ADDITION OF BUILT-IN SELF-TESTING CAPABILITY TO THE INTEL SBC 80/10A SINGLE BOARD COMPUTER

by

CHEOW FATT YEO

B. Sc., Kansas State University, 1984

A MASTER THESIS

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

Kansas State University

Manhattan, Kansas

1986

Approved by:

Donald H. Lenhert

Major Professor

LD
2668
.T4
1986
Y46
c. 2

A11206 749467

Acknowledgements

I would like to sincerely thank my major advisor Dr. D. Lenhert for his friendship and guidance during my research work. I would also like to thank Dr. A. Pahwa and Dr. R. Greechie for serving as my committee members.

Finally, I would also like to thank my family for their support and encouragement during my studies at Kansas State University.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The continuous refinement of digital integrated circuit design and production techniques has made possible the development of microcomputer systems which exhibit functional performance characteristic that had not previously been possible to obtain using a single board. These same techniques which enable the evolution of complicated computer systems with such powerful capabilities have also created challenging opportunities for the test engineer to develop parallel techniques which will provide an adequate data base necessary to insure proper functional performance.

Innovations in computer test methodology in the last decade have come in response to increases in the power and complexity of these complicated computer systems (see Figure. 1), but even then new systems were becoming increasingly tough for tester to handle and programmers have to work hard to generate test exercises that could achieve an adequate level of test comprehensiveness.

Two common approaches are used to diagnose problems at board level [8]. The in-circuit method which isolates each component on the board and test it individually. In the event of failure, the tester will generate a component level diagnostic to assist in repairing the defect. In contrast, the functional test method operates the board as a unit and checks for discrepancies between actual and ideal behavior.

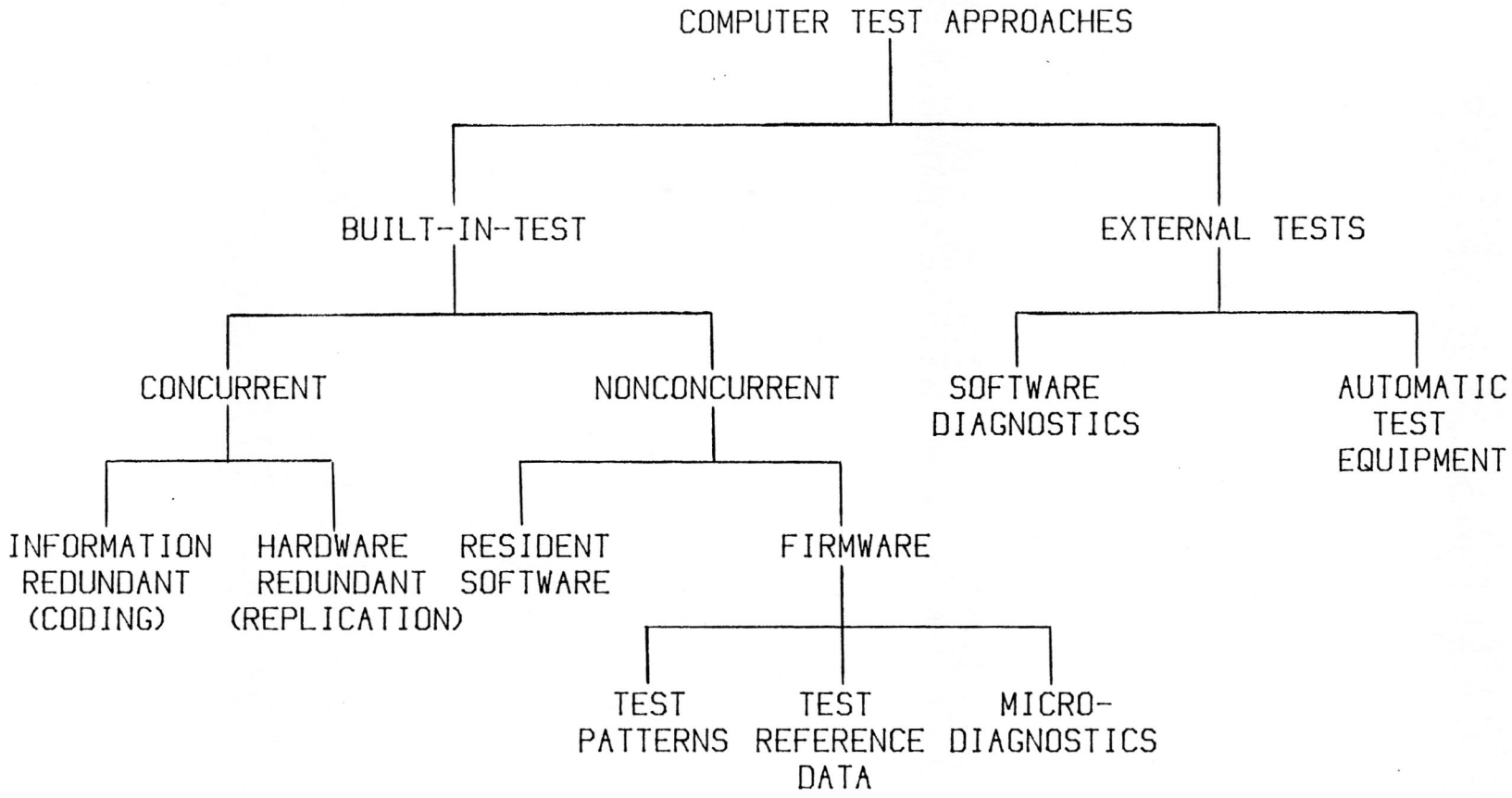


Figure 1. Computer Test Methodology

(Adapted From [11])

Both the in-circuit and functional test methods required external Automatic Test Equipment (ATE) to assist in the testing procedure.

Both in-circuit and functional testers suffer from a common problem [12]: the increasing difficulty of generating a comprehensive test problem due to the growing functional complexity of devices and board. The same complexity, however has provided the impetus for the emergence of new test techniques. Designers of the new boards also face the same complexity issues. Their response has been to impose a structure on their boards in order to make their design decisions manageable. Large scale integration (LSI)-based boards today are characterized by the increasingly pervasive microprocessor. The microprocessor allows much of the board's functionality to be defined in software, allowing designers to spend more effort on defining what that function is and less on its implemention.

Just as the increasing use of microprocessors is helping to structure the task of the board designer, so is it helping to structure the task of the test engineer. One increasingly common test approach that take advantage of this structure is the SELF-TEST, which is in essence the purpose of this project: to design and implement a BUILT-IN SELF-TESTING technique on the Intel SBC 80/10A single board computer . Upon execution or completion of the test, it would have some kind of indication (usually a GO/NO-GO

message) to indicate whether the board under test is good or bad.

2.0 REVIEW OF BUILT-IN-TEST

As computer performance has increased, testing problems have become more difficult because of increased system complexity and decreased circuit controllability and observability. Controllability and observability problems have long been the focus of struggles of the digital test engineer as the density of computer IC increased. New testing approaches have become necessary in many computer applications to provide on-line visibility into computer functional processes. A viable approach to increased on-line computer process visibility is the built-in-test [11].

In general, built-in-test refers to testing approaches which are an integral part of a particular computer design [11]. Built-in-test is therefore distinguished from external testing approaches which are executed when a computer is off-line using external automatic test equipment or software diagnostic loaded from a data storage medium (refer to Figure. 1).

Built-in-test approaches may be implemented in computer hardware, firmware or software. Some schemes can be executed during system run-time, concurrently with on-going functional operations or during convenient system idle time. In real-time applications, particularly where special-purpose processing is done, built-in-test is often implemented in hardware separate from functional computer resources to allow concurrent testing during normal computer

operation.

Concurrent built-in-test techniques are an important subcategory which may be generally classified as information redundant or hardware redundant. Information redundant built-in-tests include popular coding schemes such as parity, cyclic redundancy checks, error-correcting codes, etc. Hardware redundant built-in-test techniques range from self-testing circuits at the gate level, through duplication at the module level, to replicated computer at the system level.

Nonconcurrent built-in-test may conveniently exist in resident software which can be called by a computer operating system when needed. Software built-in-test is usually executed during times when normal functional processing is not being done. This approach to built-in-test is low-cost, since only minimum special purpose hardware is required [14].

2.1 BUILT-IN SELF-TESTING TECHNIQUE

Built-in self-testing is similar in definition to built-in-test and can also be implemented in computer hardware, firmware or software. As we know from section 2.0, built-in-test has such a broad definition that it really depends on what the applications are. We will have a somewhat restricted definition for the built-in self-testing

concept as used in this research or application. It is a test program written in the processor's own language to exercise the board's functions in order to verify that all essential parts are working correctly.

With this method, only minimum hardware is required to generate the large volume of stimulus data [14]. All it needs is some spare memory in ROM for storing the self-test program, some interconnecting logic and edge-connectors for setting up the test; see Section 5.0.

The self-test technique should be able to verify the working condition of the following parts of a computer system (Intel SBC 80/10A in this case):

1. Verify the basic operation of the address and data busses,
2. Verify operation of the central processing unit (CPU),
3. Verify the ROM(s) containing the system software,
4. Verify the ROM containing the self-test program,
5. Check the ability to read or write from RAM,
6. Perform tests to exercise the control logic,
7. Verify operation of the I/O ports (both serial and parallel),
8. Verify the external system bus (multibus interface).

Self-test can radically reduce the amount of ATE needed, because the GO/NO-GO indication can quickly identify

whether the Intel SBC 80/10A board is good or bad. It must be noted that the self-test technique verifies faults, it does not diagnose them. Diagnosis is not the scope of this research.

3.0 FUNCTIONAL DESCRIPTION OF THE INTEL SBC 80/10A BOARD

The Intel SBC 80/10A board is a complete computer system built on a single 6.75" by 12.00" printed circuit board (see Figure A-1 of Appendix A). The CPU, system clock, read/write memory, non-volatile read-only-memory, I/O ports and drivers, serial and parallel communication interface, bus control logic and drivers all reside on the board.

The circuitry on the Intel SBC 80/10A board can be divided into six functional blocks (Figure. 2):

1. CPU set,
2. System bus interface,
3. Random access memory (RAM),
4. Read-only-memory (ROM),
5. Serial I/O port,
6. Parallel I/O port.

3.1 CPU Set

The CPU set consists of the Intel 8080A central processor, the 8224 clock generator and the 8238 system controller (Figure 3). The CPU set is the brain of the Intel SBC 80/10A board. It performs all the system processing functions and provides a stable timing reference for all other circuitry in the system. The CPU set generates all of the address and control signals necessary to access

the memory and the I/O ports both on and external to the Intel SBC 80/10A board.

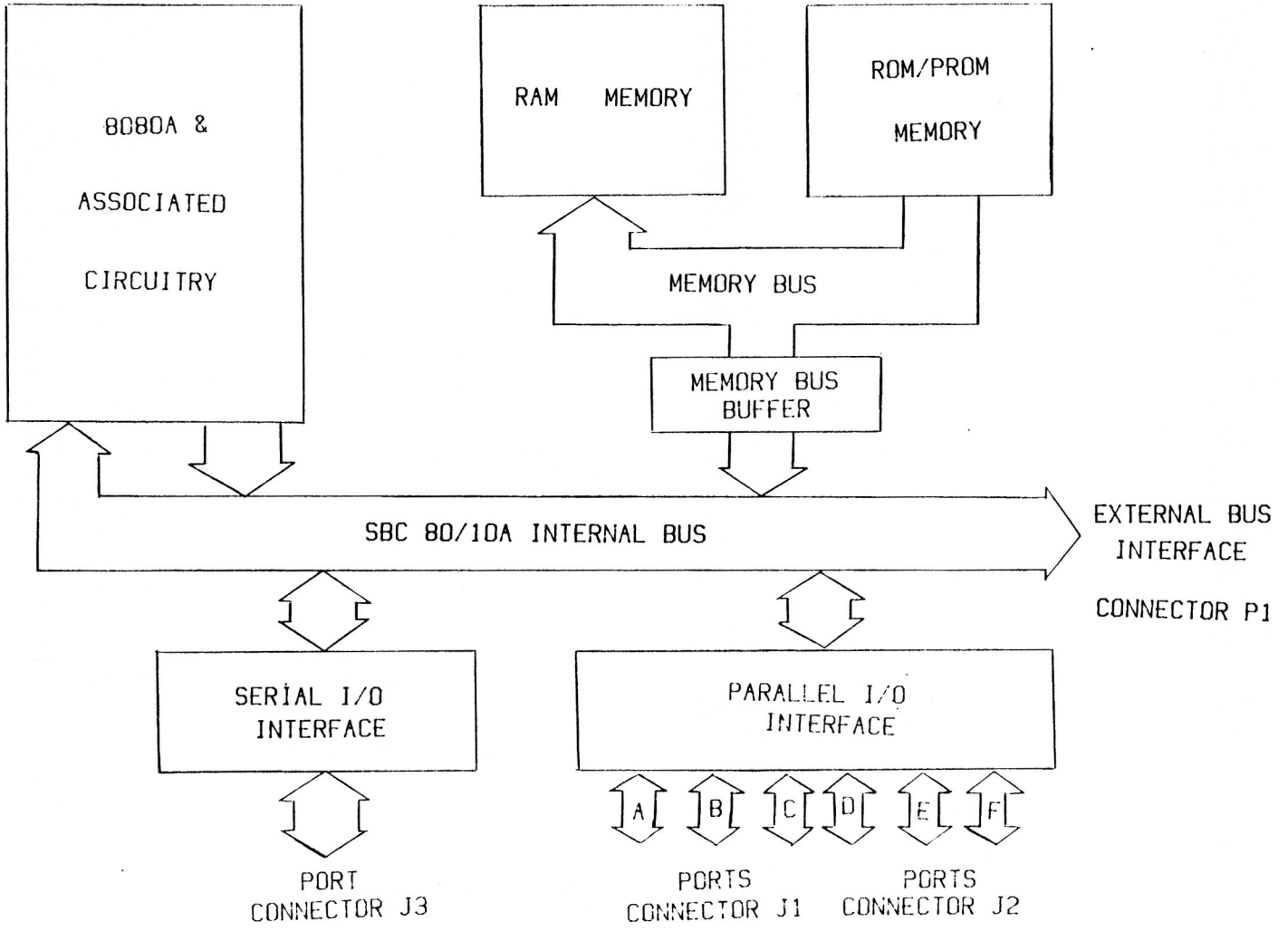


Figure 2. Functional Block Diagram of SBC 80/10A

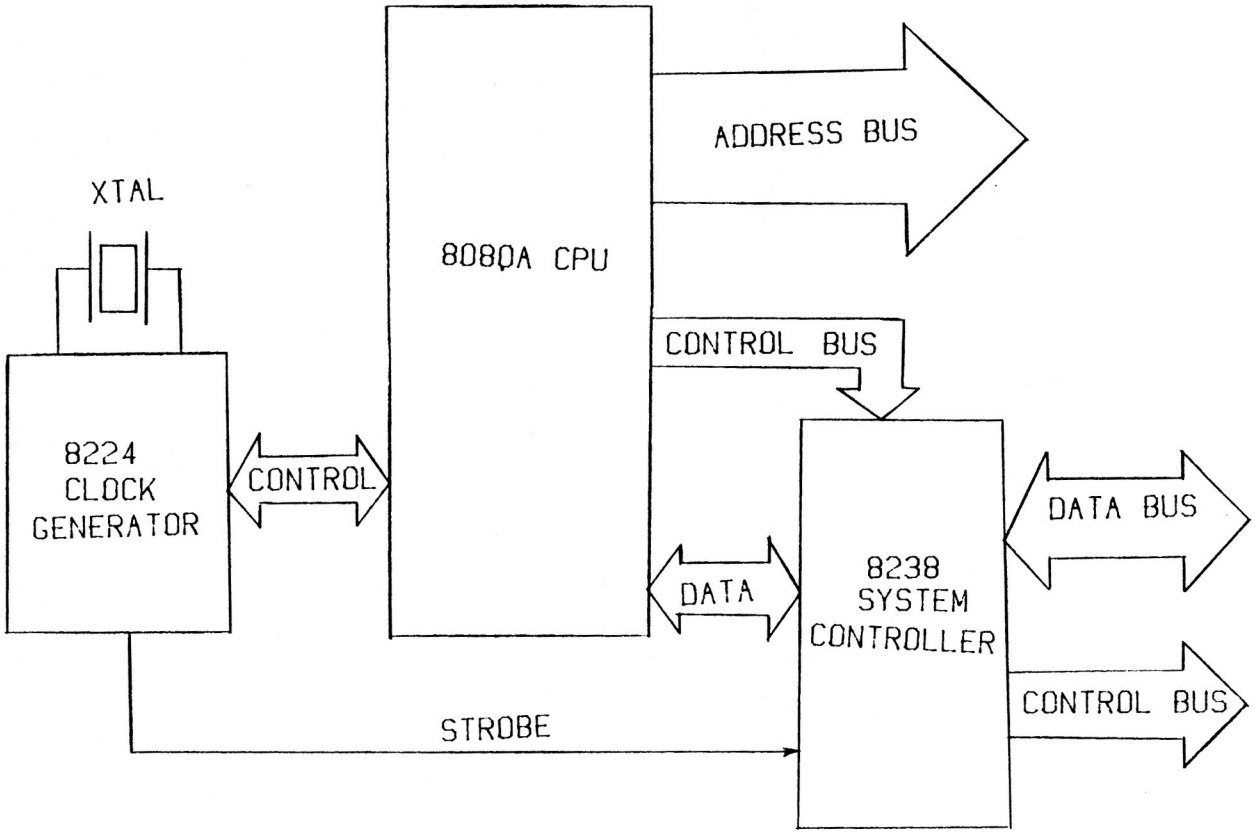


Figure 3. CPU Sets

(Adapted From [1])

3.2 The System Bus Interface

The system bus interface includes all those circuits that drive the various system control signals which also includes two Intel 8216 bi-directional bus drivers and six Intel 8226 devices which drive the memory data bus and the external system data and address busses, respectively.

3.3 The Random Access Memory (RAM)

It provides the Intel SBC 80/10A users with 1024 x 8 bits (1K) of onboard read/write storage, and is made up of eight Intel 8102 low power static RAMs (1024 x 1 bit each).

3.4 The Read-Only-Memory (ROM)

The Intel SBC 80/10A has four 24-pin sockets that can accept either Intel 8708 erasable and electrically reprogrammable ROM (EPROM) chips or Intel 8308 metal masked ROM chips, or Intel 2716 EPROM chips or Intel 2758 EPROM chips or Intel 2316 metal masked ROM chips. The total ROM/EPROM memory capacity using the 8708, 8308 or 2758 is 4K x 8 bits and 8K x 8 bits using the 2716 or 2316 chips.

3.5 The Serial I/O Interface

The Intel 8251 USART device provides a bi-directional serial data communication channel that can be programmed to operate with most of the current serial data transmission protocols: synchronous or asynchronous mode,

baud rate, character length, number of stop bits and the choice of even, odd or no parity are all program selectable.

3.6 The Parallel I/O Interface

Two Intel 8255 Programmable Peripheral interface devices provide 48 I/O lines for the transfer and control of data to or from external peripheral devices. Eight lines already have a bi-directional driver and termination network permanently installed. These bi-directional networks allow the eight lines to be configured as input, output or bi-directional and the remaining 40 lines have sockets provided for the installation of drivers or termination networks as required to meet the specific needs of the users ,see Section 4.3.1 on testing the parallel I/O ports.

4.0 TESTING OF THE INTEL SBC 80/10A SINGLE BOARD COMPUTER

The presence of the CPU on the Intel SBC 80/10A board allows it to exhibit self-testing capabilities (i.e., the intelligence of the microprocessor will be made use of since all the test programs are written in the processor language). All it needs is the addition of a self-test ROM containing all the necessary self-test software, some interfacing logic and connectors. This self-test software can be executed on a regular basis, such as each time the single board computer is powered up. In this application, basic tests such as continuity, power shorts, signal shorts, address and data pins stuck high or low, I/O pins stuck high or low, presence of clock signal, etc., would be performed. After the tests, the self-test routine would give a GO/NO-GO indication as to whether the Intel SBC 80/10A board is good or bad.

The remaining portion of this section will be devoted to detailing how the different parts of the Intel SBC 80/10A board are tested.

The minimum parts of the Intel SBC 80/10A board that are to be tested are:

1. onboard microprocessor,
2. onboard RAM,
3. onboard ROM,
4. address and data lines,

5. serial and parallel I/O ports,

6. external system bus (multibus interface).

4.1 Testing the Onboard Microprocessor

The Intel SBC 80/10A board utilized the 8080A single- chip, N-channel microprocessor for all its system processing functions (example, arithmetic and logic calculations).

The 8080A is a complete 8-bit parallel central processing unit (CPU). It contains six 8-bit general-purpose working registers and an accumulator. The six general-purpose registers may be addressed individually or in pairs, providing both single and double precision operations. The 8080A microprocessor also has an external stack feature wherein any portion of memory may be used as a last in/first out stack to store/retrieve the contents of the accumulator, flags, program counter and all of the six general-purpose registers. The 16-bit stack pointer controls the addressing of this external stack. A brief block diagram of the 8080A is shown in Figure 4.

Most of the test programs are influenced by the internal structure of the microprocessor. Some aim to exercise each functional block in the microprocessor, whereas others aim to test the device pin for stuck-at-0 and stuck-at-1 faults, and still others aim to test only the instruction sets [5]. The structural approach normally

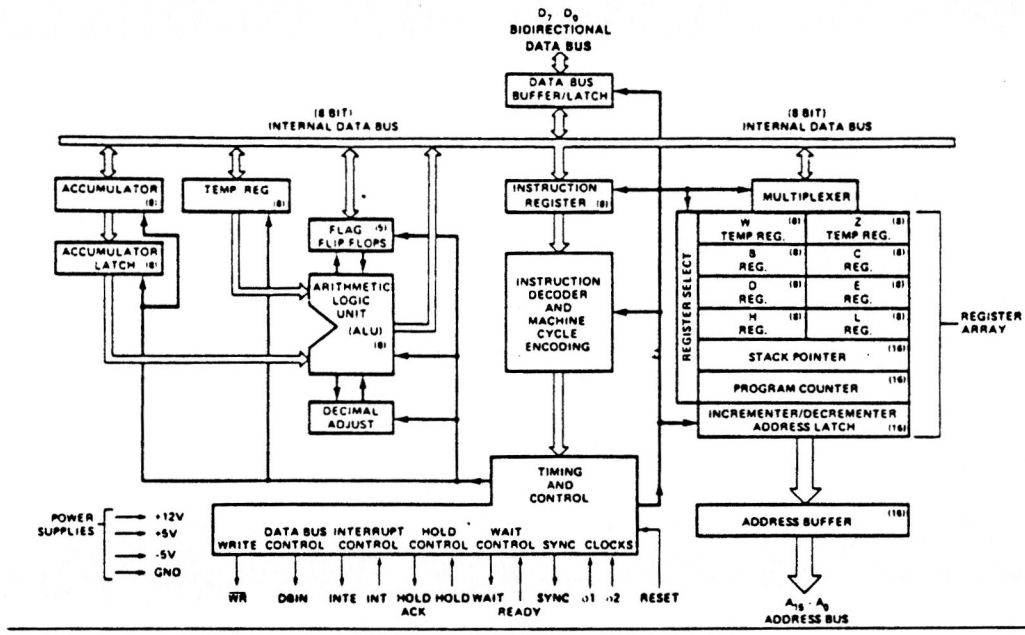


Figure 4. 8080A Functional Block Diagram

(Adapted From [29])

requires a detailed logical description of the microprocessor and this information is usually not available to the user. Even if we know the details of the circuit realization, an enormous amount of computation is needed to generate tests at the gate level, due to the large numberof gates on the chip.However, testing of the instruction sets will usually exercise both the structure and most of the pins on the microprocessor. Hence the latter method will be used to test the onboard microprocessor.

There are two possibilities for carrying out the microprocessor test [3]: either the open loop testing or the closed loop testing. In the former case, the instructions come from the tester in a sequence completely under external tester (ATE) control, whereas in the latter case, the microprocessor itself determines the instructions sequencing.

Since we are developing a program for built-in self-testing, the microprocessor can only be executed in a closed loop manner and without an external tester.

One of the most interesting methods for testing a microprocessor, particularly in the user testing environment, was suggested by Brahme and Abraham in [24] and then modified by Abraham and Parker in [23]. This method is easy to set up and we only need to know the register and instruction sets of the microprocessor. Only certains parts of the methods described in [23] and [24] were used for our

testing purposes.

The test designed for the microprocessor has the following purposes:

1. detect stuck-at-0 and stuck-at-1 faults on the microprocessor address and data lines,
2. detect stuck-at-0 and stuck-at-1 faults in the registers,
3. detect stuck-at-0 and stuck-at-1 faults in the Arithmetic and Logic unit,
4. detect stuck-at-0 and stuck-at-1 faults in its supporting circuitry.

4.1.1 Generation of test sequence

The microprocessor instruction sets were divided into four main categories:

1. Data transfer group (example, MOV R_1 R_2, MVI R_1 #, LXI R_1 address, etc.),
2. Arithmetic and logical group (example, ADD, SUB, DCR, ORA, XRA, etc.),
3. Branch control group (example, JMP, JNZ, CNZ, etc.),
4. I/O group (example, IN, OUT, etc.).

Under a fault in a microprocessor , either or both of the following may happen [23] :

1. an instruction is not executed,

2. the information stored in the microprocessor is incorrectly modified. The incorrect modification of the storage elements or registers will result in their being :

   a. reset to zero,
   b. set to one,
   c. overwritten with the contents of other registers (with the AND or OR function of the registers),
   d. exchanged,
   e. complemented.

At any time during execution of the self-test program, we assume the presence of any number of faults but only in one function described above. This assumption can be justified in our case because of the locality of failures and the frequency of tests. However, even if this assumption is not justified, it is unlikely that a test based on it will fail to detect a fault affecting two functions. Of course, if the additional instruction does not cause anything to be incorrectly modified in the microprocessor, then the fault is undetectable. However, as long as we can detect the faulty behavior (as a result of erroneous data), we really do not care whether it was caused by the execution of another extra instruction or by accessing a register or

by the AND or OR function of any register pair.

4.1.2 Test procedure

To test for the faults just described, we merely have to execute each instruction, make sure that it was correctly executed and none of the internal storage/registers was incorrectly modified.

Note that we have to execute other instructions in order to read the internal storage/registers and these instructions may themselves be faulty. The result may be that the correct (expected) data may be seen at the output pins as a result of interaction between faults. In order to ensure that such fault-masking does not exist, we will apply the tests in the following order [3].

Microprocessor Test Procedures

A. Test the reading-out and the loading of the internal registers,

B. Test as many of the "frequently used" instructions as possible.

The above tests will first ensure that the instructions can be executed correctly and that they do not destroy the information in the microprocessor.

Since the registers have to be read out after each instruction, we have to first ensure that all registers can be read out correctly and that the reading-out process

does not cause information to be changed inside the microprocessor. This test can be implemented with the data transfer group instruction.

We start with loading different code words [24] in each register. The property that the code words should satisfy is that any faults (AND or OR function of two registers) operating on code words should produce a non-code word.One of the commonly used codes is the d-out-of-w code. This code word will ensure that the AND or OR function of any set of registers will not be the same as the valid code word.

We illustrate the use of code word with the available 8080A data :

N = # of register = 7

w = width of register in bit = 8

$d \geq 1$

$$\frac{w!}{d!\,(w-d)!} \geq N$$

With the above information and formula, we could use a 1-out-of-8 code. The code words are listed in Table 1.

Table 1. Code Word Assignment to Registers for Test Procedure A

Register	Bit Number 0 1 2 3 4 5 6 7
A	0 1 1 1 1 1 1 1
B	1 0 1 1 1 1 1 1
C	1 1 0 1 1 1 1 1
D	1 1 1 0 1 1 1 1
E	1 1 1 1 0 1 1 1
H	1 1 1 1 1 0 1 1
L	1 1 1 1 1 1 0 1

Verification of Faults

CASE 1. Consider the fault if register A is AND with register C.

R_A:01111111 AND R_C:11011111 ---> R_A:01011111

CASE 2. Consider the fault if register D is OR with register H.

R_D:11101111 OR R_H:11111011 ---> R_D:11111111

and so on.

Microprocessor Test Procedure A (Register Check Test)

1. Load different code words into each register.
2. For i = 0 to 6 do

   For j = 0 to 6 do

   begin

   Read and compare (i)
   Read and compare (j)
   Read and compare (i)
   Read and compare (j)

   end

The error will be detected when the registers are checked against their corresponding code words during the test. If step 1 does not load the registers correctly, we will detect it when the registers are read out. The two reads are necessary to ensure that reading a register does not change the content of another register.

Note that with the above test, if two adjacent cells of a register are bridged, then the fault will not be detected and will probably be propagated to other tests. In order to avoid this undetected fault, a marching 1's or 0's test pattern can be used as the code word, i.e., the rotation of a 1 or 0 around the register. Hence an extra step, rotate the register to the right, will be inserted after step 2.

With the 8080A instruction sets only the accumulator can be read out directly, other registers (B, C, D, E, H and L) have to be read out through the accumulator. Again we have to ensure that the internal transfer instructions do not cause the code words to be modified in other registers. The test conducted will therefore be in stages, with the register which can be read out directly tested first, followed by the registers which can only be read out in two steps.

The above would utilize about 80% of the data transfer group instructions and 10% of the other groups (example, CMP, CPI, RAL, RAR, etc., of the arithmetic and

logical group, and JMP, CALL, CNZ, CZ, etc., of the branch control group).

Once the register read test is completed, we can be sure that all the registers can be safely read out. Although the program counter and the stack pointer are not directly tested, nevertheless it must be pointed out that the correct execution of the test program verifies indirectly or implicitly, at least to some extent, that these registers are functioning properly.

An exhaustive test of all the 8080A instructions is practically impossible because it might make the test several times longer than it should be. Hence we must choose to test only the most frequently used instructions. A list of the frequently used instructions can be found in Table 3-1, page 3-5 in [17]. Of course the term "frequently used" might be subject to different interpretations depending on the kind of application that the 8080A microprocessor is used for, but a lot of the "frequently used" instructions described here constitute at least 90% of any software package [17].

The group of instructions used in microprocessor Test Procedure A is used to test the remainder of the instruction set, each tested instruction is then added to this group, and so on. When testing the other instructions, it is desirable to test the most "reliable" instructions first [5], so that the above group only contains the most "reliable" instructions. One definition of "reliable" is given by [30] and depends upon the minimum sum of the number of machine cycles taken to execute the instruction.

Microprocessor Test Procedure B (other instructions Test)

1. ADD and ADC instructions

   (Initial Code Words for Registers)

   A = 10000000

   B = 01000000

   C = 00100000

   D = 00010000

   E = 00001000

   H = 00000100

   L = 00000010

   Test Sequence

   1. STC, ADC B, ADD C, ADD D, ADD E, ADD H, ADD L, CPI FF.
   2. STC, ADD B, ADC C, ADD D, ADD E, ADD H, ADD L, CMA, CPI 00.

3. STC, ADD B, ADD C, ADC D, ADD E, ADD H, ADD L, CPI FF.
4. STC, ADD B, ADD C, ADD D, ADC E, ADD H, ADD L, CMA, CPI 00.
5. STC, ADD B, ADD C, ADD D, ADD E, ADC H, ADD L, CPI FF,
6. STC, ADD B, ADD C, ADD D, ADD E, ADD H, ADC L, CMA, CPI 00.

2. SUB and SBB instructions

(Initial Code Words for Registers)

A = 01111111

B = 01000000

C = 00100000

D = 00010000

E = 00001000

H = 00000100

L = 00000010

Test Sequence

1. STC, SBB B, SUB C, SUB D, SUB E, SUB H, SUB L, CMA, CPI FF.
2. STC, SUB B, SBB C, SUB D, SUB E, SUB H, SUB L, CPI 00.
3. STC, SUB B, SUB C, SBB D, SUB E, SUB H, SUB L, CMA, CPI FF.
4. STC, SUB B, SUB C, SUB D, SBB E, SUB H, SUB L,

CPI 00.

5. STC, SUB B, SUB C, SUB D, SUB E, SBB H, SUB L, CMA, CPI FF.
6. STC, SUB B, SUB C, SUB D, SUB E, SUB H, SBB L, CPI 00.

3. ANA instruction

(Initial Code Words for Registers)

A = 01111111

B = 10111111

C = 11011111

D = 11101111

E = 11110111

H = 11111011

L = 11111101

Test Sequence

1. ANA B, ANA C, ANA D, ANA E, ANA H, ANA L, ANI 00, CPI 00, CMA, CPI FF.

4. ORA instruction

(Initial Code Words for Registers)

A = 10000000

B = 01000000

C = 00100000

D = 00010000

E = 00001000

H = 00000100

L = 00000010

Test Sequence

1. ORA B, ORA C, ORA D, ORA E, ORA H, ORA L, ORI 01, CPI FF, CMA, CPI 0.

5. XRA instruction

(Initial Code Words for Registers)

A = 01111111

B = 10111111

C = 11011111

D = 11101111

E = 11110111

H = 11111011

L = 11111101

Test Sequence

1. XRA B, XRA C, XRA D, XRA E, XRA H, XRA L, XRI 01, CPI FF, CMA, CPI 00.

Finally, other instructions which are not used in testing the microprocessor are used extensively in other modules tests. These microprocessor tests together with other modules tests have utilized at least 80% of the microprocessor instruction sets. However, if the remaining 20% untested instructions were in fact never used at all, (depending on the applications), then this 80% instructions

coverage should be considered excellent, if not optimal.

4.2 Testing the Onboard Memory

The purpose of this test is two-fold:

1. detect stuck-at-0 and stuck-at-1 faults on data in/data out lines of the RAM chip,
2. detect stuck-at-0 and stuck-at-1 on the address lines,
3. detect stuck-at-0 and stuck-at-1 fault in the row and column (address) decoder of the RAM chip,
4. detect stuck-at-0 and stuck-at-1 fault in the storage elements,
5. to verify that the supporting circuits (interfacing logic), i.e., all logic that ties the memory and the processor together, are functioning properly.

4.2.1 Testing the RAM (address $3C00 - $3FFF)

This is accomplished by initializing the address bus to the starting address of the RAM ($3C00). Each location within the RAM is then sequentially written into and after the memory is fully written into, it is read and the information or pattern being read from each location is compared with the information written into it

earlier. Thus the RAM is verified. In the following section, we will develop an algorithm for testing the RAM. The thoroughness of this test is directly related to the complexity of the patterns stored and read from each location.

4.2.1.1 Generation of test sequence

In the built-in-test structure described in [26] and [27], the authors present a test algorithm which provides a minimal length test for stuck-at-fault in RAM. The test algorithm requires four accesses per addressline, with additional hardware to access the RAM.

The algorithm (named ATS) divides the address into groups of three (modulo-3) and uses the result (0, 1 or 2) to assign each address to one of the three groups GP_0, GP_1 and GP_2. The test itself consists of writing and reading the memory in the manner shown in Table 2.

Table 2. Algorithm Test Sequence (ATS)

STEP \ PART.	GP_0	GP_1	GP_2
1		WrW_0	WrW_0
2	WrW_1		
3		RdW_0	
4		WrW_1	
5			RdW_0
6	RdW_1	RdW_1	
7	WrW_0 RdW_0		
8			WrW_1 RdW_1

W_0 = test data 01010101 Wr = write

W_1 = test data 10101010 Rd = read

This test algorithm is shown to detect al single and multiple stuck-at-faults in memory addres register (MAR), memory data register (MDR), decoder an memory array. A summarized proof that the ATS detects thes faults will be considered later. For a detailed proof, th reader is referred to [26] and [27].

Derivation of Test Algorithm

Let A_j be the memory address of location j, $0 \leqslant j < 2^n$, n = # of address lines (i.e., width of address lines).

$GP_0 = \{ A_j | j = 0 \text{ (modulo-3)} \}$

$GP_1 = \{ A_j | j = 1 \text{ (modulo-3)} \}$

$GP_2 = \{ A_j | j = 2 \text{ (modulo-3)} \}$

Refer to Table 2

Step 1: Write the W_0 word at all locations

$A_j \in GP_1$ and $A_k \in GP_2$

Step 2: Write the W_1 word at all locations

$A_i \in GP_0$

Step 3: Read all locations $A_j \in GP_1$

if $_{output} = W_0$; no fault indicated

$\neq W_0$; RAM fault indicated

Step 4: Write the W_1 word at all locations

$A_j \in GP_1$

Step 5: Read all locations $A_k \in GP_2$

if $_{output} = W_0$; no fault indicated

$\neq W_0$; RAM fault indicated

Step 6: Read all locations $A_i \in GP_0$ and $A_j \in GP_1$

if $_{output} = W_1$; no fault indicated

$\neq W_1$; RAM fault indicated

Step7: Write and then read the W_0 word at locations $A_i \in GP_0$

if $_{output} = W_0$; no fault indicated

$\neq W_0$; RAM fault indicated

Step 8: Write and then read the W_1 word at all locations $A_k \in GP_2$

if $_{output} = W_1$; no fault indicated

$\neq W_1$; RAM fault indicated

Verification of Faults

Definition:

Hamming distance - it is the number of binary bits in which two code word or binary memory addresses disagree [26].

Theorem 1:

If A_x and A_y are two distinct binary memory addresses which belong to the same group GP_i, $0 \leqslant i \leqslant 2$, then the Hamming distance, denoted $d(A_x,A_y)$, is at least $\geqslant 2$, [26] and [27].

MAR Faults (Assuming all other sub-units are fault-free)

Assume that the kind of fault caused by the MAR s-a-0 or s-a-1 is the single accessing of the wrong memory array word. Therefore, a faulty bit in the MAR can cause an access to any address that is of Hamming distance 1 from the desired memory location. Therefore the test algorithm required is to write W_0 in one location and W_1 in all locations that are Hamming distance 1 away. This is done many times over in steps 1 through 5 of the ATS algorithm.

Analysis of Fault Using the ATS Algorithm (Refer to Table 2)

Step 1: Write the W_0 word in all locations belonging to GP_1 and GP_2

Step 2: Write the W_1 word in all locations belonging to GP_0

Step 3: Read W_0 at all locations in GP_1; this will detect all single MAR faults between all addresses in GP_0 and GP_1

Step 4: Write the W_1 word at all locations in GP_1

Step 5: Read the W_0 word at all locations in GP_2; this will detect all single MAR faults between all addresses in GP_2 and GP_1 and also between all addresses in GP_1 and GP_0, etc.

As a matter of fact, this algorithm will also detect multiple MAR faults (i.e., Hamming distance > 1).

Decoder Faults (Assume all other sub-units are fault-free)

Assumption:

1. An address accessing two or more locations will result in the wired-ORing of these locations in the MDR or the data bus,
2. Inputs of decoder are fault-free. Since faults in the input lines of the decoder are indistinguishable from faults in the MAR and any address always accesses some

the ATS algorithm. Thus, we can at this point consider the inputs of the decoder to be fault-free.

3. Decoder circuit is combinational and remains so in the event of multiple stuck-at faults.

Let us illustrate the decoder input/selected memory array word line.

Table 3. Decoder Fault-free Table

LOC. / ADDR.	L_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7
A_0	1	0	0	0	0	0	0	0
A_1	0	1	0	0	0	0	0	0
A_2	0	0	1	0	0	0	0	0
A_3	0	0	0	1	0	0	0	0
A_4	0	0	0	0	1	0	0	0
A_5	0	0	0	0	0	1	0	0
A_6	0	0	0	0	0	0	1	0
A_7	0	0	0	0	0	0	0	1

The following shows the equivalent decoder input/selected memory array.

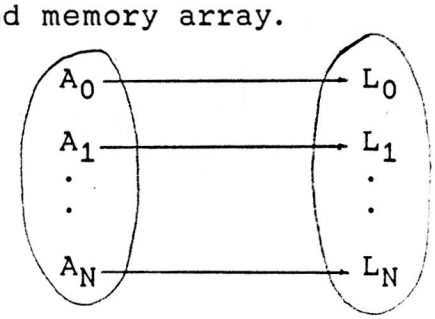


Figure 5. Equivalent Fault-free Diagram

There are 2^{N^2} possible tables (N = number of words in memory), of which N! tables have exactly one 1 in every row and every column. Since these correspond to the cases where each address accesses a unique location and every location is accessed, they can be ignored because the functioning of the memory is not affected by such a fault in the decoder, see Table 4. Table 5 shows a typical fault model for a decoder.

Table 4. Decoder Undetectable Fault

LOC. / ADDR.	L_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7
A_0	0	0	1	0	0	0	0	0
A_1	0	0	0	0	0	0	1	0
A_2	0	0	0	0	1	0	0	0
A_3	0	0	0	0	0	1	0	0
A_4	0	0	0	0	0	0	0	1
A_5	0	0	0	1	0	0	0	0
A_6	1	0	0	0	0	0	0	0
A_7	0	1	0	0	0	0	0	0

Table 5. Decoder Stuck-at-Faults

LOC. / ADDR.	L_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7
A_0	0	1	0	0	0	0	0	0
A_1	0	1	0	0	0	0	0	0
A_2	0	0	0	0	0	0	0	0
A_3	0	0	0	1	0	0	0	0
A_4	0	0	0	0	0	1	0	0
A_5	0	0	0	0	0	1	0	0
A_6	0	0	0	0	0	0	0	0
A_7	0	0	0	0	0	0	0	1

The following is an equivalent diagram of the above decoder stuck-at-fault.

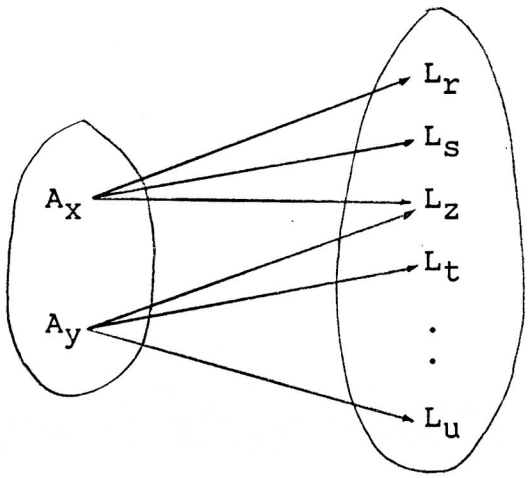


Figure 6. General Fault Model for Decoder

For the above model, there are two kinds of faults :

1. an overwrite condition where the initially stored information is destroyed by a wrong memory address access during input,
2. a logical ORing condition where the stored information is erroneously output due to a multiple memory access,

Analysis of Fault Using the ATS Algorithm (Refer to Table 2)

The following will show how the algorithm in Table 2 detects the decoder stuck-at-fault.

Case 1: $A_x \in GP_0$, then $A_y \in GP_2 \cup GP_1$ by Theorem 1.

If $A_y \in GP_1$, fault will be detected in step 3 due to reading W_1 instead of W_0 (i.e., A_x <-->

A_y),

If $A_y \in GP_2$, then fault will be detected in step 5 due to reading W_1 instead W_0,

Case 2: $A_x \in GP_1$, then $A_y \in GP_0 \cup GP_2$,

If $A_y \in GP_0$, fault will be detected in step 3,

If $A_y \in GP_2$, fault will be detected in step 5,

Case 3: $A_x \in GP_2$, then $A_y \in GP_0 \cup Gp_1$. Fault will be detected in step 5.

Memory Array Fault

Faults in the memory imply the various bit positions in some words are at some combination of s-a-1 and s-a-0 faults. Since the algorithm write and read both W_0 and W_1 in all memory locations, the test will also detect faults in the memory array.

Memory Data Register Fault

Faults in the MDR are very much the same as those in the memory array i.e., bit position s-a-0 and s-a-1 faults. Hence the MDR fault is also detected by the ATS algorithm.

4.2.1.2 Final algorithm for RAM

Referring to Table 2 again, [28] makes it clear that capabilities of the ATS algorithm remain unchanged if step 7 is moved to step 1 and step 8 to step 5

(not swapping). From the practical viewpoint, this algorithm seems to be more convenient to implement [28]. Further simplification of the present algorithm is still possible if we examine Theorem 1 again. This Theorem is valid for any partitioning as long as [28]:

$$GP_i = \{ A_j | j = i \text{ (modulo-k)} \} \; ; \; i = 0,1,\ldots k-1$$

$$j < \log N$$

$$N = \# \text{ of words in memory.}$$

Several values of k satisfy the above equation, but the case when k = N is worth noting.

The validity of the modified ATS algorithm just described does not depend on the number of partitions [28]; hence by increasing the number of partitions to N, we have the following final algorithm for the RAM as shown in Table 6.

Table 6. Final ATS Algorithm for RAM

STEP \ PART.	A_0	A_1	. . .	A_{N-2}	A_{N-1}
1	WrW_0				
2		WrW_0			
.					
.					
N-1				WrW_0	
N					WrW_0
N+1	RdW_0				
N+2	WrW_1				
N+3		RdW_0			
N+4		WrW_1			
.					
.					
3N-3				RdW_0	
3N-2				WrW_1	
3N-1					RdW_0
3N					WrW_1
3N+1	RdW_1				
3N+2		RdW_1			
.					
.					
4N-1				RdW_1	
4N					RdW_1

Comparing the first ATS algorithm and the final algorithm, we see that the latter algorithm can be implemented in a much simpler way than the former algorithm, which is complicated by the division of 3 scheme.

4.2.2 Testing the ROMs

The verification of the information that a ROM contains is commonly accomplished by sequentially reading each location and generating a 16-bit cumulative sum. After all locations have been read, the 16-bit checksum thus generated is a fairly reliable indicator [21] of the validity of the information contained in the ROM and of the ability of the CPU to access it. The entire system software occupies 1k bytes of memory space and its checksum is $BB18 (generated by PROMlink).

The checksum is formed by taking the sum of the consecutive memory data. The carry resulting from the summation of this consecutive memory data is taken care of in another register. Thus a 16-bit checksum is formed. The cumulative carry resulting from the LSByte addition forms the MSByte of the checksum. This is called the extended-precision checksum.

The self-test program is placed separately from the system program. The self-test program is placed in the ROM located at A23 (see Figure C-3, Appendix C). When the board is powered up, it will jump to address $0000

(beginning address of the ROM containing the self-test program). The system program is moved to ROM location A25 (address $800 - $FFF). The valid memory addresses are listed in Table 7.

TABLE 7. ROM/PROM Addresses
(Adapted From [1])

	CHIP ADDRESS			
	A23	A24	A25	A26
4K	0 - 3FF	400 - 7FF	800 - BFF	C00 - FFF
8K	0 - 7FF	1000 - 17FF	800 - FFF	1800 - 1FFF

4.3 Testing the I/O ports

There are three I/O ports (three parallel ports and one serial port) on the Intel SBC 80/10A board. The purpose of testing both these ports is two-fold:

1. to verify that the CPU can access these ports,
2. to verify that both the parallel and serial ports are functioning correctly,
3. to verify that there are no stuck-at-0 or stuck-at-1 faults in the data bits of the ports,
4. to verify that the supporting circuit, i.e., all logic that ties the I/O ports and the processor, is functioning properly.

4.3.1 Testing the parallel I/O port

Referring to Figure C-5 of Appendix C, we see that there are two Intel 8255 Programmable Peripheral Interfaces (PPI), one located at A19, the other at A20. For convenience the following device designations will be used: The device at A19 is called the "I/O group 1" device, while the device at A20 is referred to as the "I/O group 2" device. Each device has three eight-bit ports. The "I/O group 1" ports are designated as Ports A, B and C while the "I/O group 2" ports are designated as Ports D, E and F.

The parallel I/O interface logic on the Intel SBC 80/10A provides 48 signal lines for the transfer and control of data to or from external peripheral devices. All the 48 signal lines emanate from the I/O ports on two Intel 8255 PPI (Figure C-2), and are brought out of the Intel SBC 80/10A board via two 50-pin double-sided PC edge connectors (J1 and J2).

The Intel 8255 PPI contains three eight-bit ports (A, B and C); see Figure 7. All can be configured in a wide variety of functional characteristics by the system software and each has its own special features to further enhance its power and flexibility. The Intel 8255 PPI also allows for a wide variety of I/O configurations and its 24 I/O lines can be individually programmed in 2 groups of twelve (see Figure. 7) and used in three major modes of

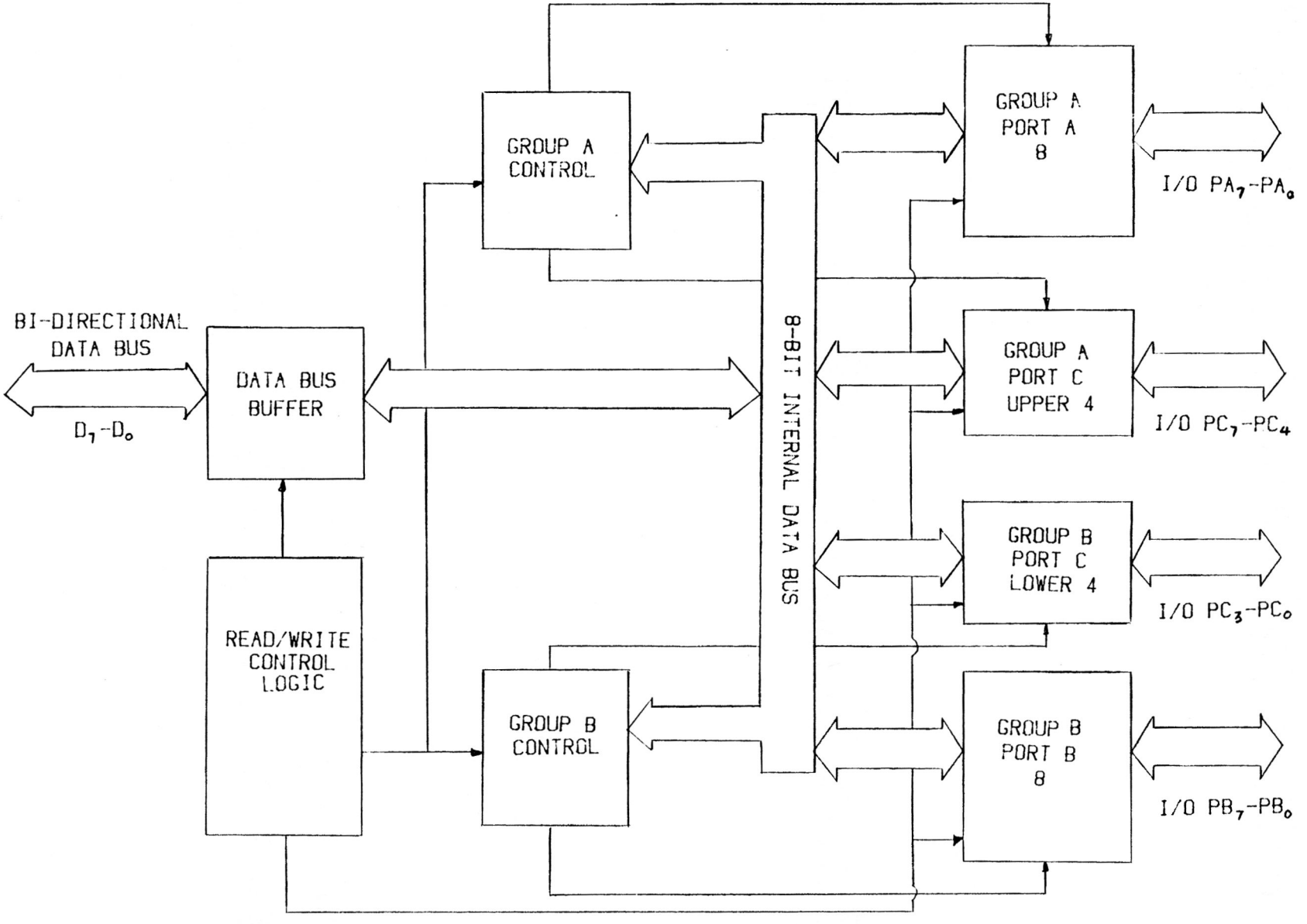


Figure 7. 8255 PPI Functional Block Diagram

(Adapted From [29])

operation. The user is referred to [29] for further detail on the general operational characteristics of the 8255 PPI.

Though both the Intel 8255's maintain the same interface (at different I/O addresses) with the CPU set, the interface between I/O group 1 device and the edge connector (J1) is significantly different from the interface between the I/O group 2 device and its associated edge connector (J2). This gives the user a great deal of flexibility when configuring the system's external parallel I/O devices. Because of these flexible "external" interfaces however, not all ports are capable of operating in each of the 8255's operational modes, though all ports can be programmed as either input or output. The I/O group 1 ports can fully utilize the 8255's multi-mode operation. The I/O group 2 ports, however, are limited to a single mode (mode 0) of operation. The allowable port configurations for both groups are summarized below [1]:

Port A (I/O Group 1)

Mode 0 input

Mode 0 output (latched)

Mode 1 input (strobed)

Mode 1 output (latched)

Mode 2 bi-directional

Port B (I/O Group 1)

Mode 0 input

Mode 0 output (latched)

Mode 1 input (strobed)

Mode 1 output (latched)

Port C (I/O Group 1)

Mode 0 8-bit input

Mode 0 8-bit output

Ports D and E (I/O Group 2)

Mode 0 input

Mode 0 output (latched)

Port F (I/O Group 2)

Mode 0 8-bit input

Mode 0 8-bit output

Mode 1 4-bit input/4-bit output (unlatched/latched)

Mode 1 4-bit output/4-bit input (latched/unlatched)

Since the objective here is to test the I/O lines for stuck-at-0 or stuck-at-1 faults and because of the limitation on the I/O group 2 mode operation [1], the mode 0 configuration seems to fit this testing purpose. All the I/O lines can be configured either as output or input lines. With this configuration, a technique called LOOPBACK [5], i.e., loop I/O port outputs to inputs, can be employed in testing the I/O ports. One group (I/O group 1) of the parallel port can be configured as the output port and the

other group(I/O group 2) as the input port and, with the proper edge connector, a word pattern is output from J1 (which is configured as the output port) and then read by J2 (which is configured as the input port). The pattern output by J1 and pattern input by J2 should agree and any discrepancy would indicate that the port is bad. See Figure 8 for the conceptual idea. Different word patterns can then be used to test the I/O lines for stuck-at-0 or stuck-at-1 faults (see Section 4.3.1.1).

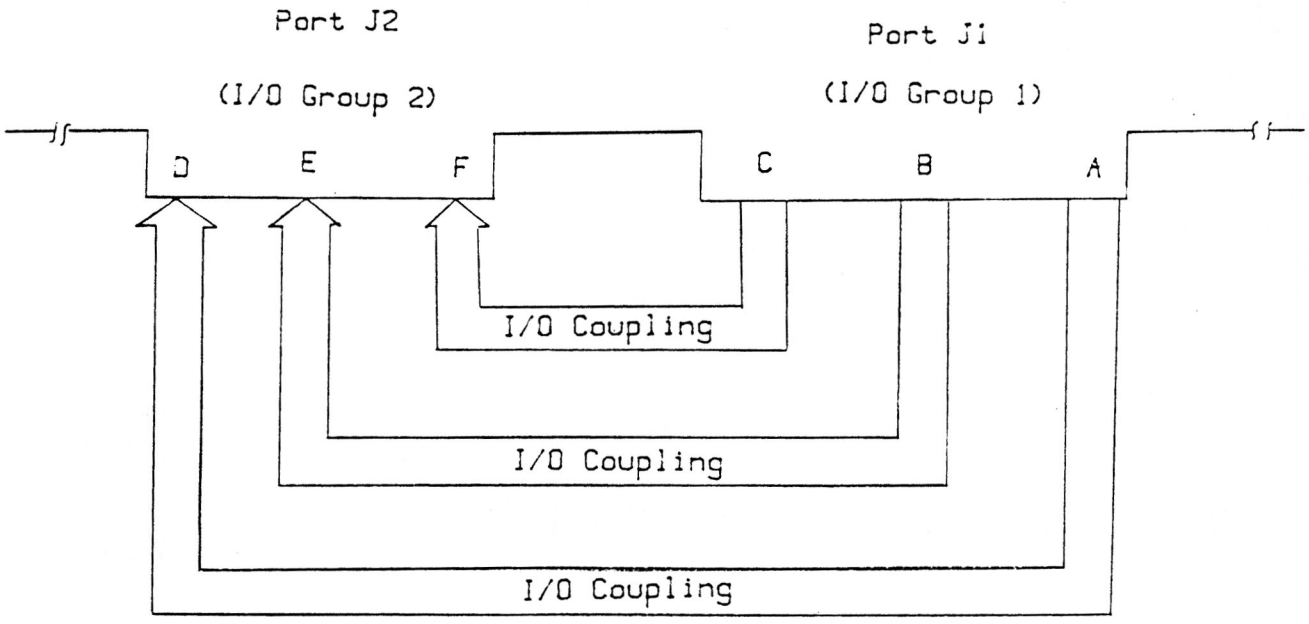


Figure 8. Loopback Technique

It was earlier suggested that the loopback test be done in both directions, i.e., configured J1 as the input port and J2 as the output port (after the first test), so as to verify that both the parallel ports can function as either input or output ports. But due to the different interfacing networks required for different port configurations (see next paragraph) , it has proved to be inconvenient to have to swap the interfacing networks every time the test is performed. Therefore the test was instead restricted to one direction only (J1 as the output port and J2 as the input port).

Note that eight lines already have been installed with driver networks (locations A1 and A2) for external interfacing and since all the ports are configured as output, therefore additional driver networks are necessary and are to be installed at locations A3, A4, A5 and A6 for proper interfacing with the outside world. Also termination networks will be installed at locations A7, A8, A9, A10, A11 and A12 (J2 is configured as input port). Typical driver networks will be those listed in Table 8 and the termination network is either the 220 ohm/330 ohm driver or the 1 k-ohm pull up (available in package); see Figure 9.

The drivers , terminators and cables used for our testing purposes will be discussed later in Section 5.0.

Table 8. Drivers (Adapted From [1])

Drivers	Characteristic	Sink Current (mA)
7438	I,OC	48
7437	I	48
7432	NI	16
7426	I,OC	16
7409	NI,OC	16
7408	NI	16
7403	I,OC	16
7400	I	16

I = inverting NI = non-invertin OC = open collector

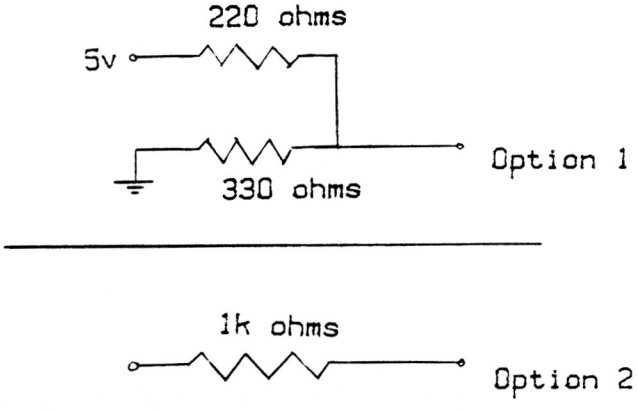


Figure 9. I/O Terminators
(Adapted From [1])

4.3.1.1 Configuring the parallel ports for testing

Tables 9 and 10 depict the pin assignments for connectors J1 and J2 respectively. In order to ensure that ports J1 (I/O group 1) and J2 (I/O group 2) has no stuck-at faults or bridging faults between adjacent bits, J1 (I/O group 1) is first configured as an output port and a word pattern of 101010....10 alternating with 010101....01 is output (one at a time) to J1. These patterns will then be read by J2 (which is configured as the input port) and compared with a known value. This verifies whether port J1 is working correctly or not. The same test is then repeated but this time with the test data 10101010 or 01010101 received at port J2 (I/O group 2). One very important thing to note from Tables 9 and 10 is that the data bits for ports A, B and C are not arranged in such a way that they are all in ascending or proper order i.e., PA0 PA1 ...PB0 PB1 ...PC0 PC1 (see Figure. 10). Hence the word patterns selected for ports A, B and C, when output to J1 should "look" like 101010....10 or 010101....01 or for the second case, when received by ports D, E and F, should also "look" like 1010....10 or 01010....01 at J2.

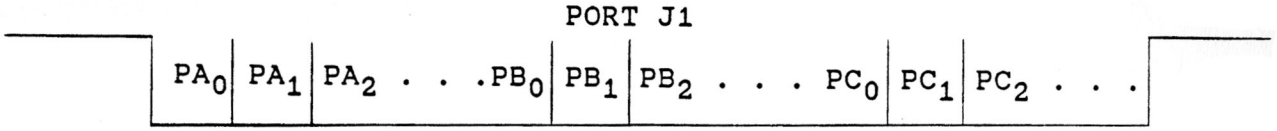


Figure 10. Desired Arrangement of Port J1

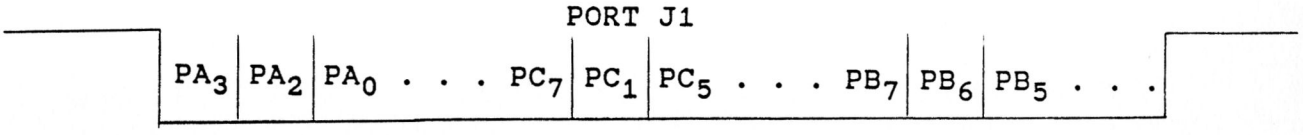


Figure 11. Actual Arrangement of Port J1

Table 9. Pin Assignments for Connector J1
(Adapted From [1])

PIN	SIGNAL	PIN	SIGNAL
1	PORT B - BIT 3	2	GND
3	PORT B - BIT 2	4	GND
5	PORT B - BIT 1	6	GND
7	PORT B - BIT 0	8	GND
9	PORT B - BIT 4	10	GND
11	PORT B - BIT 5	12	GND
13	PORT B - BIT 6	14	GND
15	PORT B - BIT 7	16	GND
17	PORT C - BIT 3	18	GND
19	PORT C - BIT 2	20	GND
21	PORT C - BIT 4	22	GND
23	PORT C - BIT 6	24	GND
25	PORT C - BIT 0	26	GND
27	PORT C - BIT 5	28	GND
29	PORT C - BIT 1	30	GND
31	PORT C - BIT 7	32	GND
33	PORT A - BIT 7	34	GND
35	PORT A - BIT 6	36	GND

37	PORT A - BIT 5	38	GND
39	PORT A - BIT 4	40	GND
41	PORT A - BIT 1	42	GND
43	PORT A - BIT 0	44	GND
45	PORT A - BIT 2	46	GND
47	PORT A - BIT 3	48	GND
49	EXT INTR 1/	50	GND

Table 10. Pin Assignments for Connector J2
(Adapted From [1])

PIN	SIGNAL	PIN	SIGNAL
1	GND	2	GND
3	PORT E - BIT 3	4	GND
5	PORT E - BIT 0	6	GND
7	PORT E - BIT 1	8	GND
9	PORT E - BIT 2	10	GND
11	PORT E - BIT 4	12	GND
13	PORT E - BIT 5	14	GND
15	PORT E - BIT 6	16	GND
17	PORT E - BIT 7	18	GND
19	PORT F - BIT 3	20	GND
21	PORT F - BIT 2	22	GND
23	PORT F - BIT 1	24	GND
25	PORT F - BIT 0	26	GND
27	PORT F - BIT 4	28	GND
29	PORT F - BIT 5	30	GND
31	PORT F - BIT 6	32	GND
33	PORT F - BIT 7	34	GND
35	PORT D - BIT 7	36	GND
37	PORT D - BIT 6	38	GND
39	PORT D - BIT 5	40	GND
41	PORT D - BIT 4	42	GND
43	PORT D - BIT 0	44	GND
45	PORT D - BIT 1	46	GND
47	PORT D - BIT 2	48	GND
49	PORT D - BIT 3	50	GND

Test Patterns for Ports A, B and C (I/O group 1) when:

1. test pattern at J1 is 101010....10

   Port A = 9A

   Port B = A5

   Port C = 27

2. test pattern at J1 is 010101....01

   Port A = 65

   Port B = 5A

   Port C = D8

Test Patterns for Ports A, B and C (I/O group 1) when :

1. test pattern received at J2 is 101010....10

   Port A = 5A

   Port B = A5

   Port C = A5

2. test pattern received at J2 is 010101....01

   Port A = A5

   Port B = 5A

   Port C = 5A

Specific I/O addresses for the six ports where the CPU set can access are listed in Table 11.

Table 11. Parallel I/O Port Addresses (Adapted From [1])

PORT	8255 DEVICE LOCATION	8-BIT ADDRESSES (HEX)
A	GROUP 1 PORT A	E4
B	GROUP 1 PORT B	E5
C	GROUP 1 PORT C	E6
-	GROUP 1 CONTROL	E7 (WRITE ONLY)
D	GROUP 2 PORT A	E8
E	GROUP 2 PORT B	E9
F	GROUP 2 PORT C	EA
-	GROUP 2 CONTROL	EB (WRITE ONLY)

4.3.1.2 Other cases of testing the parallel ports

In the preceding case, we have an equal number of output ports and I/O lines, so there is no problem in testing all ports or lines for stuck-at faults or bridging faults between adjacent bits.

For the rest of this section, we will discuss two other cases of testing the parallel I/O ports :

Case 1: Input ports or I/O lines more numerous than output ports or I/O lines,

Case 2: Input ports or I/O lines less numerous than output ports or I/O lines.

Case 1

When the number of input I/O lines is more than the

number of output I/O lines, the ports (both input and output) can still be easily tested for stuck-at faults or bridging faults. Simply connect all the available output I/O lines to all the input I/O lines as shown in Figure 12.

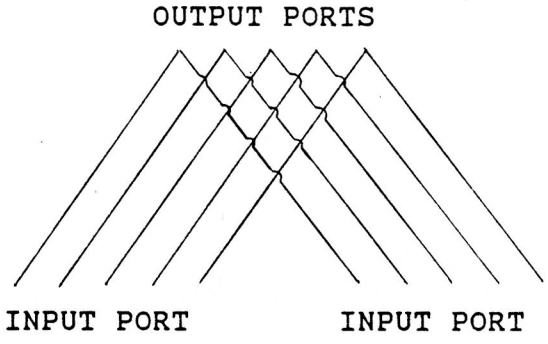


Figure 12. Configuration for Case 1

With the appropriate test data, the ports can be checked for stuck-at faults or bridging faults between adjacent bits.

Case 2

When the number of input I/O lines is less than the number of output I/O lines, then some of the output I/O lines might have to be tied together with some logic gates, so as to keep the number of output I/O lines the same as the number of input I/O lines. Figure 13 shows two possible configurations of case 2.

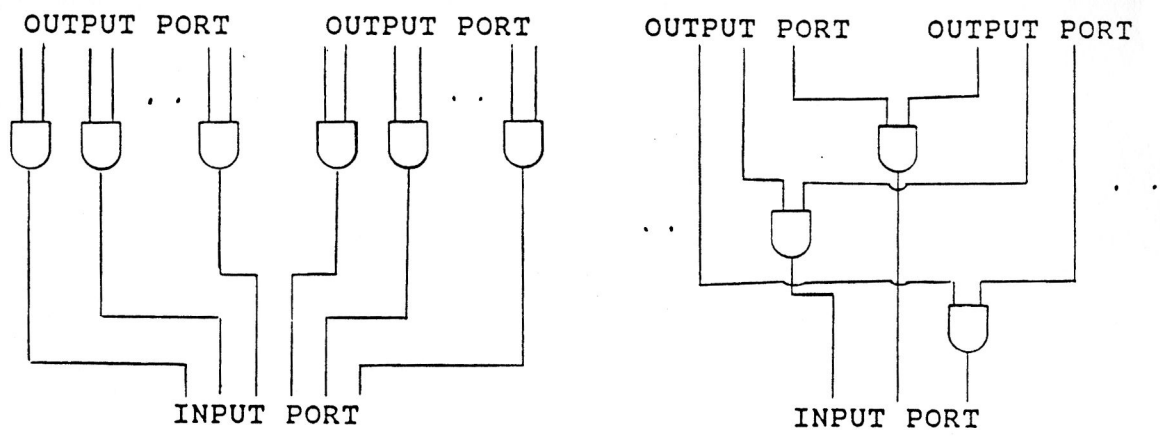


Figure 13. Configuration for Case 2

Again, with the appropriate test data, the input and output ports can be checked for stuck-at faults or bridging faults.

4.3.2 Testing the serial I/O port

The serial I/O interface logic consists primarily of an Intel 8251 USART device located at A22 in Figure C-4 of Appendix C. It is a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) designed specifically for the 8080 microcomputer systems. Like other I/O devices in the 8080 microcomputer system, its functional configuration is programmed by the system software for maximum flexibility. A block diagram of the 8251 is shown in Figure 14.

The Intel 8251 USART device converts

parallel format system data into serial format for transmission and converts incoming serial format data into parallel system data for reception. It also deletes or inserts bits or characters that are functionally unique to the communication technique.

The 8251 USART can also be used for either synchronous or asynchronous data communication. Both these formats require that framing information be added to the data to enable proper detection of the character at the receiving end. The major difference between the two formats is that the asynchronous format requires framing information to be added to each character, while the synchronous format adds framing information to blocks of data or messages. Both these formats can be transmitted in half or full duplex mode.

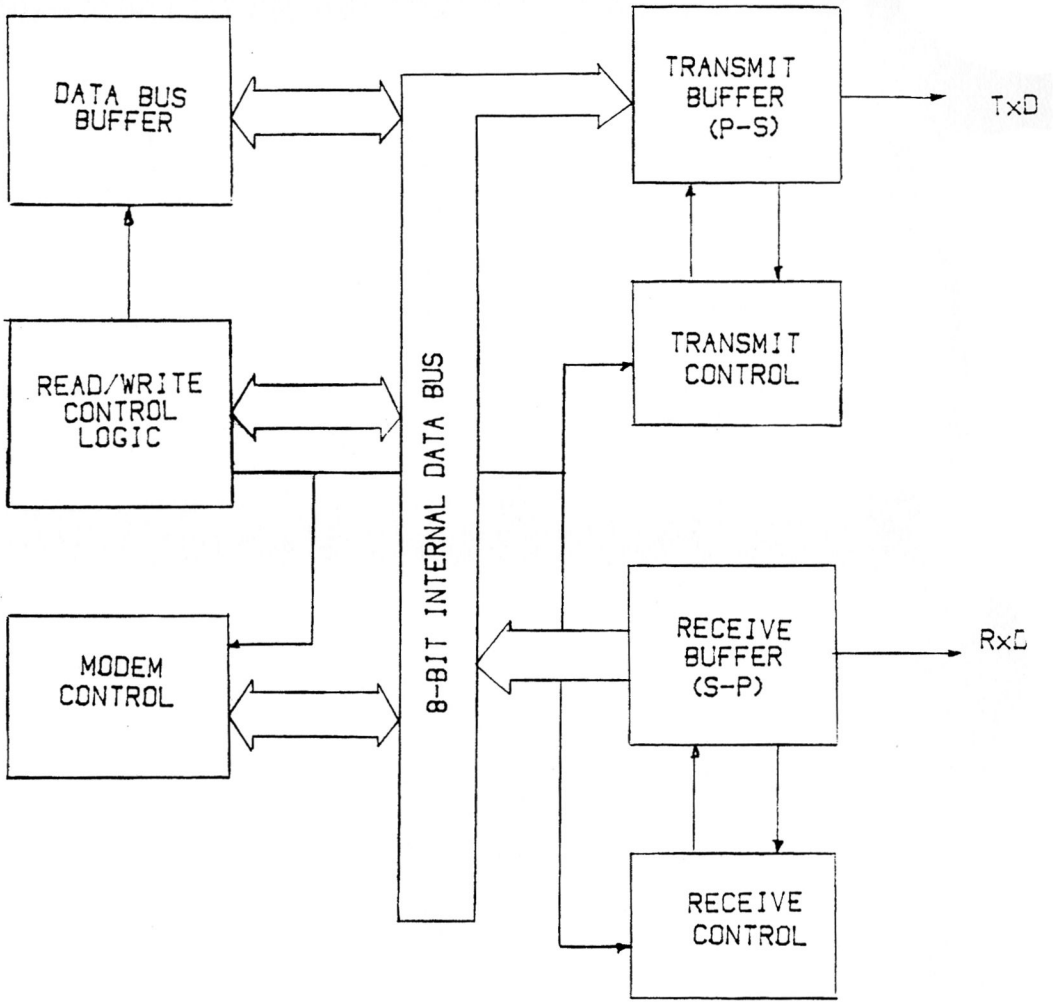


Figure 14. 8251 USART Functional Block Diagram
(Adapted From [29])

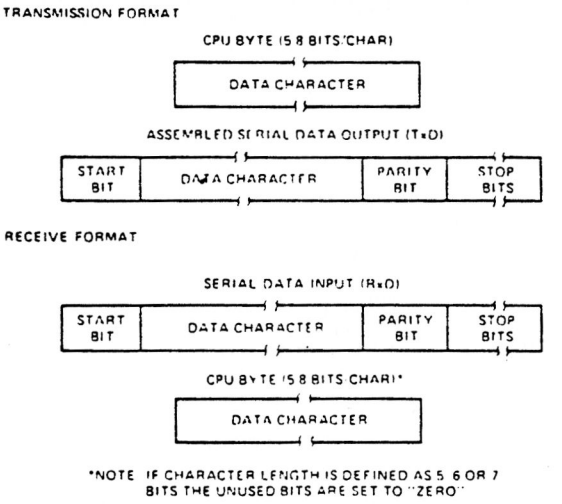


Figure 15. Asynchronous Mode (Adapted From [1])

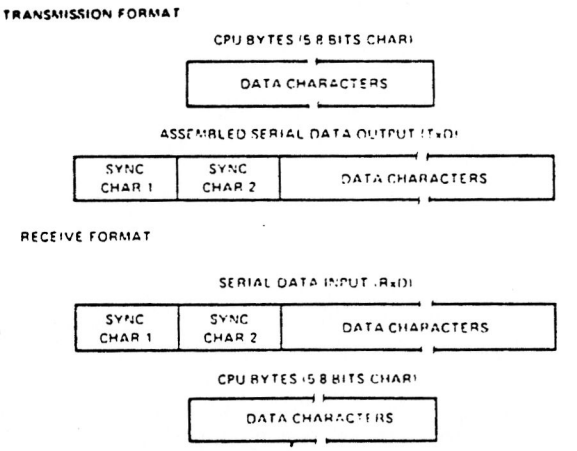


Figure 16. Synchronous Mode (Adapted From [1])

Prior to starting data transmission or reception, a set of control words generated by the CPU must be sent out to initialize the 8251 USART to support the desired communication format. These control words define the complete functional description of the 8251 USART and must immediately follow a Reset operation (internal or external). These control words will program the: baud rate, character length, number of stop bits, synchronous or asynchronous operation, even/odd parity, etc. In the synchronous mode, options are also provided for selecting either internal or external character synchronization.

The Intel 8251 USART essentially defines the character of the serial I/O interface. The user is referred to [29] for further application notes on the 8251 USART.

4.3.2.1 Configuring the serial port for testing

The serial I/O port interface communicates with an external I/O device via a 26-pin double-sided PC edge connector (J3). Table 12 provides a pin list for connector J3, and Table 13 specifies the I/O addresses which can be accessed by the CPU set.

On the connector J3, the serial I/O port is configured to perform full duplex operation, so the signal line TxD will be tied to RxD so that a closed-loop (loopback) test can be performed. A 1 and a 0 are

placed (a bit at a time) in the most significant bit of the accumulator. Then they are sent out to the serial-output-data pin (TxD) of the 8251. This data is captured by the serial-input-data pin (RxD) of the 8251. Each time the captured data is compared with what was sent earlier. If they agree, the serial port is good, otherwise it is bad.

There are two general schemes for the transmission or reception of data between the serial I/O port and the external world: the asynchronous and the synchronous modes. Both these methods will be tested on the Intel SBC 80/10A board. We will first perform the testing procedure in the asynchronous mode and then follow by the synchronous mode.

Table 12. Pin Assignments for Connector J3 (Adapted From [1])

PIN	SIGNAL NAME	PIN	SIGNAL NAME
1	CHASSIS GND	2	
3	TRANSMITTED DATA	4	
5	RECEIVE DATA	6	TTY RD CONTROL
7	REQ TO SEND	8	
9	CLEAR TO SEND	10	
11	DATA SET READY	12	
13	GND	14	Tx CLK/DATA TERM READY
15	DATA CARRIER RET	16	TTY RD CONT RET
17		18	
19		20	
21		22	RECEIVE CLK/TTY Rx DATA RET
23	TTY Rx DATA	24	TTY Tx DATA RET
25	TTY Tx DATA	26	GND

Table 13. Serial I/O Port Addresses
(Adapted From [1])

I/O ADDR. (HEX)	COMMAND	FUNCTION
ED OR EF	OUTPUT	CONTROL WORD
EC OR EE	OUTPUT	DATA
ED OR EF	INPUT	STATUS
EC OR EE	INPUT	DATA

Asynchronous Mode

Control Word

1 1 0 0 1 1 1 0 Mode Instruction

0 0 1 0 0 1 1 1 Command Instruction

Synchronous Mode

Control Word

1 0 0 0 1 1 0 0 Mode Instruction

1 0 1 0 0 1 1 1 Command Instruction

0 0 1 0 1 1 0 Sync Character

In both the asynchronous and synchronous modes, test data 1010101 (01010101) will be sent through TxD and received by RxD. The value received should be checked with the value sent; any discrepancy between

the two values indicates that the serial port is not functioning properly.

The test set-up is shown later in Section 5.0.

4.4 Testing of the External System Bus (Connector P1)

Table 14 shows each of the external system bus signals and this multibus structure provides a common element for communication between a wide variety of system modules which includes : single board computer, memory, digital and analog I/O expansion boards, and peripheral controllers [29].

These multibus signal lines allow for master-slave relationships between the various system modules described above. A bus master module can drive the command and address bus. An example of a bus master would be a single board computer. A bus slave cannot control the bus. An example of a bus slave would be a memory expansion board.

The SBC 80/10A does not provide the Bus Priority Request signal and therefore it can only be used with one other multibus master [1]. One of our goals here is try to minimize the amount of hardware use and therefore a better choice would be to use an external RAM to test the command signals (only the address, data, write and read signal lines) on the external system bus.

Table 14. Pin Assignment for Connector P1
(Adapted From [1])

	(COMPONENT SIDE)			(CIRCUIT SIDE)		
	PIN	MNEMONIC	DESCRIPTION	PIN	MNEMONIC	DESCRIPTION
POWER SUPPLIES	1	GND	Signal GND	2	GND	Signal GND
	3	VCC	+ 5VDC	4	VCC	+ 5VDC
	5	VCC	+ 5VDC	6	VCC	+ 5VDC
	7	VDD	+12VDC	8	VDD	+12VDC
	9	VBB	- 5VDC	10	VBB	- 5VDC
	11	GND	Signal GND	12	GND	Signal GND
BUS CONTROLS	13	BCLK/	Bus Clock	14	INIT/	Initialize
	15	BPRN	Bus Pri. In	16	▷	
	17	BUSY/	Bus Busy	18	▷	
	19	MRDC/	Mem Read Cmd	20	MWTC/	Mem Write Cmd
	21	IORC/	I/O Read Cmd	22	IOWC/	I/O Write Cmd
	23	XACK/	XFER Acknow	24	▷	
SPARES	25	AACK/	Special	26		
	27			28		
	29			30		
	31	CCLK/	Constant Clock	32		
	33	▷		34		
INTERRUPTS	35	▷		36	▷	
	37	▷		38	▷	
	39	▷		40	▷	
	41	▷		42	INTR1/	Interrupt request
ADDRESS	43	ADRE/	Address Bus	44	ADRF/	Address Bus
	45	ADRC/		46	ADRD/	
	47	ADRA/		48	ADRB/	
	49	ADR8/		50	ADR9/	
	51	ADR6/		52	ADR7/	
	53	ADR4/		54	ADR5/	
	55	ADR2/		56	ADR3/	
	57	ADR0/		58	ADR1/	
DATA	59	▷	Data Bus	60	▷	Data Bus
	61	▷		62	▷	
	63	▷		64	▷	
	65	▷		66	▷	
	67	DAT6/		68	DAT7/	
	69	DAT4/		70	DAT5/	
	71	DAT2/		72	DAT3/	
	73	DAT0/		74	DAT1/	
POWER SUPPLIES	75	GND	Signal GND	76	GND	Signal GND
	77	VBB ▷	-10VDC	78	VBB ▷	-10VDC
	79	VAA	-12VDC	80	VAA	-12VDC
	81	VCC	+ 5VDC	82	VCC	+ 5VDC
	83	VCC	+ 5VDC	84	VCC	+ 5VDC
	85	GND	Signal GND	86	GND	Signal GND

▷ Used by Intellec® MDS Bus.

A 2K RAM is used for testing the multibus signal lines. A memory map of the SBC 80/10A board is shown in Figure 17.

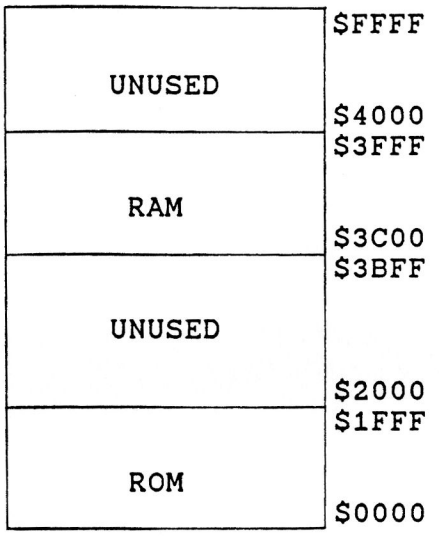


Figure 17. Memory Map of the SBC 80/10A Board

The addresses of the external RAM are set up between $6800 and $6FFF. The circuit diagram for the set-up is shown in section 5.0. The test is then carried out the same way as the onboard RAM test.

4.5 Auxiliary Connector P2

Table 5-4 in [1] shows the various auxiliary test points on the SBC 80/10A board. Most of these signals actually have been indirectly tested while performing the other module tests. Hence, this external system bus will not be tested again.

5.0 PRINCIPLE OF OPERATION

The following section describes the test set-up and the sequence of events which occurs during the testing process.

5.1 Test Set Up

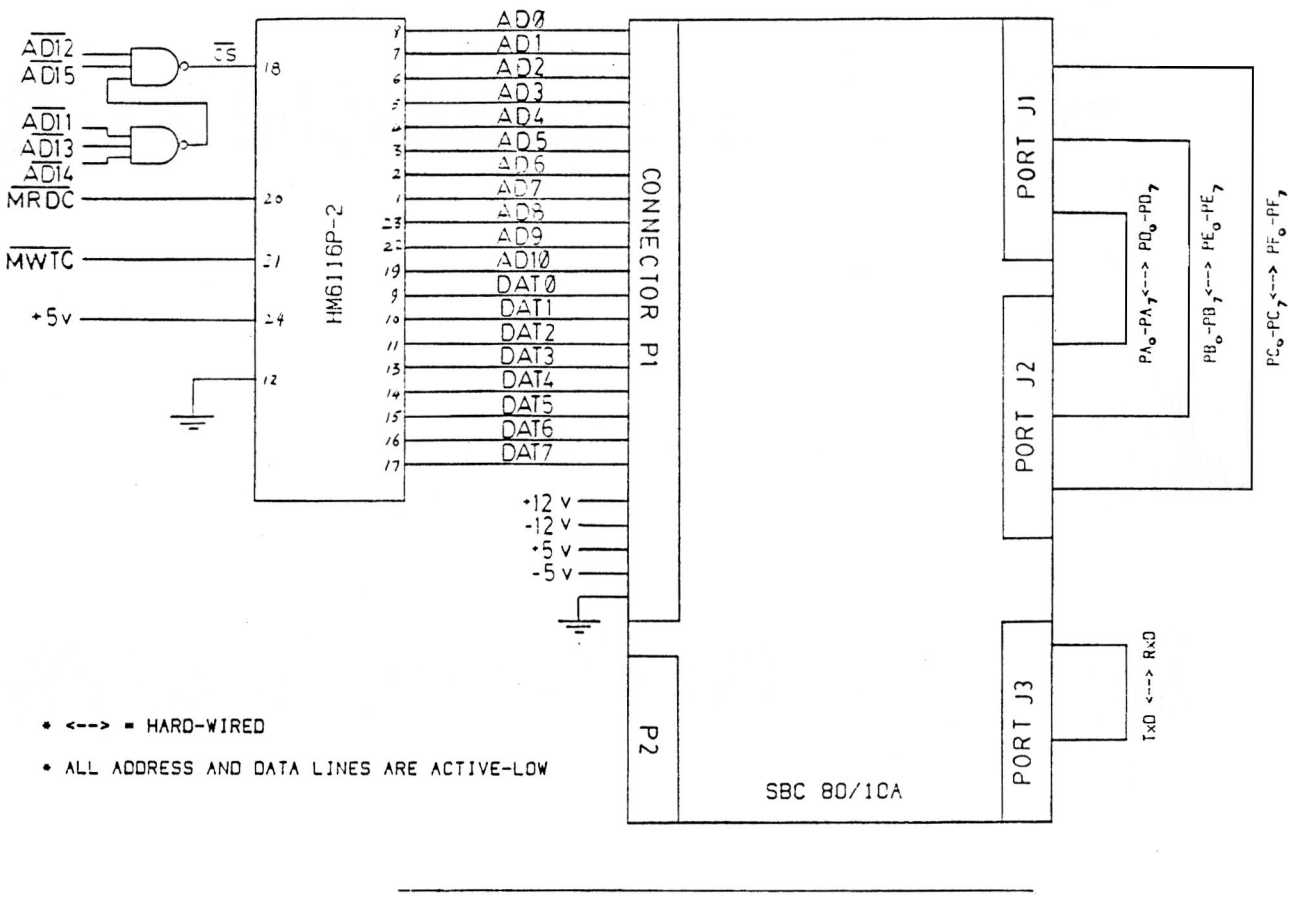
The test is set up in the user testing environment and once invoked, becomes automatic, requiring no operator intervention, and upon confronting an error the self-test program will immediately halt.

- The "GO/NO_GO" feature is indicated by an LED pair connected to the outputs of two NAND gates (7400); see Figure 18. When the test begins, LED1 will be ON and LED2 will be OFF. Upon successful execution of the program, the LED pair will toggle and, if for some reason the LED pair remains in its initial state for quite some time, the user should immediately recognize that the self-test program has halted. The location where the program execution stopped can be checked by testing the logic states of the address lines.
- Two 50-pin card edge-connectors (0.1" center) are used for the two parallel ports (I/O group 1 and I/O group 2) at J1 and J2. The I/O lines from port J1 are hard-wired to their corresponding lines at J2 (i.e PA1 to PD1, PA2 to PD2, etc.); see Figure 18.
- A 26-pin card edge-connector (0.1" center) is used for the serial port (J3). TxD and RxD are again hard-wired

together; see Figure 18.

- An 86-pin card edge-connector (0.156" center) with a 40-pin IDC connector (wire-wrap) is used for the external system bus for accessing the necessary signal lines; see Figure 18.
- Driver networks (7408) have already been inserted at locations A1, A2, A3, A4, A5 and A6. Termination networks, SBC-902 (1K-ohm pull-up resistor) are still needed for locations A7, A8, A9, A10, A11 and A21.
- Other hardware needed are : one 7410 3-input NAND gate
  two 470 ohms resistors
  HM6116P-2 2K x 8 bits RAM

Figure 18 shows the circuit diagram of the whole set-up ready to be tested.



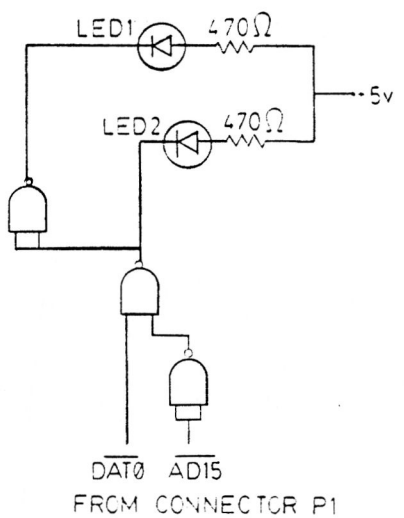


Figure 18. Board Set Up For Testing

5.2 Test Procedure

The following is the sequence of events that should occur during the testing process.

1. The power is turned ON,
2. This action activates the clock and resets the circuit, which in turn generates the Reset signal. The microprocessor automatically executes the instruction in memory location $0000 which is the beginning address of the self-test program.
3. LED1 is turned ON and LED2 is OFF, indicating that the self-test program is in progress.
4. The first module of the program executes the ROM (containing the self-test program) test.
5. If this module passes the test, program execution is passed to the microprocessor test.
6. If the microprocessor passes the test, program execution is passed to the ROM (containing the system software) test.
7. If the ROM passes the test, program execution is passed to the RAM test.
8. If the RAM passes the test, program execution is passed to the serial I/O ports test.
9. If the serial I/O ports passes the test, program execution is passed on to the parallel I/O port test.
10. If the parallel I/O port passes the test, program execution is passed on to the external system bus test.

11. After the external system bus test module has been completed, then the program will set address line A15 and data line D0 to 1.These signals will be used to toggle the LED pair, indicating that the self-test program has been executed successfully.

6.0 CONCLUSIONS AND POSSIBLE FURTHER EXPANSION

6.1 Conclusions

The basic philosophy behind incorporating a built-in self-testing concept into the Intel SBC 80/10A board is to enable the system to perform self-test and to verify that the system is operational. It is intended to be used either in a production testing environment, where thousands of incoming boards have to be screened before being integrated into more complex systems, or in a user testing environment, where the operator might want to check his malfunctioning board. The "go/no-go" capability/feature of the self-test program is best used for this purpose. It can quickly distinguish the good boards from the bad ones and hence reduce the time and cost involved in isolating the same faults in later more complex systems. The self-test program is driven by software written specially for the processor used on the board. The classes of faults are mostly restricted to pin-level stuck-at-0 and stuck-at-1 type and to the instruction sets of the microprocessor. This self-test technique verifies faults but it does not provide a complete and thorough system diagnostic capability; however, it could be further enhanced to include fault-detection and isolation down to subsystem or component level (see Section 6.2).

The design and implementation of the self-testing

program requires not only an intimate knowledge of the processor software but also an in depth familarity with the overall system design. For the test engineer who has not been responsible for the system design, and who attempts to generate adequate functional test program for the system, a large amount of time can be spent on familarizing himself with the operating environment of the microprocessor based system.

Samples of program listings can be found in Appendix B to show their complexity and the amount of ROM space needed. These programs have not been tested; hence how many faults this self-test program will actually detect is yet to be determined.

This project suggested to some extent the feasibility of incorporating the built-in self-testing concept into the Intel SBC 80/10A board and it can possibly be modified and extended to perform the same tasks for more complex systems, including the 16 and 32 bit microcomputers.

Finally, as the complexity of microprocessor-based systems increases, the tasks of testing a system and isolating faults becomes increasingly difficult. Accompanying the increase in complexity is an increase in testing cost, due to the cost of automatic test equipment (ATE). As an attempt to reduce this testing cost and at the same time retain user confidence in the systems, built-in

self-testing techniques should be attracting lots of attention in the 1980s and in the foreseeable future.

6.2 Possible Further Expansion

If, in addition to performing the self-test, we can use a dumb terminal to monitor the testing process instead of the LEDs, the concept can be expanded to include diagnosis for fault-detection and isolation down to subsystem or component levels. The resolution of the fault-location diagnosis is a function of the amount of hardware and software on the system. For the Intel SBC 80/10A board, we can decompose the board into the following subsystems or components:

1. CPU set - 8080A microprocessor
   - 8224 clock generator
   - 8238 system controller
   - and associated circuitry (see Figure C-1, Appendix C)
2. Random Access Memory - Eight 1 bit static RAM (8102)
   - 8226 bi-directional bus driver
   - 3205 address decoder
   - and associated circuitry (see Figure C-2, Appendix C)
3. Read-Only-Memory - ROM/PROM used in locations A23, A24, A25 and A26

- 3205 address decoder
- 8216 bi-directional bus driver
- and associated circuitry (see Figure C-3, Appendix C)

4. Parallel I/O Port - two 8255 PPI
   - Eight 8226 bi-directional bus drivers
   - 3205 address decoder
   - and associated circuitry (see Figure C-5, Appendix C)

5. Serial I/O Port - 8251 USART
   - LM1489 inverters
   - and associated circuitry (see Figure C-4, Appendix C)

With the above subsystems guides, diagnosis of board failure becomes very much a simple "cook-book" procedure. Since the dominant failure mode in the system is a stuck pin, to find this type of fault, it is necessary to start with a board's edge-failure and work back to locate the device with good input and bad outputs. A quicker method would be to replace the component on the path of the failure activity and re-run the self-testing program. This diagnostic algorithm of fault detection and isolation might have to wander all over the board to follow the path leading to the failure.

To alleviate the burden of wandering around the

board, the subsystem or component-level information can be stored in the computer (database) in such a way that it can be retrieved each time a fault is encountered and it can display the set of possible faulty components (depending on which phase the testing process is on).

An expert system (knowledge-based) would be a good application for the above purpose. A file can be created that would contain all the information about all the possible failures that might occur and another file can contain all the possible remedies corresponding to each failure. For example, if a failure occurs during the RAM testing stage, the computer will display all the possible components that may be related to this fault, and the process is repeated until the fault is narrowed down to a bad node.

Finally, the self-testing program can be made general-purpose for use in many systems based on a particular microprocessor. For example, the memory test for the 8080A processor can, except for the starting and the ending address parameters, be the same for every system, only the I/O devices, which tend to be more system-specific, require major modifications of the test program. However, if the I/O driver routine in the self-testing program is written as a module, then it can be called from the main program after the appropriate modifications have

been done. This way the main program won't have to go through major changes each time the program is used in another system,and also if a particular device is absent from the system, the software module for that particular device can be omitted.

Appendix A

Flowcharts and Program Description

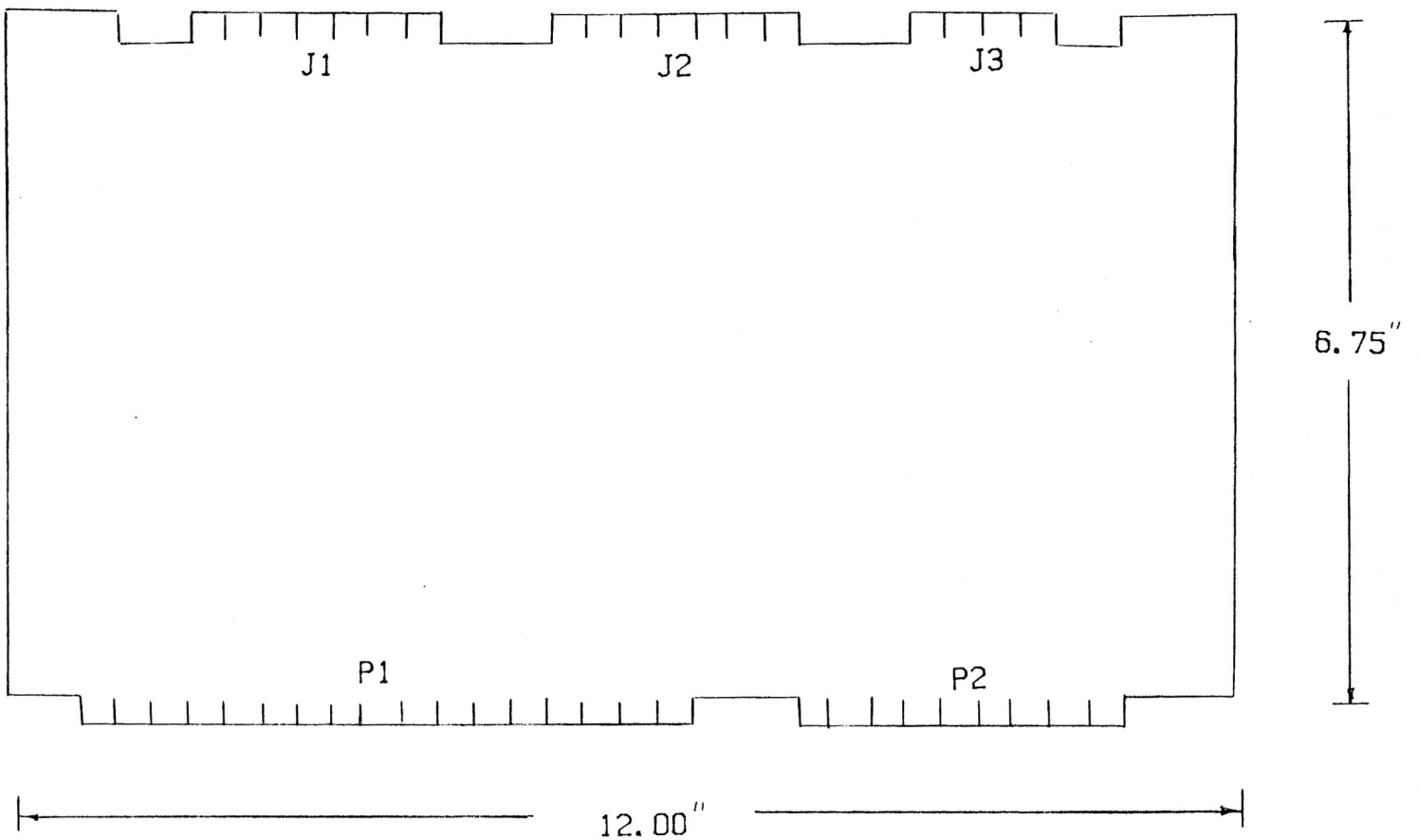


Figure A-1. SBC 80/10A Dimension Drawing

1. MAIN

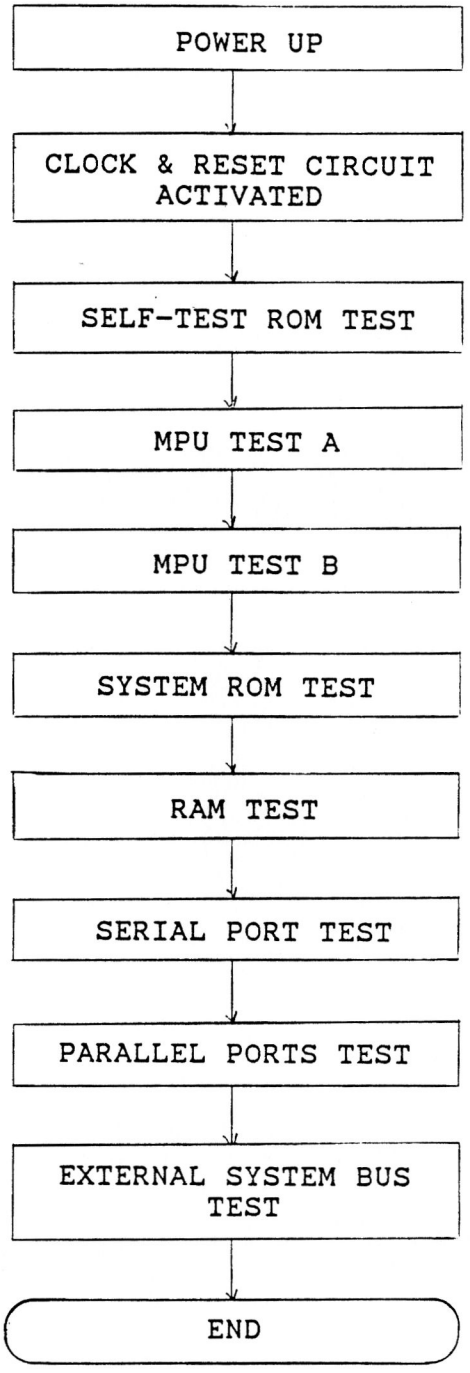


Figure A-2. MAIN Program

MAIN

This is the driver routine which calls all the appropriate routines (SELF-TEST ROM TEST, MPU TEST, SYSTEM ROM TEST, RAM TEST, SERIAL PORT TEST, PARALLEL PORTS TEST AND EXTERNAL SYSTEM BUS TEST) to perform the required tasks.

Upon encountering an error while executing one of the subroutines, the microprocessor operations will be halted until the operator has replaced the faulty component. After the replacement, the circuit is Reset and the self-testing program is re-run.

2. SELF

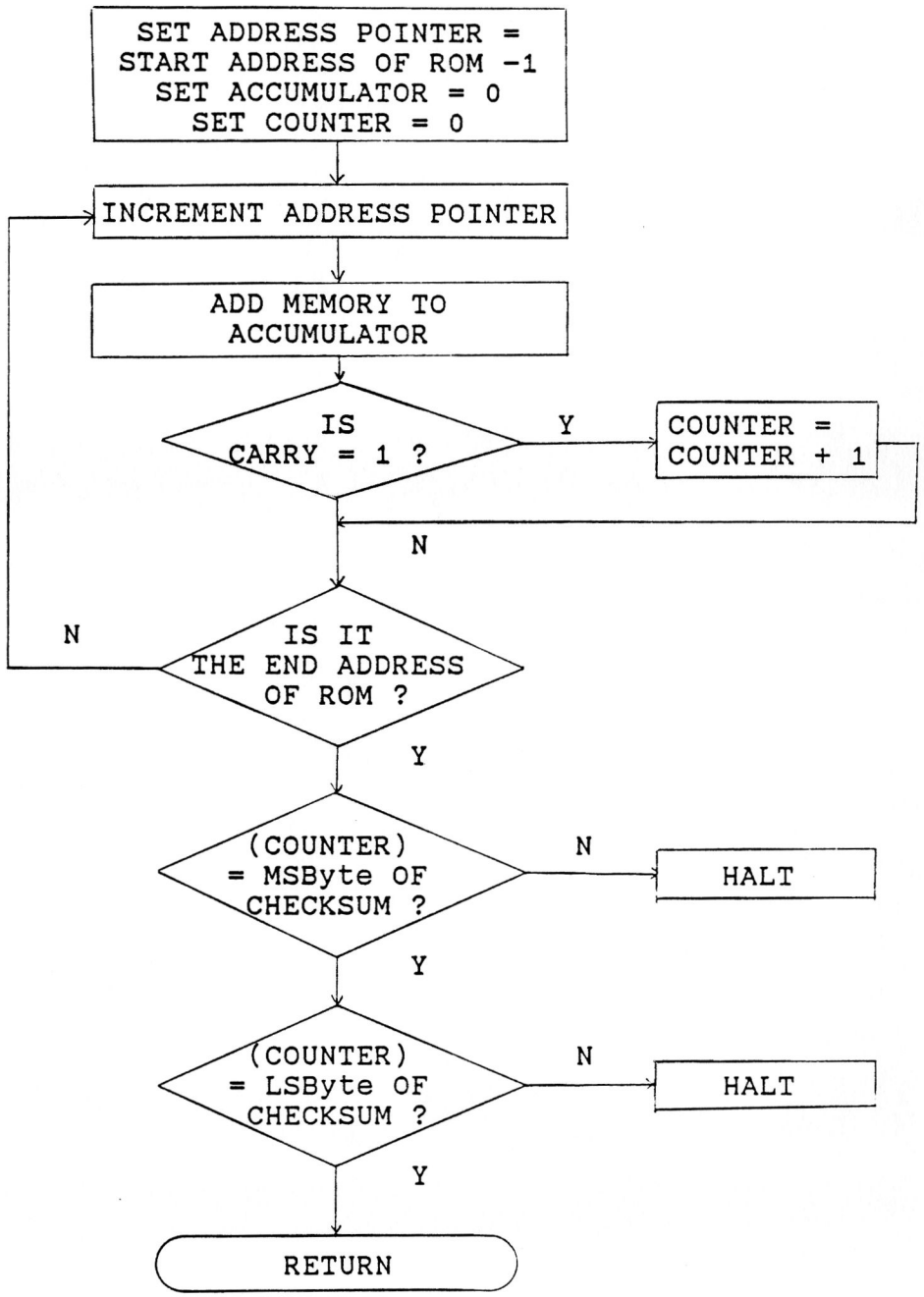


Figure A-3. Subroutine SELF

SELF

This routine constantly reads and adds the contents of the self-testing program. The carry generated from the addition is taken care of in another register and thus forms a 16-bit value checksum. This checksum should check with the predetermined checksum. Any discrepanies between the two values will force the program execution to halt.

3. MPU TEST A

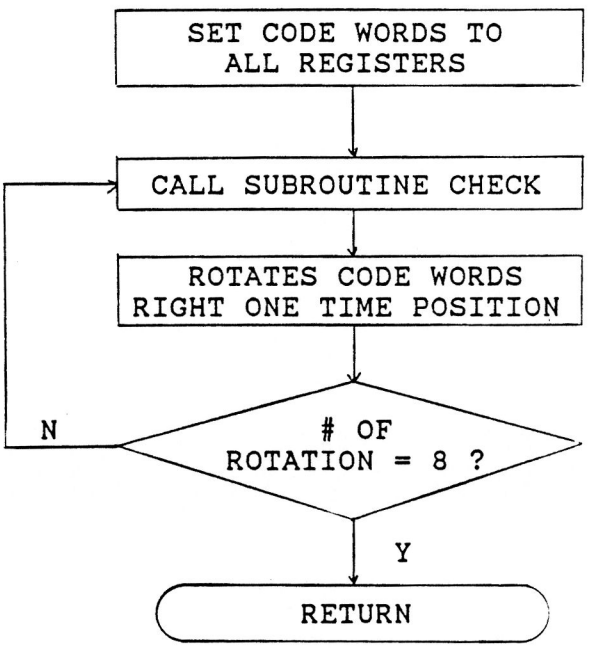


Figure A-4. Subroutine MPU TEST A

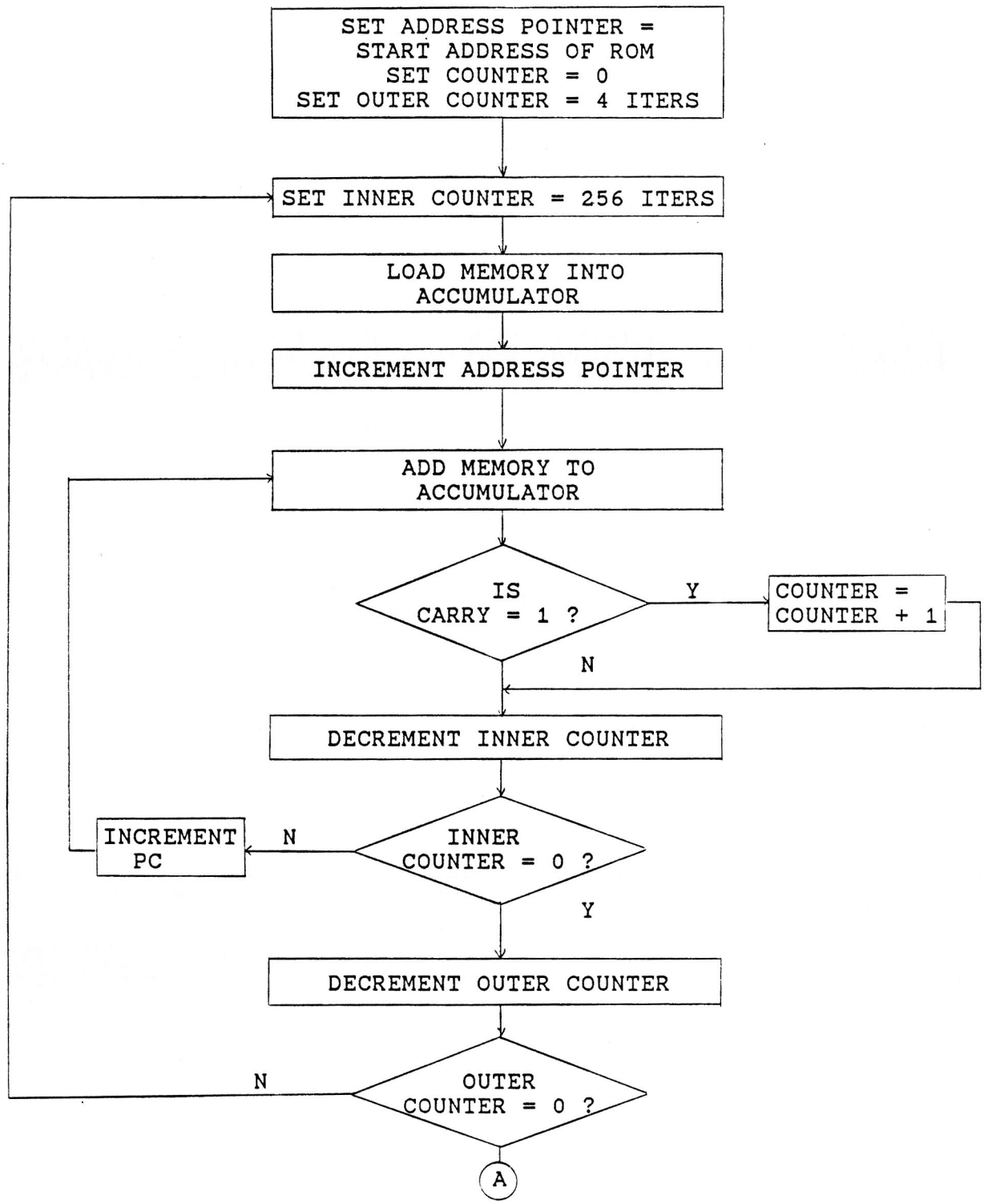
MPU TEST A

See subroutine CHECK.

MPU TEST B

The MPU Test B subroutine is functionally not doing anything except exercising as many of the 8080A instructions as possible.

4. ROM


SET ADDRESS POINTER =
START ADDRESS OF ROM
SET COUNTER = 0
SET OUTER COUNTER = 4 ITERS
SET INNER COUNTER = 256 ITERS
LOAD MEMORY INTO
ACCUMULATOR
INCREMENT ADDRESS POINTER
ADD MEMORY TO
ACCUMULATOR
IS
CARRY = 1 ?
Y
COUNTER =
COUNTER + 1
N
DECREMENT INNER COUNTER
INCREMENT
PC
N
INNER
COUNTER = 0 ?
Y
DECREMENT OUTER COUNTER
N
OUTER
COUNTER = 0 ?
A

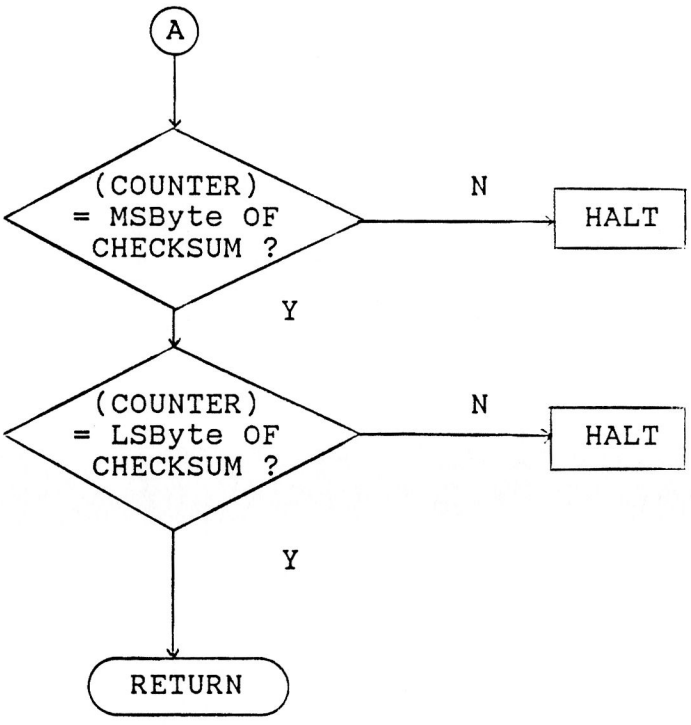
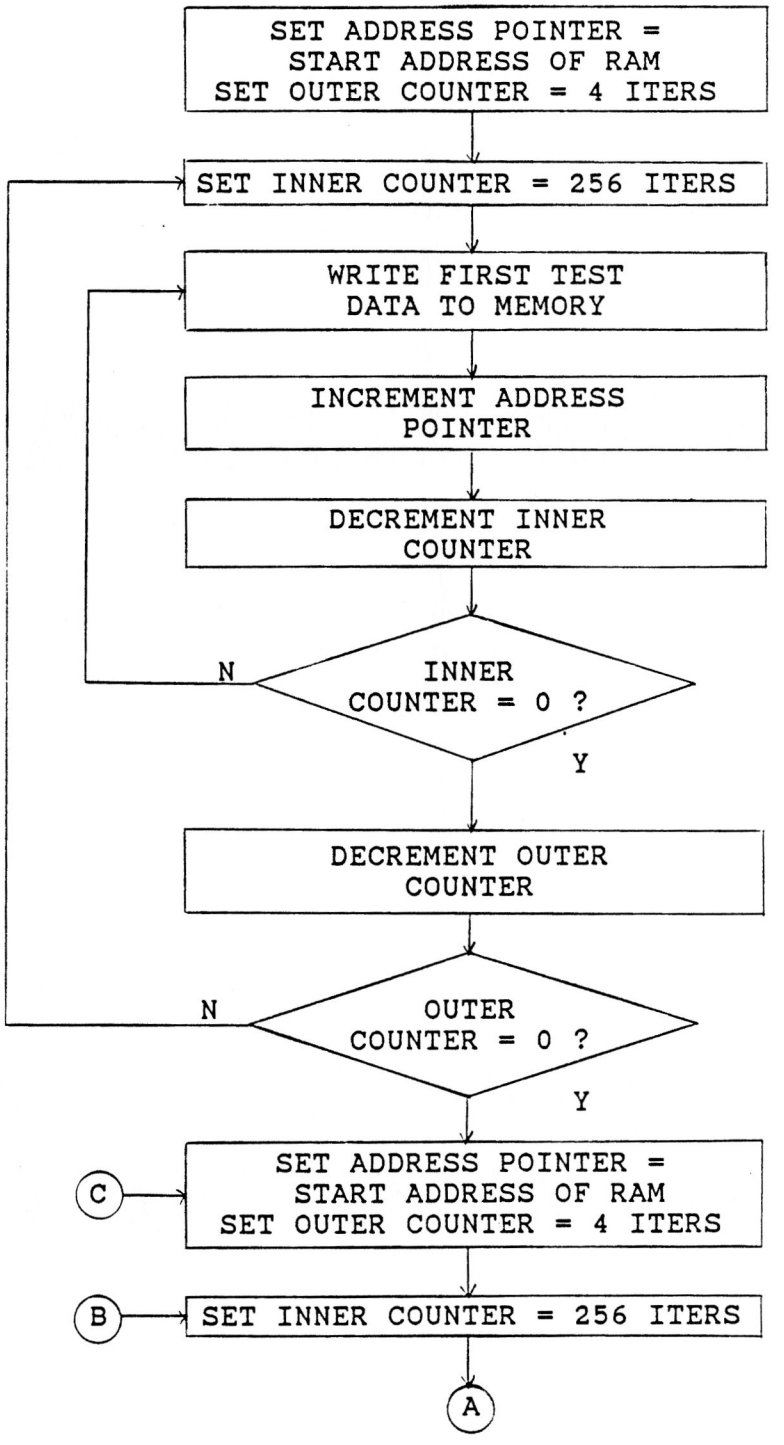


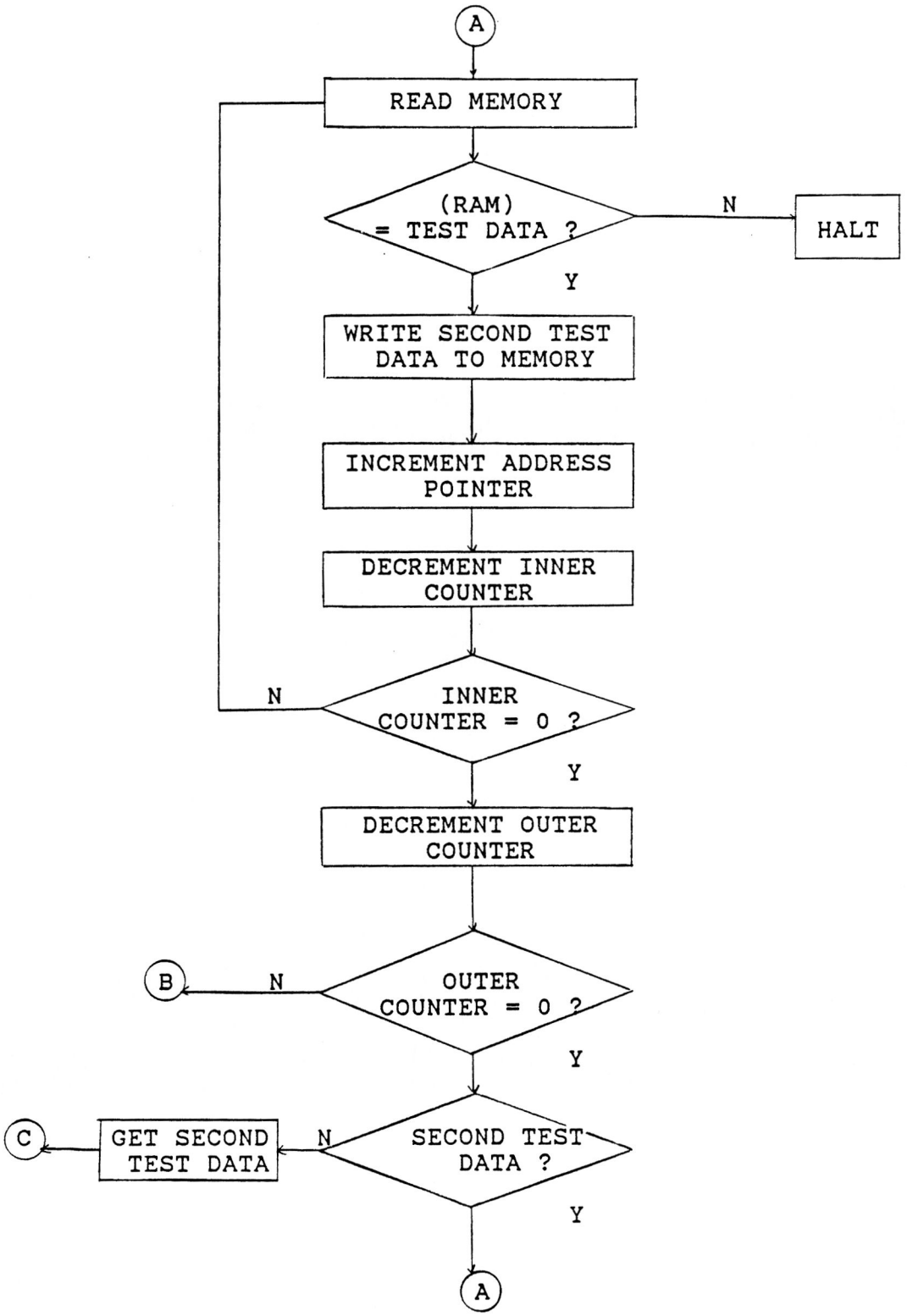
Figure A-5. Subroutine ROM

ROM

This routine constantly reads and adds the contents of the ROM. The carry generated from the addition is taken care of in another register and thus forms a 16-bit value checksum. This checksum should check with the predetermined checksum.

5. RAM


SET ADDRESS POINTER =
START ADDRESS OF RAM
SET OUTER COUNTER = 4 ITERS
SET INNER COUNTER = 256 ITERS
WRITE FIRST TEST
DATA TO MEMORY
INCREMENT ADDRESS
POINTER
DECREMENT INNER
COUNTER
INNER
COUNTER = 0 ?
N
Y
DECREMENT OUTER
COUNTER
OUTER
COUNTER = 0 ?
N
Y
C
SET ADDRESS POINTER =
START ADDRESS OF RAM
SET OUTER COUNTER = 4 ITERS
B
SET INNER COUNTER = 256 ITERS
A


A
READ MEMORY
(RAM)
= TEST DATA ?
N
HALT
Y
WRITE SECOND TEST
DATA TO MEMORY
INCREMENT ADDRESS
POINTER
DECREMENT INNER
COUNTER
N
INNER
COUNTER = 0 ?
Y
DECREMENT OUTER
COUNTER
B
N
OUTER
COUNTER = 0 ?
Y
C
GET SECOND
TEST DATA
N
SECOND TEST
DATA ?
Y
A

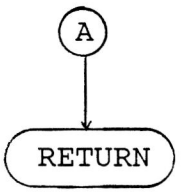


Figure A-6. Subroutine RAM

RAM

This routine constantly writes a test pattern, 01010101, in all the RAM locations. After all the RAM locations have been written into, the routine reads the RAM again and compares with the data it wrote earlier. The two data should be the same if the RAM and its associate circuitry are functioning properly.

The test is then repeated with the test data 10101010.

6. SERIAL

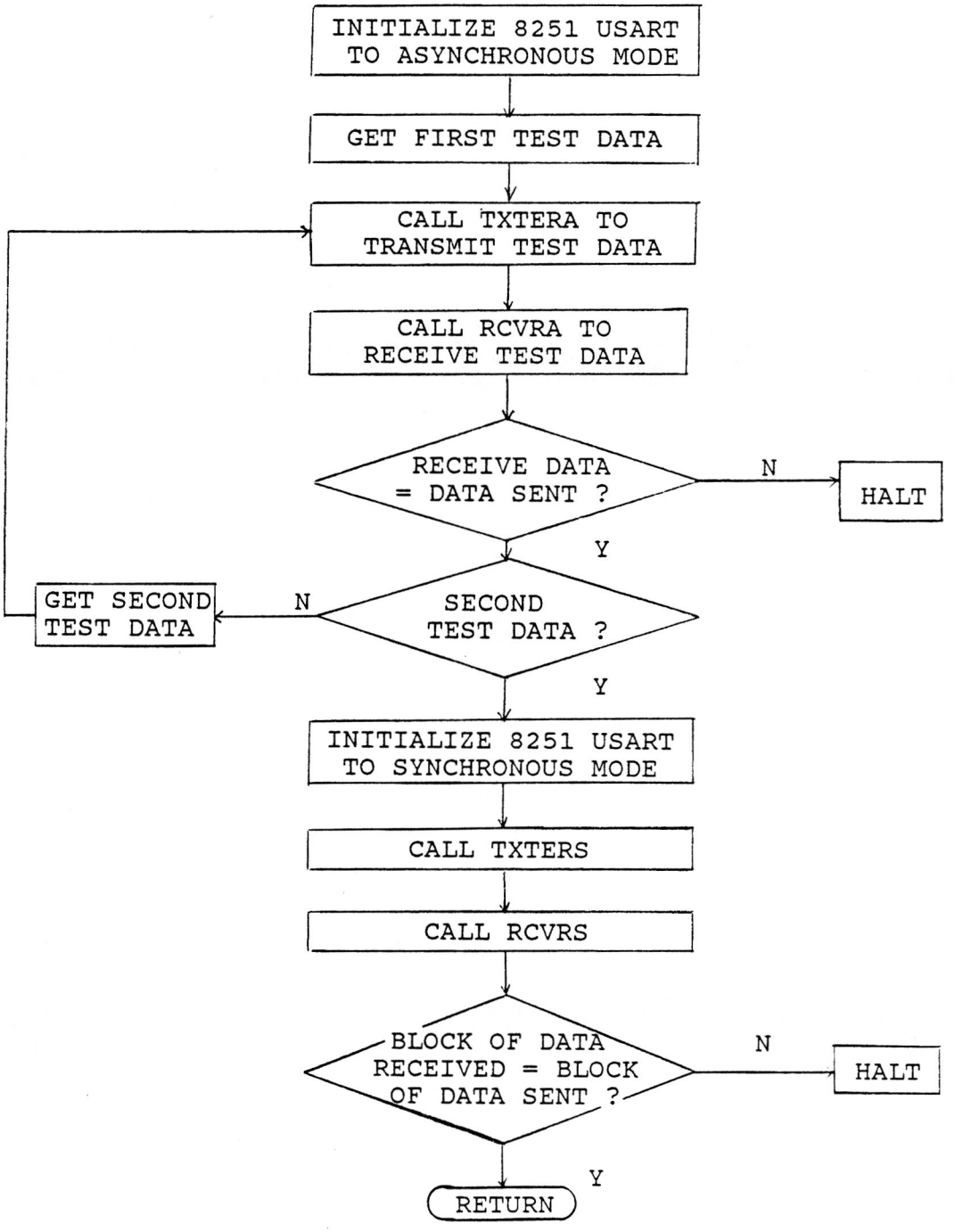


Figure A-7. Subroutine SERIAL

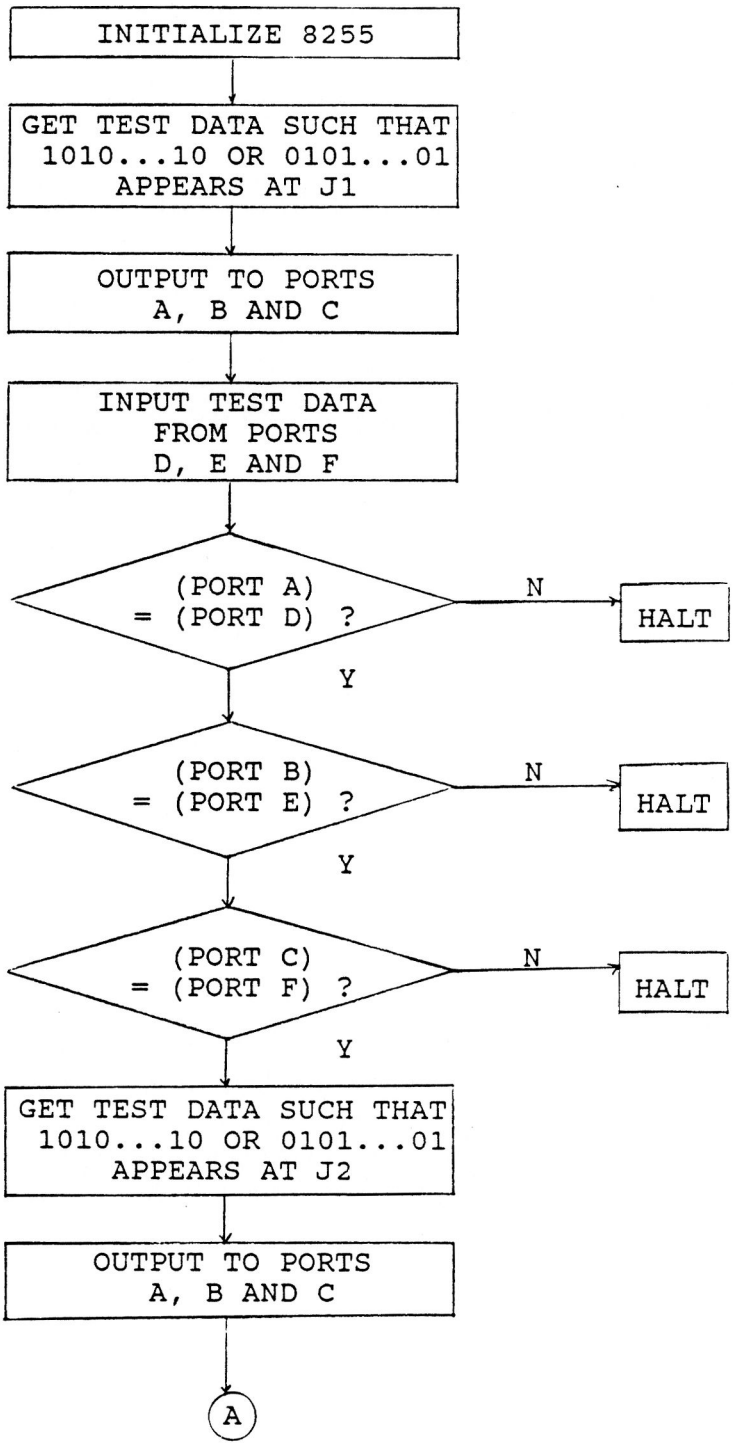
SERIAL

This routine first configures the serial port in the Asynchronous mode, and sends the test data to the TxD pin. This data is then captured by the input pin RxD. If the data sent and data received are different, then program excution will be halted.

The same test is performed with the port configure in Synchronous mode.

Other routines called by this program are TXTERA, RCVRA, TXTERS and RCVRS.

7. PARALL


INITIALIZE 8255
GET TEST DATA SUCH THAT 1010...10 OR 0101...01 APPEARS AT J1
OUTPUT TO PORTS A, B AND C
INPUT TEST DATA FROM PORTS D, E AND F
(PORT A) = (PORT D) ?
N
HALT
Y
(PORT B) = (PORT E) ?
N
HALT
Y
(PORT C) = (PORT F) ?
N
HALT
Y
GET TEST DATA SUCH THAT 1010...10 OR 0101...01 APPEARS AT J2
OUTPUT TO PORTS A, B AND C
A

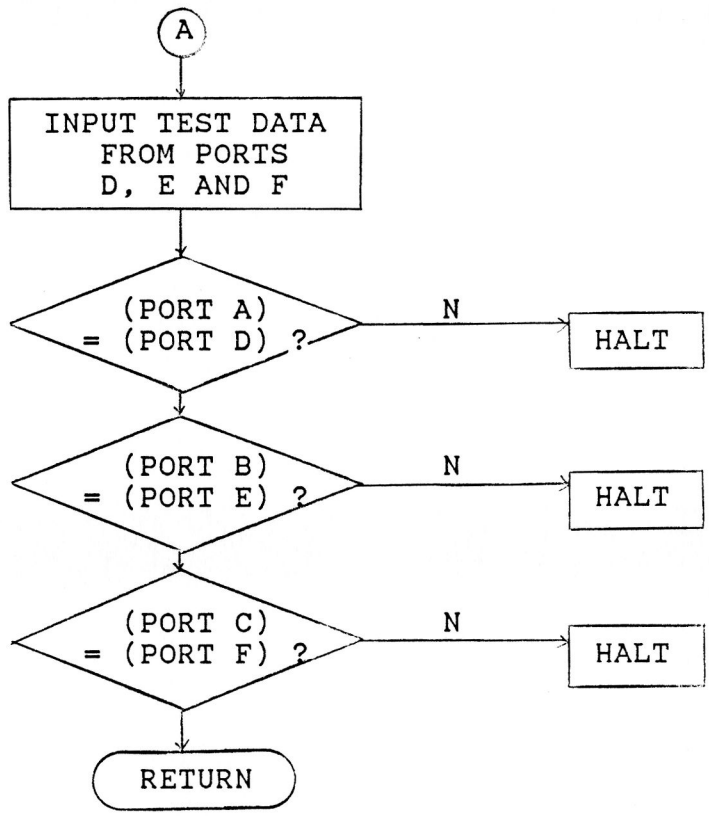


Figure A-8. Subroutine PARALL

PARALL

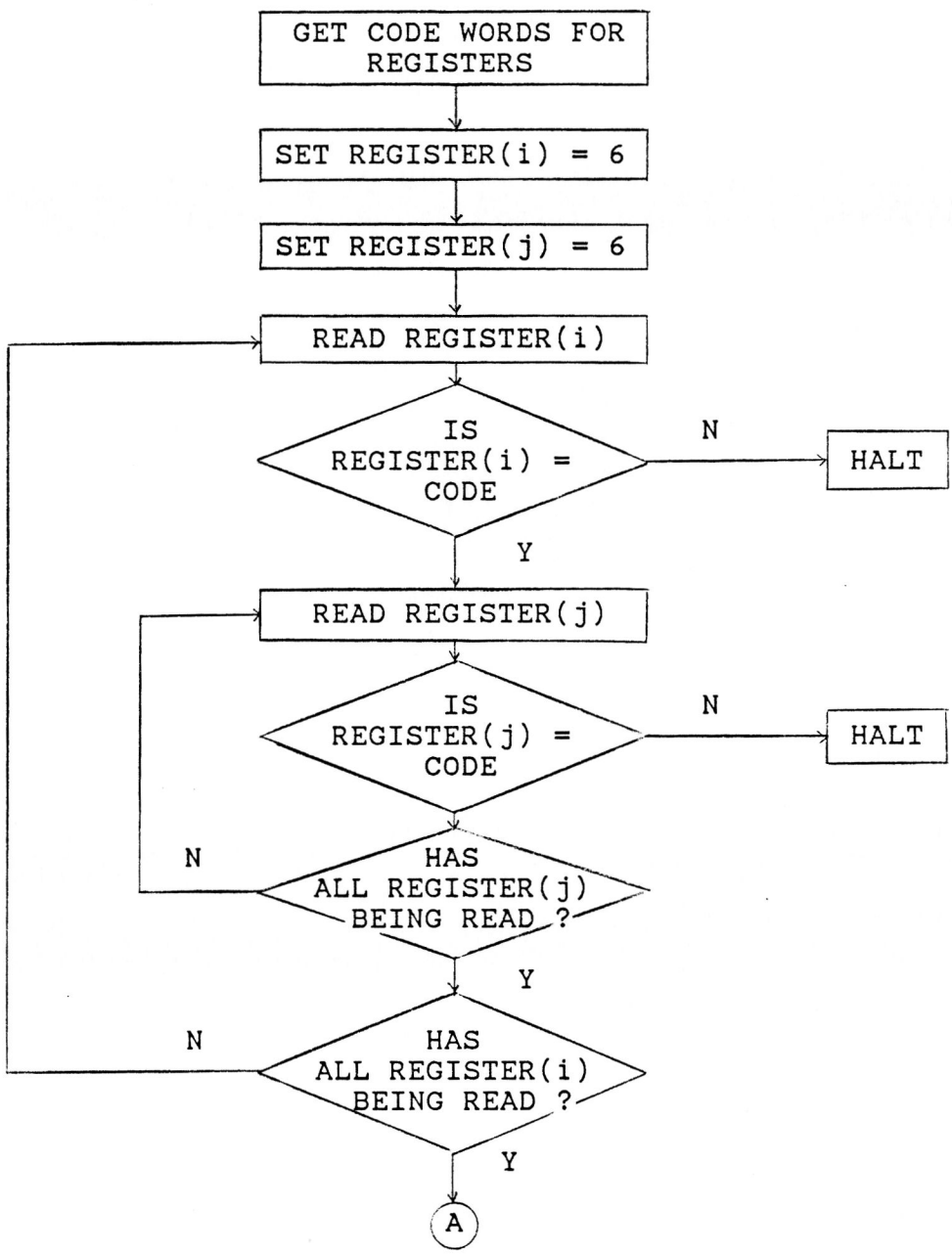
This program configures the I/O group 1 ports as outputs and the I/O group 2 ports as inputs. It then outputs test data (10101010 and 01010101, one at a time) to the I/O group 1 ports. The data sent is captured by the I/O group 2 ports. They are then compared to see if they are the same. Any discrepanies between the data sent and the data received would result in stopping the program execution.

The test is repeated but this time with the test data 1010....10 and 0101...01, one at a time) received at J2 or the I/O group 2 ports.

8. EXBUS

This routine performs the same service as the RAM test except that the address parameters are different. The starting address is $6800 and the ending address is $6FFF.

9. CHECK



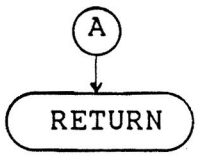


Figure A-9. Subroutine CHECK

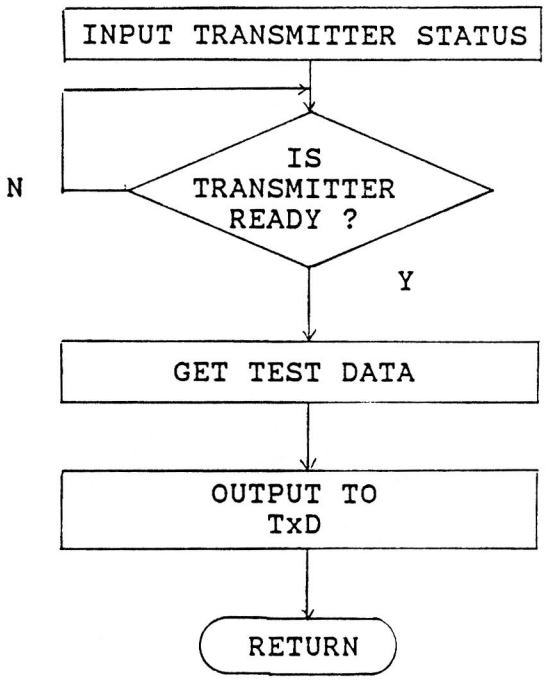
CHECK

The CHECK routine is functionally not doing anything, but merely confirming that all the registers can be written into and read from correctly.

This routine is called by MPU TEST A only.

10. TXTERA & RCVRA

A.



B.

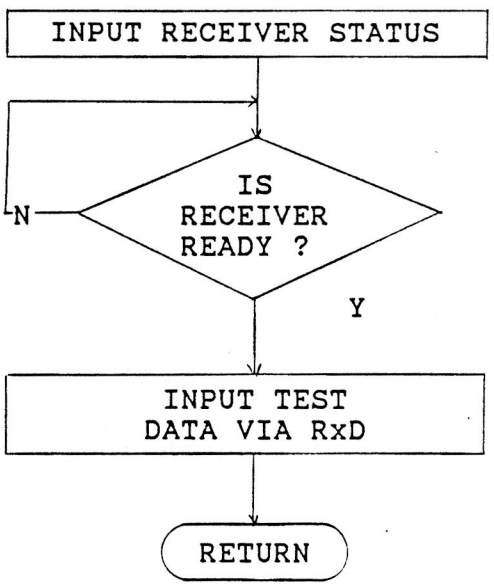


Figure A-10. A. Subroutine TXTERA B. Subroutine RCVRA

TXTERA & TXTERS

These routines are called by SERIAL. They transmit the test data sent by SERIAL to the output pin TxD. They also constantly check to see if the transmitter is ready before transmission occurs.

TXTERA is used when the serial port is configured in the Asynchronous mode, while TXTERS is used for Synchronous mode.

RCVRA & RCVRS

These routines are the same as above except that they are used for data reception.

APPENDIX B

PROGRAM LISTINGS

```
A8H80 MAIN


ISIS II 8080/8085 MACRO ASSEMBLER, V4.0        MODULE   PAGE   1

 LOC  OBJ          LINE        SOURCE STATEMENT

                      1 ;
                      2 ;*************************************************************************
                      3 ; PROGRAM NAME  :  MAIN                                                 *
                      4 ; DATE CREATED  :  JULY, 2 1986                                         *
                      5 ; VERSION       :  2.0                                                  *
                      6 ; PROGRAM INTENT:  THIS IS THE MAIN DRIVER PROGRAM WHICH CALLED ALL    *
                      7 ;                  THE OTHER TESTING ROUTINES LIKE SELF, MPU, ROM, RAM *
                      8 ;                  SERIAL, PARALL, EXBUS.                               *
                      9 ;*************************************************************************
                     10 ;
                     11 ;/////////////////////////////////////////////////////////////////////////
                     12 ;                    VARIABLES DECLARATIONS                             /
                     13 ;/////////////////////////////////////////////////////////////////////////
                     14 ;
 00EE                15          USARTD  EQU  0EEH          ; OUTPUT DATA
 00EF                16          USARTC  EQU  0EFH          ; OUTPUT CONTROL
 00EE                17          USARTI  EQU  0EEH          ; INPUT DATA
 00EF                18          USARTS  EQU  0EFH          ; INPUT STATUS
 00E4                19          PORTA   EQU  0E4H          ; PORTA ADDRESS
 00E5                20          PORTB   EQU  0E5H          ; PORTB ADDRESS
 00E6                21          PORTC   EQU  0E6H          ; PORTC ADDRESS
 00E7                22          GRP1C   EQU  0E7H          ; GROUP 1 CONTROL WORD ADDRESS
 00E8                23          PORTD   EQU  0E8H          ; PORTD ADDRESS
 00E9                24          PORTE   EQU  0E9H          ; PORTE ADDRESS
 00EA                25          PORTF   EQU  0EAH          ; PORTF ADDRESS
 00EB                26          GRP2C   EQU  0EBH          ; GROUP 2 CONTROL WORD ADDRESS
 3C00                27          START   EQU  3C00H         ; STARTING ADDRESS OF RAM
 0000                28          STARTS  EQU  0000H         ; STARTING ADDRESS OF SELF-TEST ROM
 0000                29          STARTR  EQU  0000H         ; STARTING ADDRESS OF ROM
 6800                30          STARTE  EQU  6800H         ; STARTING ADDRESS OF EXTERNAL RAM
                     31 ;
                     32 ;/////////////////////////////////////////////////////////////////////////
                     33 ;
                     34                      CSEG
 0000                35                      ORG   0000H
                     36 ;
 0000 CD1E00    C    37 MAIN:  CALL   SELF              ; TEST THE ROM WITH SELF-TEST PROGRAM
 0003 CD4B00    C    38        CALL   MPU               ; MICROPROCESSOR TEST
 0006 CDE500    D    39        CALL   ROM               ; TEST THE ROM WITH SYSTEM PROGRAM
 0009 CDAB00    D    40        CALL   RAM               ; TEST THE ONBOARD RAM
 000C CD7805    C    41        CALL   SERIAL            ; TEST THE SERIAL PORT
 000F CD0400    D    42        CALL   PARALL            ; TEST THE PARALLEL PORT
 0012 CD0D01    D    43        CALL   EXBUS             ; TEST EXTERNAL SYSTEM BUS
                     44 LIT:                          ; TOGGLE THE LED PAIR
 0015 210000        45        LXI    H,0000H           ; SET ADDRESS LINE #15
 0018 3E01          46        MVI    A,00000001B       ; SET DATA BIT 0
 001A 77            47        MOV    M,A               ; MOVE 00000001 TO ADDR. 0000H
 001B C31500    C   48        JMP    LIT               ; INFINITE LOOP JUST TO LIT LED
                     49 ;
                     50 ;*************************************************************************
                     51 ;                       SUBROUTINE: SELF                                 *
                     52 ;THIS ROUTINE TESTS THE ROM CONTAINING THE SELF TEST PROGRAM.            *
                     53 ;*************************************************************************
                     54 ;
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                    55 SELF:
001E 21FFFF         56          LXI     H,STARTS-1          ; BEGINNING ADDR. OF ROM MINUS ONE
0021 E600           57          ANI     00H                 ; CLEAR ACC., AND CARRY BIT
0023 47             58          MOV     B,A                 ; CLEAR REGISTER B
0024 4F             59          MOV     C,A                 ; CLEAR REGISTER C
                    60 REPETM:
0025 79             61          MOV     A,C
0026 23             62          INX     H                   ; INCREMENT ADDR., POINTER
0027 86             63          ADD     M                   ; ACC. = ACC. + DATA = LSByte OF CHECKSUM
0028 D22C00   C     64          JNC     NEXTM
002B 04             65          INR     B                   ; MSByte OF 16 BIT CHECKSUM
                    66 NEXTM:
002C 4F             67          MOV     C,A                 ; PUT (A) IN TEMPORARY STORAGE
002D EB             68          XCHG                        ; EXCHANGE REGISTER PAIRS HL AND DE
002E 7A             69          MOV     A,D
002F FE01           70          CPI     01H                 ; IS (A) = MSByte OF ENDING ADDRESS
0031 C22500   C     71          JNZ     REPETM              ; NO, GO AND FETCH MORE DATA
0034 7B             72          MOV     A,E
0035 FE4C           73          CPI     4CH                 ; IS (A) = LSByte OF ENDING ADDRESS
0037 C22500   C     74          JNZ     REPETM              ; NO, GO AND FETCH MORE DATA
003A 79             75          MOV     A,C                 ; RESTORE CONTENT OF REGISTER A
003B FE33           76          CPI     033H                ; THIS 16-BIT CHECKSUM IS YET TO BE DETER.,
003D C24900   C     77          JNZ     LED1                ; HALT EXECUTION IF NOT EQUAL
0040 78             78          MOV     A,B
0041 FE33           79          CPI     033H                ; THIS 16-BIT CHECKSUM IS YET TO BE DETER.,
0043 C24900   C     80          JNZ     LED1
0046 C34A00   C     81          JMP     SELEND              ; JUMP TO END OF SUBROUTINE SELF
                    82 LED1:
0049 76             83          HLT
                    84 SELEND:
004A C9             85          RET                         ; END OF SUBROUTINE SELF
                    86 ;
                    87 ;*************************************************************************
                    88 ;                          SUBROUTINE: MPU                              *
                    89 ;THIS ROUTINE TESTS THE MICROPROCESSOR IN TWO STEPS:                    *
                    90 ;   MICROPROCESSOR TEST PROCEDURE A: REGISTERS CHECK TEST              *
                    91 ;   MICROPROCESSOR TEST PROCEDURE B: OTHER INSTRUCTIONS TEST           *
                    92 ;*************************************************************************
                    93 ;
                    94 MPU:
004B CD5200   C     95          CALL    MPU1                ; MICROPROCESSOR REGISTER READ TEST
004E CD6B00   C     96          CALL    MPU2                ; MICROPROCESSOR OTHER INSTR., TEST
                    97 ;
0051 C9             98          RET                         ; EXIST MPU ROUTINE
                    99 ;
                   100 ;*************************************************************************
                   101 ;                          SUBROUTINE: MPU1                             *
                   102 ;THIS ROUTINE PERFORMS THE REGISTER READ TEST.                         *
                   103 ;*************************************************************************
                   104 ;
                   105 MPU1:
007F               106          CODEA   SET   01111111B     ; CODE WORD FOR REGISTER A
00BF               107          CODEB   SET   10111111B     ; CODE WORD FOR REGISTER B
00DF               108          CODEC   SET   11011111B     ; CODE WORD FOR REGISTER C
00EF               109          CODED   SET   11101111B     ; CODE WORD FOR REGISTER D
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

00F7               110         CODEE   SET   11110111B    ; CODE WORD FOR REGISTER E
00FB               111         CODEH   SET   11111011B    ; CODE WORD FOR REGISTER H
00FD               112         CODEL   SET   11111101B    ; CODE WORD FOR REGISTER L
                   113 ;
0052 CD7501    C   114         CALL    CHECK              ; CALL REGISTER TEST ROUTINE
                   115 ;
00BF               116         CODEA   SET   10111111B
00DF               117         CODEB   SET   11011111B
00EF               118         CODEC   SET   11101111B
00F7               119         CODED   SET   11110111B
00FB               120         CODEE   SET   11111011B
00FD               121         CODEH   SET   11111101B
00FE               122         CODEL   SET   11111110B
                   123 ;
0055 CD7501    C   124         CALL    CHECK
                   125 ;
00DF               126         CODEA   SET   11011111B
00EF               127         CODEB   SET   11101111B
00F7               128         CODEC   SET   11110111B
00FB               129         CODED   SET   11111011B
00FD               130         CODEE   SET   11111101B
00FE               131         CODEH   SET   11111110B
007F               132         CODEL   SET   01111111B
                   133 ;
0058 CD7501    C   134         CALL    CHECK
                   135 ;
00EF               136         CODEA   SET   11101111B
00F7               137         CODEB   SET   11110111B
00FB               138         CODEC   SET   11111011B
00FD               139         CODED   SET   11111101B
00FE               140         CODEE   SET   11111110B
007F               141         CODEH   SET   01111111B
00BF               142         CODEL   SET   10111111B
                   143 ;
005B CD7501    C   144         CALL    CHECK
                   145 ;
00F7               146         CODEA   SET   11110111B
00FB               147         CODEB   SET   11111011B
00FD               148         CODEC   SET   11111101B
00FE               149         CODED   SET   11111110B
007F               150         CODEE   SET   01111111B
00BF               151         CODEH   SET   10111111B
00DF               152         CODEL   SET   11011111B
                   153 ;
005E CD7501    C   154         CALL    CHECK
                   155 ;
00FD               156         CODEA   SET   11111011B
00FD               157         CODEB   SET   11111101B
00FE               158         CODEC   SET   11111110B
007F               159         CODED   SET   01111111B
00BF               160         CODEE   SET   10111111B
00DF               161         CODEH   SET   11011111B
00EF               162         CODEL   SET   11101111B
                   163 ;
0061 CD7501    C   164         CALL    CHECK
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                   165 ;
00FD               166         CODEA   SET   11111101B
00FE               167         CODEB   SET   11111110B
007F               168         CODEC   SET   01111111B
00BF               169         CODED   SET   10111111B
00DF               170         CODEE   SET   11011111B
00EF               171         CODEH   SET   11101111B
00F7               172         CODEL   SET   11110111B
                   173 ;
0064 CD7501    C   174         CALL    CHECK
                   175 ;
00FE               176         CODEA   SET   11111110B
007F               177         CODEB   SET   01111111B
00BF               178         CODEC   SET   10111111B
00DF               179         CODED   SET   11011111B
00EF               180         CODEE   SET   11101111B
00F7               181         CODEH   SET   11110111B
00FB               182         CODEL   SET   11111011B
                   183 ;
0067 CD7501    C   184         CALL    CHECK
                   185 ;
006A C9            186         RET                         ; EXIST THE MPU1 ROUTINE
                   187 ;
                   188 ;*************************************************************************
                   189 ;                          SUBROUTINE: MPU2                             *
                   190 ;THIS ROUTINE TESTS THE OTHER INSTRUCTION SETS OF THE MICROPROCESSOR.  *
                   191 ;*************************************************************************
                   192 ;
                   193 MPU2:
0080               194         CODEA   SET   10000000B     ; CODE WORD FOR REGISTER A
0040               195         CODEB   SET   01000000B     ; CODE WORD FOR REGISTER B
0020               196         CODEC   SET   00100000B     ; CODE WORD FOR REGISTER C
0010               197         CODED   SET   00010000B     ; CODE WORD FOR REGISTER D
0008               198         CODEE   SET   00001000B     ; CODE WORD FOR REGISTER E
0004               199         CODEH   SET   00000100B     ; CODE WORD FOR REGISTER H
0002               200         CODEL   SET   00000010B     ; CODE WORD FOR REGISTER L
                   201 ;
006B 3E80          202         MVI   A,CODEA
006D 0640          203         MVI   B,CODEB
006F 0E20          204         MVI   C,CODEC
0071 1610          205         MVI   D,CODED
0073 1E08          206         MVI   E,CODEE
0075 2604          207         MVI   H,CODEH
0077 2E02          208         MVI   L,CODEL
                   209 ;
                   210 ;THIS PORTION OF THE MPU2 PROGRAM EXERCISE THE ADD AND ADC INSTRUCTIONS
                   211 ;
0079 37            212         STC                         ; SET CARRY BIT
007A 88            213         ADC   B                     ; ADD (B) TO ACCUMULATOR WITH CARRY
007B 81            214         ADD   C                     ; ADD (C) TO ACCUMULATOR
007C 82            215         ADD   D                     ; ADD (D) TO ACCUMULATOR
007D 83            216         ADD   E                     ; ADD (E) TO ACCU.,
007E 84            217         ADD   H                     ; ADD (H) TO ACCU.,
007F 85            218         ADD   L                     ; ADD (L) TO ACCU.,
0080 FEFF          219         CPI   0FFH                  ; COMPARE (A) WITH 11111111
```

```
LOC  OBJ          LINE      SOURCE STATEMENT

0082 C27301   C    220       JNZ   ERROR              ; IF NOT EQUAL, HALT EXECUTION
                   221 ;
0085 3E80          222       MVI   A, CODEA           ; RESTORE REGISTER A CODE WORD
0087 37            223       STC                      ; REPEAT THE ABOVE PROCESS EXCEPT THE
0088 80            224       ADD   B                  ; ADC INSTR., IS SWITCHED TO A DIFFERENT
0089 89            225       ADC   C                  ; REGISTER AND SO ON.
008A 82            226       ADD   D
008B 83            227       ADD   E
008C 84            228       ADD   H
008D 85            229       ADD   L
008E 2F            230       CMA
008F FE00          231       CPI   00H
0091 C27301   C    232 *     JNZ   ERROR
                   233 ;
0094 3E80          234       MVI   A,CODEA
0096 37            235       STC
0097 80            236       ADD   B
0098 81            237       ADD   C
0099 8A            238       ADC   D
009A 83            239       ADD   E
009B 84            240       ADD   H
009C 85            241       ADD   L
009D FEFF          242       CPI   0FFH
009F C27301   C    243       JNZ   ERROR
                   244 ;
00A2 3E80          245       MVI   A,CODEA
00A4 37            246       STC
00A5 80            247       ADD   B
00A6 81            248       ADD   C
00A7 82            249       ADD   D
00A8 8B            250       ADC   E
00A9 84            251       ADD   H
00AA 85            252       ADD   L
00AB 2F            253       CMA
00AC FE00          254       CPI   00H
00AE C27301   C    255       JNZ   ERROR
                   256 ;
00B1 3E80          257       MVI   A,CODEA
00B3 37            258       STC
00B4 80            259       ADD   B
00B5 81            260       ADD   C
00B6 82            261       ADD   D
00B7 83            262       ADD   E
00B8 8C            263       ADC   H
00B9 85            264       ADD   L
00BA FEFF          265       CPI   0FFH
00BC C27301   C    266       JNZ   ERROR
                   267 ;
00BF 3E80          268       MVI   A,CODEA
00C1 37            269       STC
00C2 80            270       ADD   B
00C3 81            271       ADD   C
00C4 82            272       ADD   D
00C5 83            273       ADD   E
00C6 84            274       ADD   H
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

00C7 8D             275         ADC   L
00C8 2F             276         CMA
00C9 FEC0           277         CPI   C0H
00CB C27301   C     278         JNZ   ERROR
                    279 ;
                    280 ;EXERCISE THE SUB AND ABB INSTRUCTIONS
                    281 ;
007F                282         CODEA   SET   01111111B    ; SET NEW CODE WORD FOR REGISTER A
                    283 ;
00CE 3E7F           284         MVI   A,CODEA
00D0 37             285         STC
00D1 98             286         SBB   B
00D2 91             287         SUB   C
00D3 92             288         SUB   D
00D4 93             289         SUB   E
00D5 94             290         SUB   H
00D6 95             291         SUB   L
00D7 2F             292         CMA
00D8 FEFF           293         CPI   0FFH
00DA C27301   C     294         JNZ   ERROR
                    295 ;
00DD 3E7F           296         MVI   A,CODEA
00DF 37             297         STC
00E0 90             298         SUB   B
00E1 99             299         SBB   C
00E2 92             300         SUB   D
00E3 93             301         SUB   E
00E4 94             302         SUB   H
00E5 95             303         SUB   L
00E6 FE00           304         CPI   00H
00E8 C27301   C     305         JNZ   ERROR
                    306 ;
00EB 3E7F           307         MVI   A,CODEA
00ED 37             308         STC
00EE 90             309         SUB   B
00EF 91             310         SUB   C
00F0 9A             311         SBB   D
00F1 93             312         SUB   E
00F2 94             313         SUB   H
00F3 95             314         SUB   L
00F4 2F             315         CMA
00F5 FEFF           316         CPI   0FFH
00F7 C27301   C     317         JNZ   ERROR
                    318 ;
00FA 3E7F           319         MVI   A,CODEA
00FC 37             320         STC
00FD 90             321         SUB   B
00FE 91             322         SUB   C
00FF 92             323         SUB   D
0100 9B             324         SBB   E
0101 94             325         SUB   H
0102 95             326         SUB   L
0103 FE00           327         CPI   00H
0105 C27301   C     328         JNZ   ERROR
                    329 ;
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0108 3E7F          330        MVI   A,CODEA
010A 37            331        STC
010B 90            332        SUB   B
010C 91            333        SUB   C
010D 92            334        SUB   D
010E 93            335        SUB   E
010F 9C            336        SBB   H
0110 95            337        SUB   L
0111 2F            338        CMA
0112 FEFF          339        CPI   0FFH
0114 C27301    C   340        JNZ   ERROR
                   341 ;
0117 3E7F          342        MVI   A,CODEA
0119 37            343        STC
011A 90            344        SUB   B
011B 91            345        SUB   C
011C 92            346        SUB   D
011D 93            347        SUB   E
011E 94            348        SUB   H
011F 9D            349        SBB   L
0120 FE00          350        CPI   00H
0122 C27301    C   351        JNZ   ERROR
                   352 ;
                   353 ;EXERCISE THE ORA AND ORI INSTRUCTIONS
                   354 ;
0080               355        CODEA   SET   10000000B    ; SET NEW CODE WORD FOR REGISTER A
                   356 ;
0125 3E80          357        MVI   A,CODEA
0127 B0            358        ORA   B
0128 B1            359        ORA   C
0129 B2            360        ORA   D
012A B3            361        ORA   E
012B B4            362        ORA   H
012C B5            363        ORA   L
012D F601          364        ORI   01H
012F FEFF          365        CPI   0FFH
0131 C27301    C   366        JNZ   ERROR
0134 2F            367        CMA
0135 FE00          368        CPI   00H
0137 C27301    C   369        JNZ   ERROR
                   370 ;
007F               371        CODEA   SET   01111111B
00BF               372        CODEB   SET   10111111B
00DF               373        CODEC   SET   11011111B
00EF               374        CODED   SET   11101111B
00F7               375        CODEE   SET   11110111B
00FB               376        CODEH   SET   11111011B
00FD               377        CODEL   SET   11111101B
                   378 ;
013A 3E7F          379        MVI   A,CODEA
013C 06BF          380        MVI   B,CODEB
013E 0EDF          381        MVI   C,CODEC
0140 16EF          382        MVI   D,CODED
0142 1EF7          383        MVI   E,CODEE
0144 26FB          384        MVI   H,CODEH
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0146 2EFD          385        MVI   L,CODEL
                   386 ;
                   387 ;EXERCISE THE ANA AND ANI INSTRUCTIONS
                   388 ;
0148 A0            389        ANA   B
0149 A1            390        ANA   C
014A A2            391        ANA   D
014B A3            392        ANA   E
014C A4            393        ANA   H
014D A5            394        ANA   L
014E E600          395        ANI   00H
0150 FE00          396        CPI   00H
0152 C27301    C   397        JNZ   ERROR
0155 2F            398        CMA
0156 FEFF          399        CPI   0FFH
0158 C27301    C   400        JNZ   ERROR
                   401 ;
                   402 ;EXERCISE THE XRA AND XRI INSTRUCTIONS
                   403 ;
015B 3E7F          404        MVI   A,CODEA              ; SET NEW CODE WORD FOR REGISTER A
                   405 ;
015D A8            406        XRA   B
015E A9            407        XRA   C
015F AA            408        XRA   D
0160 AB            409        XRA   E
0161 AC            410        XRA   H
0162 AD            411        XRA   L
0163 EE01          412        XRI   01H
0165 FEFF          413        CPI   0FFH
0167 C27301    C   414        JNZ   ERROR
016A 2F            415        CMA
016B FE00          416        CPI   00H
016D C27301    C   417        JNZ   ERROR
                   418 ;
0170 C37401    C   419        JMP   MPUEND
                   420 ;
                   421 ERROR:
0173 76            422        HLT
                   423 MPUEND:
0174 C9            424        RET                        ; EXIST THE MPU2 ROUTINE
                   425 ;
                   426 ;*************************************************************************
                   427 ;                        SUBROUTINE: CHECK (CALLED BY MPU)               *
                   428 ;THIS ROUTINE PERFORMS THE REGISTER CHECK TEST. THE SEQUENCE OF TEST     *
                   429 ;IS AS FOLLOWS:                                                         *
                   430 ;      1. ABAB,ACAC, . . . ,ALAL    2. BABA,BCBC, . . . ,BLBL         *
                   431 ;      3. CACA,CBCB, . . . ,CLCL    4. DADA,DBDB, . . . ,DLDL         *
                   432 ;                                                                      *
                   433 ;       ETC                                                            *
                   434 ;*************************************************************************
                   435 ;
                   436 CHECK:
0175 3E7F          437        MVI   A,CODEA              ; MOVE CODEA TO REGISTER A
0177 06BF          438        MVI   B,CODEB              ; MOVE CODEB TO REGISTER B
0179 0EDF          439        MVI   C,CODEC              ; MOVE CODEC TO REGISTER C
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

017B 16EF           440       MVI   D,CODED
017D 1EF7           441       MVI   E,CODEE
017F 26FB           442       MVI   H,CODEH
0181 2EFD           443       MVI   L,CODEL
                    444 ;
                    445 ;     ADAD REGISTER READ
                    446 ;
0183 FE7F           447       CPI   CODEA               ; READ (A) AND COMPARE WITH CODEA
0185 C27605    C    448       JNZ   LED
0188 78             449       MOV   A,B                 ; READ (B)
0189 B8             450       CMP   B                   ; COMPARE (B) WITH CODEB
018A C27605    C    451       JNZ   LED
018D 47             452       MOV   B,A
018E 3E7F           453       MVI   A,CODEA             ; READ (A)
0190 BF             454       CMP   A                   ; COMPARE (A) WITH CODEA
0191 C27605    C    455       JNZ   LED
0194 78             456       MOV   A,B                 ; READ (B)
0195 B8             457       CMP   B                   ; COMPARE (B) WITH CODEB
0196 C27605    C    458       JNZ   LED
                    459 ;
                    460 ;     ACAC REGISTER READ
                    461 ;
0199 3E7F           462       MVI   A,CODEA
019B FE7F           463       CPI   CODEA
019D C27605    C    464       JNZ   LED
01A0 79             465       MOV   A,C
01A1 B9             466       CMP   C
01A2 C27605    C    467       JNZ   LED
01A5 4F             468       MOV   C,A
01A6 3E7F           469       MVI   A,CODEA
01A8 FE7F           470       CPI   CODEA
01AA C27605    C    471       JNZ   LED
01AD 79             472       MOV   A,C
01AE B9             473       CMP   C
01AF C27605    C    474       JNZ   LED
                    475 ;
                    476 ;     ADAD REGISTER READ
                    477 ;
01B2 3E7F           478       MVI   A,CODEA
01B4 FE7F           479       CPI   CODEA
01B6 C27605    C    480       JNZ   LED
01B9 7A             481       MOV   A,D
01BA BA             482       CMP   D
01BB C27605    C    483       JNZ   LED
01BE 57             484       MOV   D,A
01BF 3E7F           485       MVI   A,CODEA
01C1 FE7F           486       CPI   CODEA
01C3 C27605    C    487       JNZ   LED
01C6 7A             488       MOV   A,D
01C7 BA             489       CMP   D
01C8 C27605    C    490       JNZ   LED
                    491 ;
                    492 ;     ACAE REGISTER READ
                    493 ;
01CB 3E7F           494       MVI   A,CODEA
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

01CD BF             495         CMP   A
01CE C27605   C     496         JNZ   LED
01D1 7B             497         MOV   A,E
01D2 BB             498         CMP   E
01D3 C27605   C     499         JNZ   LED
01D6 5F             500         MOV   E,A
01D7 3E7F           501         MVI   A,CODEA
01D9 FE7F           502         CPI   CODEA
01DB C27605   C     503         JNZ   LED
01DE 7B             504         MOV   A,E
01DF BB             505         CMP   E
01E0 C27605   C     506         JNZ   LED
                    507 ;
                    508 ;       AHAH REGISTER READ
                    509 ;
01E3 3E7F           510         MVI   A,CODEA
01E5 FE7F           511         CPI   CODEA
01E7 C27605   C     512         JNZ   LED
01EA 7C             513         MOV   A,H
01EB BC             514         CMP   H
01EC C27605   C     515         JNZ   LED
01EF 67             516         MOV   H,A
01F0 3E7F           517         MVI   A,CODEA
01F2 BF             518         CMP   A
01F3 C27605   C     519         JNZ   LED
01F6 7C             520         MOV   A,H
01F7 BC             521         CMP   H
01F8 C27605   C     522         JNZ   LED
                    523 ;
                    524 ;       ALAL REGISTER READ
                    525 ;
01FB 3E7F           526         MVI   A,CODEA
01FD FE7F           527         CPI   CODEA
01FF C27605   C     528         JNZ   LED
0202 7D             529         MOV   A,L
0203 BD             530         CMP   L
0204 C27605   C     531         JNZ   LED
0207 6F             532         MOV   L,A
0208 3E7F           533         MVI   A,CODEA
020A FE7F           534         CPI   CODEA
020C C27605   C     535         JNZ   LED
020F 7D             536         MOV   A,L
0210 BD             537         CMP   L
0211 C27605   C     538         JNZ   LED
                    539 ;
                    540 ;       DADA REGISTER READ
                    541 ;
0214 78             542         MOV   A,B
0215 B8             543         CMP   B
0216 C27605   C     544         JNZ   LED
0219 47             545         MOV   B,A
021A 3E7F           546         MVI   A,CODEA
021C BF             547         CMP   A
021D C27605   C     548         JNZ   LED
0220 78             549         MOV   A,B
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0221 B8             550        CMP   B
0222 C27605    C    551        JNZ   LED
0225 3E7F           552        MVI   A,COMEA
0227 BF             553        CMP   A
0228 C27605    C    554        JNZ   LED
                    555 ;
                    556 ;      BCBC REGISTER READ
                    557 ;
022B 78             558        MOV   A,B
022C B8             559        CMP   B
022D C27605    C    560        JNZ   LED
0230 47             561        MOV   B,A
0231 79             562        MOV   A,C
0232 B9             563        CMP   C
0233 C27605    C    564        JNZ   LED
0236 4F             565        MOV   C,A
0237 78             566        MOV   A,B
0238 B8             567        CMP   B
0239 C27605    C    568        JNZ   LED
023C 47             569        MOV   B,A
023D 79             570        MOV   A,C
023E B9             571        CMP   C
023F C27605    C    572        JNZ   LED
0242 4F             573        MOV   C,A
                    574 ;
                    575 ;      DDDD REGISTER READ
                    576 ;
0243 78             577        MOV   A,B
0244 B8             578        CMP   B
0245 C27605    C    579        JNZ   LED
0248 47             580        MOV   B,A
0249 7A             581        MOV   A,D
024A BA             582        CMP   D
024B C27605    C    583        JNZ   LED
024E 57             584        MOV   D,A
024F 78             585        MOV   A,B
0250 B8             586        CMP   B
0251 C27605    C    587        JNZ   LED
0254 47             588        MOV   B,A
0255 7A             589        MOV   A,D
0256 BA             590        CMP   D
0257 C27605    C    591        JNZ   LED
025A 57             592        MOV   D,A
                    593 ;
                    594 ;      BEBE REGISTER READ
                    595 ;
025B 78             596        MOV   A,B
025C B8             597        CMP   B
025D C27605    C    598        JNZ   LED
0260 47             599        MOV   B,A
0261 7B             600        MOV   A,E
0262 BB             601        CMP   E
0263 C27605    C    602        JNZ   LED
0266 5F             603        MOV   E,A
0267 78             604        MOV   A,B
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0268 B8           605         CMP   B
0269 C27605   C   606         JNZ   LCD
026C 47           607         MOV   B,A
026D 7D           608         MOV   A,E
026E BB           609         CMP   E
026F C27605   C   610         JNZ   LCD
0272 5F           611         MOV   E,A
                  612 ;
                  613 ;       BHBH REGISTER READ
                  614 ;
0273 78           615         MOV   A,B
0274 B8           616         CMP   B
0275 C27605   C   617         JNZ   LCD
0278 47           618         MOV   B,A
0279 7C           619         MOV   A,H
027A BC           620         CMP   H
027B C27605   C   621         JNZ   LCD
027E 67           622         MOV   H,A
027F 78           623         MOV   A,B
0280 B8           624         CMP   B
0281 C27605   C   625         JNZ   LCD
0284 47           626         MOV   B,A
0285 7C           627         MOV   A,H
0286 BC           628         CMP   H
0287 C27605   C   629         JNZ   LCD
028A 67           630         MOV   H,A
                  631 ;
                  632 ;       BLBL REGISTER READ
                  633 ;
028B 78           634         MOV   A,B
028C B8           635         CMP   B
028D C27605   C   636         JNZ   LCD
0290 47           637         MOV   B,A
0291 7D           638         MOV   A,L
0292 BD           639         CMP   L
0293 C27605   C   640         JNZ   LCD
0296 6F           641         MOV   L,A
0297 78           642         MOV   A,B
0298 B8           643         CMP   B
0299 C27605   C   644         JNZ   LCD
029C 47           645         MOV   B,A
029D 7D           646         MOV   A,L
029E BD           647         CMP   L
029F C27605   C   648         JNZ   LCD
02A2 6F           649         MOV   L,A
                  650 ;
                  651 ;       CACA REGISTER READ
                  652 ;
02A3 79           653         MOV   A,C
02A4 B9           654         CMP   C
02A5 C27605   C   655         JNZ   LCD
02A8 4F           656         MOV   C,A
02A9 3E7F         657         MVI   A,CODCA
02AB BF           658         CMP   A
02AC C27605   C   659         JNZ   LCD
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

02AF 79            660        MOV   A,C
02B0 B9            661        CMP   C
02B1 C27605   C    662        JNZ   LED
02B4 4F            663        MOV   C,A
02B5 3E7F          664        MVI   A,CODEA
02B7 BF            665        CMP   A
02B8 C27605   C    666        JNZ   LED
                   667 ;
                   668 ;      CBCD REGISTER READ
                   669 ;
02BB 79            670        MOV   A,C
02BC B9            671        CMP   C
02BD C27605   C    672        JNZ   LED
02C0 4F            673        MOV   C,A
02C1 78            674        MOV   A,B
02C2 B8            675        CMP   B
02C3 C27605   C    676        JNZ   LED
02C6 47            677        MOV   B,A
02C7 79            678        MOV   A,C
02C8 B9            679        CMP   C
02C9 C27605   C    680        JNZ   LED
02CC 4F            681        MOV   C,A
02CD 78            682        MOV   A,B
02CE B8            683        CMP   B
02CF C27605   C    684        JNZ   LED
02D2 47            685        MOV   B,A
                   686 ;
                   687 ;      CBCD REGISTER READ
                   688 ;
02D3 79            689        MOV   A,C
02D4 B9            690        CMP   C
02D5 C27605   C    691        JNZ   LED
02D8 4F            692        MOV   C,A
02D9 7A            693        MOV   A,D
02DA BA            694        CMP   D
02DB C27605   C    695        JNZ   LED
02DE 57            696        MOV   D,A
02DF 79            697        MOV   A,C
02E0 B9            698        CMP   C
02E1 C27605   C    699        JNZ   LED
02E4 4F            700        MOV   C,A
02E5 7A            701        MOV   A,D
02E6 BA            702        CMP   D
02E7 C27605   C    703        JNZ   LED
02EA 57            704        MOV   D,A
                   705 ;
                   706 ;      CECE REGISTER READ
                   707 ;
02EB 79            708        MOV   A,C
02EC B9            709        CMP   C
02ED C27605   C    710        JNZ   LED
02F0 4F            711        MOV   C,A
02F1 7B            712        MOV   A,E
02F2 BB            713        CMP   E
02F3 C27605   C    714        JNZ   LED
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

02F6 5F             715         MOV   E,A
02F7 79             716         MOV   A,C
02F8 B9             717         CMP   C
02F9 C27605    C    718         JNZ   LED
02FC 4F             719         MOV   C,A
02FD 7D             720         MOV   A,E
02FE BD             721         CMP   E
02FF C27605    C    722         JNZ   LED
0302 5F             723         MOV   E,A
                    724 ;
                    725 ;       CHCH REGISTER READ
                    726 ;
0303 79             727         MOV   A,C
0304 B9             728         CMP   C
0305 C27605    C    729         JNZ   LED
0308 4F             730         MOV   C,A
0309 7C             731         MOV   A,H
030A BC             732         CMP   H
030B C27605    C    733         JNZ   LED
030E 67             734         MOV   H,A
030F 79             735         MOV   A,C
0310 B9             736         CMP   C
0311 C27605    C    737         JNZ   LED
0314 4F             738         MOV   C,A
0315 7C             739         MOV   A,H
0316 BC             740         CMP   H
0317 C27605    C    741         JNZ   LED
031A 67             742         MOV   H,A
                    743 ;
                    744 ;       CLCL REGISTER READ
                    745 ;
031B 79             746         MOV   A,C
031C B9             747         CMP   C
031D C27605    C    748         JNZ   LED
0320 4F             749         MOV   C,A
0321 7D             750         MOV   A,L
0322 BD             751         CMP   L
0323 C27605    C    752         JNZ   LED
0326 6F             753         MOV   L,A
0327 79             754         MOV   A,C
0328 B9             755         CMP   C
0329 C27605    C    756         JNZ   LED
032C 4F             757         MOV   C,A
032D 7D             758         MOV   A,L
032E BD             759         CMP   L
032F C27605    C    760         JNZ   LED
0332 6F             761         MOV   L,A
                    762 ;
                    763 ;       BABA REGISTER READ
                    764 ;
0333 7A             765         MOV   A,B
0334 BA             766         CMP   B
0335 C27605    C    767         JNZ   LED
0338 57             768         MOV   B,A
0339 3E7F           769         MVI   A,7FH
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

033B BF           770         CMP   A
033C C27605   C   771         JNZ   LED
033F 7A           772         MOV   A,D
0340 BA           773         CMP   D
0341 C27605   C   774         JNZ   LED
0344 57           775         MOV   D,A
0345 3E7F         776         MVI   A,CODEA
0347 BF           777         CMP   A
0348 C27605   C   778         JNZ   LED
                  779 ;
                  780 ;       DDDD REGISTER READ
                  781 ;
034B 7A           782         MOV   A,D
034C BA           783         CMP   D
034D C27605   C   784         JNZ   LED
0350 57           785         MOV   D,A
0351 78           786         MOV   A,B
0352 B8           787         CMP   B
0353 C27605   C   788         JNZ   LED
0356 47           789         MOV   B,A
0357 7A           790         MOV   A,D
0358 BA           791         CMP   D
0359 C27605   C   792         JNZ   LED
035C 57           793         MOV   D,A
035D 78           794         MOV   A,B
035E B8           795         CMP   B
035F C27605   C   796         JNZ   LED
0362 47           797         MOV   B,A
                  798 ;
                  799 ;       DCBC REGISTER READ
                  800 ;
0363 7A           801         MOV   A,D
0364 BA           802         CMP   D
0365 C27605   C   803         JNZ   LED
0368 57           804         MOV   D,A
0369 79           805         MOV   A,C
036A B9           806         CMP   C
036B C27605   C   807         JNZ   LED
036E 4F           808         MOV   C,A
036F 7A           809         MOV   A,D
0370 BA           810         CMP   D
0371 C27605   C   811         JNZ   LED
0374 57           812         MOV   D,A
0375 79           813         MOV   A,C
0376 B9           814         CMP   C
0377 C27605   C   815         JNZ   LED
037A 4F           816         MOV   C,A
                  817 ;
                  818 ;       DCDE REGISTER READ
                  819 ;
037B 7A           820         MOV   A,D
037C BA           821         CMP   D
037D C27605   C   822         JNZ   LED
0380 57           823         MOV   D,A
0381 7B           824         MOV   A,E
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0382 BD            825        CMP   E
0383 C27605    C   826        JNZ   LED
0386 5F            827        MOV   E,A
0387 7A            828        MOV   A,D
0388 BA            829        CMP   D
0389 C27605    C   830        JNZ   LED
038C 57            831        MOV   D,A
038D 7D            832        MOV   A,E
038E BB            833        CMP   E
038F C27605    C   834        JNZ   LED
0392 5F            835        MOV   E,A
                   836 ;
                   837 ;       DIDH REGISTER READ
                   838 ;
0393 7A            839        MOV   A,D
0394 BA            840        CMP   D
0395 C27605    C   841        JNZ   LED
0398 57            842        MOV   D,A
0399 7C            843        MOV   A,H
039A BC            844        CMP   H
039B C27605    C   845        JNZ   LED
039E 67            846        MOV   H,A
039F 7A            847        MOV   A,D
03A0 BA            848        CMP   D
03A1 C27605    C   849        JNZ   LED
03A4 57            850        MOV   D,A
03A5 7C            851        MOV   A,H
03A6 BC            852        CMP   H
03A7 C27605    C   853        JNZ   LED
03AA 67            854        MOV   H,A
                   855 ;
                   856 ;       DLDL REGISTER READ
                   857 ;
03AB 7A            858        MOV   A,D
03AC BA            859        CMP   D
03AD C27605    C   860        JNZ   LED
03B0 57            861        MOV   D,A
03B1 7D            862        MOV   A,L
03B2 BD            863        CMP   L
03B3 C27605    C   864        JNZ   LED
03B6 6F            865        MOV   L,A
03B7 7A            866        MOV   A,D
03B8 BA            867        CMP   D
03B9 C27605    C   868        JNZ   LED
03BC 57            869        MOV   D,A
03BD 7D            870        MOV   A,L
03BE BD            871        CMP   L
03BF C27605    C   872        JNZ   LED
03C2 6F            873        MOV   L,A
                   874 ;
                   875 ;       EAEA REGISTER READ
                   876 ;
03C3 7B            877        MOV   A,E
03C4 BB            878        CMP   E
03C5 C27605    C   879        JNZ   LED
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

03C8 5F             880        MOV   E,A
03C9 3E7F           881        MVI   A,CODEA
03CB BF             882        CMP   A
03CC C27605   C     883        JNZ   LED
03CF 7B             884        MOV   A,E
03D0 BB             885        CMP   E
03D1 C27605   C     886        JNZ   LED
03D4 5F             887        MOV   E,A
03D5 3E7F           888        MVI   A,CODEA
03D7 BF             889        CMP   A
03D8 C27605   C     890        JNZ   LED
                    891 ;
                    892 ;      EBEB REGISTER READ
                    893 ;
03DB 7B             894        MOV   A,E
03DC BB             895        CMP   E
03DD C27605   C     896        JNZ   LED
03E0 5F             897        MOV   E,A
03E1 78             898        MOV   A,B
03E2 B8             899        CMP   B
03E3 C27605   C     900        JNZ   LED
03E6 47             901        MOV   B,A
03E7 7B             902        MOV   A,E
03E8 BB             903        CMP   E
03E9 C27605   C     904        JNZ   LED
03EC 5F             905        MOV   E,A
03ED 78             906        MOV   A,B
03EE B8             907        CMP   B
03EF C27605   C     908        JNZ   LED
03F2 47             909        MOV   B,A
                    910 ;
                    911 ;      ECEC REGISTER READ
                    912 ;
03F3 7B             913        MOV   A,E
03F4 BB             914        CMP   E
03F5 C27605   C     915        JNZ   LED
03F8 5F             916        MOV   E,A
03F9 79             917        MOV   A,C
03FA B9             918        CMP   C
03FB C27605   C     919        JNZ   LED
03FE 4F             920        MOV   C,A
03FF 7B             921        MOV   A,E
0400 BB             922        CMP   E
0401 C27605   C     923        JNZ   LED
0404 5F             924        MOV   E,A
0405 79             925        MOV   A,C
0406 B9             926        CMP   C
0407 C27605   C     927        JNZ   LED
040A 4F             928        MOV   C,A
                    929 ;
                    930 ;      EDED REGISTER READ
                    931 ;
040B 7B             932        MOV   A,E
040C BB             933        CMP   E
040D C27605   C     934        JNZ   LED
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0410 5F            935        MOV   E,A
0411 7A            936        MOV   A,D
0412 BA            937        CMP   D
0413 C27605    C   938        JNZ   LED
0416 57            939        MOV   D,A
0417 7B            940        MOV   A,E
0418 BB            941        CMP   E
0419 C27605    C   942        JNZ   LED
041C 5F            943        MOV   E,A
041D 7A            944        MOV   A,D
041E BA            945        CMP   D
041F C27605    C   946        JNZ   LED
0422 57            947        MOV   D,A
                   948 ;
                   949 ;       EIEH REGISTER READ
                   950 ;
0423 7B            951        MOV   A,E
0424 BB            952        CMP   E
0425 C27605    C   953        JNZ   LED
0428 5F            954        MOV   E,A
0429 7C            955        MOV   A,H
042A BC            956        CMP   H
042B C27605    C   957        JNZ   LED
042E 67            958        MOV   H,A
042F 7B            959        MOV   A,E
0430 BB            960        CMP   E
0431 C27605    C   961        JNZ   LED
0434 5F            962        MOV   E,A
0435 7C            963        MOV   A,H
0436 BC            964        CMP   H
0437 C27605    C   965        JNZ   LED
043A 67            966        MOV   H,A
                   967 ;
                   968 ;       ELEL REGISTER READ
                   969 ;
043B 7B            970        MOV   A,E
043C BB            971        CMP   E
043D C27605    C   972        JNZ   LED
0440 5F            973        MOV   E,A
0441 7D            974        MOV   A,L
0442 BD            975        CMP   L
0443 C27605    C   976        JNZ   LED
0446 6F            977        MOV   L,A
0447 7B            978        MOV   A,E
0448 BB            979        CMP   E
0449 C27605    C   980        JNZ   LED
044C 5F            981        MOV   E,A
044D 7D            982        MOV   A,L
044E BD            983        CMP   L
044F C27605    C   984        JNZ   LED
0452 6F            985        MOV   L,A
                   986 ;
                   987 ;       HAHA REGISTER READ
                   988 ;
0453 7C            989        MOV   A,H
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0454 BC            990        CMP   H
0455 C27605    C   991        JNZ   LED
0458 67            992        MOV   H,A
0459 3E7F          993        MVI   A,CODEA
045B BF            994        CMP   A
045C C27605    C   995        JNZ   LED
045F 7C            996        MOV   A,H
0460 BC            997        CMP   H
0461 C27605    C   998        JNZ   LED
0464 67            999        MOV   H,A
0465 3E7F         1000        MVI   A,CODEA
0467 BF           1001        CMP   A
0468 C27605    C  1002        JNZ   LED
                  1003 ;
                  1004 ;       HDHB REGISTER READ
                  1005 ;
046B 7C           1006        MOV   A,H
046C BC           1007        CMP   H
046D C27605    C  1008        JNZ   LED
0470 67           1009        MOV   H,A
0471 78           1010        MOV   A,B
0472 B8           1011        CMP   B
0473 C27605    C  1012        JNZ   LED
0476 47           1013        MOV   B,A
0477 7C           1014        MOV   A,H
0478 BC           1015        CMP   H
0479 C27605    C  1016        JNZ   LED
047C 67           1017        MOV   H,A
047D 78           1018        MOV   A,B
047E B8           1019        CMP   B
047F C27605    C  1020        JNZ   LED
0482 47           1021        MOV   B,A
                  1022 ;
                  1023 ;       HCHC REGISTER READ
                  1024 ;
0483 7C           1025        MOV   A,H
0484 BC           1026        CMP   H
0485 C27605    C  1027        JNZ   LED
0488 67           1028        MOV   H,A
0489 79           1029        MOV   A,C
048A B9           1030        CMP   C
048B C27605    C  1031        JNZ   LED
048E 4F           1032        MOV   C,A
048F 7C           1033        MOV   A,H
0490 BC           1034        CMP   H
0491 C27605    C  1035        JNZ   LED
0494 67           1036        MOV   H,A
0495 79           1037        MOV   A,C
0496 B9           1038        CMP   C
0497 C27605    C  1039        JNZ   LED
049A 4F           1040        MOV   C,A
                  1041 ;
                  1042 ;       HDHB REGISTER READ
                  1043 ;
049D 7C           1044        MOV   A,H
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

049C BC           1045        CMP   H
049D C27605   C   1046        JNZ   LED
04A0 67           1047        MOV   H,A
04A1 7A           1048        MOV   A,D
04A2 BA           1049        CMP   D
04A3 C27605   C   1050        JNZ   LED
04A6 57           1051        MOV   D,A
04A7 7C           1052        MOV   A,H
04A8 BC           1053        CMP   H
04A9 C27605   C   1054        JNZ   LED
04AC 67           1055        MOV   H,A
04AD 7A           1056        MOV   A,D
04AE BA           1057        CMP   D
04AF C27605   C   1058        JNZ   LED
04B2 57           1059        MOV   D,A
                  1060 ;
                  1061 ;      HEHE REGISTER READ
                  1062 ;
04B3 7C           1063        MOV   A,H
04B4 BC           1064        CMP   H
04B5 C27605   C   1065        JNZ   LED
04B8 67           1066        MOV   H,A
04B9 7B           1067        MOV   A,E
04BA BB           1068        CMP   E
04BB C27605   C   1069        JNZ   LED
04BE 5F           1070        MOV   E,A
04BF 7C           1071        MOV   A,H
04C0 BC           1072        CMP   H
04C1 C27605   C   1073        JNZ   LED
04C4 67           1074        MOV   H,A
04C5 7B           1075        MOV   A,E
04C6 BB           1076        CMP   E
04C7 C27605   C   1077        JNZ   LED
04CA 5F           1078        MOV   E,A
                  1079 ;
                  1080 ;      HLHL REGISTER READ
                  1081 ;
04CB 7C           1082        MOV   A,H
04CC BC           1083        CMP   H
04CD C27605   C   1084        JNZ   LED
04D0 67           1085        MOV   H,A
04D1 7B           1086        MOV   A,L
04D2 BD           1087        CMP   L
04D3 C27605   C   1088        JNZ   LED
04D6 6F           1089        MOV   L,A
04D7 7C           1090        MOV   A,H
04D8 BC           1091        CMP   H
04D9 C27605   C   1092        JNZ   LED
04DC 67           1093        MOV   H,A
04DD 7D           1094        MOV   A,L
04DE BD           1095        CMP   L
04DF C27605   C   1096        JNZ   LED
04E2 6F           1097        MOV   L,A
                  1098 ;
                  1099 ;      LALA REGISTER READ
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                  1100 ;
04E3 7D           1101        MOV   A,L
04E4 BD           1102        CMP   L
04E5 C27605   C   1103        JNZ   LED
04E8 6F           1104        MOV   L,A
04E9 3E7F         1105        MVI   A,CODEA
04EB BF           1106        CMP   A
04EC C27605   C   1107        JNZ   LED
04EF 7D           1108        MOV   A,L
04F0 BD           1109        CMP   L
04F1 C27605   C   1110        JNZ   LED
04F4 6F           1111        MOV   L,A
04F5 3E7F         1112        MVI   A,CODEA
04F7 BF           1113        CMP   A
04F8 C27605   C   1114        JNZ   LED
                  1115 ;
                  1116 ;      LDLB REGISTER READ
                  1117 ;
04FB 7D           1118        MOV   A,L
04FC BD           1119        CMP   L
04FD C27605   C   1120        JNZ   LED
0500 6F           1121        MOV   L,A
0501 78           1122        MOV   A,B
0502 B8           1123        CMP   B
0503 C27605   C   1124        JNZ   LED
0506 47           1125        MOV   B,A
0507 7D           1126        MOV   A,L
0508 BD           1127        CMP   L
0509 C27605   C   1128        JNZ   LED
050C 6F           1129        MOV   L,A
050D 78           1130        MOV   A,B
050E B8           1131        CMP   B
050F C27605   C   1132        JNZ   LED
0512 47           1133        MOV   B,A
                  1134 ;
                  1135 ;      LCLC REGISTER READ
                  1136 ;
0513 7D           1137        MOV   A,L
0514 BD           1138        CMP   L
0515 C27605   C   1139        JNZ   LED
0518 6F           1140        MOV   L,A
0519 79           1141        MOV   A,C
051A B9           1142        CMP   C
051B C27605   C   1143        JNZ   LED
051E 4F           1144        MOV   C,A
051F 7D           1145        MOV   A,L
0520 BD           1146        CMP   L
0521 C27605   C   1147        JNZ   LED
0524 6F           1148        MOV   L,A
0525 79           1149        MOV   A,C
0526 B9           1150        CMP   C
0527 C27605   C   1151        JNZ   LED
052A 4F           1152        MOV   C,A
                  1153 ;
                  1154 ;      LDLD REGISTER READ
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                  1155 ;
052B 7D           1156        MOV   A,L
052C BD           1157        CMP   L
052D C27605   C   1158        JNZ   LED
0530 6F           1159        MOV   L,A
0531 7A           1160        MOV   A,D
0532 BA           1161        CMP   D
0533 C27605   C   1162        JNZ   LED
0536 57           1163        MOV   D,A
0537 7D           1164        MOV   A,L
0538 BD           1165        CMP   L
0539 C27605   C   1166        JNZ   LED
053C 6F           1167        MOV   L,A
053D 7A           1168        MOV   A,D
053E BA           1169        CMP   D
053F C27605   C   1170        JNZ   LED
0542 57           1171        MOV   D,A
                  1172 ;
                  1173 ;      LELE REGISTER READ
                  1174 ;
0543 7D           1175        MOV   A,L
0544 BD           1176        CMP   L
0545 C27605   C   1177        JNZ   LED
0548 6F           1178        MOV   L,A
0549 7B           1179        MOV   A,E
054A BB           1180        CMP   E
054B C27605   C   1181        JNZ   LED
054E 5F           1182        MOV   E,A
054F 7D           1183        MOV   A,L
0550 BD           1184        CMP   L
0551 C27605   C   1185        JNZ   LED
0554 6F           1186        MOV   L,A
0555 7B           1187        MOV   A,E
0556 BB           1188        CMP   E
0557 C27605   C   1189        JNZ   LED
055A 5F           1190        MOV   E,A
                  1191 ;
                  1192 ;      LHLH REGISTER READ
                  1193 ;
055B 7D           1194        MOV   A,L
055C BD           1195        CMP   L
055D C27605   C   1196        JNZ   LED
0560 6F           1197        MOV   L,A
0561 7C           1198        MOV   A,H
0562 BC           1199        CMP   H
0563 C27605   C   1200        JNZ   LED
0566 67           1201        MOV   H,A
0567 7D           1202        MOV   A,L
0568 BD           1203        CMP   L
0569 C27605   C   1204        JNZ   LED
056C 6F           1205        MOV   L,A
056D 7C           1206        MOV   A,H
056E BC           1207        CMP   H
056F C27605   C   1208        JNZ   LED
0572 67           1209        MOV   H,A
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0573 C37705    C  1210        JMP   CHKEND
                  1211 ;
                  1212 LED:
0576 76           1213        HLT
                  1214 CHKEND:
0577 C9           1215        RET                         ; EXIST THE CHECK ROUTINE
                  1216 ;
                  1217 ;*****************************************************************
                  1218 ;                        SUBROUTINE: SERIAL                      *
                  1219 ;THIS ROUTINE TESTS INTEL SBC 80/10A SERIAL PORT. THE TRANSMITTER *
                  1220 ;OUTPUT IS ROUTED TO THE RECEIVER. THE TEST DATA ARE 1010101010 AND *
                  1221 ;01010101, TO ENSURE THAT THE PORT IS ABLE TO FUNCTION IN DIFFERENT *
                  1222 ;MODE OF OPERATIONS AND THERE IS NO STUCK-AT FAULTS.          .     *
                  1223 ;*****************************************************************
                  1224 ;
                  1225 ;///////////////////////////////////////////////////////////////
                  1226 ;                          TEST PART 1                          /
                  1227 ; THIS PORTION INITIALIZES THE SERIAL PORT TO ASYNCHRONOUS MODE.  /
                  1228 ;///////////////////////////////////////////////////////////////
                  1229 ;
                  1230 SERIAL:
0578 3E40         1231        MVI   A,01000000B         ; RESET/RETURN TO MODE INPUT
057A D3EF         1232        OUT   USARTC
057C 3ECE         1233        MVI   A,11001110B         ; 2 STOP/16X RATE/8-BIT DATA (MODE INSTR.)
057E D3EF         1234        OUT   USARTC
0580 3E27         1235        MVI   A,00100111B         ; ENABLE TxEN/DTR/RxE/RTS (COMMAND INSTR.)
0582 D3EF         1236        OUT   USARTC
                  1237 BEGIN:
0584 3A0000    D  1238        LDA   DATA                ; GET TEST PATTERTN
0587 4F           1239        MOV   C,A                 ; STORE IN REGISTER
0588 CDC705    C  1240        CALL  TXTERA              ; SERIAL PORT TRANSMIT
058B CDD205    C  1241        CALL  RCVRA               ; SERIAL PORT RECEIVE
058E FEAA         1242        CPI   0AAH                ; COMPARE TxD AND RxD DATA
0590 C20200    D  1243        JNZ   LED4                ; IF NOT EQUAL, DISPLAY ERROR MESSAGE
0593 3A0100    D  1244        LDA   DATA+1              ; GET NEXT TEST PATTERN
0596 CDC705    C  1245        CALL  TXTERA
0599 CDD205    C  1246        CALL  RCVRA
059C FE55         1247        CPI   55H
059E C20200    D  1248        JNZ   LED4
                  1249 ;
                  1250 ;///////////////////////////////////////////////////////////////.
                  1251 ;                          TEST PART 2                          /
                  1252 ; THIS PORTION INITIALIZES THE SERIAL PORT TO SYNCHRONOUS MODE.  /
                  1253 ;///////////////////////////////////////////////////////////////
                  1254 ;
                  1255 INIT2:
05A1 3E40         1256        MVI   A,01000000B         ; RESET/RETURN TO MODE INPUT
05A3 D3EF         1257        OUT   USARTC
05A5 3E8C         1258        MVI   A,10001100B         ; 1 SYNC CHAR/SYNDET OUTPUT/8-BIT DATA
05A7 D3EF         1259        OUT   USARTC
05A9 3E16         1260        MVI   A,00010110B         ; SYNC CHARACTER
05AB D3EF         1261        OUT   USARTC
05AD 3EA7         1262        MVI   A,10100111B         ; ENABLE HUNT/RxE/DTR/TxEN/RTS
05AF D3EF         1263        OUT   USARTC
                  1264 BEGIN2:
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

05D1 3A0000    D  1265         LDA     DATA                ; GET TEST PATTERN
05D4 4F           1266         MOV     C,A
05D5 CDDC05    C  1267         CALL    TXTERS
05D8 CDE405    C  1268         CALL    RCVRS
05DB 3A0100    D  1269         LDA     DATA+1              ; GET NEXT TEST PATTERN
05DE CDDC05    C  1270         CALL    TXTERS
05E1 CDE405    C  1271         CALL    RCVRS
05E4 C30300    D  1272         JMP     SEREND
                  1273 ;
                  1274 ;//////////////////////////////////////////////////////////////////////
                  1275 ; SUBROUTINES:  TXTERA - THIS ROUTINE CHECKS TO SEE IF THE TRANSMITTER /
                  1276 ;                        IS READY, THEN OUTPUT THE TEST DATA TO TxD.   /
                  1277 ;                RCVRA - THIS ROUTINE CHECKS TO SEE IF THE RECEIVER IS  /
                  1278 ;                        IS READY, THEN INPUT THE DATA OUPUT BY TxD VIA /
                  1279 ;                        RxD.                                          /
                  1280 ;                BOTH THESE ROUTINES ARE CALLED SERIAL ONLY.           /
                  1281 ;//////////////////////////////////////////////////////////////////////
                  1282 ;
                  1283 TXTERA:
05E7 DBEF         1284         IN      USARTS              ; CHECK INPUT STATUS
05E9 E601         1285         ANI     00000001B           ; IS TXTER READY ?
05EB CAE705    C  1286         JZ      TXTERA              ; KEEP CHECKING UNTIL IT'S READY
05EE 79           1287         MOV     A,C                 ; GET DATA
05EF D3EE         1288         OUT     USARTO              ; OUTPUT DATA TO TxD
05F1 C9           1289         RET
                  1290 ;
                  1291 RCVRA:
05F2 DBEF         1292         IN      USARTS
05F4 E602         1293         ANI     00000010B           ; IS RCVR READY ?
05F6 CAF205    C  1294         JZ      RCVRA
05F9 DBEE         1295         IN      USARTI              ; INPUT DATA
05FB C9           1296         RET
                  1297 ;
                  1298 ;//////////////////////////////////////////////////////////////////////
                  1299 ; SUBROUTINES:  TXTERS _ SAME AS ABOVE EXCEPT OPERATE IN SYNCHRONOUS MODE/
                  1300 ;                RCVRS - SAME AS ABOVE EXCEPT OPERATE IN SYNCHRONOUS MODE /
                  1301 ;                BOTH THESE ROUTINES ARE CALLED BY SERIAL ONLY.          /
                  1302 ;//////////////////////////////////////////////////////////////////////
                  1303 ;
                  1304 TXTERS:
05FC DBEF         1305         IN      USARTS
05FE E601         1306         ANI     00000001B
0600 CAFC05    C  1307         JZ      TXTERS
0603 C9           1308         RET
                  1309 RCVRS:
0604 DBEF         1310         IN      USARTS
0606 E602         1311         ANI     00000010B
0608 CA0406    C  1312         JZ      RCVRS
                  1313 AGAIN:
060B DBEF         1314         IN      USARTS              ; CHECK FOR SYNC CHARACTER
060D E640         1315         ANI     01000000B
060F CA0B06    C  1316         JZ      AGAIN
0612 DBEE         1317         IN      USARTI              ; INPUT DATA
0614 FEAA         1318         CPI     0AAH                ; COMPARE WITH TEST DATA 10101010
0616 C20306    C  1319         JNZ     SYNERR
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

05F9 DBEE         1320        IN      USARTI              ; INPUT DATA
05FB FE55         1321        CPI     55H                 ; COMPARE WITH TEST DATA 01010101
05FD C20306   C   1322        JNZ     SYNERR
0600 C30406   C   1323        JMP     SYNEND
                  1324 ;
                  1325 SYNERR:
0603 76           1326        HLT
                  1327 SYNEND:
0604 C9           1328        RET
                  1327 ;
                  1330                                DSEG
                  1331 ;
                  1332 DATA:
0000 AA           1333        DB      0AAH,55H
0001 55
                  1334 ;
                  1335 LED4:
0002 76           1336        HLT
                  1337 SEREND:
0003 C9           1338        RET                         ; EXIST SERIAL ROUTINE
                  1339 ;
                  1340 ;*****************************************************************************
                  1341 ;                         SUBROUTINE:  PARALL                                *
                  1342 ;THIS ROUTINE TESTS THE INTEL SBC 80/10A PARALLEL PORTS. ALL THE GROUP *
                  1343 ;1 PORTS ARE CONFIGURED AS OUTPUTS AND GROUP 2 PORTS AS INPUTS.        *
                  1344 ;*****************************************************************************
                  1345 ;
                  1346 PARALL:
0004 3E80         1347        MVI     A,10000000B         ; MODE 0/PORTS A, B AND C OUTPUTS
0006 D3E7         1348        OUT     GRP1C               ; OUTPUT TO GROUP 1 CONTROL WORD
0008 3E9B         1349        MVI     A,10011011B         ; MODE 0/PORTS D, E AND F INPUTS
000A D3EB         1350        OUT     GRP2C               ; OUTPUT TO GROUP 2 CONTROL WORD
                  1351 ;
000C 3AA000   D   1352        LDA     DATAJ1              ; GET FIRST TEST DATA
000F D3E4         1353        OUT     PORTA               ; OUTPUT TO PORT A
0011 3AA100   D   1354        LDA     DATAJ1+1
0014 D3E5         1355        OUT     PORTB
0016 3AA200   D   1356        LDA     DATAJ1+2
0019 D3E6         1357        OUT     PORTC
001B DBE8         1358        IN      PORTD
001D FE9A         1359        CPI     09AH                ; COMPARE OUTPUT AND RECEIVED DATA
001F C29F00   D   1360        JNZ     LED5                ; IF NOT EQUAL, DISPLAY ERROR MESSAGE
0022 DBE9         1361        IN      PORTE
0024 FEA5         1362        CPI     0A5H
0026 C29F00   D   1363        JNZ     LED5
0029 DBEA         1364        IN      PORTF
002B FE27         1365        CPI     027H
002D C20200   D   1366        JNZ     LED4
0030 3AA300   D   1367        LDA     DATAJ2              ; GET SECOND TEST DATA
0033 D3E4         1368        OUT     PORTA               ; OUTPUT TO PORT A
0035 3AA400   D   1369        LDA     DATAJ2+1
0038 D3E5         1370        OUT     PORTB
003A 3AA500   D   1371        LDA     DATAJ2+2
003D D3E6         1372        OUT     PORTC
003F DBE8         1373        IN      PORTD
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

0041 FE65         1374        CPI     065H
0043 C29F00   D   1375        JNZ     LED5
0046 DBE9         1376        IN      PORTE
0048 FE5A         1377        CPI     05AH
004A C29F00   D   1378        JNZ     LED5
004D DBEA         1379        IN      PORTF
004F FED8         1380        CPI     0D8H
0051 C29F00   D   1381        JNZ     LED5
0054 3AA600   D   1382        LDA     DATAJ3              ; GET THIRD TEST DATA
0057 D3E4         1383        OUT     PORTA
0059 3AA700   D   1384        LDA     DATAJ3+1
005C D3E5         1385        OUT     PORTB
005E 3AA800   D   1386        LDA     DATAJ3+2
0061 D3E6         1387        OUT     PORTC
0063 DBE8         1388        IN      PORTD
0065 FE5A         1389        CPI     05AH
0067 C29F00   D   1390        JNZ     LED5                ; GET FOURTH TEST DATA
006A DBE9         1391        IN      PORTE
006C FEA5         1392        CPI     0A5H
006E C29F00   D   1393        JNZ     LED5
0071 DBEA         1394        IN      PORTF
0073 FE5A         1395        CPI     05AH
0075 C29F00   D   1396        JNZ     LED5
0078 3AA900   D   1397        LDA     DATAJ4
007B D3E4         1398        OUT     PORTA
007D 3AAA00   D   1399        LDA     DATAJ4+1
0080 D3E5         1400        OUT     PORTB
0082 3A0200   D   1401        LDA     DATA+2
0085 D3E6         1402        OUT     PORTC
0087 DBE8         1403        IN      PORTD
0089 FEA5         1404        CPI     0A5H
008B C29F00   D   1405        JNZ     LED5
008E DBE9         1406        IN      PORTE
0090 FE5A         1407        CPI     05AH
0092 C29F00   D   1408        JNZ     LED5
0095 DBEA         1409        IN      PORTF
0097 FE5A         1410        CPI     05AH
0099 C29F00   D   1411        JNZ     LED5
009C C3AC00   D   1412        JMP     PAREND
                  1413 ;
                  1414 LED5:
009F 76           1415        HLT
                  1416 ;
                  1417            DSEG
                  1418 ;
                  1417 DATAJ1:
00A0 9A           1420        DB      09AH,0A5H,027H
00A1 A5
00A2 27
                  1421 DATAJ2:
00A3 65           1422        DB      065H,05AH,0D8H
00A4 5A
00A5 D8
                  1423 DATAJ3:
00A6 5A           1424        DB      05AH,0A5H,0A5H
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

00A7 A5
00A8 A5
                  1425 DATAJ4:
00A9 A5           1426        DB    0A5H,05AH,05AH
00AA 5A
00AB 5A
                  1427 ;
                  1428 PARENB:
00AC C9           1429        RET                         ; EXIST PARALL ROUTINE
                  1430
                  1431
                  1432 ;*****************************************************************
                  1433 ;                       SUBROUTINE: RAM                         *
                  1434 ;THIS ROUTINE TESTS THE 1K BYTES OF RAM ON THE INTEL SBC 80/10A BOARD. *
                  1435 ;THE FIRST TEST DATA 10101010 IS WRITTEN TO ALL THE RAM LOCATIONS AND  *
                  1436 ;THEN IS READ AGAIN. THE TEST IS REPEATED WITH THE SECOND TEST DATA    *
                  1437 ;01010101. THESE TESTS ARE DONE TO ENSURE THAT ALL RAM MEMORY CELLS    *
                  1438 ;ARE ACCESSIBLE AND THAT THERE IS NO STUCK AT FAULT.                   *
                  1439 ;*****************************************************************
                  1440 ;
                  1441 RAM:
00AD 21003C       1442        LXI   H,START             ; STARTING ADDRESS OF RAM
00B0 3E55         1443        MVI   A,01010101B         ; FIRST TEST DATA
                  1444 OUTER:
00B2 0604         1445        MVI   B,04H               ; 4 ITERATIONS OF OUTER LOOP
                  1446 INNER:
00B4 0E00         1447        MVI   C,00H               ; 256 ITERATIONS OF INNER LOOP
                  1448 WRITE:
00B6 77           1449        MOV   M,A                 ; WRITE TEST DATA TO MEMORY
00B7 23           1450        INX   H                   ; INCREMENT ADDRESS POINTER
00B8 0D           1451        DCR   C                   ; DECREMENT INNER LOOP
00B9 C2B600    D  1452        JNZ   WRITE               ; REPEAT WRITE PROCEDURE
00BC 05           1453        DCR   B                   ; DECREMENT OUTER LOOP
00BD C2B400    D  1454        JNZ   INNER
                  1455 READ:
00C0 21003C       1456        LXI   H,START
                  1457 LOOP1:
00C3 0604         1458        MVI   B,04H
                  1459 LOOP2:
00C5 0E00         1460        MVI   C,00H
                  1461 COMP:
00C7 BE           1462        CMP   M                   ; IS (MEMORY) = TEST DATA ?
00C8 C2E300    D  1463        JNZ   LED3                ; NO, DISPLAY ERROR MESSAGE
00CB 36AA         1464        MVI   M,10101010B         ; WRITE SECOND TEST DATA TO MEMORY AFTER READ
00CD 23           1465        INX   H                   ; NEXT MEMORY CELL
00CE 0D           1466        DCR   C
00CF C2C700    D  1467        JNZ   COMP
00D2 05           1468        DCR   B
00D3 C2C500    D  1469        JNZ   LOOP2
00D6 FEAA         1470        CPI   10101010B
00D8 C2DE00    D  1471        JNZ   GET
00DB C3E400    D  1472        JMP   RAMEND
                  1473 GET:
00DE 3EAA         1474        MVI   A,10101010B         ; SECOND TEST DATA
00E0 C3C0C0    D  1475        JMP   READ
```

```
LOC  OBJ          LINE        SOURCE STATEMENT

                  1476 ;
                  1477 LED3:
00E3 76           1478        HLT
                  1479 RAMEND:
00E4 C9           1480        RET                         ; EXIST RAM ROUTINE
                  1481 ;
                  1482 ;*****************************************************************************
                  1483 ;                              SUBROUTINE: ROM                               *
                  1484 ;THIS ROUTINE TESTS THE ROM ON THE INTEL SBC 80/10A BOARD. THE EXTENDED*
                  1485 ; PRECISION CHECKSUM IS GENERATED BY SUMMING  ALL THE DATA BYTE IN THE *
                  1486 ;ROM PLUS ANY CARRY IN A DIFFERENT REGISTER. THIS FORMED A 16 BIT VALUE*
                  1487 ;CHECKSUM. THE GENERATED CHECKSUM IS THEN COMPARED WITH A PREDETERMINED*
                  1488 ;VALUE DD1O.                                                             *
                  1489 ;*****************************************************************************
                  1490 ROM:
00E5 21FF07       1491        LXI   H,STARTR-1            ; STARTING ADDRESS OF ROM MINUS ONE
00E8 E600         1492        ANI   00H                   ; CLEAR ACC., AND CARRY BIT OF PSW
00EA 47           1493        MOV   B,A                   ; HOLD CARRY
00EB 0E04         1494        MVI   C,04H
                  1495 ITER:
00ED 1600         1496        MVI   D,00H
                  1497 REPEAT:
00EF 23           1498        INX   H
00F0 86           1499        ADD   M                     ; ACC., = ACC., + DATA = LSByte OF CHECKSUM
00F1 D2F500    D  1500        JNC   NEXT
00F4 04           1501        INR   B
                  1502 NEXT:
00F5 15           1503        DCR   D
00F6 C2EF00    D  1504        JNZ   REPEAT
00F9 0D           1505        DCR   C
00FA C2ED00    D  1506        JNZ   ITER
00FD FE10         1507        CPI   010H                  ; COMPARE LSByte OF CHECKSUM
00FF C20B01    D  1508        JNZ   LED2
0102 78           1509        MOV   A,B
0103 FEBB         1510        CPI   0BBH                  ; COMPARE MSByte OF CHECKSUM
0105 C20B01    D  1511        JNZ   LED2
0108 C30C01    D  1512        JMP   ROMEND
                  1513 ;
                  1514 LED2:
                  1515
010B 76           1516       HLT
                  1517 ROMEND:
010C C9           1518        RET                          ; EXIST ROM ROUTINE
                  1519 ;
                  1520 ;*****************************************************************************
                  1521 ;                              SUBROUTINE: EXBUS                             *
                  1522 ;THIS ROUTINE TESTS THE 2K BYTES OF EXTERNAL RAM CONNECTED TO THE EXTER- *
                  1523 ;NAL BUS. THE FIRST TEST DATA 01010101 IS WRITTEN TO ALL THE RAM LOCAT-  *
                  1524 ;IONS AND THEN IS READ AGAIN. THE TEST IS REPEAT WITH THE SECOND TEST    *
                  1525 ;DATA 10101010. THESE TESTS ARE DONE TO ENSURE SOME OF THE COMMAND       *
                  1526 ;SIGNALS EXTERNAL SYSTEM BUS ARE FUNCTIONING PROPERLY.                   *
                  1527 ;*****************************************************************************
                  1528 ;
                  1529 EXBUS:
010D 210068       1530        LXI   H,STARTE               ; BEGINNING ADDRESS OF EXTERNAL RAM
```

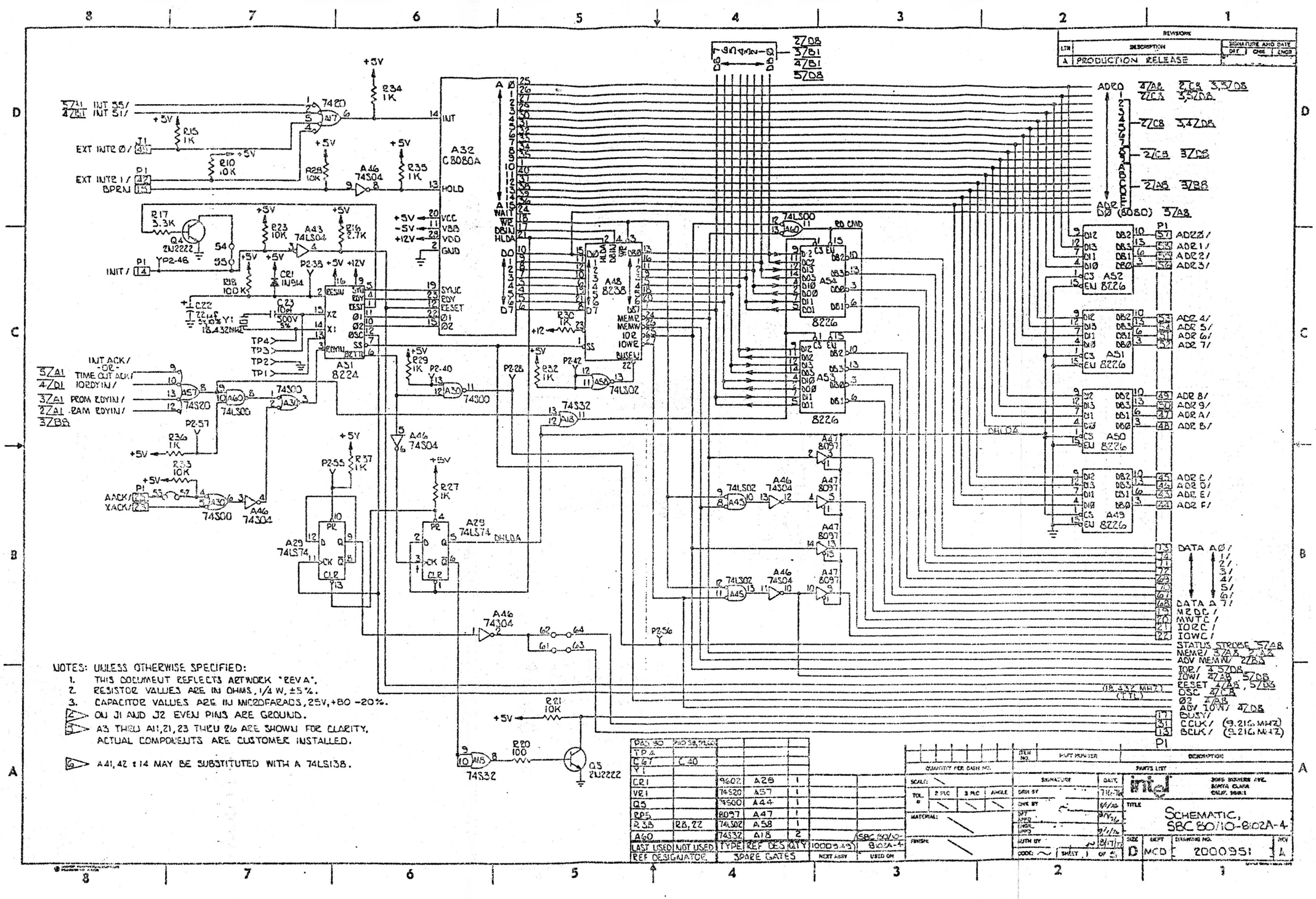
```
 LOC  OBJ          LINE        SOURCE STATEMENT

 0110 3E55         1531        MVI   A,01010101B          ; FIRST TEST DATA
 0112 1602         1532        MVI   D,02H                ; 2 ITERATIONS OF OUTERE LOOP
                   1533 OUTERE:
 0114 0604         1534        MVI   B,04H                ; 4 ITERATIONS OF INNERE LOOP
                   1535 INNERE:
 0116 0E00         1536        MVI   C,00H                ; 256 ITERATIONS OF WRITEE LOOP
                   1537 WRITEE:
 0118 77           1538        MOV   M,A                  ; WRITE FIRST TEST DATA TO MEMORY
 0119 23           1539        INX   H                    ; INCREMENT ADDRESSPOINTER
 011A 0D           1540        DCR   C                    ; DECREMENT INNERE LOOP
 011B C21801   D   1541        JNZ   WRITEE               ; REPEAT WRITEE LOOP IF C IS NOT ZERO
 011E 05           1542        DCR   B
 011F C21601   D   1543        JNZ   INNERE
 0122 15           1544        DCR   D
 0123 C21401   D   1545        JNZ   OUTERE
                   1546 READE:
 0126 210068       1547        LXI   H,STARTE
 0129 1602         1548        MVI   D,02H
                   1549 LOOP1E:
 012B 0604         1550        MVI   B,04H
                   1551 LOOP2E:
 012D 0E00         1552        MVI   C,00H
                   1553 COMPE:
 012F BE           1554        CMP   M                    ; IS (MEMORY) = TEST DATA ?
 0130 C24F01   D   1555        JNZ   LED6                 ; NO, DISPLAY ERROR
 0133 36AA         1556        MVI   M,10101010B
 0135 23           1557        INX   H                    ; INCREMENT ADDRESS POINTER
 0136 0D           1558        DCR   C
 0137 C22F01   D   1559        JNZ   COMPE
 013A 05           1560        DCR   B
 013B C22D01   D   1561        JNZ   LOOP2E
 013E 15           1562        DCR   D
 013F C22B01   D   1563        JNZ   LOOP1E
 0142 FEAA         1564        CPI   10101010B
 0144 C24A01   D   1565        JNZ   GETE
 0147 C35001   D   1566        JMP   BUSEND
                   1567 GETE:
 014A 3EAA         1568        MVI   A,10101010B
 014C C32601   D   1569        JMP   READE
                   1570 ;
                   1571 LED6:
 014F 76           1572        HLT
                   1573 BUSEND:
 0150 C9           1574        RET                        ; EXIST EXBUS ROUITNE
                   1575 ;
                   1576        END                        ; END OF MAIN PROGRAM

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS
```

AGAIN	C 05EB	BEGIN	C 0504	BEGIN2	C 05D1	BUSEND	D 0150	CHECK	C 0175	CHKEND	C 0577	CODEA	A 007F
CODEB	A 00BF	CODEC	A 00DF	CODED	A 00EF	CODEE	A 00F7	CODEH	A 00FB	CODEL	A 00FD	COMP	D 00C7
COMPE	D 012F	DATA	D 0000	DATAJ1	D 00A0	DATAJ2	D 00A3	DATAJ3	D 00A6	DATAJ4	D 00A9	ERROR	C 0173
EXBUS	D 010D	GET	D 00DE	GETE	D 014A	GRP1C	A 00E7	GRP2C	A 00EB	INIT2	C 05A1	INNER	D 00D4
INNERE	D 0116	ITER	D 00ED	LED	C 0576	LED1	C 0049	LED2	D 010B	LED3	D 00E3	LED4	D 0002
LED5	D 009F	LED6	D 014F	LIT	C 0015	LOOP1	D 00C3	LOOP1E	D 012B	LOOP2	D 00C5	LOOP2E	D 012D
MAIN	C 0000	MPU	C 004B	MPU1	C 0052	MPU2	C 006B	MPUEND	C 0174	NEXT	D 00F5	NEXTM	C 002C
OUTER	D 00D2	OUTERE	D 0114	PARALL	D 0004	PAREND	D 00AC	PORTA	A 00E4	PORTB	A 00E5	PORTC	A 00E6
PORTD	A 00E8	PORTE	A 00E9	PORTF	A 00EA	RAM	D 00AD	RAMEND	D 00E4	RCVRA	C 05D2	RCVRS	C 05E4
READ	D 00C0	READE	D 0126	REPEAT	D 00EF	REPETM	C 0025	ROM	D 00E5	ROMEND	D 010C	SELEND	C 004A
SELF	C 001E	SEREND	D 0003	SERIAL	C 0578	START	A 3C00	STARTE	A 6800	STARTR	A 0800	STARTS	A 0000
SYNEND	C 0604	SYNERR	C 0603	TXTERA	C 05C7	TXTERS	C 05DC	USARTC	A 00EF	USARTI	A 00EE	USARTO	A 00EE
USARTS	A 00EF	WRITE	D 00B6	WRITEE	D 0118								

ASSEMBLY COMPLETE, NO ERRORS

APPENDIX C

SCHEMATICS FOR THE INTEL SBC 80/10A



NOTES: UNLESS OTHERWISE SPECIFIED:

1. THIS DOCUMENT REFLECTS ARTWORK "REV A".
2. RESISTOR VALUES ARE IN OHMS, 1/4 W, ±5%.
3. CAPACITOR VALUES ARE IN MICROFARADS, 25V, +80 -20%.

(A1) ON J1 AND J2 EVEN PINS ARE GROUND.

(A2) A3 THRU A11, 21, 23 THRU 26 ARE SHOWN FOR CLARITY. ACTUAL COMPONENTS ARE CUSTOMER INSTALLED.

(A5) A41, 42 & 14 MAY BE SUBSTITUTED WITH A 74LS138.

LAST USED	NOT USED	TYPE	REF DES	QTY
REF DESIGNATOR		SPARE GATES		
CR1		9602	A28	1
VR1		74S20	A57	1
Q5		74S00	A44	1
RP5		8097	A47	1
R38	R8, 22	74LS02	A58	1
A60		74S32	A18	2

(SBC 80/10-8102A-4)

Figure C-1. Schematic of CPU Sets & Associate Circuitry (Sheet 1 of 5)

(Adapted From [1])

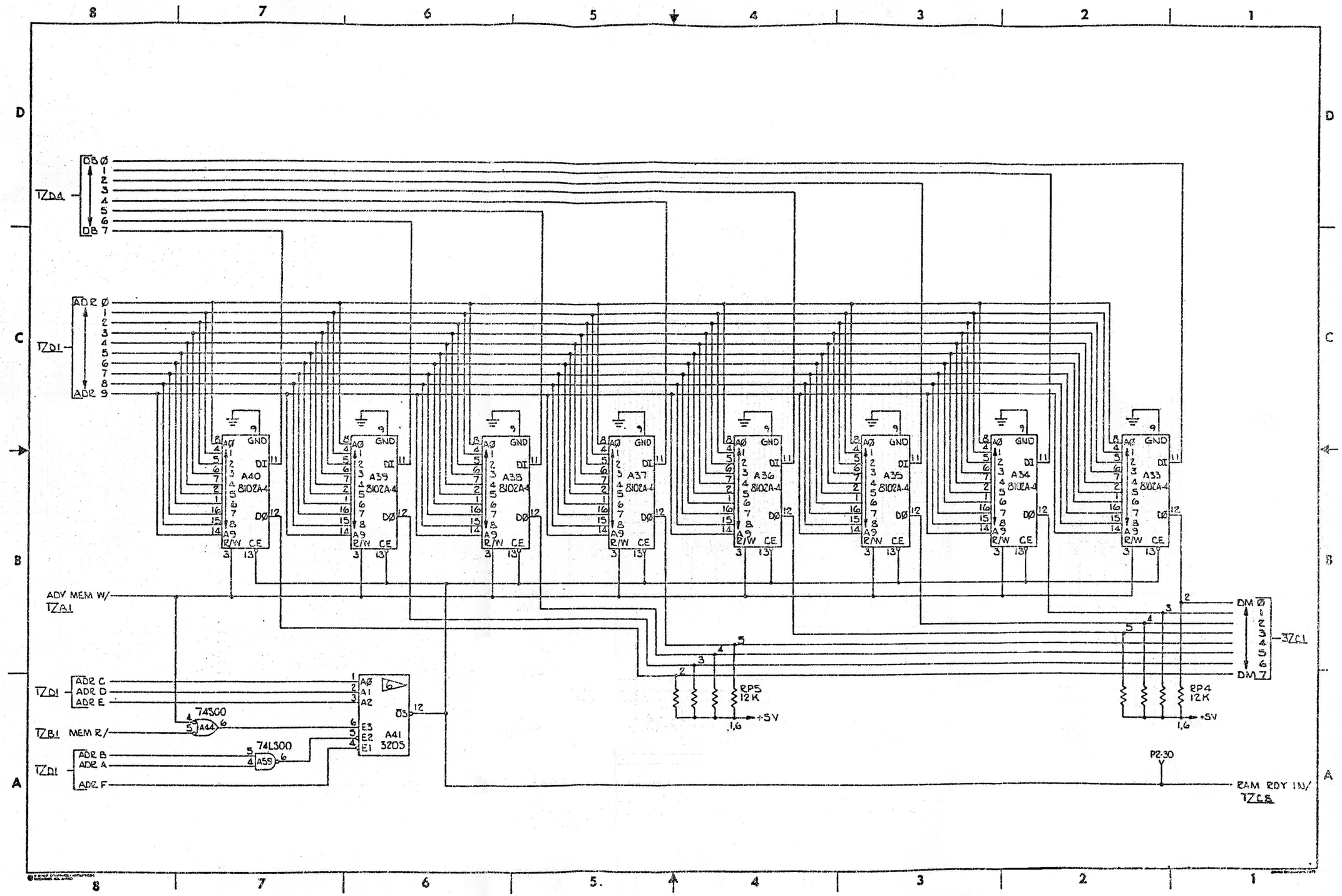


Figure C-2. Schematic of RAM & Associate Circuitry (Sheet 2 of 5)
(Adapted From [1])

JUMPER TABLE				
2716	66-67	69-70	74-75	77-78
2708	65-66	68-69	73-74	76-78

	A23	A24	A25	A26
2716	0-7FF	1000-17FF	800-FFF	1800-1FFE
2708	0-3FF	400-7FF	800-BFF	C00-FFF

Figure C-3. Schematic of ROM & Associate Circuitry (Sheet 3 of 5)
(Adapted From [1])

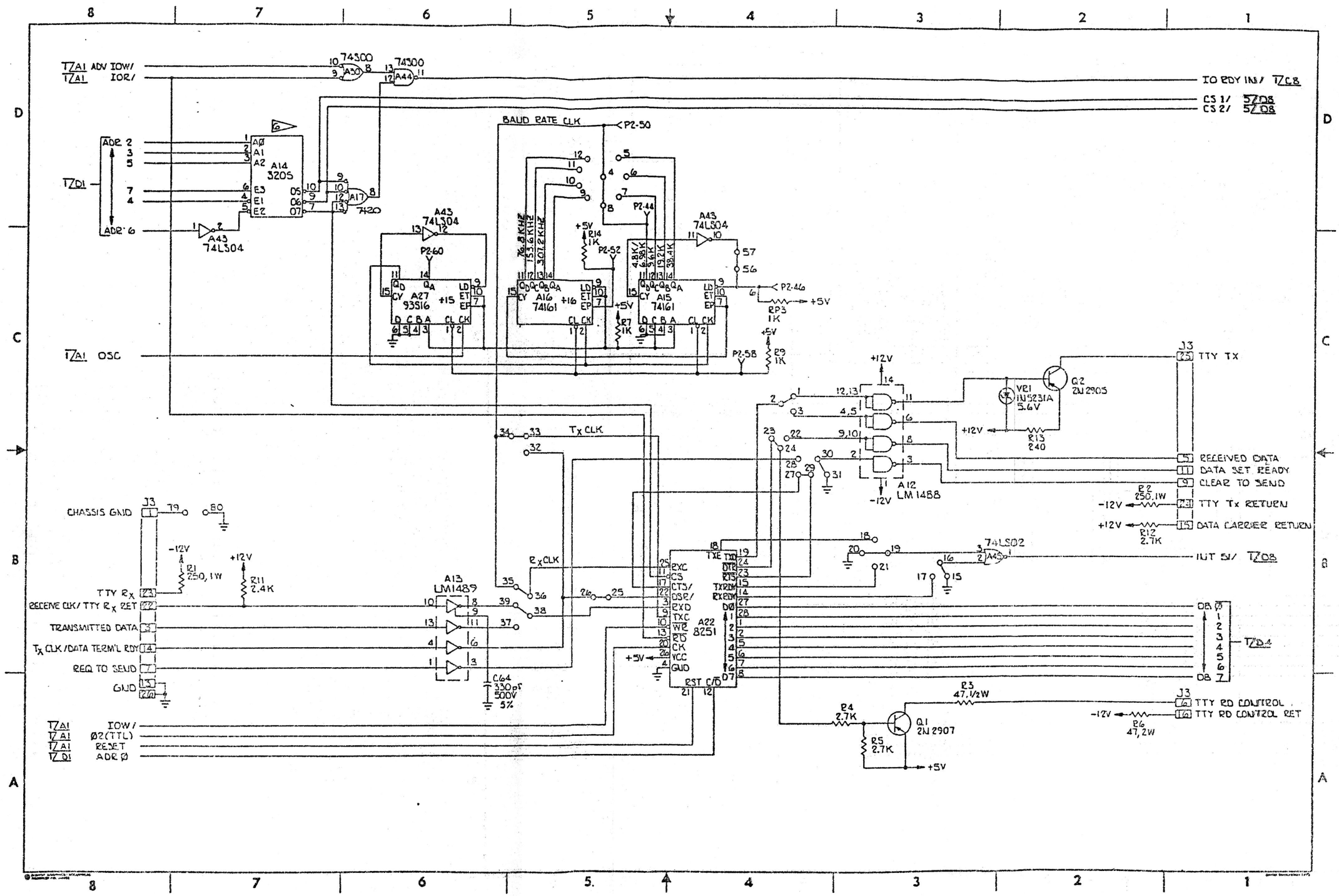


Figure C-4. Schematic of Serial I/O Interface & Associate Circuitry (Sheet 4 of 5)
(Adapted From [1])

Figure C-5. Schematic of Parallel I/O Interface & Associate Circuitry (Sheet 5 of 5)

(Adapted From [1])

REFERENCES

[1]. Intel SBC 80/10A Single Board Computer Hardware Reference Manual.

[2]. Patrick P.Fasang " A Fault Detection and Isolation Technique for Microcomputers", 1982 Test Conference, pp. 214-219, Oct. 1982.

[3]. J. Abadir and Y. Deswarte " Run-Time Program for Self-Checking Single Board Computer", 1982 Test Conference, pp. 205-213, Oct. 1979.

[4]. Mary E. Thiel " Improving Microprocessor Testability", Vol. 6, Electronics Test, pp. 30, Oct. 1983.

[5]. J. M. Bilton " A Survey of Self-Test and BITE Program Generation", Euromicro Journal, pp. 168-174, June 1980.

[6]. Rodham E. Tulloss " Automated Board Testing: Coping with Complex Circuits", IEEE Spectrum, Vol. 20, pp. 38-43, July 1983.

[7]. Patrick P. Fasang " Microbit Brings Self-Testing on Board Complex Microcomputers", Vol. 56, Electronics, pp. 116-119, March 10, 1983.

[8]. Eric L. Paul " Measuring the Effectiveness of PCB Test Systems", Vol. 6, Electronics Test, pp. 98, Oct. 1983.

[9]. Joel Boney and Ed Rupp " Let Your Next Microcomputer Check Itself and Cut Down Your Testing Overhead",

Electronic Design,Vol. 27, pp. 100-105, Sept. 1, 1979.

[10]. Lawrence E. Getgen " ROM-Resident Software Self-Test Microcomputers", Electronic Design, Vol. 28, No. 13, pp. 131-134, June 21, 1980.

[11]. J. B. Clary and R. A. Sacane " Self-Testing Computers" , IEEE Computer, pp. 49-59, Oct. 1979.

[12]. John Masciola and Gary Roberts " Testing Microprocessor Boards and Systems A New Approach", 1983 IEEE Test Conference, pp. 46-50, Oct. 1983.

[13]. Micheal D. Lippman " Designing Testable Microcomputer Systems", ATE Seminar/Exhibit, pp. 123-128, Jan 23, 1979.

[14]. Micheal D. Lippman and Edward S. Donn "Design Forethought Promotes Easier Testing of Microcomputer Boards", Vol. 52, Electronics, pp. 113-119, Jan 18, 1979.

[15]. Philip Edward Hodur " Enhancing Testability of Complex LSI-Based Digital Systems Using Self-Verification Techniques", AUTOTESTCON 82, pp. 146-151, 1982.

[16]. The Future of Test, International Test Conference, pp. 352-355, 1985.

[17]. Lance A. Leventhal " 8080A/8085 Assembly Language Programming", Osborne & Associates, Berkeley, CA 94702.

[18]. Intel ISIS-II User's Guide.

[19]. Intel ISIS-II 8080/8085 Macro Assembler Operator's Manual.

[20]. Intel ISIS-II (CRT-Based Text Editor) User's Guide.

[21]. Daniel P. Siewiorek, Robert S. Swartz " The Theory and Practice of Reliable System Design", Digital Press, Educational Services, Digital Equipment Corp., Bedford, MA 01730.

[22]. Parag K. Lala " Fault Tolerant & Fault Testable Hardware Design", Prentice Hall, Inc., Englewood Cliffs, NJ 07632.

[23]. Jacob A. Abraham and Kenneth P. Parker "Practical Microprocessor Testing: Open and Closed Loop Approaches", IEEE Trans. on Computer, pp. 308-311, 1981.

[24]. Dhanajay Brahme and Jacob A. Abraham "Functional Testing of Microprocessor", IEEE Trans. on Computer, pp. 1-35, Oct. 1983.

[25]. P. H. Bardell and Jr., W. H. McAnney "Self-Testing of Random Access Memory", Test & Measurement World, pp. 26-29, MArch 1983.

[26]. J. Knaizuk, Jr. and C. R. P. Hartman "An Algorithm for Testing Random Access Memory", IEEE Trans. Computer, Vol. C-26, pp. 414-416, April 1977.

[27]. J. Knaizuk, Jr. and C. R. P. Hartman "An Optimal Algorithm for Testing Stuck-at-faults in Random Access Memory ", IEEE Trans. Computer, Vol. C-26, pp. 1141-1144, Nov 1977.

[28]. R. Nair "Comments on an Algorithm for Testing Stuck-at-faults in Random Access Memories", IEEE Trans. Computer, Vol. C-28, pp. 258-261, March 1979.

[29]. Intel Data Book 1977.

[30]. Vason P. Srini " Fault Diagnosis of Microprocessor Systems",Computer, Vol. 10, pp.60, 1977.

ADDITION OF BUILT-IN SELF-TESTING CAPABILITY TO THE INTEL SBC 80/10A SINGLE BOARD COMPUTER

by

CHEOW FATT YEO

B. Sc., Kansas State University, 1984

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

Kansas State University

Manhattan, Kansas

1986

ABSTRACT

This thesis discusses the application of built-in self-cesting technique at the board level (Intel SBC 80/10A single board computer). The test technologies used in testing microprocessor-based boards are briefly explained. It examines a spectrum of problems which are presenting the test engineers with some very formidable testing challenges and are also becoming increasingly hard for current automatic test equipment (ATE) test technology to deal with.

A functional description of the Intel SBC 80/10A board was briefly studied so that every essential part of the board is tested. The concept of built-in self-testing was then introduced and it was later explained why this concept is becoming an increasingly attractive solution to the complex problem of testing microprocessor-based boards.

Different testing methods were then designed to test the board's microprocessor, self-test ROM, system software ROM, RAM, parallel I/O ports, serial I/O port and the external system bus.

A summary of the test set up in the user testing environment was given and it also listed a series of events that should occur when the test was run.

Finally, fault-detection and isolation capability were explored as an expansion to the existing self-testing program.