DCC	title	prefix/class-number.revision
	Micro-Processor Acceptance Procedure	MPAP/W-18.2

checked	authors	approval date	revision date
Wm Hodge	Jesse T. Quatse	10-16-69	2-18-70
checked: Bo Lewendal		classification: Working Paper	
approved: Jesse T. Quatse		distribution: Company Private	pages: 44

ABSTRACT and CONTENTS

Micro-Processor electronics will be final-tested with special ROM boards for which acceptance routines will be written. The testing strategy is described. The symbolic code for the acceptance routine for the adder/cycler card is described in detail.

Introduction

As part of the testing procedure for Model-1 system components, the electronics of each micro-processor must be checked as thoroughly as possible before the final ROM boards are installed. A criterion is necessary in order to determine whether or not a micro-processor is "debugged" and ready for final ROM installations. (Another criterion is required in order to determine whether or not the ROM is operating properly, etc.). Towards these ends, a set of Acceptance Routines will be developed which will stand as a reasonable verification that the tested micro-processor is operating properly.

These routines will consist of ROM boards which are installed whenever a micro-processor seems to be operating perfectly. They are oriented towards exhaustive testing; they assume that no faults exist; and they are tightly coded in order to fit into the smallest micro-processor. Consequently, they will not necessarily satisfy the requirements of a good debugging tool. They will give useful information concerning the nature of each detected fault; but their organizations may result in overly complicated tests for the purpose of exercising mal-functioning micro-processors. Also, a multiplicity of hardware faults can lead to confusing results. The main purpose

of the acceptance routines will be to answer yes or no to the question of whether the tested micro-processor is operating properly.

Information concerning the nature of the detected fault will be provided by the organization of the acceptance routines. Each test will loop if a fault is detected. The address of the loop gives some indication of the problem. Where possible, test loops will be small and uncomplicated so that debugging with an oscilliscope is reasonable. A manual switch will be provided which overrides the test loop and progresses the micro-processor to the next test. In this way, it is possible to sequence through the set of tests, gathering data which can be important to fault diagnosis.

The Test Method

In general, the routines will consist of three parts:

1) a comprehensive logic test which detects a "solid" failure of any circuit under any input condition which is independent of time,

2) a similar test on circuits whose inputs are time dependent, and

3) a pseudo-random test which attempts to check performance in many operating environments.

These parts can best be described by example. The example given here is the adder of the basic micro-processor.

The first part should detect almost all faulty components. The testing method is to create a PC card input condition designed to establish a particular input condition on gates within the card, and then to compare outputs from the card with expected patterns. In the case of the adder, sets of operand and input carry states are generated, each of which tests a set of gating conditions within the adder. For each set, the sum is compared to an expected sum for fault detection. If an error is detected, the process is repeated. Continued errors cause continuous looping until the termination condition is met. This condition is caused by manually closing a switch contact, thereby generating a pulse which sets a flip-flop. The switch, called ADVANCE, is a push-botton located on the local control panel. The flip-flop, called ATTN1, is located on the branch condition card. It is test-and-resetable by branch conditions 26(ATTN1 = $\emptyset$) and 36(ATTN1 $\neq \emptyset$). Another switch will set RSLAT2; or it will hold RSLAT2 true. The acceptance routines will loop on a single card test if RSLAT2 = $\emptyset$, or proceed to the test of the next card if RSLAT2 = 1.

This testing method is made comprehensive by providing the following test conditions for each NAND gate.

1) each input term is held false while the remaining input terms are held true.

2) All input terms are held true.

For these tests, the output state must be "detectable". That is, for each gate input condition, a signal path must exist which begins at the output of the given gate, and terminates at the X bus, such that for each gate in the signal path, all input terms other than the signal path term are true. In other words, the state of the given gate is detectable on the X bus.

The second part should detect faults in the small percentage of the total micro-processor which requires some time dependency during testing. Otherwise, the test method is identical to that of part 1. An example is the adder carry accelerating circuitry, which is redundant logically and serves only to predict conditions which are about to arise at a later time. Another example is the input/output circuitry which is driven by independently timed devices such as teletypes.

The third part is intended to detect subtle correlations in the behavior of apparently independent circuits. For example, the simultaneous setting of all flip-flops of a register might generate a power line transient which causes an erroneous output of some other-wise unrelated circuit which was a marginal threshold. The only way to detect such subtle faults is to either generate every state of the micro-processor, or to generate enough states to give a reasonably high probability that the subtle fault will not go undetected.

The Adder/Cycler Test Routine

The BCC Micro Language (MICRO/M-8) is used wherever possible. Since these routines perform hardware testing, the instruction word sometimes requires bit configurations which are not readily expressable in standard form. In these cases, and in cases where the standard form is not explicit, a free-style syntax and semantics is used.

The adder/cycler test requires 6Ø ROM instructions organized as follows (addresses are relative to the board address):

(0-3) CONST: four word test of the constant field gating onto the X bus, and of the no-gating condition.

(4-14B) LCYSF: nine word test of the special function left cycle transfer enables.

(15B-17B) LCYZ: three-word test of the Z controlled left cycle gating and transfer enables (used LCYZS subroutine for the actual testing).

(20B-26B) AP1: seven word adder test for cases where VCY=Ø.

(27B-47B) AP3P4: seventeen word adder test for cases where VCY=1 (uses ADDSUB1 subroutine for the actual testing).

(50B-55B) ACC1: seven word test of carry accelerators.

(56B-57B) EXIT: two word branch to loop or to transfer to the next acceptance routine.

(6ØB-63B) SPARE: four words left as spares for future changes.

(64B-67B) LCYZS: four word subroutine for LCYZ testing.

(70B-77B) ADDSUB1: eight word subroutine for AP3P4 testing.

The Cycler and X Bus Tests

(0-3) CONST: the constant field and all x bus gate enabling signals.

(0) CONST: Q←X←(CONSTANT = -1),

GOTO LCYSF IF ATTN1;

All ones are gated from the constant field to Q as a test for the DATA/GATE condition of the constant field gates on the adder/cycler card. The branch is executed when the ADVANCE push-button is depressed. Errors detected anywhere in the four instruction loop, starting at CONST, will cause a branch to CONST. The branch to LCYSF terminates the loop when ADVANCE is depressed.

(1) M←X←(CONSTANT = Ø),

GOTO CONST IF X ≠ Ø;

All zeros are gated from the constant field to M as a test for the DATA/GATE condition of the constant field gates. The branch is a redundant check, on the all zero test, which will be checked again later.

(2) (CONSTANT = -1), (BL = -1), (BR = Ø),

GOTO CONST IF X ≠ Ø;

No gating signals are specified; and all data inputs to all X gates on the adder cycler card are true. All gating terms are checked unless the entire gate is faulty. The all zero test will detect such a fault.

(3) Q←Y←(CONSTANT = 1),

GOTO CONST ON Q EQV M ≠ Ø;

The all zero and all one tests are checked by comparing the results of the two transfers to Q and M. The Q register is initialized for the next set of tests.

(4-14B) LCYSF: the left cycle gate enabling signals which are controlled by special functions.

```
(4)   LCYSF:  Q←X←Q  LCY 1, M←Y←(CONSTANT = 1B6)
              GOTO LCYZ IF ATTN1;
(5)           Q←X←Q LCY 2;
(6)           Q←X←Q LCY 3;
(7)           Q←X←Q LCY 4;
(10B)         Q←X←Q LCY 8;
(11B)         Q←X←Q LCY 12;
(12B)         Q←X←Q LCY 16;
(13B)         Q←X←Q LCY 2Ø;
(14B)         Q←Y←(CONSTANT = 1)
              GOTO LCYSF ON M EOR Q ≠ Ø;
```

The +1 pattern, initially loaded into Q by the previous test loop, is shifted by each of the special functions provided for cycling. The last instruction (14B)

checks the accumulated shift which should be 66 bits or 1B6 (mod 24). The branch of (14B) returns to LCYSF if the accumulated shift is not 1B6. Simultaneously, Q is again initialized to +1 so that faults are detectable. The first command (4) has the exit branch for the ADVANCE push button. This test loop checks the gate enabling signals only (the control gates). The actual cycle gates are checked by the next test which is controlled by chcle counts in the Z register.

(64B-67B) LCYZS: Subroutine used by LCYZ to obtain left cycles.

```
(64B)  LCYZS:  M←Y←RØ,
               Q←X←(CONSTANT = 437777Ø1B)
               RETURN IF M EOR Q = Ø;

(65B)  LOOP:   Z←X←Q+Z, VCY,
               Q←Y←RØ,
               GOTO LCYZS IF X< Ø;

(66B)  ENTER:  M←X←M  LCYZ_H,
               RETURN IF ATTN1;

(67B)  LAST:   M←X←M  LCYZ_L,
               Q←Y←(CONSTANT = 4000011B);
               GO TO LOOP;
```

The subroutine is initialized by loading a shift pattern into M and RØ and by calling ENTER. The main loop, which begins at ENTER, shifts the contents of M according to the high and low bits of Z in order. The instruction at LOOP increments Z according to the table of Figure 1. When the loop terminates, by virtue of X(Ø) =1, the next instruction is LCYZS, where the cycled pattern in M is compared with the original pattern loaded from RØ to Q by LOOP. In case of an unsuccessful test, M is initialized to RØ and Q is initialized to 437777Ø1B. The instruction

at LOOP will add the value of Q to 4ØØØØ1Ø4B, which must be the value remaining in Z from the previous execution of LOOP. The value of Z is thereby initialized to 4ØØØØØ5B.

PASS THRU LOOP	CYCLE COUNTS $LCYZ_H$	$LCYZ_L$	NON-ZERO BITS OF Z Ø 1 2 3	18 19 20 21	22 23
1	4	1	0 0 0 1	0 0 0 1	0 1
2	12	2	0 0 1 0	0 0 1 1	1 0
3	20	3	0 0 1 1	0 1 0 1	1 1
4	0	0	0 1 0 0	1 0 0 0	0 0
5	8	1	0 1 0 1	1 0 1 0	0 1
6	16	2	0 1 1 0	1 1 0 0	1 0
7	0	3	0 1 1 1	1 1 1 0	1 1
	60	12	1 0 0 0 = TERMINAL CONDITION		

72 = 0 MOD 24

FIGURE 1

TABLE OF CYCLE COUNTS

According to the table, the shift pattern is presented to every cycle gate at least once. Successive cycle gates are specified by adding Q=11B to Z at LOOP. This particular incrementing constant is chosen so that all gates are selected at least once and so that the total resulting shift is Ø mod 24. A net shift of zero simplifies testing at LCYZS. The loop branches to LCYZS when the upper bits of Z have incremented to a value where ZØ=1. If the test at LCYZS fails, the loop is initialized and entered again. Continued failures will cause indefinite looping until ATTN1 is set by the ADVANCE switch. Initialization requires that M is returned to its initial value (because an error must have changed the pattern) which is retained intact in RØ. The count in Z also is initialized by LCYZS. The Q register serves two purposes. It is loaded from RØ at LOOP for the comparison at LCYZS; and it is loaded from the constant field at LAST for the purpose of incrementing Z at LOOP.

(15B-17B) LCYZ: the left cycle gates which are controlled by the Z register.

The calling sequence for LCYZS places the pattern to be cycled in M and RØ , initializes Z to 4ØØØØØ5B, and calls ENTER. Each of the three commands provides a pattern which is necessary for complete checking.

```
(15B)  LCYZ:  Z←Y←(CONSTANT = 4ØØØØØ5B),
              (M, RØ) ← X←Q,
              CALL ENTER;
```

The previous test loop leaves either +1 or +2 in Q, depending upon the outcome of the LCYSF test. This call on LCYZS tests the cycle enable gates, by cycling a single non-zero bit, in a manner similar to LCYSF.

```
(16B)         Z←Y←(CONSTANT = 4ØØØØØ5B),
              (M, RØ)←X←(BL = -1),
              CALL ENTER;
(17B)         Z←Y←(CONSTANT = 4ØØØØØ5B),
              (M, RØ)←X←(BL = Ø),
              CALL ENTER;
```

All cycle gates are tested for both the true and false conditions, in these two instructions. The first cycles all ones and the second all zeros. They test the same gates that are used by LCYSF. Therefore, the cycle gate and cycle control gate tests are comprehensive.

The Adder Test Patterns

Twelve addend and augend pairs are sufficient to generate all input conditions for all gates of the adder (excluding the carry accelerator patterns of part 3.) Eight bits of each pattern are shown in Figure 2. The other 16 bits are repetitions of the

	VCY		LPAT(n)								RPAT(n)								LOC	SPAT(n)							
	0	1	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	0	1	1	0	1
	0	2	0	0	0	0	1	0	0	0	1	1	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0
	1	3	1	1	0	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
	1	4	1	1	1	1	1	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0
	1	5	0	0	0	1	0	0	0	1	1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	1	1
	1	6	0	0	0	1	0	0	0	1	1	1	1	0	1	1	0	1	0	1	1	1	1	1	1	1	0
	1	7	0	0	1	0	0	0	1	0	1	1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	0
	1	8	0	0	1	0	0	0	1	0	1	0	1	1	1	1	1	1	0	1	1	1	0	0	0	0	1
*	1	9	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0
*	1	10	1	1	1	0	1	1	1	0	0	1	0	0	0	1	0	0	1	0	0	1	1	0	0	1	1
*	1	11	1	1	0	1	1	1	0	1	1	0	1	1	1	0	1	1	1	1	0	0	1	1	0	0	1
*	1	12	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1

* See Text for Explanation

Figure 2

The adder test patterns

given eight bits. LOC is the low order carry bit and VCY is a bit of the ROM microcode.

Each pattern regenerates its own input conditions over an eight-bit pattern unless marked by (*) in the figure. In these four cases, the pattern actually repeats every four bits, although eight bit patterns are shown in the figure.

The first two patterns require that VCY = Ø. They are time independent; but they check the circuitry associated with the VCY bit of the instruction. In addition, they check adder logic as specified below.

Figure 3 describes the adder logic checked by each pattern. Other patterns may redundantly check the same conditions. However, if the checks proceed in the order specified by Figure 3, the conditions shown will be checked for the first time in the order specified. (Whatever order turns out to be, the complete check is comprehensive.) For brevity and clarity, the figure refers to a "typical" four bit interval of the adder, with bit Ø being the leftmost and bit 3 being the rightmost bit of the interval. All other four bit intervals can be generated by aligning bit Ø of the typical interval with bits Ø, 4, 8, 12, 16, or 2Ø of the adder.

The test patterns for part 3 are intended to create conditions where all accelerator circuits must

TEST PATTERN (1)

BIT	ø	ø	ø	ø	ø	ø	ø	1	1		1
LOCATION	4F	4G	5H	5H	5H	4E	4E	4H	4H		5G
OUTPUT PIN	8	8	11	6	8	12	8	11	6		8
INPUT PIN	ALL	2,3,4*	ALL	5	9	13	ALL	ALL	5		9
INPUT TEST	1	ø	1	ø	ø	ø	1	1	ø		ø

TEST PATTERN (1)

BIT	1	1	2	2	2	2	2	2	3	3	3
LOCATION	5F	5F	3H	3H	3H	3G	3G	3F	2H	2H	2H
OUTPUT PIN	6	8	6	8	3	6	8	8	6	8	3
INPUT PIN	4	ALL	ALL	9	1	ALL	ALL	11	ALL	9	1
INPUT TEST	ø	1	1	ø	ø	1	1	ø	1	ø	ø

TEST PATTERN (1)

BIT	3	3	3								
LOCATION	2F	2E	2E								
OUTPUT PIN	8	12	8								
INPUT PIN	4	1	ALL								
INPUT TEST	ø	ø	1								

means that entire gate is now checked

TEST PATTERN (2)

BIT	ø	ø	ø	ø	ø	ø	1	1	1	1	2
LOCATION	5H	5H	5H	4F	4E	4E	4H	5G	5F	5F	3H
OUTPUT PIN	11	3	6	8	12	8	8	6	6	8	3
INPUT PIN	12,13*	1,2*	ALL	11,12*	ALL	11	All	1	ALL	11	ALL
INPUT TEST	ø	ø	1	ø	1	ø	1	ø	1	ø	1

* A Multiplicity of input terms wired together as a single term

Figure 3 Conditions Tested by Each Test Pattern

TEST PATTERN (2)

BIT	2	2	2	3	3	3	3				
LOCATION	3G	3F	3F	2H	2F	2E	2E				
OUTPUT PIN	6	6	8	3	8	12	8				
INPUT PIN	1	ALL	9	ALL	1,2*	ALL	11				
INPUT TEST	Ø	1	Ø	1	Ø	1	Ø				

TEST PATTERN (3)

BIT	Ø	1	1	1	1	2	2				
LOCATION	4F	4H	4H	4H	5G	3G	3F				
OUTPUT PIN	8	11	3	6	6	8	8				
INPUT PIN	3	12,13*	1,2*	ALL	2	12	ALL				
INPUT TEST	Ø	Ø	Ø	1	Ø	Ø	1				

TEST PATTERN (4)

BIT	Ø	Ø	Ø	Ø	1	1	1	1	1	1	2
LOCATION	5H	5H	4G	4E	4H	4H	4H	5G	5G	5F	3H
OUTPUT PIN	3	6	8	8	3	6	8	8	6	8	11
INPUT PIN	ALL	4	ALL	1Ø	ALL	4	1Ø	ALL	5	1Ø	ALL
INPUT TEST	1	Ø	1	Ø	1	Ø	Ø	1	Ø	Ø	1

TEST PATTERN (4)

BIT	2	2	2	3	3	3	3	3	3		
LOCATION	3H	3H	3G	2H	2H	2H	2G	2F	2E		
OUTPUT PIN	8	3	6	11	8	3	8	8	8		
INPUT PIN	1Ø	2	5	ALL	1Ø	2	ALL	3	9		
INPUT TEST	Ø	Ø	Ø	1	Ø	Ø	1	Ø	Ø		

* A multiplicity of input terms wired as a single term

Figure 3 Continued

TEST PATTERN 5

BIT	Ø	1	1	1	1	2	3	3	3	3
LOCATION	4G	4H	5G	5G	5F	3F	2H	2H	2H	2F
OUTPUT PIN	8	8	8	6	6	6	6	11	8	8
INPUT PIN	6	9	1Ø	ALL	3	4	4,5*	12,13*	ALL	11,12*
INPUT TEST	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø	1	Ø

TEST PATTERN 6

BIT	2	2	3	3	3
LOCATION	3G	3F	2G	2F	2E
OUTPUT PIN	8	6	8	8	12
INPUT PIN	9	3	4	4	13
INPUT TEST	Ø	Ø	Ø	Ø	Ø

TEST PATTERN 7

BIT	Ø	1	2
LOCATION	4F	5G	3G
OUTPUT PIN	8	6	6
INPUT PIN	4,5,6*	4	2
INPUT TEST	Ø	Ø	Ø

TEST PATTERN 8

BIT	Ø	1	1	2	2	2	2
LOCATION	4G	5G	5F	3H	3H	3H	3G
OUTPUT PIN	8	8	6	6	11	8	8
INPUT PIN	12	13	5	4,5*	12,13*	ALL	1Ø
INPUT TEST	Ø	Ø	Ø	Ø	Ø	1	Ø

* A multiplicity of input terms wired as a single term

Figure 3 Continued

TEST PATTERN 9

BIT	Ø	Ø	Ø	1
LOCATION	5H	4G	4E	5G
OUTPUT PIN	8	8	12	8
INPUT PIN	ALL	5	2	12
INPUT TEST	1	Ø	Ø	Ø

TEST PATTERN 1Ø

BIT	2	3	3
LOCATION	3G	2G	2E
OUTPUT PIN	8	8	12
INPUT PIN	13	1	2
INPUT TEST	Ø	Ø	Ø

TEST PATTERN 11

BIT	Ø	Ø	2	3
LOCATION	4G	4E	3G	2G
OUTPUT PIN	8	12	6	8
INPUT PIN	11	1	4	11,12*
INPUT TEST	Ø	Ø	Ø	Ø

TEST PATTERN 12

BIT	Ø	3
LOCATION	4F	2G
OUTPUT PIN	8	8
INPUT PIN	2	3
INPUT TEST	Ø	Ø

* A multiplicity of input terms wired as a single term

Figure 3 Continued

operate properly in order to produce a correct sum in 2ØØ ns. This means that all bits are in the "pass" state in which carry-in emerges as carry-out and not-carry-in emerges as not-carry-out. The pass state is obtained by one of the two operand inputs being true and the other being false. A carry is started with VCY = 1, and a no-op with VCY = 1 follows, so that sufficient time is allowed for carry propagation. The next instruction allows only one command, with VCY = 1 and not-carry, so that the carry accelerators are essential. The same test is rerun with opposite carry conditions for the test of the complement set of carry accelerators.

THE ADDER TESTS

(20B - 26B) AP1: seven word adder test for cases where VCY = Ø.

The first seven instructions of the adder test use pattern 1 and pattern 2. They require VCY = Ø. All other patterns require VCY = 1. Therefore, the test subroutine ADDSUB is not applicable.

```
(20B) AP1:      M←X←(CONSTANT = LPAT1),
(21B)           Z←X←(CONSTANT = RPAT1),
(22B)           M←X←M+Z, LOC = 1, VCY = Ø,
                Q←Y←(CONSTANT = SPAT1)
                GOTO AP2 IF ATTN 1;
```

```
(23B)              M←Y←(CONSTANT = LPAT1),
                   GOTO 22B ON M EOR Q ≠ Ø;
(24B) AP2:         M←X←M LCY4,
                   Z←Y←(CONSTANT = RPAT2);
(25B)              M←X←M+Z, LOC = Ø, VCY = Ø,
                   Q←Y←(CONSTANT = SPAT2),
                   GOTO AP3P4 IF ATTN1;
(26B)              M←Y←(CONSTANT = LPAT2)
                   GOTO 25B ON M EOR Q ≠ Ø;
```

The LPAT1 and RPAT1 are loaded into M and Z in preparation for the adder test. The test is done in 22B, with VCY = Ø, while Q is loaded with the SPAT1 in preparation for the comparison. Since LPAT1 is destroyed by the addition, 23B must reload M while comparing with Q. An adder error will cause indefinite looping on just the add and test instructions until ATTN1 is set by the ADVANCE push button.

Pattern 2 is tested in the same way, beginning at AP2. One instruction is saved by deriving the second test patterns from a left shift of the first. A relationship between patterns occurs frequently in the adder test in order to minimize the number of instructions required.

(70B - 77B) ADDSUB1: the subroutine for checking adder test patterns.

Another minimization is the ADDSUB subroutine. Patterns 3 through 12 make use of this subroutine to test the patterns which are generated by instructions 27B through 47B. The purpose of the subroutine organization, and for other complications in the adder test routine organization, is to obtain compactness. (In the most obvious form, the organization would require one instruction to load the left operand from the constant field, one for the right operand, one for the sum, one for the addition, and one for the equality test. For ten patterns, this organization would require 50 instructions, rather than the 25 used here.)

The subroutine occupies words 7ØB through 77B. It is parameterized to perform one test, in the case of pattern 9 through pattern 12; or it will perform one test, then shift the operands left four bits, then perform a second test, in order to handle pattern 3 through pattern 8. This testing mode is controlled by RØ. With RØ $\geq$ Ø only one test is performed. If RØ =-1, the subroutine increments RØ and loops. Therefore, RØ $\geq$ Ø at the end of each test, and need not be set by single test calls.

Another parameterization is the input carry condition. If the subroutine is entered at ADDSUB1, the zero order carry will be true (LOC = 1). If the entry point is ADDSUBØ, then LOC = Ø. For double test entries (RØ =-1), the carry condition of the second test is controlled by

FLAG A. If FLAG A = Ø, then the second test is performed with LOC = Ø.

```
(70B)  ADDSUB1:     M←X←M+Z, LOC = 1, VCY = 1,
                    GOTO ADLOOP IF ATTN1;
(71B)               M←Y←R6,
                    GOTO ADDSUB1 ON M EOR Q ≠ Ø;
(72B)  ADLOOP:      R6←X←M LCY4,
                    M←Y←R5,
(73B)               Z←X←M LCY4,
                    M←Y←R6,
                    RETURN IF RØ ≥ Ø;
(74B)               Q←X←Q LCY4,
                    RØ←Y←RØ+1,
                    GOTO ADDSUB1 IF FLAG A;
(75)  ADDSUBØ:      M←X←M+Z, LOC = Ø, VCY = 1,
                    GOTO ADLOOP IF ATTN1;
(76B)               M←Y←R6,
                    GOTO ADDSUBØ ON M EOR Q ≠ Ø;
(77B)               GOTO ADLOOP;
```

The tests for pattern 1 and pattern 2 cannot use the subroutine because they require VCY = Ø. In order to further generalize the subroutine to handle this case also, more instructions are needed in total than are required by treating the first two patterns as special cases.

(27B-47B) AP3P4: Adder test patterns requiring VCY = 1.

Pattern 3 through pattern 8 are arranged in pairs. The second member of each pair is equivalent to the first member shifted four bits to the left. One set of tests is performed on each four bit interval by each test of the pair. After the shifting, each eight bit interval is subjected to the same test conditions. Pattern 9 through pattern 12 are designed to save instruction space by being related to each other.

The left operand for pattern 10 (LPAT(10)) is obtained from LPAT(9) by shifting left one bit. This allows the left pattern to be generated from the previous left pattern during the same instruction that loads the right pattern from the constant field. The same relation holds between pattern 1Ø and pattern 11. The left operand for 12 is the complement of the left pattern for 11.

```
(27B) AP3P4:    (R6,M)←Y←(CONSTANT = LPAT3),
                SET FLAG A;
(30B)           (R5, Z)←Y←(CONSTANT = RPAT3);
(31B)           Q←Y←(CONSTANT=SPAT3),
                RØ←X←(BL = -1),
                CALL ADDSUB1;
(32B) AP5P6:    (R6, M)←Y←(CONSTANT = LPAT5),
                RESET FLAG A;
```

```
(33B)                (R5, Z)←Y←(CONSTANT = RPAT5);
(34B)                Q←Y←(CONSTANT = SPAT5),
                     RØ←X←(BL = -1),
                     CALL ADDSUBØ;
(35B) AP7P8:         (R6, M)←Y←(CONSTANT = LPAT7),
                     RESET FLAG A;
(36B)                (R5, Z)←Y←(CONSTANT = RPAT7);
(37B)                Q←Y←(CONSTANT = SPAT7),
                     RØ←X←(BL = -1),
                     CALL ADDSUB1;
(40B) AP9:           (R5, Z, M)←Y←(CONSTANT = LPAT9);
(41B)                Q←Y←(CONSTANT = SPAT9),
                     R6←X←M,
                     CALL ADDSUBØ;
(42B) APlØ:          M←X←M LCY1,
                     (R5, Z)←Y←(CONSTANT = RPATlØ)
(43B)                Q←Y←(CONSTANT = SPATlØ),
                     R6←X←M,
                     CALL ADDSUB1;
(44B) AP11:          M←X←M LCY1,
                     (R5, Z)←Y←(CONSTANT = RPAT11);
(45B)                Q←Y←(CONSTANT = SPAT11),
                     R6←X←M,
                     CALL ADDSUB1;
(46B) AP12:          M←X←NOT M,
                     (R5, Z)←Y←(CONSTANT = RPAT12),
```

```
(47B)                   Q←Y←(CONSTANT = SPAT12),
                        R6←X←M,
                        CALL ADDSUBØ;
(50B-56B)  ACC:         Carry accelerator tests
```

The accelerator tests are performed separately by instructions 50B through 56B because the ADDSUB subroutine is not applicable.

The carry accelerators used in the Model 1 are redundant logically. If any accelerator gates are faulty in the direction of true outputs, only timing tests will detect the fault. For this reason, the accelerator tests must be run at the maximum clock frequency.

The first two instructions present the adder with a pattern which enables all accelerator gates. Both instructions have VCY = 1 because two double instruction cycles are required to stabalize the carry circuits if the accelerators are faulty. A false low order carry is provided so that the test instruction which follows can check the propagation delay of a true low order carry. The test instruction reverses the values of BL and BR so that all gates must stabalize again.

```
(50B)  ACC1:            BL←7(BL output = -1), BR←1ØB(BR output
                        =Ø),
                        VCY = 1, LOC = Ø,
                        GOTO ACCØ IF ATTN1;
(51B)                   BL←7, BR←1ØB,
                        VCY = 1, LOC = Ø;
```

```
(52B)                BL←1ØB, BR←7,
                     VCY = 1, LOC = 1, TAX
                     GOTO ACC1 IF X ≠ Ø;
(53B)  ACCØ:         BL←7, BR←1ØB,
                     VCY = 1, LOC = 1,
                     GOTO EXIT IF ATTN1;
(54B)                BL←7, BR←1ØB,
                     VCY = 1, LOC = 1;
(55B)                BL←1ØB, BR←7,
                     VCY = 1, LOC = Ø, TAX
                     GOTO ACCØ IF X > Ø;
```

The last three instructions test the accelerators for not-carry. As with ACC1, the two instructions starting at ACCØ present an unchanging input pattern to the adder over a long enough period of time to allow carry stabalization, even if the carry accelerators are faulty. The third command retains the same input pattern, again with BL and BR reversed, while the input carry changes from one to zero. The sum at the most significant bit of X is checked as a verification of the worse carry path.

(56B-57B) EXIT: the terminating instructions.

The last two instructions cause the adder/cycler acceptance routine to either loop or to proceed, according to the state of RSLAT2.

```
(56B)  EXIT:        GOTO Ø IF RSLAT2 = Ø;
(57B)               GOTO 100B;
```

If RSLAT = 1, each acceptance test is performed in sequence. If RSLAT = Ø, each acceptance test is self contained and loops indefinitely. The last acceptance test loops to the first, if RSLAT =1 (GO TO ABSOLUTEØ).

```
PAGE   1    /N.EJESSE    17 OCTOBER 1969   23:24

*  ADDER ACCEPTANCE ROUTINE
*  MACROS

       MACRO MO<MACRO;
       MACRO PM<DEFINE PARAMETER *1*<*2*;
       MACRO SC<DEFINE SCONDITION *1*<*2*,(*3*)*4*;
       MACRO BC<DEFINE BCONDITION *1*<*2*,(*3*)*4*;
       MACRO LPAT<LPAT*1*;
       MACRO RPAT<RPAT*1*;
       MACRO SPAT<SPAT*1*;

*  BRANCH CONDITION DEFINITIONS

       DEFINE BCONDITION ATTN1<36B,();
       DEFINE BCONDITION FLAGA<33B,();
       DEFINE BCONDITION NOTRSLAT2<31B,();
       DEFINE BCONDITION BL=0,BLEQ0<22B,();
       DEFINE BCONDITION R0>=0,R0GE0<12B,();
       DEFINE BCONDITION X#0,XNE0<3,();
       DEFINE BCONDITION X<0,XLT0<4,();
       DEFINE BCONDITION X>0,XGT0<6,();

*  SPECIAL CONDITION DEFINITIONS
       DEFINE SCONDITION RESA<31B,();
       DEFINE SCONDITION SETA<30B,();

*  PARAMETERS

       DEFINE PARAMETER START<300B;
       DEFINE PARAMETER NEXTBOARD<START+100B;

       DEFINE PARAMETER LPAT1<40100200B;
       DEFINE PARAMETER LPAT2<2004010B;
       DEFINE PARAMETER LPAT3<67757737B;
       DEFINE PARAMETER LPAT4<77376775B;
       DEFINE PARAMETER LPAT5<4210421B;
       DEFINE PARAMETER LPAT6<LPAT5;
       DEFINE PARAMETER LPAT7<10421042B;
       DEFINE PARAMETER LPAT8<LPAT7;

       DEFINE PARAMETER LPAT9<35673567B;
       DEFINE PARAMETER LPAT10<73567356B;
       DEFINE PARAMETER LPAT11<67356735B;
       DEFINE PARAMETER LPAT12<10421042B;

       DEFINE PARAMETER RPAT1<43707617B;
       DEFINE PARAMETER RPAT2<76174370B;
       DEFINE PARAMETER RPAT3<20040100B;
       DEFINE PARAMETER RPAT4<1002004B;
       DEFINE PARAMETER RPAT5<67557336B;
       DEFINE PARAMETER RPAT6<7366755B;
       DEFINE PARAMETER RPAT7<76775773B;
       DEFINE PARAMETER RPAT8<57737677B;
       DEFINE PARAMETER RPAT9<35673567B;
```

```
PAGE   2    /N.EJESSE    17 OCTOBER 1969   23:24

        DEFINE PARAMETER RPAT10<21042104B;
        DEFINE PARAMETER RPAT11<56735673B;
        DEFINE PARAMETER RPAT12<14631463B;

        DEFINE PARAMETER SPAT1<3206415B;
        DEFINE PARAMETER SPAT2<64150320B;
        DEFINE PARAMETER SPAT3<10020040B;
        DEFINE PARAMETER SPAT4<401002B;
        DEFINE PARAMETER SPAT5<73767757B;
        DEFINE PARAMETER SPAT6<77577376B;
        DEFINE PARAMETER SPAT7<7417036B;
        DEFINE PARAMETER SPAT8<70360741B;
        DEFINE PARAMETER SPAT9<73567356B;
        DEFINE PARAMETER SPAT10<6615433B;
        DEFINE PARAMETER SPAT11<46314631B;
        DEFINE PARAMETER SPAT12<25252525B;

*  PROGRAM PROPER

        ORG START;


300:  CONST: .C<-1, .TCX, .LQX, GOTO LCYSF IF ATTN1 .LZY*
.TCX = 1    43
.LQX = 1    74
.MC = 36    ,4,3,2,1
.B = 304     10,11,15,
.C = -1    18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39
 LZY = 1  77                                                          40,41

301:  .C<0, .TCX, .LMX, GOTO CONST IF X#0
.TCX = 1    43
.LMX = 1    72
.MC = 3    5,4
.B = 300     10,11,


302:  .C<-1, .BL<7, .BR<10B, GOTO CONST IF X#0
.BR = 10    ,82
.BL = 7    81,80,79
.MC = 3    5,4
.B = 300     10,11,
.C = -1    18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
                                                                      40,41

303:  .C<1, .TCY, .LQY, GOTO CONST ON Q EQV M # 0   (BL)
.BR = 10    ,82
.BL = 1    81
.TCY = 1    44
.TAX = 1    49
.LQY = 1    75
.MC =23    5,4, 1
.B = 300     10,11,
.C = 1    41,
```

```
PAGE   3    /N.EJESSE    17 OCTOBER 1969   23:24

304:  LCYSF: Q≤X≤Q LCY 1, M≤Y≤1B6, GOTO LCYZ IF ATTN1
.BR = 10    ,82
.BL = 2     ,80
.MS = 1     65
.TCY = 1    44
.LMY = 1    73
.LQX = 1    74
.MC = 36    ,4,3,2,1
.B = 315     10,11,14,15,17,
.C = 1000000   23,


305:  Q≤X≤Q LCY 2
.BR = 10    ,82
.BL = 2     ,80
.MS = 2     ,64
.LQX = 1    74
.VCY = 1    86


306:  Q≤X≤Q LCY 3
.BR = 10    ,82
.BL = 2     ,80
.MS = 3     65,64
.LQX = 1    74
.VCY = 1    86


307:  Q≤X≤Q LCY 4
.BR = 10    ,82
.BL = 2     ,80
.MS = 4     ,63
.LQX = 1    74
.VCY = 1    86


310:  Q≤X≤Q LCY 8
.BR = 10    ,82
.BL = 2     ,80
.MS = 5     65,63
.LQX = 1    74
.VCY = 1    86


311:  Q≤X≤Q LCY 12
.BR = 10    ,82
.BL = 2     ,80
.MS = 6     ,64,63
.LQX = 1    74
.VCY = 1    86


312:  Q≤X≤Q LCY 16
.BR = 10    ,82
```

```
PAGE   4    /N.EJESSE    17 OCTOBER 1969   23:24

.BL = 2    ,80
.MS = 7    65,64,63
.LQX = 1    74
.VCY = 1    86


313:  Q<X<Q LCY 20
.BR = 10    ,82
.BL = 2    ,80
.MS = 10    ,62
.LQX = 1    74
.VCY = 1    86


314:  .C<1, .TCY, .LQY, GOTO LCYSF ON M EOR Q # 0  (BL)
.BR = 10    ,82
.BL = 16    ,80,79,78
.TCY = 1    44
.TAX = 1    49
.LQY = 1    75
.MC =23    5,4, 1
.B = 304     10,11,15,
.C = 1    41,


315:  LCYZ:  Z<Y<40000005B, M<R0<X<Q, CALL ENTER
.MCONT = 1    7
.BR = 10    ,82
.BL = 2    ,80
.TCY = 1    44
.TXW = 1    47
.TAX = 1    49
.LMX = 1    72
.LZY = 1    77
.LR0 = 1    58
.MC = 1    5
.VCY = 1    86
.B = 366     10,11,12,13,15,16,
.C = 4000005    21,39,41,


316:  Z<Y<40000005B, M<R0<X<-1, CALL ENTER
.MCONT = 1    7
.BR = 10    ,82
.BL = 7    81,80,79
.TCY = 1    44
.TXW = 1    47
.TAX = 1    49
.LMX = 1    72
.LZY = 1    77
.LR0 = 1    58
.MC = 20    ,1
.VCY = 1    86
```

```
PAGE   5    /N.EJESSE    17 OCTOBER 1969   23:24

.B = 366     10,11,12,13,15,16,
.C = 4000005    21,39,41,


317:  Z←Y←4000005B, M←R0←X←0, CALL ENTER
.MCONT = 1    7
.TCY = 1    44
.TXW = 1    47
.LMX = 1    72
.LZY = 1    77
.LR0 = 1    58
.MC = 1    5
.VCY = 1    86
.B = 366     10,11,12,13,15,16,
.C = 4000005    21,39,41,


320:  AP1:    M←X←LPAT1
.TCX = 1    43
.LMX = 1    72
.VCY = 1    86
.C = -37677600    18,26,34,


321:  Z←X←RPAT1
.TCX = 1    43
.LZX = 1    76
.VCY = 1    86
.C = -34070161    18,22,23,24,25,26,30,31,32,33,34,38,39,40,41,


322:  M←X←M+Z, .VCY, .LOC, Q←Y←SPAT1, GOTO AP2 IF ATTN1
.BR = 4    ,83
.BL = 4    ,79
.LOC = 1    50
.TCY = 1    44
.TAX = 1    49
.LMX = 1    72
.LQY = 1    75
.MC = 36    ,4,3,2,1
.B = 324     10,11,13,15,
.C = 3206415    22,23,25,30,31,33,38,39,41,


323:  M←Y←LPAT1, GOTO *-1 ON M EOR Q # 0    (BL)
.BR = 10    ,82
.BL = 16    ,80,79,78
.TCY = 1    44
.TAX = 1    49
.LMY = 1    73
.MC =23    5,4, 1
.B = 322     10,11,13,16,
.C = -37677600    18,26,34,
```

```
PAGE   6    /N.EJESSE    17 OCTOBER 1969   23:24

324:  AP2:   M<X<M LCY 4, Z<Y<RPAT2
.BR = 10    ,82
.BL = 4     ,79
.MS = 4     ,63
.TCY = 1    44
.LMX = 1    72
.LZY = 1    77
.VCY = 1    86
.C = -1603410    18,19,20,21,22,26,27,28,29,30,34,35,36,37,38,


325:  M<X<M+Z, .VCY, Q<Y<SPAT2, GOTO AP3P4 IF ATTN1
.BR = 4     ,83
.BL = 4     ,79
.TCY = 1    44
.TAX = 1    49
.LMX = 1    72
.LQY = 1    75
.MC = 36    ,4,3,2,1
.B = 327     10,11,13,15,16,17,
.C = -13627460    18,19,21,26,27,29,34,35,37,


326:  M<Y<LPAT2, GOTO *-1 ON M EOR Q # 0    (BL)
.BR = 10    ,82
.BL = 16    ,80,79,78
.TCY = 1    44
.TAX = 1    49
.LMY = 1    73
.MC =23    5,4, 1
.B = 325     10,11,13,15,17,
.C = 2004010    22,30,38,


327:  AP3P4: R6<M<Y<LPAT3, SETA
.MS = 30    ,62,61
.TCY = 1    44
.TYW = 1    48
.LMY = 1    73
.LRN = 6    ,70,69
.C = -10020041    18,19,21,22,23,24,25,26,27,29,30,31,32,33,34,35,37,38,39,
                                                                  40,41

330:  R5<Z<Y<RPAT3
.TCY = 1    44
.TYW = 1    48
.LZY = 1    77
.LRN = 5    71,69
.VCY = 1    86
.C = 20040100    19,27,35,


331:  Q<Y<SPAT3, R0<X<-1, CALL ADDS1
```

```
PAGE   7    /N.EJESSE    17 OCTOBER 1969   23:24

.MCONT = 1    7
.BR = 10     ,82
.BL = 7     81,80,79
.TCY = 1     44
.TXW = 1     47
.TAX = 1     49
.LQY = 1     75
.LRO = 1     58
.MC = 20     ,1
.VCY = 1     86
.B = 370      10,11,12,13,14,
.C = 10020040    20,28,36,


332:  AP5P6: R6<M<Y<LPAT5, RESA
.MS = 31     65,62,61
.TCY = 1     44
.TYW = 1     48
.LMY = 1     73
.LRN = 6     ,70,69
.C = 4210421     21,25,29,33,37,41,


333:  R5<Z<Y<RPAT5  ,  GOTO 36Ø
.TCY = 1     44
.TYW = 1     48
.LZY = 1     77
.LRN = 5     71,69
.VCY = 1     86
.C = -10220442    18,19,21,22,23,24,26,27,29,30,31,32,34,35,37,38,39,40,
 MC = 1       5
 B = 36Ø      10, 11, 12, 13
334:              , R0<X<-1, CALL ADDS0
.MCONT = 1    7
.BR = 10     ,82
.BL = 7     81,80,79
.
.TXW = 1     47
.TAX = 1     49
.
.LRO = 1     58
.MC = 1     5
.VCY = 1     86
.B = 375      10,11,12,13,14,15,17,
.C = -4010021    18,19,20,22,23,24,25,26,27,28,30,31,32,33,34,35,36,38,39,
                                                                  40, 41


335:  AP7P8: R6<M<Y<LPAT7, RESA
.MS = 31     65,62,61
.TCY = 1     44
.TYW = 1     48
.LMY = 1     73
```

```
PAGE   8    /N.EJESSE    17 OCTOBER 1969   23:24

.LRN = 6    ,70,69
.C = 10421042    20,24,28,32,36,40,


336:  R5<Z<Y<RPAT7
.TCY = 1    44
.TYW = 1    48
.LZY = 1    77
.LRN = 5    71,69
.VCY = 1    86
.C = -1002005    18,19,20,21,22,24,25,26,27,28,29,30,32,33,34,35,36,37,38,
                                                                    40,41

337:  Q<Y<SPAT7, R0<X<-1, CALL ADDS1
.MCONT = 1    7
.BR = 10    ,82
.BL = 7    81,80,79
.TCY = 1    44
.TXW = 1    47
.TAX = 1    49
.LQY = 1    75
.LR0 = 1    58
.MC = 20    ,1
.VCY = 1    86
.B = 370     10,11,12,13,14,
.C = 7417036    21,22,23,24,29,30,31,32,37,38,39,40,


340:  AP9:    R5<Z<M<Y<LPAT9
.TCY = 1    44
.TYW = 1    48
.LMY = 1    73
.LZY = 1    77
.LRN = 5    71,69
.VCY = 1    86
.C = 35673567    19,20,21,23,24,25,27,28,29,31,32,33,35,36,37,39,40,41,


341:  Q<Y<SPAT9, R6<X<M, CALL ADDS0
.MCONT = 1    7
.BR = 10    ,82
.BL = 4    ,79
.TCY = 1    44
.TXW = 1    47
.TAX = 1    49
.LQY = 1    75
.LRN = 6    ,70,69
.MC = 1    5
.VCY = 1    86
.B = 375     10,11,12,13,14,15,17,
.C = -4210422    18,19,20,22,23,24,26,27,28,30,31,32,34,35,36,38,39,40,
```

```
PAGE   9    /N.EJESSE    17 OCTOBER 1969   23:24

342:  AP10:  M<X<M LCY 1, R5<Z<Y<RPAT10
.BR = 10    ,82
.BL = 4     ,79
.MS = 1    65
.TCY = 1    44
.TYW = 1    48
.LMX = 1    72
.LZY = 1    77
.LRN = 5    71,69
.VCY = 1    86
.C = 21042104    19,23,27,31,35,39,


343:  Q<Y<SPAT10, R6<X<M, CALL ADDS1
.MCONT = 1    7
.BR = 10    ,82
.BL = 4     ,79
.TCY = 1    44
.TXW = 1    47
.TAX = 1    49
.LQY = 1    75
.LRN = 6    ,70,69
.MC = 20    ,1
.VCY = 1    86
.B = 370     10,11,12,13,14,
.C = 6615433    21,22,24,25,29,30,32,33,37,38,40,41,


344:  AP11:  M<X<M LCY 1, R5<Z<Y<RPAT11
.BR = 10    ,82
.BL = 4     ,79
.MS = 1    65
.TCY = 1    44
.TYW = 1    48
.LMX = 1    72
.LZY = 1    77
.LRN = 5    71,69
.VCY = 1    86
.C = -21042105    18,20,21,22,24,25,26,28,29,30,32,33,34,36,37,38,40,41,


345:  Q<Y<SPAT11, R6<X<M, CALL ADDS1
.MCONT = 1    7
.BR = 10    ,82
.BL = 4     ,79
.TCY = 1    44
.TXW = 1    47
.TAX = 1    49
.LQY = 1    75
.LRN = 6    ,70,69
.MC = 1    5
.VCY = 1    86
.B = 370     10,11,12,13,14,
```

```
PAGE  10    /N.EJESSE    17 OCTOBER 1969   23:24

.C = -31463147    18,21,22,25,26,29,30,33,34,37,38,41,


346:  AP12:  M←X←NOT M, R5←Z←Y←RPAT12
.BR = 10    ,82
.BL = 13    81,80,78
.TCY = 1    44
.TYW = 1    48
.TAX = 1    49
.LMX = 1    72
.LZY = 1    77
.LRN = 5    71,69
.VCY = 1    86
.C = 14631463    20,21,24,25,28,29,32,33,36,37,40,41,


347:  Q←Y←SPAT12, R6←X←M, CALL ADDS0
.MCONT = 1    7
.BR = 10    ,82
.BL = 4    ,79
.TCY = 1    44
.TXW = 1    47
.TAX = 1    49
.LQY = 1    75
.LRN = 6    ,70,69
.MC = 20    ,1
.VCY = 1    86
.B = 375     10,11,12,13,14,15,17,
.C = 25252525    19,21,23,25,27,29,31,33,35,37,39,41,


350:  ACC1:  .BL←7, .BR←10B, GOTO ACC0 IF ATTN1
.BR = 10    ,82
.BL = 7    81,80,79
.MC = 36    ,4,3,2,1
.B = 353     10,11,12,14,16,17,


351:  .BL←7, .BR←10B  Z←RØ*
.BR = 10    ,82
.BL = 7    81,80,79
.VCY = 1    86
.THY = 1    46
.LZY = 1    77
352:  .BL←10B, .BR←7, .LOC, .TAX, GOTO ACC1 IF X#0, RØ ← X *
.BR = 7    85,84,83
.BL = 10    ,78
.LOC = 1    50
.TAX = 1    49
.MC = 3    5,4
.B = 350     10,11,12,14,
.TXW = 1     47
.LRØ = 1     58
```

PAGE 11 /N.EJESSE 17 OCTOBER 1969 23:24

```
353:  ACC0:  .BL←7, .BR←10B, .LOC, GOTO EXIT IF ATTN1
.BR = 10    ,82
.BL = 7    81,80,79
.LOC = 1    50
.MC = 36    ,4,3,2,1
.B = 356     10,11,12,14,15,16,

354:  .BL←7, .BR←10B, .LOC, RESET RSL#2, Z←R1 *
.BR = 10    ,82          .RRN = 1      66
.BL = 7    81,80,79      .THY = 1      46
.LOC = 1    50           .LZY = 1      77
.VCY = 1    86
 MS = 32

355:  .BL←10B, .BR←7, .TAX, GOTO ACC0 IF X>0 , R1 ← X  *
.BR = 7    85,84,83      .TXW = 1      47
.BL = 10    ,78          .LRN = 1      69
.TAX = 1    49
.MC = 6    ,4,3
.B = 353     10,11,12,14,16,17,

356:  EXIT:  GOTO START IF NOTRSLAT2
.MC = 31    5,2,1
.B = 300     10,11,

357:  GOTO NEXTBOARD
.MC = 1    5
.VCY = 1    86
.B = 400     9,

360:  Q←Y←SPAT5'
TCY = 1    44
LQX = 1    75
C = 4010020    21,29,37

361:  Q←Q̄, GOTO 334
TAX = 1    49
BR = 15    82,83,85
BL = 1Ø    78
MC = 2Ø    1
MCONT = 2  6
LQX = 1    74
VCY = 1    86
B = 334    10,11,13,14,15

364:  LCYZS:  M←Y←R0, Q←X← -34000077B, .BL←16B, RETURN IF BL=0
.MCONT = 2     ,6
.BL = 16     ,80,79,78
.TCX = 1     43
.THY = 1     46
.LMY = 1     73
.LQX = 1     74
.MC = 22     ,4,1
.C = 43777701B  18,22,23,24,25,26,27,28,29,30,31,32,33,34,35,41

365:  LOOP:  Z←X←Q+Z, Q←Y←R0, GOTO LCYZS IF X<0    *
.BR = 4    ,83
```

```
PAGE  12    /N.EJESSE    17 OCTOBER 1969   23:24

.BL = 2    ,80
.THY = 1    46
.TAX = 1    49
.LQY = 1    75
.LZX = 1    76
.MC = 4    ,3                For CPU/I only: M = 17    2,3,4,5    *
.B = 364     10,11,12,13,15,


366:  ENTER: M<X<M LCH Z, RETURN IF ATTN1
.MCONT = 2    ,6
.BR = 10    ,82
.BL = 4    ,79
.MS = 12    ,64,62
.LMX = 1    72
.MC = 36    ,4,3,2,1


367:  LAST:  M<X<M LCL Z, Q<Y<4000011B, GOTO LOOP
.BR = 10    ,82
.BL = 4    ,79
.MS = 11    65,62
.TCY = 1    44
.LMX = 1    72
.LQY = 1    75
.MC = 20    ,1
.VCY = 1    86
.B = 365     10,11,12,13,15,17,
.C = 4000011    21,38,41,


370:  ADDS1: M<X<M!Z, .LOC, GOTO ADLOOP IF ATTN1
.BR = 4    ,83
.BL = 4    ,79
.LOC = 1    50
.TAX = 1    49
.LMX = 1    72
.MC = 36    ,4,3,2,1
.B = 372     10,11,12,13,14,16,


371:  M<Y<R6, GOTO ADDS1 ON M EOR Q # 0    (BL)
.RRN = 6    ,67,66
.BR = 10    ,82
.BL = 16    ,80,79,78
.THY = 1    46
.TAX = 1    49
.LMY = 1    73
.MC =23    5,4, 1
.B = 370     10,11,12,13,14,


372:  ADLOOP: R6<X<M LCY 4, M<Y<R5
```

```
PAGE  13    /N.EJESSE    17 OCTOBER 1969   23:24

.RRN = 5    68,66
.BR = 10    ,82
.BL = 4     ,79
.MS = 4     ,63
.THY = 1    46
.TXW = 1    47
.LMY = 1    73
.LRN = 6    ,70,69
.VCY = 1    86


373:  Z<X<M LCY 4, M<Y<R6, RETURN IF R0>=0
.MCONT = 2    ,6
.RRN = 6    ,67,66
.BR = 10    ,82
.BL = 4     ,79
.MS = 4     ,63
.THY = 1    46
.LMY = 1    73
.LZX = 1    76
.MC = 12    ,4,2


374:  Q<X<Q LCY 4, R0<Y<R0+1, GOTO ADDS1 IF FLAGA
.BR = 10    ,82
.BL = 2     ,80
.MS = 4     ,63
.IHR = 1    42
.THY = 1    46
.TYW = 1    48
.LQX = 1    74
.LR0 = 1    58
.MC = 33    5,4,2,1
.B = 370     10,11,12,13,14,


375:  ADDS0: M<X<M!Z, GOTO ADLOOP IF ATTN1
.BR = 4     ,83
.BL = 4     ,79
.TAX = 1    49
.LMX = 1    72
.MC = 36    ,4,3,2,1
.B = 372     10,11,12,13,14,16,


376:  M<Y<R6, GOTO ADDS0 ON M EOR Q # 0   (BL)
.RRN = 6    ,67,66
.BR = 10    ,82
.BL = 16    ,80,79,78
.THY = 1    46
.TAX = 1    49
.LMY = 1    73
.MC = 23    5,4, 1
.B = 375     10,11,12,13,14,15,17,
```

```
PAGE  14    /N.EJESSE    17 OCTOBER 1969   23:24

377:  GOTO ADLOOP
.MC = 1   5
.VCY = 1   86
.B = 372    10,11,12,13,14,16,
```

Modifications for CPU - Phase I

In order to apply this acceptance procedure for the CPU-Phase I some changes had to be made, since CPU/I branches on the Z-Bus instead of the X-Bus (MC = 2 through MC = 6). The changes are marked with an astrix (*) in the listing. All these modifications do in no way affect the idea of the test nor the sequence of the execution of the instructions, as far as the general Microprocessor is concerned. CPU/I will deviate from the execution sequence in the following way:

1.) Instruction 300 to 302:

LZY = 1 in 300 loads Z with Ø. Thus the branches Z = Ø in 301 and 302 are never true, i.e. these tests are not done by the CPU/I.

Added bit LZY (bit 77) in 300.

2.) Test ACC1 (instruction 350-352):

Branch condition MC = 3 branches in CPU/I on Z ≠ Ø, compared with X≠0 in the other Microprocessors. Since RØ = 1 (from previous subroutine) has been transferred in to Z at instruction 351, MC = 3 causes a branch to 350 during the first execution of 352.

But if 352 is reached for the second time, Z will be loaded with whatever was on the X-Bus during the first time and thus allows to proceed to the next test if ACC1 is successful.

Permanent added bits: 351: LZY (77), THY (46)

352: TXW (47), LRØ (58)

3.) Test ACC0 (instruction 353-355):

Branch condition MC = 6 branches in CPU/I on Z > Ø.

Similar to the previous modification, the value of X is stored through R1 in Z during the first execution of 355, while Z still contains 1 (from R1 = 1 in 374). During the second time a successful ACC0 - Test (during the first time) has made Z = 0 and stops the test loop.

Permanent added bits: 354: RRN2 (66), THY (46), LZY (77)

355: TXW (47), LRN2 (69)

4.) Loop (instruction 365):

Branch condition X < Ø is also on the CPU/I available, however as MC = 17 rather than MC = 4. Thus during the time the CPU/I is using this test the following bits have to be temporary added:

Temporary added bits: 365: MC2 (2), MC4 (4), MC5 (5)

With this exception, the ROM-Board does not need any change when switched from CPU/I to a regular Microprocessor.