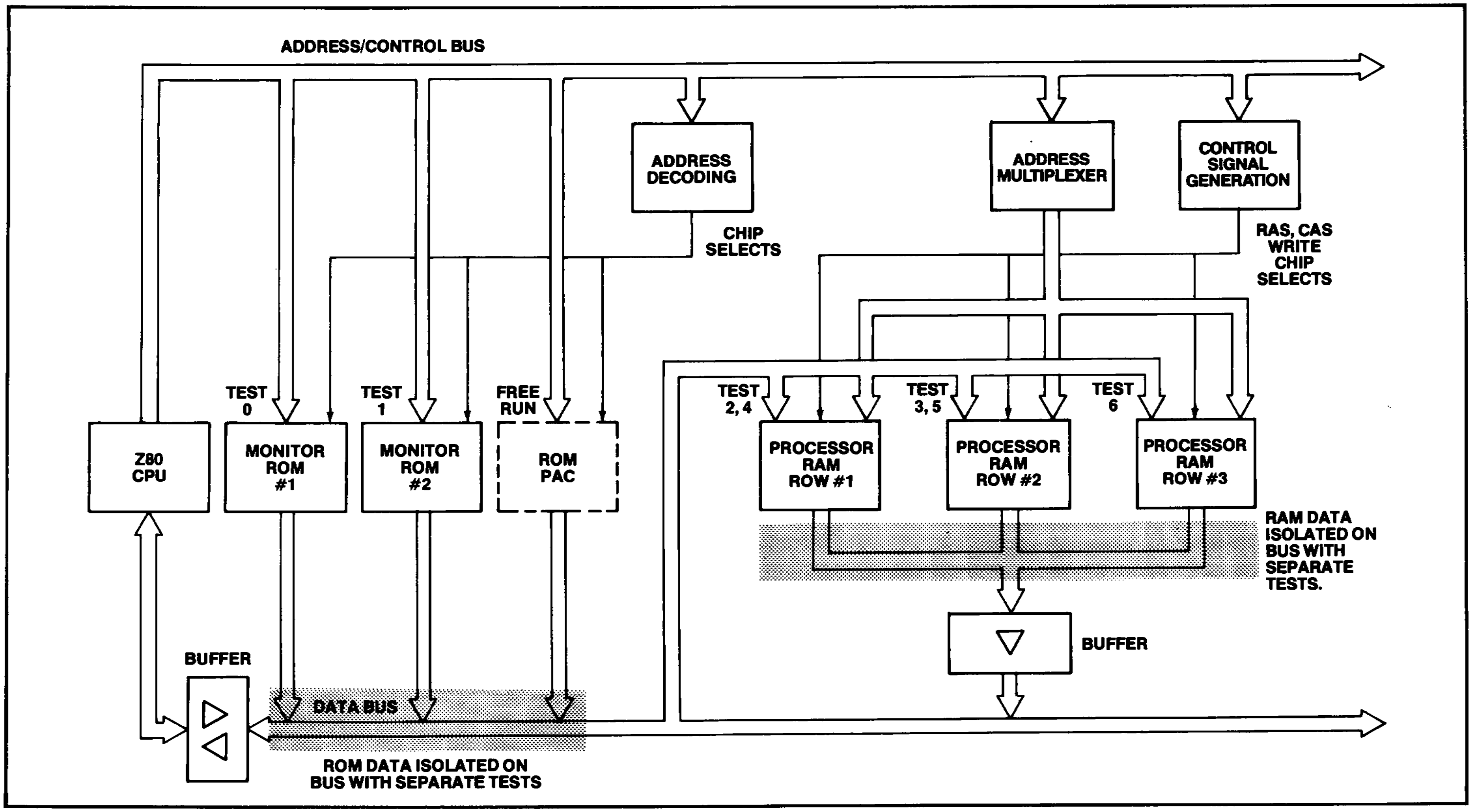


Figure 13. Each MONITOR ROM and each row of PROCESSOR RAM is tested separately to allow easier fault isolation of these bused devices. This ensures that the only data on any node comes from the device being tested and nothing else. (Except for the program that is executing from ROM PAC, which has to 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. $\overline{\text{RD}}$ and $\overline{\text{WR}}$ as CLOCKS also sort out faults. When incorrect signatures appear on the outputs of the RAM during a test using $\overline{\text{RD}}$ 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 $\overline{\text{SELECT}}$), or the data was incorrectly written into RAM. To isolate, $\overline{\text{WR}}$ 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

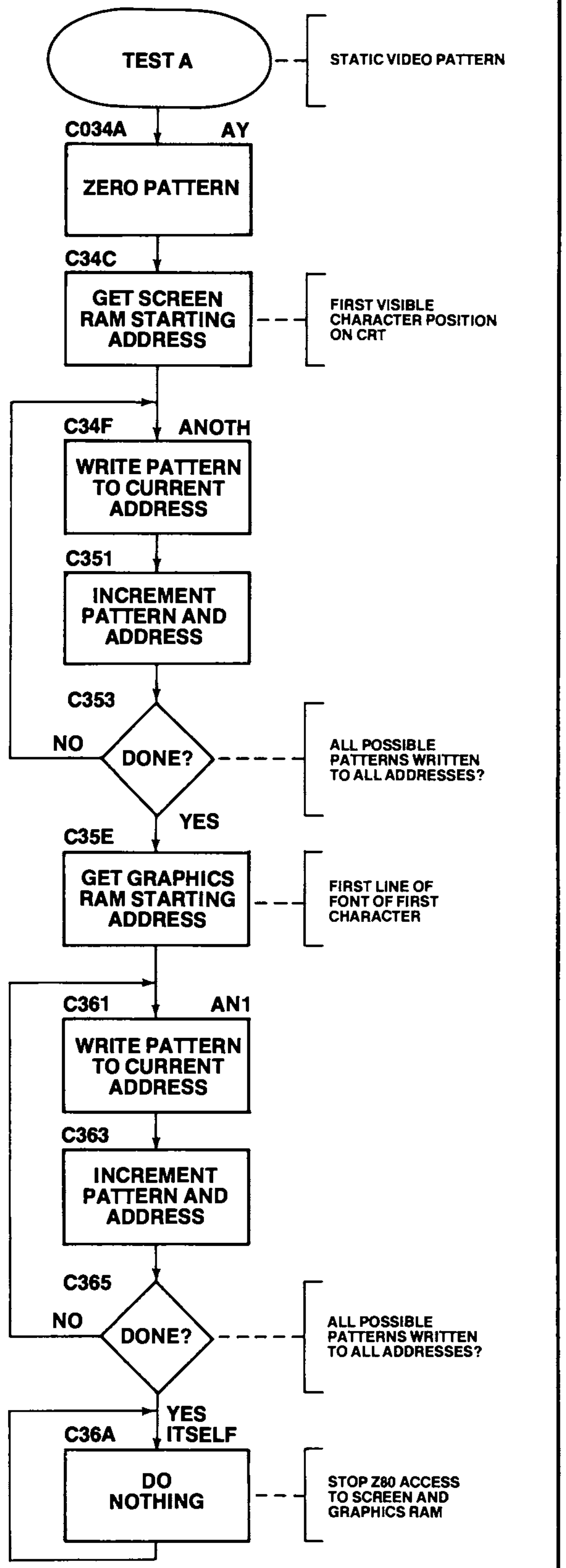

TEST A
STATIC VIDEO PATTERN
C034A
AY
ZERO PATTERN
C34C
GET SCREEN RAM STARTING ADDRESS
FIRST VISIBLE CHARACTER POSITION ON CRT
C34F
ANOTH
WRITE PATTERN TO CURRENT ADDRESS
C351
INCREMENT PATTERN AND ADDRESS
C353
NO
DONE?
ALL POSSIBLE PATTERNS WRITTEN TO ALL ADDRESSES?
YES
C35E
GET GRAPHICS RAM STARTING 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
C36A
ITSELF
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
```

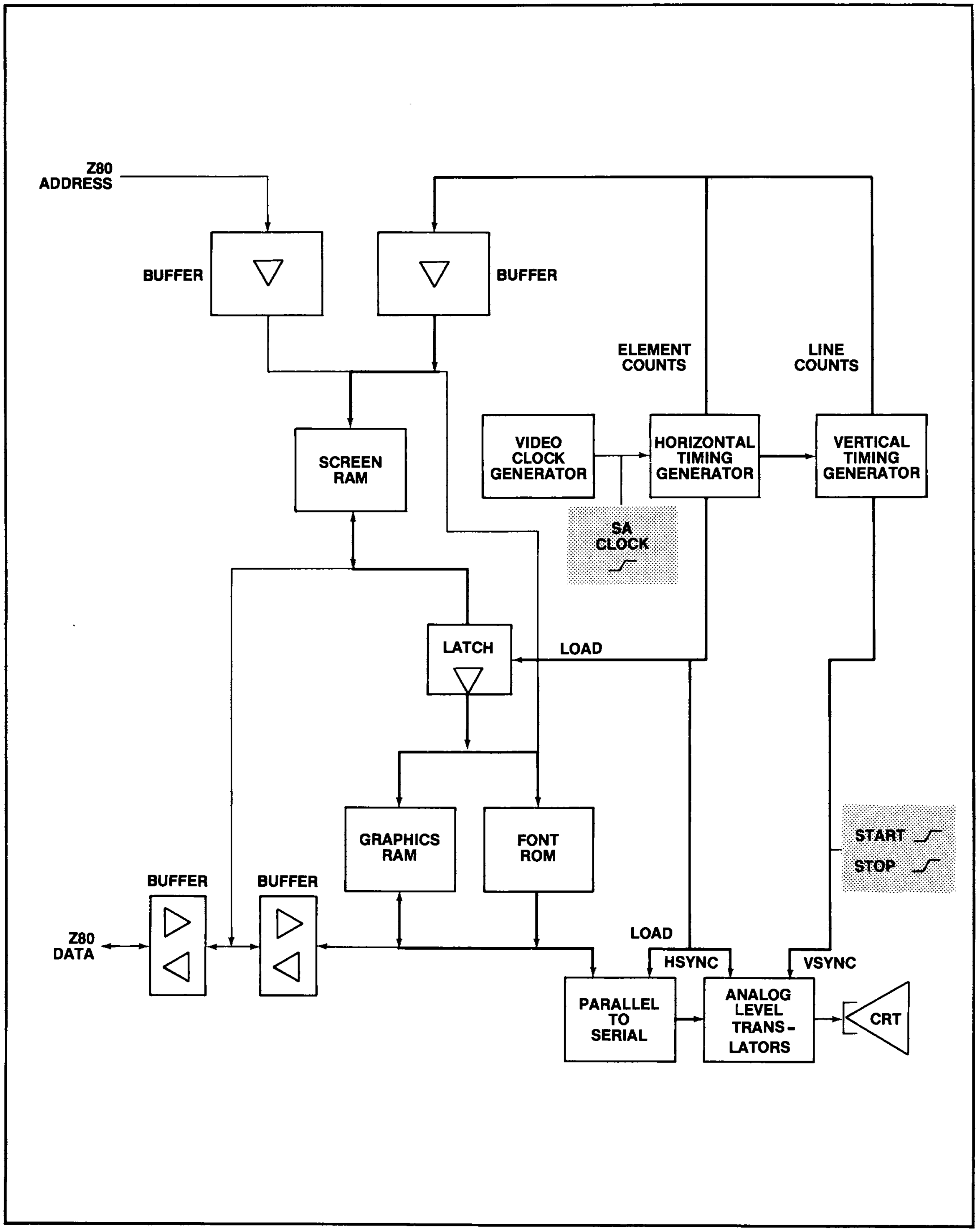


Figure 15. These nodes are stimulated by the CRT refresh process while the static video pattern is displayed on the CRT. These START, STOP and CLOCK connections form a GATE that is open for one complete refresh cycle of the CRT, allowing the signature analyzer to detect errors in any of the circuits 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.

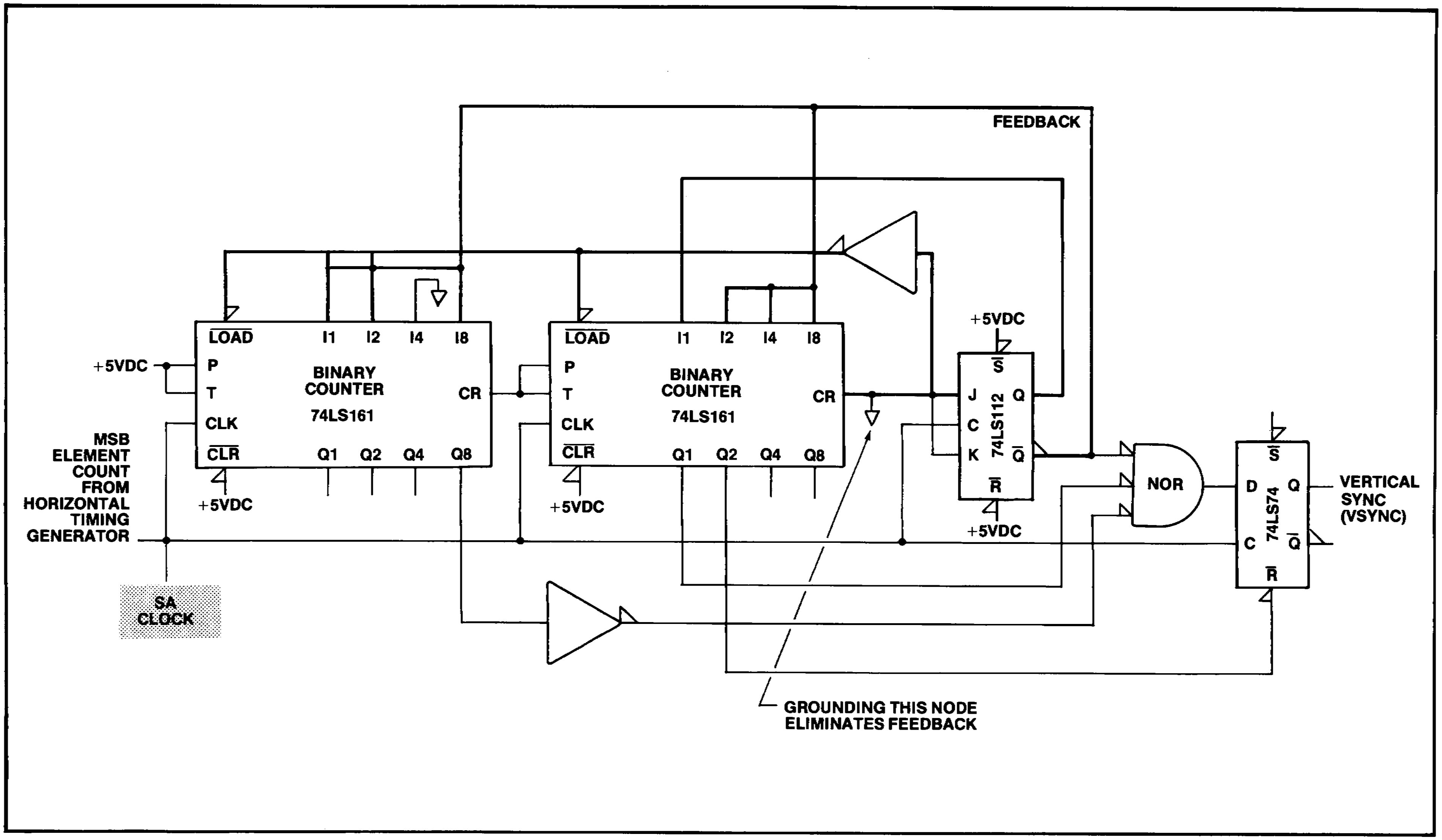


Figure 16. Grounding one point causes these VERTICAL TIMING GENERATOR counters to "FREERUN" through all possible states instead of counting CRT vertical lines. It does this by eliminating feedback loops. While grounding the output of a TTL device is not a recommended procedure, it was not possible to 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.

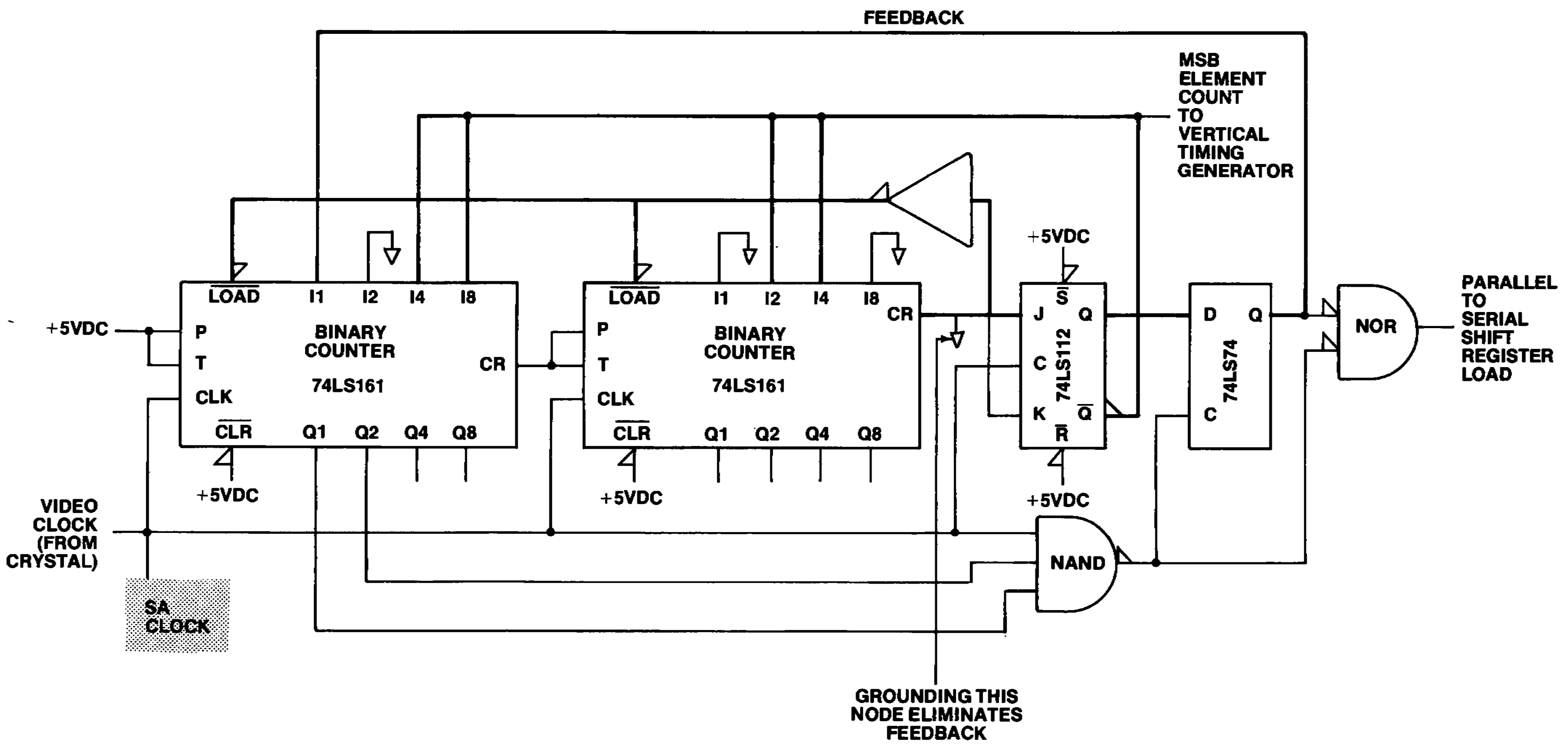
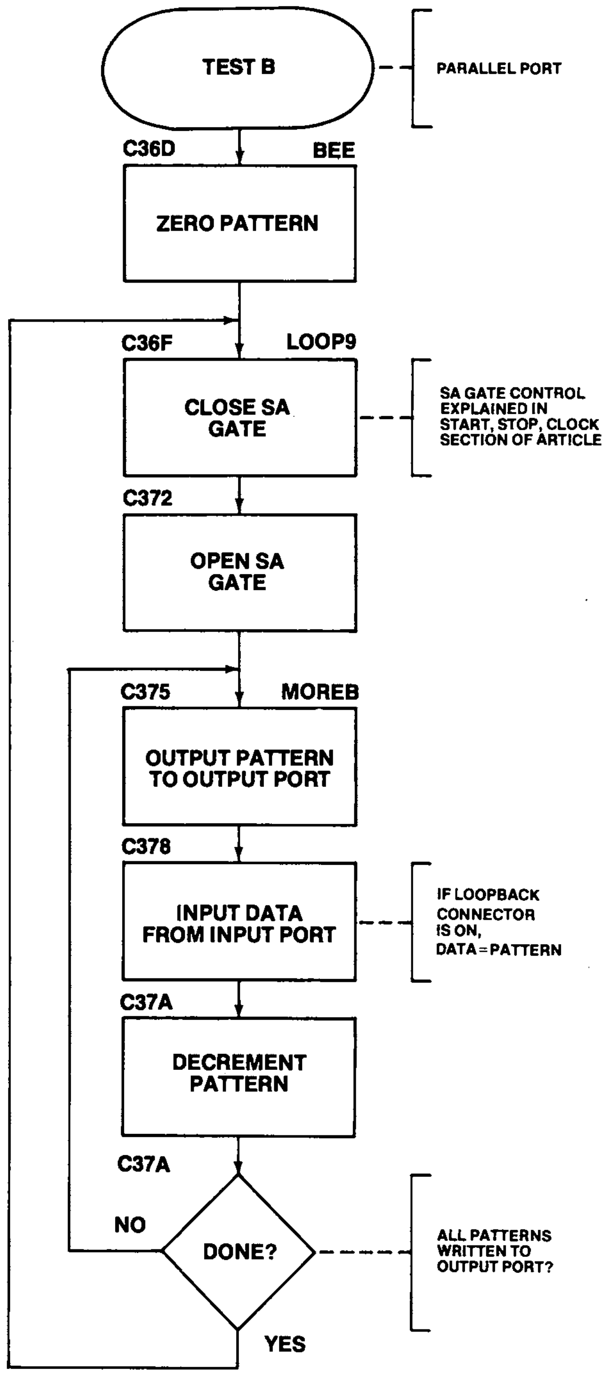

FEEDBACK
MSB ELEMENT COUNT TO VERTICAL TIMING GENERATOR
+5VDC
LOAD
I1
I2
I4
I8
P
T
CLK
CLR
BINARY COUNTER
74LS161
CR
Q1
Q2
Q4
Q8
+5VDC
J
C
K
S
R
Q
74LS112
D
Q
C
74LS74
NOR
PARALLEL TO SERIAL SHIFT REGISTER LOAD
NAND
VIDEO CLOCK (FROM CRYSTAL)
SA CLOCK
GROUNDING THIS NODE ELIMINATES FEEDBACK

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
```

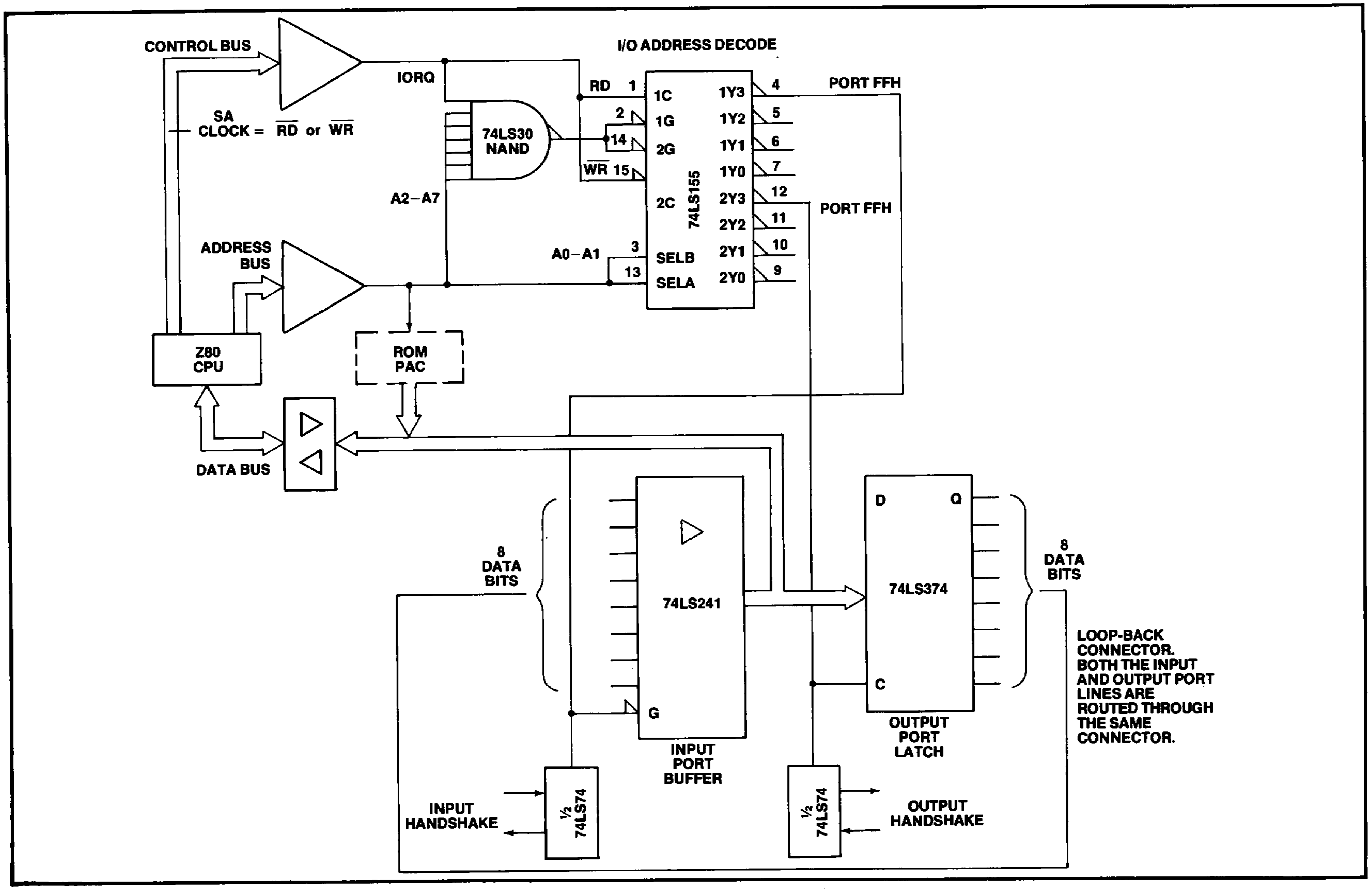


Figure 19. These circuits are stimulated by the PARALLEL PORT test. START and STOP are connected to a bit in the KEYBOARD SCAN LATCH and are controlled by the program as shown in the section called THE HARDWARE AND SOFTWARE BEHIND START, STOP AND CLOCK. With the loop-back connector on, signatures are 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

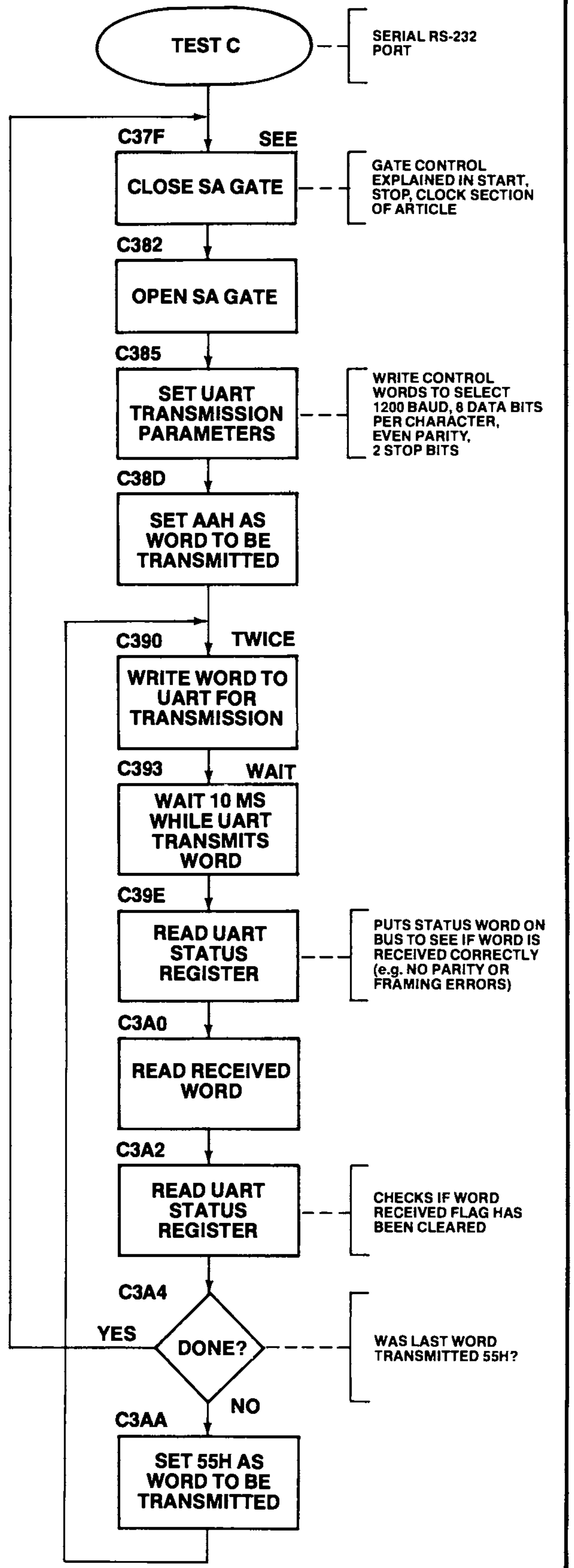
This test stimulates the UART for SA troubleshooting. UART's are generally considered to be test problems because of the lack of synchronization between the parallel side and the serial side of the part. However, the UART in this circuit (and in most applications) can be checked with SA as follows.

1. The serial output is connected to the serial input to take advantage of the loop-back technique for port testing, described in Section G. Figure 21 shows the UART loop-back configuration.

2. The stimulus program for Test C writes checkerboard patterns to the UART, then reads them back onto the data bus, allowing a fixed time for loop-back transmission of the serial words. This program is described in Figure 20.

3. The first test setup allows a go/no go check on the UART and support circuits. Connect START and STOP to the keyboard scan latch (explained in Figure 9). Connect CLOCK to $\overline{\text{RD}}$. Take signatures on data bus lines. If bus signatures are correct, then the UART and decoder are OK. If incorrect, then take signatures on the decoder outputs. If decoder signatures are incorrect, suspect the decoder. If correct, then set up the second test.

4. The second test setup allows verification of a UART chip failure. Connect START to the serial output line, TSO, which will open the GATE on the first serial output bit (the start bit). Connect STOP to the TBE output, which indicates the end of a serial output word. Connect CLOCK to the UART clock, pin TCP. Take signatures on the serial side of the UART for node-level fault isolation.


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?
WAS LAST WORD TRANSMITTED 55H?
NO
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.

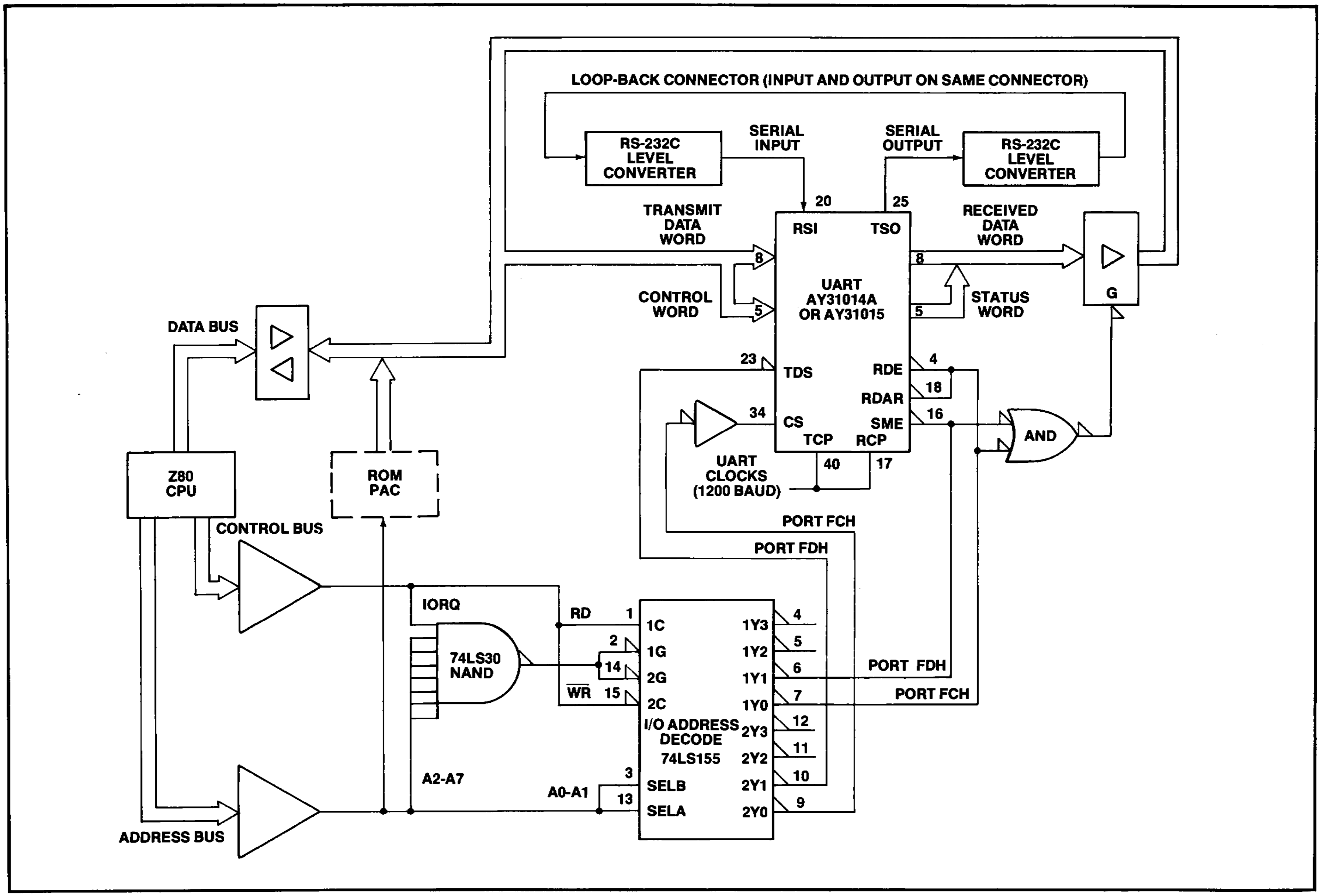


Figure 21. In Test C, the UART serial output is looped back into the serial input, as shown in this schematic. The stimulus exercises the entire loop. START, STOP and CLOCK connections for troubleshooting both the parallel and serial sides of the UART are outlined in the text.

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.

HEWLETT PACKARD

For more information, call your local HP Sales Office or nearest Regional Office: **Eastern** (201) 265-5000; **Midwestern** (312) 255-9800; **Southern** (404) 955-1500; **Western** (213) 970-7500; **Canadian** (416) 678-9430. Ask the operator for instrument sales. Or write Hewlett-Packard, 1501 Page Mill Road, Palo Alto, CA 94304. **In Europe**: Hewlett-Packard S.A., 7, rue du Bois-du-Lan, P.O. Box, CH 1217 Meyrin 2, Geneva, Switzerland. **In Japan**: Yokogawa-Hewlett-Packard Ltd., 29-21, Takaido-Higashi 3-chome, Suginami-ku, Tokyo 168.

02-5952-7591 OCTOBER 1980 Printed in U.S.A.