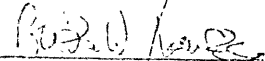


BCC  
MICROPROCESSOR  
MANUAL

50010

1 APRIL 70

Concurred By:

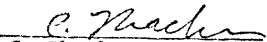


B. Lampson  
Programming

Approved By:



J. T. Quatse  
Vice-President



C. Thacker  
Engineering



A. Montoya  
Publications

TABLE OF CONTENTS

SECTION 1. GENERAL DESCRIPTION	1
A. INTRODUCTION.	3
B. PURPOSE OF EQUIPMENT.	3
C. BCC 500 SYSTEM.	3
1. Central Processing Unit 1	3
2. Central Processing Unit 2	3
3. Character Input and Output	3
4. Microscheduler	3
5. Auxiliary Memory Control	4
D. PHYSICAL DESCRIPTION.	4
E. LEADING PARTICULARS.	7
F. NON-STANDARD ABBREVIATIONS.	7
SECTION 2. GENERAL THEORY OF OPERATION	9
A. INTRODUCTION	10
1. Data Flow	10
2. Read-Only Memory	10
3. I Register	10
4. O Register	15
5. OS Register	15
6. M, Q, and Z Registers	15
7. Holding Registers	16
8. Scratchpad	16
9. Adder/Cycler	16
10. Typical Instruction Cycle	16

TABLE OF CONTENTS (cont'd)

B. MICROPROCESSOR INSTRUCTIONS.	17
1. Unsuccessful Branch Instructions	17
2. Stretched Unsuccessful Branch Instructions	20
3. Successful Branches	20
4. Stretched Successful Branch Instructions	20
5. Subroutine Calls	21
6. Deferred Branch Instruction	21
SECTION 3. DETAILED THEORY OF OPERATION	22
A. CONTROL LOGIC CARD.	23
1. Timing	23
2. State Counter	25
3. Register Clock Enable (RCE)	28
4. I2CLOCK	29
5. I3CLOCK	29
6. R0 thru R6 CLOCKS	30
7. MCLOCK, QCLOCK, ZCLOCK	30
8. K2 CLOCK	30
9. K3 CLOCK	31
10. K2X CLOCK	31
11. O Register Control	31
12. OS Register Control	32
13. BRTEST	33
14. Write Strobe	33
15. Priority Latching	34

TABLE OF CONTENTS (cont'd)

B. I REGISTER CARDS.	34
C. READ-ONLY MEMORY.	36
1. BSELECT Flip-Flop	36
2. BSS Flip-Flop	37
3. Address Decoding	37
D. O and OS REGISTER.	38
1. O Register	38
2. O Register Incrementer	41
3. OS Register	41
E. BRANCH CONDITIONS CARD.	42
1. Branches	42
2. Attention Latches	44
F. SPECIAL FUNCTIONS CARD.	46
1. POT/PIN	49
2. Protect System	50
3. Request Strobe	51
G. MQZ REGISTRATION CARD.	51
1. M Register	52
2. Q Register	53
3. Z Register	53
4. Bool Boxes	53
H. ADDER/CYCLER CARD.	55
1. Adder	56
a. The Carry Portion	56

TABLE OF CONTENTS (cont'd)

H.	ADDER/CYCLER CARD. (cont'd)	
	b. Anticipated Carry	57
	c. FLUSH	58
	2. Cyclor	58
I.	PARITY CHECKER	58
	1. Parity Register	59
	2. Addressing Parity	61
J.	SCRATCH PAD	62
K.	HOLDING REGISTERS	63
	1. RDX	66
	2. Y-bus Gating	66
L.	REQUEST STROBE	66
M.	MEMORY INTERFACE	67
	1. Request Mode and Priority Latches	70
	2. Central Memory Request Section	71
	a. Requests to Central Memory Via the MPMBM.	71
	b. Requests Direct to Central Memory.	74
	c. Priority Requests, Clocks, and Load Signals.	74
	3. Private Memory Control Section	75
N.	I/O INTERFACE	76
	1. ATTENTION	77
	2. TUCLR	78
	3. Timing and Programming	78

LIST OF TABLES

1.	Reference Designations and Common Names.	4
2.	Leading Particulars.	7
3.	Non-Standard Abbreviations.	8
4.	Clock Generation.	23
5.	Branch Logic.	43
6.	Special Functions Logic.	46
7.	Left (Right) Bool Box Logic.	55
A1.	90-Bit Microinstruction Word.	A2
A2.	Branch Conditions.	A7
A3.	Special Functions.	A9
A4.	Bool Box Control.	A11
A5.	Summary of Signal Mnemonics	A12

LIST OF ILLUSTRATIONS

Figure 1.	BCC 500, System Block Diagram.	2
Figure 2.	Card Location.	5/6
Figure 3.	BCC Microprocessor.	11/12, 13/14
Figure 4.	Instruction Cycles.	18/19
Figure 5.	Clock Distribution.	24
Figure 6.	State Counter Set Terms.	26/27
Figure 7.	I Register, Functional Block Diagram.	35
Figure 8.	O & OS Register, Functional Block Diagram.	39/40
Figure 9.	Holding Registers, Functional Block Diagram.	64/65
Figure 10.	State Counter, Memory Interface.	72

INTRODUCTION

The BCC 500 system will accommodate a large number of terminals attached by means of data multiplexers and a network of high-speed communication lines. The data multiplexers can accommodate a variety of terminals and some local peripheral equipment such as medium-speed line printers and tape units. Although the first system produced by BCC will accommodate a very large number of terminals (over 2,000), the number of users who can simultaneously make use of the system is, of course, dependent on their computing requirements. For users of BASIC and similar conversational languages who are working on problems of complexity and size similar to those now being accommodated by remote access systems, this number is approximately 500. The system has thus been designated the Model 500.

The BCC 500 system consists of two CPU's, several special purpose management processors (handling scheduling, swapping, storage allocation, character I/O, error detection and recovery), drum and disk transfer units, and a multiple module core memory, all attached to a high-speed, high-bandwidth integrated circuit memory system. Up to eight drums (120 million bytes) and up to eight disks (three billion bytes) may be attached to the system. Peripherals such as printers, magnetic tapes, and card equipment are treated as remote devices which can be attached by means of a special purpose processor.

The communications and multiplexing facility, through which the terminals are brought into the system, is centered around a special multiplexer designed and produced by BCC as a part of the system. These multiplexers can accommodate either full- or half-duplex operation of from fifty low-speed terminals to a lesser number with line printing and card reading equipment. The terminals can be of disparate types and can operate at different rates and use different codes. The remote multiplexers which communicate with the system's character input/output processor provides for the automatic detection and correction of errors arising in the communications lines.

Complex resource allocation algorithms built into the special purpose processor in the system manage and control, the dynamic allocation of processor and memory resources. These algorithms permit minimum and maximum figures to be placed on many critical system resources demanded by users while operating on the system. Such a facility can guarantee the performance of the system as seen by a user or a group of users, independent of the load presented by other users or groups of users. This guarantee includes the activation of a user's program at a specified time.

SECTION 1  
GENERAL DESCRIPTION




50010


5. Auxiliary Memory Controller (AMC) - Controls data transfers from the drum storage files and the disc storage files to central memory.

D. PHYSICAL DESCRIPTION.

The BCC Microprocessor is comprised of two rows of cards. Figure 2 shows the locations of the individual assemblies. Table 1 is a list of common names, reference designations, and assembly numbers for the individual cards. Each row of cards is mounted on a panel 24 inches wide, 12 inches high, and 14 inches deep. Each panel contains 33 connectors (136-pin).

Table 1. Reference Designations and Common Names.

REFERENCE DESIGNATION	ASSEMBLY NOMENCLATURE	ASSEMBLY NUMBER	QUANTITY
A1	Clock & Local Control		1
A4	Memory Interface	AC00240	1
A5	Control Logic	AC00241	1
A7	Special Functions	AC00242	1
A9,	Branch Conditions		1
A10	Request Strobe (Breakpoint) 	AC00359	1
A11	O & OS Register	AC00245	1
A12, A18, A23	M,Q,& Z Registers	AC00246	3
A13, A19, A24	Adder/Cycler	AC00247	3
A14, A20, A25	Holding Register	AC00248	3
A15, A21, A26	Scratchpad	AC00299	3

 Test Purpose only

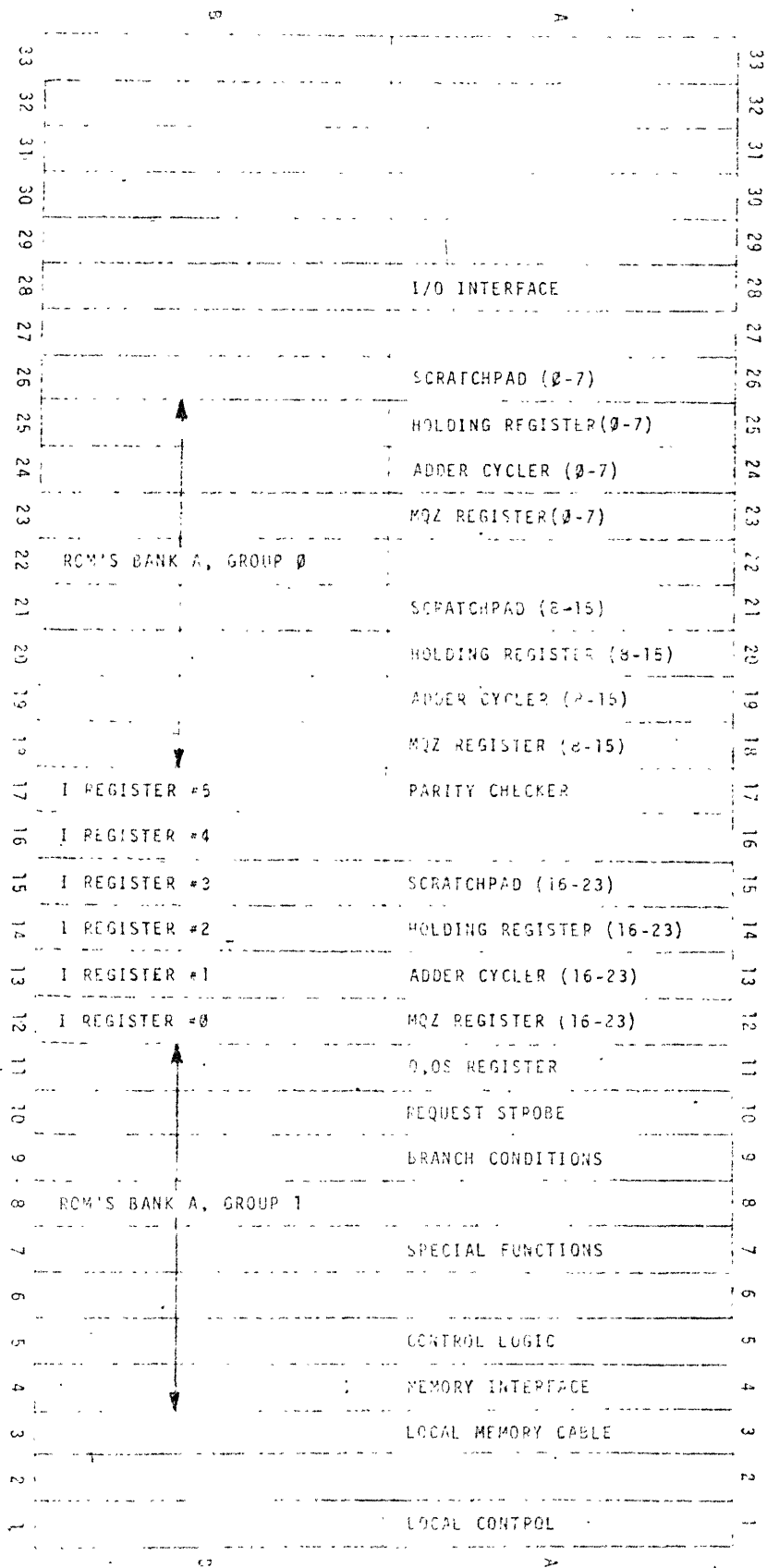


Figure 2. Card Location.



Table 1. Reference Designations and Common Names. (cont'd)

REFERENCE DESIGNATION	ASSEMBLY NOMENCLATURE	ASSEMBLY NUMBER	QUANTITY
A17	Parity Checker	AC00359	1
A28	I/O Interface	AC00751A	1
B1 thru B11 and B18 thru B33	Read-Only Memory (ROM)		as req'd
B12 thru B17	I Register	AC00252	6

## E. LEADING PARTICULARS.

Table 2 is a listing of the major physical and electrical characteristics of the BCC Microprocessor.

Table 2. Leading Particulars

PHYSICAL CHARACTERISTICS					
Height:	24 inches				
Width:	24 inches				
Depth:	14 inches				
ELECTRICAL CHARACTERISTICS					
General:	Extensive use of integrated circuits employing transistor-transistor logic (ttl). Most integrated circuits are contained in 14-lead dual in-line packaging.				
Programming:	Hardware programmed via diode memory (read-only).				
Power Required:	<table> <tr> <td>Pilot Power</td> <td> <math>\left\{ \begin{array}{l} +5 \text{ vdc} \\ -5 \text{ vdc} \\ +12 \text{ vdc} \end{array} \right.</math> </td> </tr> <tr> <td>Operating Power</td> <td> <math>\left\{ \begin{array}{l} +5 \text{ vdc} \\ -5 \text{ vdc} \\ +10 \text{ vdc} \end{array} \right.</math> </td> </tr> </table>	Pilot Power	$\left\{ \begin{array}{l} +5 \text{ vdc} \\ -5 \text{ vdc} \\ +12 \text{ vdc} \end{array} \right.$	Operating Power	$\left\{ \begin{array}{l} +5 \text{ vdc} \\ -5 \text{ vdc} \\ +10 \text{ vdc} \end{array} \right.$
Pilot Power	$\left\{ \begin{array}{l} +5 \text{ vdc} \\ -5 \text{ vdc} \\ +12 \text{ vdc} \end{array} \right.$				
Operating Power	$\left\{ \begin{array}{l} +5 \text{ vdc} \\ -5 \text{ vdc} \\ +10 \text{ vdc} \end{array} \right.$				
Signal Levels:	High: +2.2 vdc to +5.5 vdc Low: -0.5 vdc to +0.5 vdc				

## F. NON-STANDARD ABBREVIATIONS.

Table 3 is a listing of non-standard abbreviations applicable to the BCC Microprocessor and the BCC 500 System.

Table 3. Non-Standard Abbreviations.

ABBREVIATION OR MNEMONIC	DEFINITION
360/30	IBM <u>360</u> model <u>30</u> processor
AMC	<u>A</u> uxiliary <u>M</u> emory <u>C</u> ontrol
AMTU	<u>A</u> uxiliary <u>M</u> emory <u>T</u> ransfer <u>U</u> nit
BCC	<u>B</u> erkeley <u>C</u> omputer <u>C</u> orporation
CHIO	<u>C</u> haracter <u>I</u> nput/ <u>O</u> utput
CPU	<u>C</u> entral <u>P</u> rocessor(ing) <u>U</u> nit
CTR	<u>C</u> ompute <u>T</u> ime <u>R</u> egister
EOR	<u>E</u> xclusive <u>O</u> R
FM	<u>F</u> ast <u>M</u> emory
I/O	<u>I</u> nput- <u>O</u> utput
ITR	<u>I</u> nterval <u>T</u> imer <u>R</u> egister
MAP	Logical memory address to physical memory address <u>MAP</u>
MP	<u>M</u> icro <u>P</u> rocessor
MPMBM	<u>M</u> icro <u>P</u> rocessor <u>M</u> emory <u>B</u> us <u>M</u> ultiplexer
mpxr	<u>M</u> ultiplex <u>eR</u>
MSCH	<u>M</u> icro <u>S</u> cheduler
NDRO	<u>N</u> on- <u>D</u> estructive <u>R</u> ead- <u>O</u> ut
ROM	<u>R</u> ead- <u>O</u> nly <u>M</u> emory
RTR	<u>R</u> eal <u>T</u> ime <u>R</u> egister
System WR	<u>S</u> YSTEM <u>W</u> arning <u>R</u> egisters
TSU	<u>T</u> ransfer <u>S</u> ub- <u>U</u> nit
TUIM	<u>T</u> ransfer <u>U</u> nit <u>I</u> nterface <u>M</u> ultiplexer
UNG	<u>U</u> nique <u>N</u> ame <u>G</u> enerator

SECTION 2  
GENERAL THEORY OF OPERATION

SECTION 2  
GENERAL THEORY OF OPERATION

A. INTRODUCTION

The ECC Microprocessor is a synchronous, 24-bit digital computer. The flow of data between the functional elements of the computer is controlled by terms generated by the microprocessor 90-bit instruction word. Table A1, in Appendix A of this manual, is a listing of the bits in the instruction word and a definition of their function.

1. Data Flow. (See Figure 3.)

The X-bus and the Y-bus are the two principal intramicroprocessor data transfer busses. Data transfer between the microprocessor and ancillary devices is accomplished by the E1-bus and the E2-bus for a programmed-input (pin), and by the Z-bus for a programmed-output transfer (pot). Data transfer between the microprocessor and the core memory is accomplished by the M1-bus and the M2-bus for input data, and by the M-bus for output data. The data busses are 24-parallel transfer lines gated by terms derived from the microprocessor instruction word.

2. Read-Only Memory (ROM).

The ROM is a diode memory containing the 90-bit instruction words. The ROM is addressed by the O register and outputs the selected 90-bit instruction word to the I register.

3. I Register.

The I register is a 90-bit register and contains the current microprocessor instruction being executed. The



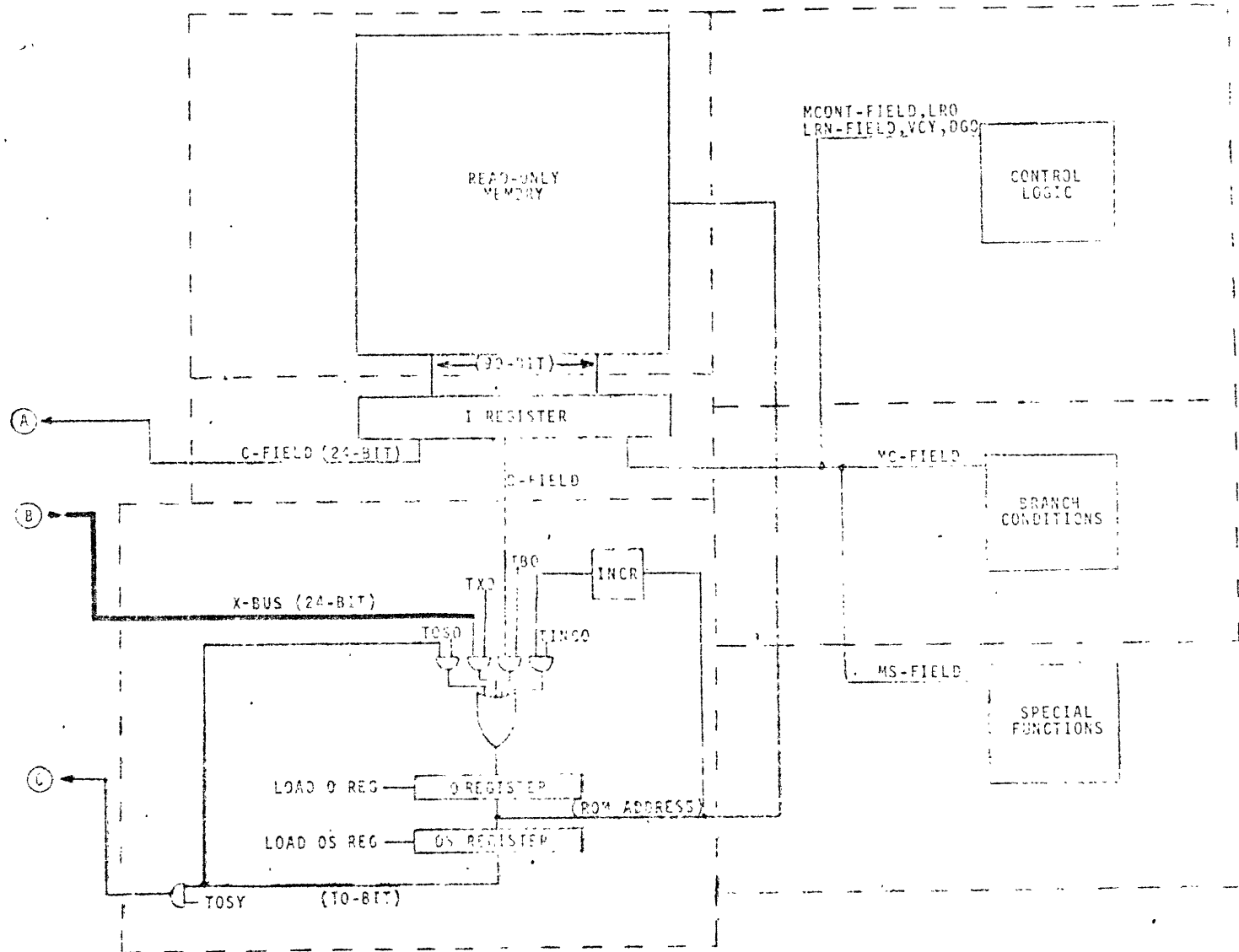


Figure 3b. BCC Microprocessor. (Sheet 2 of 2)

I register is normally loaded at the end of each machine cycle, from the ROM.

#### 4. O Register.

The O register is a 10-bit register which holds the address of the instruction word to be executed in the next cycle. It can be loaded from the B-field of the instruction word, the least significant 10 bits of the X-bus; the incremented contents of the O register; or the OS register:

#### 5. OS Register:

The OS register is a 10-bit register used to save the return address for subroutine calls. The contents of the OS register can be transferred to the Y-bus.

#### 6. M, Q, and Z Registers.

The M, Q, and Z registers are 24-bit registers, and are loaded from the X-bus or the Y-bus. The M register is also loaded, independently, from the local or central memory under control of the central memory interface. The buffered outputs of the Z register are used for parallel-output-transfers (pot) to ancillary devices via buffers on the I/O interface card. Two boolean (bool) boxes associated with the M, Q, and Z registers provide inputs to the adder/cycler. The output of the left or right bool box is the sixteen logical combinations of two variables M and Q, or the Z and Q, respectively. The logical functions are specified by the instruction word.

#### 7. Holding Registers.

The Holding registers, R<sub>H</sub> thru R<sub>G</sub>, are 24-bit registers which are loaded from the X-bus or the Y-bus. The output of the Holding register can be incremented, and is gated by the instruction word to the Y-bus.

#### 8. Scratchpad.

The scratchpad is a 24-bit by 64-word register, loaded from the X-bus; and read into the Y-bus. The address of the word to be fetched from the scratchpad is contained in the instruction word or the least significant six bits of the Z register.

#### 9. Adder/Cycler.

The adder portion of the adder/cycler is a 24-bit full-adder with an anticipated carry. The adder sums the output of the left and right bool boxes. The resultant sum is transferred to the X-bus. A low-order-carry input to the adder may be generated by the instruction word. The cycler portion of the adder/cycler is controlled by the instruction word or by the Z register, and left cycles the output of the left bool box to the X-bus.

#### 10. Typical Instruction Cycle.

Every instruction in the microprocessor is a conditional branch. The MCNT field (2 bits) of the instruction word defines the location of the branch address. The MC field of the instruction word specifies one of 64 conditions which, if satisfied, will cause a branch to occur. If the branch condition is not satisfied, the contents of the O

register (present address incremented) are used as the address of the next instruction word; and at the end of the machine cycle, the next instruction is fetched from the ROM and the O register is incremented. When the branch condition is satisfied, the O register is not incremented at the end of the machine cycle; the register clocks are inhibited. The cycle time of a successful conditional branch is, therefore, extended. A success causes the O register to be loaded from the source specified by the MCONT field of the instruction word. During the extended interval, the O register fetches the word in the branch destination address.

B. MICROPROCESSOR INSTRUCTIONS. (See Figure 4.)

A microprocessor instruction may require one, two, or three machine cycles to be executed. The control logic contains two flip-flops, XXB and XXC, which comprise the state counter (See Paragraph 3A2). The state counter determines from VCY, DCO, and BRANCH in the instruction whether an instruction will take one, two, or three machine cycles. The three possible states are state A (XXB'XXC'), state B (XXB·XXC'), and state C (XXB'·XXC). The length of time required to complete an instruction depends on the type of instruction.

1. Unsuccessful Branch Instructions.

These instructions do not branch and do not have the VCY bit in the instruction set. Execution of the instruction occurs in state A and requires only one machine

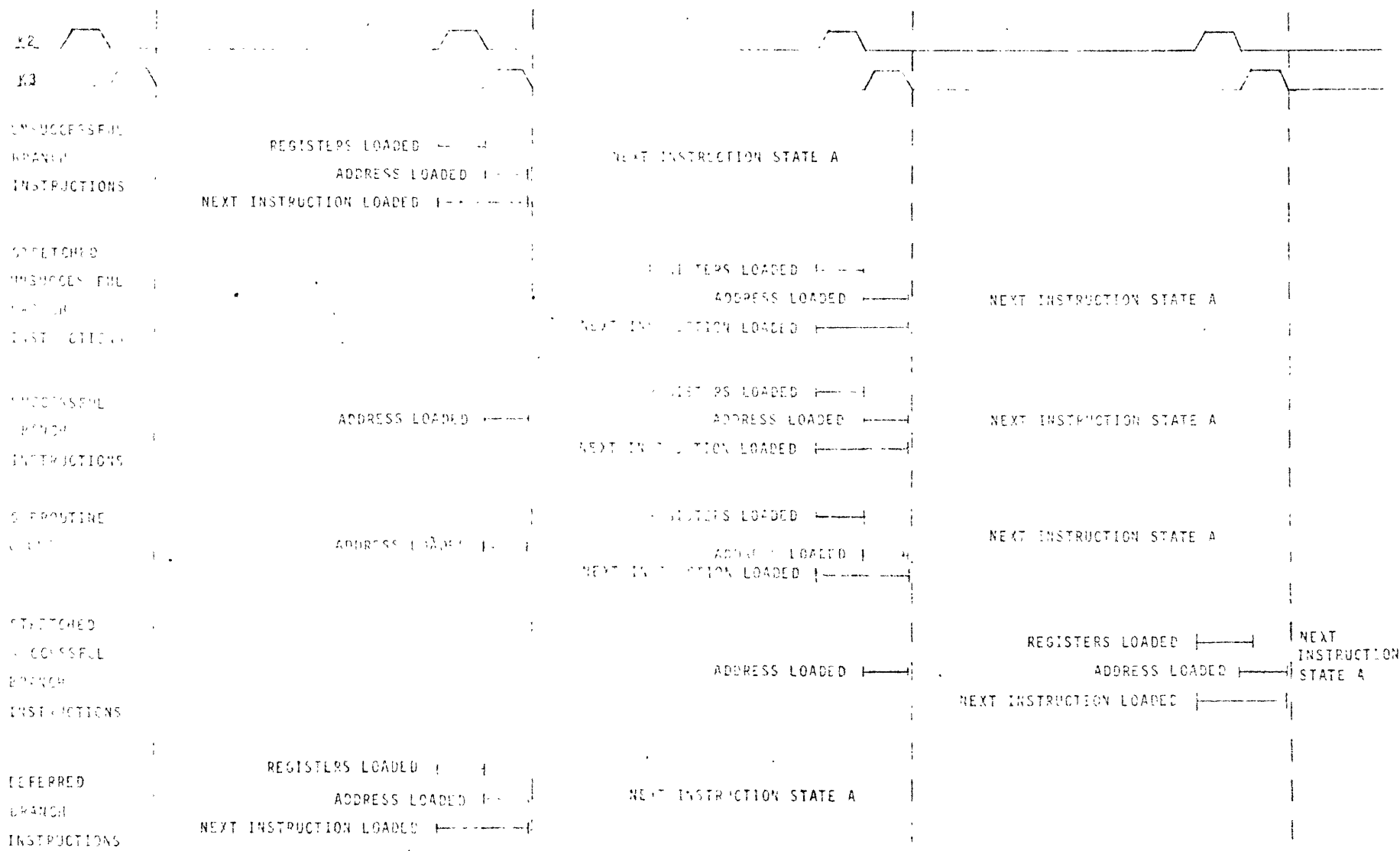


Figure 4. Instruction Cycles

cycle.

2. Stretched Unsuccessful Branch Instructions.

These instructions do not branch and have the VCY bit in the instruction set. Execution of the instruction occurs at the end of state B and, therefore, requires two machine cycles. State A is a waiting period that allows signals to propagate through long paths such as scratchpad, adder, and tests of X.

3. Successful Branches.

These are instructions where the branch condition is satisfied. They require two machine cycles and, therefore, use both state A and state B. Register loading is done at the end of state B, but the O register is loaded with the branch destination address at the end of state A. At the end of state B, the O register is loaded again, this time with the branch destination address plus one. Simultaneously, the I register is loaded with the instruction contained at the branch address.

4. Stretched Successful Branch Instructions.

These are instructions for which the branch condition is satisfied, and the VCY bit in the instruction is set. These instructions use three machine cycles and, therefore, require states A, B, and C. This type of instruction is used when the branch address or condition requires the time to be generated. Loading of any register specified in the instruction occurs at the end of state C.

5. Subroutine Calls.

A subroutine call stores the contents of the O register (the address of the instruction being executed plus 1) in the OS register, and loads the O register with the subroutine address. This instruction requires two machine cycles, state A and state B, and must, therefore, have the VCY bit set in the instruction.

6. Deferred Branch Instruction.

Deferred branch instructions cause the instruction after the deferred branch instruction (current address plus 1) to be executed before the branch occurs. In order to execute a deferred branch, the DGO bit in the instruction is set. This instruction uses state A only and requires one machine cycle. If the VCY is set in the instruction, an additional machine cycle will be available to prepare the branch condition or address, and the instruction will use state A and state B. In a deferred branch, the O register is loaded from the B-field of the instruction, the least significant ten bits of the X-bus, or the OS register at the end of state A if VCY is set.



50010

SECTION 3  
DETAILED THEORY OF OPERATION

50010

SECTION 3  
DETAILED THEORY OF OPERATION

A. CONTROL LOGIC CARD

The control logic card contains the timing, state counter, and the register clock circuits.

1. Timing.

The control logic card generates the clocks for the microprocessor from the system clock (K1). The microprocessor clocks are independently adjustable in width and delay by connecting jumpers on the control logic card between the delay line and pulse amplifiers used to generate the clock. The leading edge of the clock is determined by the start-tap position of the jumper. The trailing edge of the clock pulse is determined by the stop-tap position of the jumper. Each start-tap and stop-tap is identified on the control logic card by a letter. The clocks and their respective start- and stop-taps are listed in table 4. The clocks are used to gate data and control signals to the individual cards. Figure 5 shows the distribution of the clocks.

Table 4. Clock Generation.

NAME	START TAP	STOP TAP	FUNCTION
K10	N	M	Clocks state counter flop XXB and XXC.
K11	O	I	Gated to produce R6CLOCK thru R6CLOCK, MCLOCK, QCLOCK, ZCLOCK, PFL.
K12			Gated to produce 12CLOCKA and 12CLOCKB.

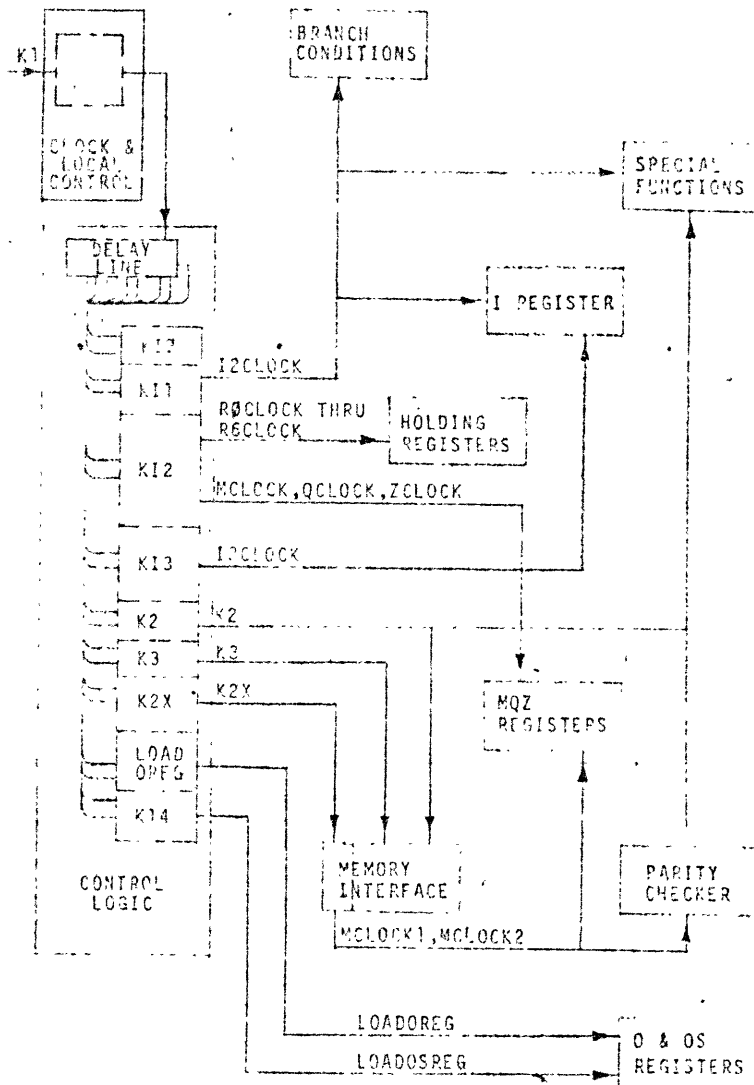


Figure 5. Clock Distribution.

Table 4. Clock Generation. (cont'd)

NAME	START TAP	STOP TAP	FUNCTION
KI3	J	H	Gated to produce I3CLOCKA and I3CLOCKB.
KI4	B	L	Gated to produce LOAD OSREG.
K2	D	E	Used on special function card, parity card, and memory interface.
K2X	K	G	Used in memory interface.
K3	C	F	Used in memory interface.
LOAD OREG	P	R	Generates LOAD OREG.

## 2. State Counter. (See Figure 6.)

The inputs to the state counter are the VCY bit and the DGO bit of the instruction word, and the BRANCH term from the branch conditions card. (BRANCH is generated when the condition specified in the branch condition field is satisfied.) The state counter is comprised of two flip-flops, XXB and XXC. State A is the initial state of all instructions (state A=XXB'·XXC'). State B (state B=XXB·XXC') is entered if the condition specified in the branch condition field is satisfied and DGO is not set (sXXB=XXB'·XXC'·BRANCH·DGO' ...), or if the VCY bit in the instruction is set (sXXB=...+XXB'·XXC'·VCY+...). The XXB flip-flop will be reset one machine cycle after it has been set (rXXB=XXB·...). State C (state C=XXB'·XXC)

	VCY	DSO	BRANCH	NEXT STATE	REGISTER CLOCKS	O REGISTER CLOCK	DS REGISTER CLOCK	O REGISTER DATA SOURCE
STATE	0	0	0	A	1	1	0	INCREMENTER
A	0	0	1	B	0	1	1; if a call	MCOUNT-FIELD
	0	1	0	A	1	1	0	INCREMENTER
	0	1	1	A	1	1	1; if a call	MCOUNT-FIELD
	1	0	0	E	1	0	0	-
	1	0	1	B	0	0	0	-
	1	1	0	B	0	0	0	-
	1	1	1	D	1	0	0	-
STATE	-	-	-	-	-	-	-	-
B	0	0	1	A	1	1	0	INCREMENTER
	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-
	1	0	0	A	1	1	0	INCREMENTER
	1	0	1	C	0	1	1; if a call	MCOUNT-FIELD
	1	1	0	A	1	1	0	INCREMENTER
	1	1	1	A	1	1	1; if a call	MCOUNT-FIELD
STATE	-	-	-	-	-	-	-	-
C	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-
	1	0	1	A	1	1	0	INCREMENTER
	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-

Figure 6. State Counter, Set Terms.

is entered after state B if both conditions to set XXB were satisfied ( $sXXC=XXB \cdot VCY \cdot BRANCH \cdot DGO' \dots$ ). The XXC flip-flop is reset one machine cycle after it has been set ( $rXXC=XXC \dots$ ). The state counter is clocked by  $KI\bar{0}$ , and is inhibited during a memory store or fetch by the SC term from the memory interface card ( $sXXB, rXXB, sXXC, rXXC = \dots \cdot KI\bar{0} \cdot SC'$ ). All the system clocks are gated with combinations of the state counter and the VCY, DGO, and BRANCH signals. Figure 6 shows VCY, DGO, BRANCH, and the state counter; and the status of the associated clock terms to the registers.

### 3. Register Clock Enable (RCE).

The RCE (register clock enable) gates the clocks for the M, Q, Z registers, the seven holding registers, and the I register. RCE is also sent to the memory interface. It signifies that the current machine cycle is the last cycle of the instruction. State A is the last cycle of an instruction if VCY is not set in the instruction and the branch condition was not satisfied ( $RCE=XXB' \cdot XXC' \cdot VCY' \cdot BRANCH' + \dots$ ), or if VCY is not set in the instruction and DGO is set in the instruction ( $RCE = \dots + XXB' \cdot XXC' \cdot VCY' \cdot DGO + \dots$ ). State B is the last cycle of an instruction if VCY is not set in the instruction ( $RCE = \dots + XXB \cdot VCY' + \dots$ ), or if DGO is set in the instruction ( $RCE = \dots + XXB \cdot DGO \dots$ ). State C is always the last cycle for three cycle instructions ( $RCE = \dots + XXC$ ).

### 4. I2CLOCK.

The I2CLOCK provides timing for the I register branch conditions, and special functions. The I2CLOCK signals from the Read-Only Memory (ROM) to the I register. The read-only memory is divided into two areas, bank A and B; each area contains an identical addressing structure comprised of 1024 words from 0000 to 1777B. I2CLOCK is divided into two clocks, I2CLOCKA and I2CLOCKB. I2CLOCKA is the clock for the data from the ROM bank A, and I2CLOCKB is the clock for the data from ROM bank B. I2CLOCKA is generated by  $KI\bar{1}$  if the ESELECT flip-flop is not set during the last cycle in an instruction ( $RCE$ ) and the instruction would not interfere with a store or fetch operation ( $SC'$ ): ( $I2CLOCKA=KI\bar{2} \cdot BSELECT' \cdot RCE \cdot SC'$ ). BSELECT is set by a special function and remains set until cleared by a special function. I2CLOCKB is generated by  $KI\bar{2}$  if the BSELECT flip-flop is set during the last cycle of an instruction ( $RCE$ ) and the instruction would not interfere with a store or fetch operation ( $SC'$ ): ( $I2CLOCKB=KI\bar{1} \cdot BSELECT \cdot RCE \cdot SC'$ ).

### 5. I3 CLOCK.

The I3CLOCK provides timing for the I register. The I3CLOCK clocks signals from the ROM to the I register, and is similar to the I2CLOCK with the exception that I3CLOCK is generated by  $KI\bar{3}$  instead of  $KI\bar{2}$ , ( $I3CLOCKA=KI\bar{3} \cdot BSELECT' \cdot RCE \cdot SC'$ ) ( $I3CLOCKB=KI\bar{3} \cdot BSELECT \cdot RCE \cdot SC'$ ).

6. R0 thru R6 CLOCKS.

The clocks for the holding registers are generated by KI1 during the last cycle of an instruction (RCE) if the instruction would not interfere with a store or fetch operation (SC') ( $Rn \text{ CLOCK} = KI2 \cdot RCE \cdot SC' \dots$ ). The specific clock is determined by the address specified in the LRN of the instruction for R1 thru R6. ( $R1 = \dots \cdot LRN0' \cdot LRN2'$ ;  $R2 = \dots \cdot LRN0' \cdot LRN1 \cdot LRN2'$ ; ...;  $R6 = \dots \cdot LRN0' \cdot LRN1 \cdot LRN2'$ ). R0 is clocked independently by LRO in the instruction which allow R0 to be loaded at the same time as one of the other registers ( $R0 \text{ CLOCK} = \dots \cdot LRO$ ). The holding registers must be clocked for a loading operation. A read operation uses the RRN field of the instruction.

7. MCLOCK3, QCLOCK, ZCLOCK.

The clock for the M, Q, or Z register are generated by KI2 during the last cycle of an instruction (RCE) if the instruction does not interfere with a store or fetch (SC') and a bit in the instruction specifies a load operation from the X-bus or the Y-bus for the M, Q, or Z register (LMX, LMY, LQX, LQY, LZQ, LZY respectively). The MCLOCK is generated by  $MCLOCK = KI2 \cdot RCE \cdot SC' \cdot (LMX + LMY)$ . The QCLOCK is generated by  $QCLOCK = KI2 \cdot RCE \cdot SC' \cdot (LZX + LZY)$ . The M, Q, and Z registers must be clocked for a loading operation.

8. K2 CLOCK.

The K2 clock provides timing for the special functions, parity checker, and memory interface. The

K2 clock is generated by the control logic for each machine cycle (K2=K2A from UK1 delayed).

9. K3 CLOCK.

The K3 clock provides timing for the memory interface. The K3 clock is generated by the control logic for each machine cycle (K3=K3' from UK1 delayed).

10. K2X CLOCK.

The K2X clock is used by the memory interface to generate MCLOCK1 and MCLOCK2. K2X is generated for each machine cycle (K2X=K2X' from UK1 delayed).

11. O Register Control.

The LOAD OREG signal is the clocking term for the O register. The O register clock is generated by the control logic if the instruction would not interfere with a store or fetch operation during state B, during state C, or during state A if VCY is not set ( $LOAD \ OREG = SC' \cdot (XXB \cdot XXC + VCY) \cdot UK1 \text{ delayed}$ ). The control logic generates the transfer gating signals TINCO, TBO, TOSO and TXO which load the O register. TINCO transfers the output of the incrementer into the O register if a branch condition specified in the MC field of the instruction is not satisfied ( $TINCO = BRANCH' + \dots$ ), if VCY is not set during state B ( $TINCO = \dots + XXB \cdot VCY' + \dots$ ), or during state C ( $TINCO = \dots + XXC$ ). TEO transfers the B field of the instruction word into the O register during state A if the MCONT field contains a 0 or 1 (MCONT0 is not set),

the branch condition is satisfied; VCY is not set in the instruction, and the instruction does not interfere with a store (TBO=XXB'·XXC'·MCONT'·BRANCH·SC'+...); or during state B when the branch condition is satisfied, MCONT is not set in the instruction, and VCY is set in the instruction (TBO=...+XXB·BRANCH·MCONT'·VCY). TOSO transfers the contents of the OS register to the O register during state A if the branch condition is satisfied, the MCONT field is equal to 2, VCY is not set, and the instruction would not interfere with a store or fetch operation (TOSO=XXB'·XXC'·BRANCH·MCONT'·MCONT1'·VCY'·SC'+...); or during state B if the MCONT field is equal to 2, the branch condition is satisfied, and VCY is set (TOSO = ...+XXB·MCONT'·MCONT1'·BRANCH·VCY). TXO will transfer the contents of the contents of the least significant ten bits of the X-bus to the O register during state A if the branch condition is satisfied, the MCONT field is equal to three, VCY is not set in the instruction, and the instruction would not interfere with a store or fetch operation (TXO=XXB'·XXC'·BRANCH·MCONT'·MCONT1·VCY'·SC+...); or during state B if the branch condition is satisfied, the MCONT field is equal to three, and VCY is set (TXO=...+XXB·BRANCH·MCONT'·MCONT1·VCY).

#### 12. OS Register Control.

The contents of the O register are clocked into the OS register by the LOAD OSREG clock. The clock is generated in the control logic by the K14 clock. LCLD

OSREG clocks the data to the OS register during state A if the branch condition specified in the MC field of the instruction is satisfied, the MCONT field is equal to one, VCY is not set, and the instruction would not interfere with a fetch or store (LOAD OSREG=XXB'·XXC'·BRANCH·MCONT'·MCONT1·VCY'·SC'+...); or during state B if the branch condition is satisfied, the MCONT field is equal to one, VCY is set, and the instruction would not interfere with a fetch or store (LOAD OSREG=...+XXB·BRANCH·MCONT'·MCONT1·VCY·SC').

#### 13. Branch Test.

BRTEST is generated by the control logic during the interval in which BRANCH is being checked. BRTEST is used by the branch conditions to reset attention latches. BRTEST is high during state A if the instruction would not interfere with a store or fetch operation, and VCY is not set in the instruction (BRTEST=XXB'·XXC'·SC'·VCY'+...); or during state B if VCY is set (BRTEST = XXB·VCY).

#### 14. Write Strobe.

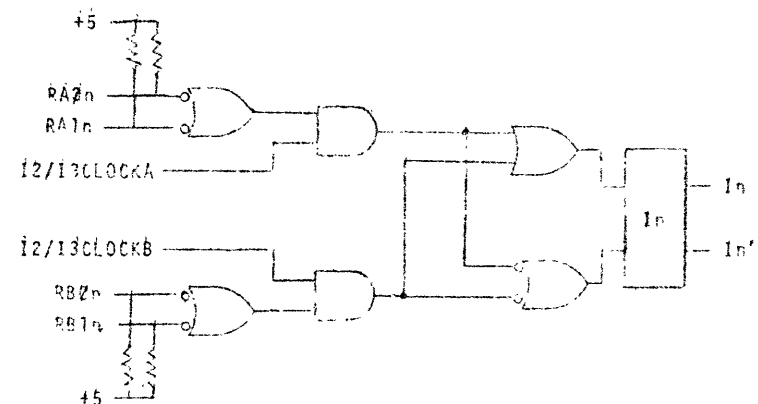
The write strobe for the scratchpad memory is generated by the control logic. The scratchpad write strobe (WS) is derived from the K3 during state A when the LSPX (load scratchpad from X-bus) bit is set in the instruction word (SWS=XXB'·XXC'·LSPX·K3; rWS=K12·state A').

15. Priority Latching:

PPL is generated by the control logic by KI2 clock to load the priority latches for central memory requests from the memory interface. PPL occurs during the last machine cycle of an instruction (RCE' if the special conditions field in the instruction contains an LPP (MS0 thru MS5-22B) and the memory interface is not in the process of doing a fetch or a store (PPL =  $KI2 \cdot RCE \cdot LPP \cdot SC'$ );

## B. I REGISTER CARDS. (See Figure 7.)

The I register is a 90-bit register which contains the instruction currently being executed by the micro-processor. The I register is loaded from the read-only memory (ROM) at the end of an instruction cycle by the I2CLOCK or I3CLOCK, depending on the timing requirements for the particular bit in the instruction (See Table A1 in Appendix A for timing a listing and functions of the individual instructions bits). The I register bit is set by  $RA0n$  or  $RA1n$  (where:  $n = 0$  thru 39.) and the corresponding I2/3CLOCKA ( $sIn = RA0n \cdot I2/3CLOCKA \cdot RA1n \cdot I2/3CLOCKA + \dots$ ); or by  $RB0n$  or  $RB1n$  and the corresponding I2/3CLOCKB ( $sIn = \dots + RB0n \cdot I2/3CLOCKB \cdot RB1n \cdot I2/3CLOCKB$ ). ( $RA0n$  is the bit from ROM card 0 thru 7 in the A bank;  $RA1n$  is the bit from ROM card 8 thru 15 in the A bank;  $RB0n$  is a bit from ROM card 0 thru 7 in the B bank;  $RB1n$  is a bit from ROM card 8 thru 15 in the B bank.



WHERE:  $n=0$  THRU 39  
AND  $RA0n, RA1n, RB0n, RB1n$ : SET=GROUND  
NOT SET=OPEN

Figure 7. I Register, Functional Block Diagram.

Only one of these signals is selected to be read out of the ROM during an instruction.) (See Paragraph 3C for a description of the ROM.) The I register bit is reset at the end of the instruction if the corresponding bit from the ROM is not set when clocked ( $rIn = RA/n \cdot RAln \cdot I2/3CLOCKA + RA/n \cdot RBl n \cdot I2/3CLOCKB$ ).

### C. READ-ONLY MEMORY.

The Read-Only Memory (ROM) is a diode memory containing the operating program for the microprocessor. The ROM is comprised of two groups of cards referred to as bank A and bank B. Each bank is capable of having up to 16 ROM cards. Each card contains sixty-four 20-bit words. The contents of the ROM address specified by 0 register, are loaded into the I register. Both banks of the ROM are addressed by the 0 register.

#### 1. BSELECT Flip-Flop.

The bank selected is determined by the BSELECT flip-flop. The BSELECT flip-flop is essentially the eleventh bit of the address, with the exception that is not incremented and must be set or reset by an instruction containing a SELECT B special condition or a SELECT A special condition. The BSELECT flip-flop is set if the special condition field specifies SELECT B, is cleared if the special condition field of the instruction specifies SELECT A, and is not changed if the special condition field does not specify a SELECT A or SELECT B.

The BSELECT flip-flop is set or reset at the same time that the 0 register is loaded from the B-field of the instruction. When the microprocessor is initialized, the BSELECT flip-flop is cleared.

#### 2. BSS Flip-Flop.

A bank select saved (BSS) flip-flop is used to store the status of BSELECT when an instruction is a subroutine call, and the contents of the 0 Register are transferred to the OS register. When the subroutine returns to the main program the BSELECT flip-flop is restored to the previous state by the BSS flip-flop. If the subroutine transfers the contents of the OS register to the Y-bus, the status of the BSS is also transferred to the Y-bus; and when the address is recalled, the BSELECT flip-flop is set from the X-bus.

#### 3. Address Decodings.

The contents of the 0 register are decoded by a logic matrix on the 0 and OS registers card. Bit 0 thru 3 of the 0 register generate the card selected signals, CS0 thru CS7 and CS10 thru CS17. Bits 4, 5, and 6 of the 0 register generate the Y-line selected signals, YS0 thru YS7. Bits 7, 8, and 9 of the 0 register generate the X-line selected signals, XS0 thru XS7. The word on the ROM card is selected by the X-and Y-lines. The particular card is selected by the CSn signals. The bank read into the I register



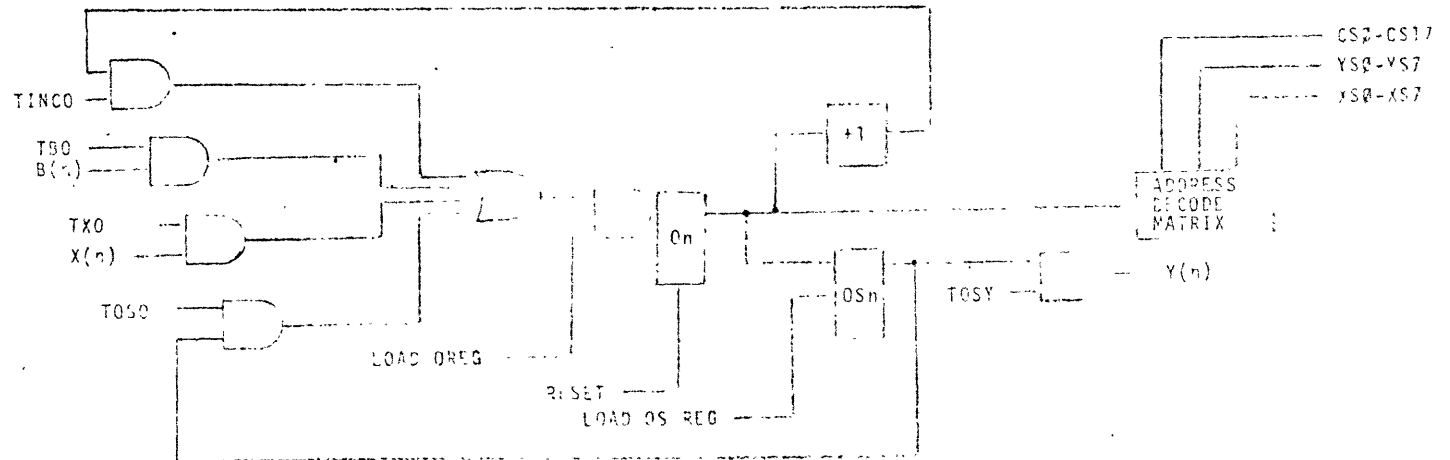
is determined by the clock signal gates by the status of the BSELECT flip-flop.

D. O & OS REGISTER CARD. (See Figure 8.)

The O and OS register card contains the address register (O register), the address storage register (OS register), and the address decoding matrix for the Read-Only Memory.

1. O Register.

The O register is a ten-bit register which contains the address of the next instruction to be executed. The O register is loaded from the location specified by the MCONT field of the instruction if the branch condition specified by the MC field of the instruction is satisfied. If the branch condition is not satisfied, the contents of the O register are incremented by one. The MCONT field is decoded by the control logic to provide the gating for the O register set terms. If a branch is satisfied, the O register is loaded by the LOAD OREG clock and the contents of the B field of the instruction if the MCONT field equals 0 or 1 (sOn = LOAD OREG·Bn TH0+...) or by the LOAD OREG clock and the contents of the OS register if the MCONT field of the instruction equals 2 (sOn ...LOAD OREG·OSH·TOSC...), or by the LOAD OREG clock and the least significant 10 bits in the X-bus if the MCONT field of the instruction equals



WHERE B, O, & OS: n=0 THRU 9  
 AND X & Y: n=14 THRU 23

Figure 8. O & OS Register, Functional Block Diagram.

3 (SON=...+LOAD OSREG·Xn·TINCO). (See paragraph 3A11. - O Register Control-for a detailed description of the control terms and clock from the control logic card.) The contents of the O register are distributed to the decode matrix to generate the addressing terms for the ROM, to the incrementer where they are incremented by one if a branch does not succeed, and to the OS register. (See Paragraph 3C3 for a description of the decode matrix.)

### 2. O Register Incrementer.

The O register is incremented by complementing the least significant bit; and generating a carry input to the next bit, if the least significant bit is high prior to being complemented. The remaining bits,  $\phi_7$  thru  $\phi_8$ , are set by conditional sum half adders. The half adder provides a sum output used to set the associated bit, and a carry to the next bit. The inputs to the half adder are the carry from the previous bit, and the associated bit. To increase the speed of the incrementing cycle, the O register is divided into three groups of bits with an carry from the previous group.

### 3. The OS Register.

The OS register is a ten-bit register used to store the contents of the O register during a branch. If the MCONT field of the instruction is a 1, and the branch condition specified by the MC field of the instruction is satisfied, the OS register is loaded with the contents of the O register by the LOAD OSREG clock

from the control logic (sCSn=On·LOAD OSREG). The contents of the OS register are transferred to the least significant ten bits of the Y-bus, if TOSY is set in the instruction (Y=TOSY·OS). (See Paragraph 3A12. - OS Register Control-for a description of the LOAD OSREG clock.)

### NOTE

OS register is implemented with flip-flops triggered by the leading edge of the clock.

### E. BRANCH CONDITIONS CARD.

The branch conditions card decodes the contents of the MC field in the instruction, and examines the data associated with the decoded branch condition. If the branch condition is satisfied, a BRANCH signal is generated. (See Table A2 in Appendix A for a listing of the branch condition codes and functions.)

#### 1. Branches.

The MC-field is decoded in two groups. The first group is comprised of bits  $\phi_0$ , 1, and 2 in the MC-field, and is decoded to generate  $S\phi$  thru  $S7$ . The second group is comprised of bits 3, 4, and 5 in the MC-field, and is decoded to generate  $T\phi$  thru  $T7$ . There are 64 possible combinations of branch conditions. Only 36 of these combinations are used conditions. The rest of the combinations are not decoded; and, therefore, will never branch. Every decoded branch is the logical product of the branch condition specified and the decoded MC-field for the condition. The BRANCH signal is the

logical sum of all the decoded branches. Table 5 is a list of the logic equations for each branch condition and a description of the terms.

Table 5 . BRANCH Logic

BRANCH=	DESCRIPTION
S $\beta$ ·T1	MC= $\beta$ 1 (will always branch)
+S $\beta$ ·T2·(X(0)'·X(1)'·...·X(23)')	(MS= $\beta$ 2)·(X= $\beta$ )
+S $\beta$ ·T3·(X(0)+X(1)+...+X(23))	(MC= $\beta$ 3)·(X $\neq$ $\beta$ )
+S $\beta$ ·T4·X( $\beta$ )	(MC= $\beta$ 4)·(X $\leq$ $\beta$ )
+S $\beta$ ·T5·X( $\beta$ )'	(MC= $\beta$ 5)·(X $\geq$ $\beta$ )
+S $\beta$ ·T6·X $\beta$ '·(X( $\beta$ )+X(1)+...+X(23))	(MC= $\beta$ 6)·(X $\geq$ $\beta$ )
+S $\beta$ ·T7·Y( $\beta$ )'	(MC= $\beta$ 7)·(Y $\geq$ $\beta$ )
+S1·T4·(X(6)·X(7)·...·X(23))	(MS=14)·(X(6)-X(23)= 777777B)
+S1·T5·(X(6)'+X(7)'+...+X(23)')	(MC=15)·(X(6)+X(23) $\neq$ 777777B)
+S1·T6·Z( $\beta$ )'	(MS=16)·(Z $\geq$ $\beta$ )
+S1·T7·Z( $\beta$ )	(MC=17)·(Z $\leq$ $\beta$ )
+S2·T $\beta$	MC=2 $\beta$ (will always branch)
+S2·T1·(Y(21)+Y(22)+Y(23))	(MC=21)·(Y $\neq$ 7/ $\beta$ )
+S2·T2·(BL0'·BL1'·...·BL23')	(MC=22)·(BL= $\beta$ )
+S2·T3·(BI $\beta$ ·BL1+...+BL23)	(MS=23)·(BL $\neq$ $\beta$ )
+S2·T4·Y(23)'	(MC=24)·(Y(23)= $\beta$ )
+S2·T5·Y(23)	(MC=25)·(Y(23) $\neq$ $\beta$ )
+S2·T6·ATLAT1	(MC=26)·(Attention latch 1= $\beta$ )

Table 5 . BRANCH Logic. (cont'd)

BRANCH=	DESCRIPTION
+S2·T7·RS1'·RS2'	(MC=27)·(RSLAT1= $\beta$ )·RSLAT2= $\beta$ )
+S3·T $\beta$ ·PNEX	(MC=3 $\beta$ )·(Protect,X)
+S3·T1·RS2'	(MC=31)·(RSLAT2= $\beta$ )
+S3·T2·SPAG'	(MC=32)·(Special flag A= $\beta$ )
+S3·T3·SPAG	(MC=33)·(Special flag A= $\beta$ )
+S3·T4·ATLAT2'	(MC=34)·(Attention latch2= $\beta$ )'
+S3·T5·ATLAT3'	(MC=35)·(Attention latch3= $\beta$ )'
+S3·T6·ATLAT1	(MC=36)·(Attention latch1/ $\beta$ )'
+S4·T $\beta$ ·SB1	(MC=40)·(Undefined)
+S4·T1·SB2	(MC=41)·(Undefined)
+S4·T2·LMPE	(MC=42)·(Local memory parity error=1)
+S4·T3·SB3	(MC=43)·(Undefined)
+S4·T4·CMPE	(MC=44)·(Central memory parity error=1)
+S4·T5·BP	(MC=45)·(Breakpoint $\neq$ $\beta$ )

## 2. Attention Latches.

The branch conditions card contains three latching flip-flops, attention latch 1 (ATLAT1), attention latch 2 (ATLAT2), and attention latch 3 (ATLAT3). The attention latches are set by the corresponding SATLA1', SATLA2', or SATLA3' signal from the I/O interface card and the K2

clock from the control logic ( $sATLAT1 = SATLA1 \cdot K2$ ; ...;  $sATLAT2 = ATLAT2 \cdot K2$ ;  $sATLAT3 = SATLA3 \cdot K2$ ): Attention latch 1 is also set by the ADVANCE switch on the local control panel. (This switch is used exclusively by the diagnostic microprocessor.) The ADVANCE switch generates two signals, ADVNO and ADVNC; where  $ADVNO = ADNC'$ . In the quiescent state; ADVNO is high, and ADVNC is low. Two flip-flops ADV1 and ADV2, are associated with the input from the switch, and are used to generate a pulse each time that the switch is pressed. ADV1 and ADV2 are initially set ( $sADV = ADVNC \cdot I2CLOCK$ ;  $sADV2 = ADV1 \cdot I2CLOCK$ ). When the ADVANCE switch is pressed, ADV1 is reset ( $rADV1 = ADVNC \cdot I2CLOCK$ ). One instruction cycle later, attention latch 1 is set ( $sATLAT1 = ADV1' \cdot ADV2 \cdot ADVNO \cdot I2CLOCK$ ), and ADV2 is reset ( $rADV2 = ADV1' \cdot I2CLOCK$ ). ADV1 remains reset until the switch is released. ( $sADV1 = ADVNC \cdot I2CLOCK$ ), and ADV2 remains reset for one instruction cycle longer ( $sADV2 = ADV1 \cdot I2CLOCK$ ). The attention latches are reset by branch condition in the MC field when the condition of the respective latches is being tested, and the latching signals are not in the process of setting or holding the attention latches ( $rATLAT1 = ADV1' \cdot SATLA1 \cdot I2CLOCK \cdot (S2 \cdot T6) + (S3 \cdot T6) + ADV2' \cdot SATLA1 \cdot I2CLOCK \cdot [(S2 \cdot T6) + (S3 \cdot T5)] + ADVNO \cdot SATLA \cdot I2CLOCK \cdot [(S2 \cdot T6) + (S3 \cdot T6)]$ ;  $rATLAT2 = SATLA2 \cdot S3 \cdot T4 \cdot K2$ ;  $rATLAT3 = SATLA3 \cdot S3 \cdot T5 \cdot K2$ ).

## F: SPECIAL FUNCTIONS CARD.

The special functions card decodes the contents of the MS field in the instruction, and generates the control signals required for the function. (See Table A3 in Appendix A for a listing of the special function.) The MS field is decoded in two groups. The first group is comprised of bits 0, 1, and 2 in the MS field, and is decoded to generate  $U0$  thru  $U7$ . The second group is comprised of bits 3, 4, and 5 in the MS field, and is decoded to generate  $V0$  thru  $V7$ . The logical product of thirty eight of the sixty four possible combinations of  $U0$  thru  $U7$ , and  $V0$  thru  $V7$  are used to generate the control terms. The remaining combinations are available for future expansion. Table 6. is a list of the logical equations for each special function and a description of the terms.

Table 6 . Special Functions Logic

Function	Equation	Definition of Terms	
Left cycle from the data from the left bool box to the X-bus... (See Paragraph (3A2 .))	...1 place.	$LCY1B = U0 \cdot V1$	$MS = 01$
	...2 places.	$LCY2B = U0 \cdot V2$	$MS = 02$
	...3 places.	$LCY3B = U0 \cdot V3$	$MS = 03$
	...4 places.	$LCY4B = U0 \cdot V4$	$MS = 04$
	...8 places.	$LCY8B = U0 \cdot V5$	$MS = 05$
	...12 places.	$LCY12B = U0 \cdot V6$	$MS = 06$
...16 places.	$LCY16B = U0 \cdot V7$	$MS = 07$	

Table 6 . Special Functions Logic (cont'd)

Function	Equation	Definition of Terms
Left cycle the data from the left pool box to the X-bus (See Paragraph 3A2 .)	...20 places. LCY20B = U1·V0	MS=10
	...as specified by Z22 and Z23 CCFZA = U1·V1	MS=11
	...as specified by Z19, Z20; and Z21 CCFZB = U1·V2	MS=12
scratchpad address from Z (See Paragraph 3J .)	SPFZ = U1·V3	MS=13
	sALERT = U1·V4 I2CLOCK	MS=14
	rALERT = U1·K2 + V4'·K2	MS=14
Set Parallel Output (See Paragraph 3F1.)	sPOT = U1·V5 I2CLOCK	MS=15
	rPOT = U1'·K2 + V5'·K2	MS=15
Set Parallel Input (See Paragraph 3F1.)	sPIN = I2CLOCK·U1·V6	MS=16
Reset Parallel Input (See Paragraph 3F1.)	rPIN = K2·U1' + K2·V6'	MS=16
Request Strobe 1	RSOUT 1 = U1·V7·BRTEST	(MS=17) (interval during which the branch condition is tested.)

Table 6 . Special Functions Logic (cont'd)

Function	Equation	Definition of Terms
Unprotect as specified by X-bus mask (See Paragraph 3F2.)	UPXn = U2·V0·Xn (n = 20, 21, 22, or 23)	(MS=20)
Clock memory interface to load priority latches for central memory requests	LPF = U2·V2	MS=22
Reset request strobe latch 1	rRS1 = I2CLOCK·U2·V3	MS=23
Reset Central Memory request	MI RESET = U2·V4 + MR0 + MR1	MS=24
Set protect as specified X-bus mask (See Paragraph 3F2.)	PXn = Us·V5·Xn (n = 20, 21, 22, or 23)	(MS=25)
Reset used by TSU	RESETTU = U2·V6	MS=26
Set special flag	sSPAG = U3·V0·I2CLOCK	MS=30
Reset special flag	rSPAG = U3·V1·I2CLOCK	MS=31
Reset request strobe latch 2	rRS2 = I2CLOCK·U3·V2	MS=32
Request strobe 2	RSOUT2 = U3·V3·BRTEST	(MS=33) (interval in which the branch condition is tested)
Undefined	SMS 1 = U3·V4	MS=34
Release Prestore Store & Hold Fetch Fetch & Hold Prefetch	Special functions 40 thru 47 are control terms for data transfer between the Microprocessor and central or local memory (see Paragraph 3M.)	MS=40 MS=41 MS=42 MS=43 MS=44 MS=45 MS=47

Table 6 : Special Functions Logic (cont'd)

Function	Equation	Definition of Terms
Select ROM bank A (See Paragraph 3L1.)	$rBSELECT = K2 \cdot U6 \cdot V0 \cdot TBO$ + $Ks \cdot BSS \cdot TOSO$ + $K2 \cdot X13 \cdot TXO$	MS=60
Select ROM bank B (See Paragraph 3L1.)	$sBSELECT = K2 \cdot U6 \cdot V1 \cdot TBO$ + $K2 \cdot BSS \cdot TOSO$ + $K2 \cdot X13 \cdot TXO$	MS=61) (Contents of B field transferred to D Register
Clear CPU maps	$CMAP = U6 \cdot V2$	MS=62
Oddword Fetch Oddword Fetch & Hold	Control terms used by memory interface (see Paragraph 3M.)	MS=64
		MS=65

### 1. POT/PIN.

The POT/PIN System allows the microprocessor to communicate with external devices. (See Paragraph 3N for a detailed description of the I/O functions). When the microprocessor wishes to transfer data to an external register (POT), it puts the address of the external register to be loaded (hardware defined) into the Z Register and sends an ALERT strobe to all external devices. The external device takes the address from the Z-bus to the specified register. The microprocessor will then load the Z-bus with the data to be sent, and send a POT strobe. The external device will use this POT strobe to load the selected register from the Z-bus. When an external device wishes to send data to a microprocessor (PIN), the

devices send an attention signal, SATLAN, to the microprocessor. This signal is latched in the microprocessor and is tested by a branch condition ATLAN=1. The microprocessor will, via programmed instructions, read a register in the external device by sending an alert to the device. The device will set up a path between the selected register and the E1-bus or E2-bus. The microprocessor will then transfer the E1-bus or E2-bus to the Y-bus; and send a PIN strobe, via programmed instruction, signifying that it has read the data.

### 2. Protect System.

The protect mechanism allows a microprocessor to have undisturbed use of certain facilities of the system. There are four identical protect units, each protecting a specific item. When a microprocessor wishes to protect, the microprocessor will gate a bit mask onto the four low order bits of the X-bus. This mask will have a one in the position corresponding to the protect to be selected. Simultaneously, the microprocessor will set a special function, request protect. The instruction which requests the protect must have VCY set. The protect system examines the X-bus and the protect request lines from all microprocessors during each cycle and resolves conflicts between units on a positional priority basis. The protect mechanism sets a latch peculiar to the type of protect, and to the microprocessor initiating the request. The protect system has a comparator for each

microprocessor which compares the low order four bits of the X-bus with the four protect latches peculiar to that microprocessor, and returns a one if they are not equal. A branch condition, Branch on X-bus not equal to protect field, is provided to allow execution of a loop until a request is satisfied. When a microprocessor wishes to unprotect, it gates a 4-bit mask to the X-bus least significant bits, and sends an unprotect signal to the protect system. The X bus mask specifies which of the four types of protect are to be released. Unprotect takes one cycle and need not be tested, as it will always succeed. A protect remains in effect until the microprocessor, which initiated it, releases it.

### 3. Request Strobe.

Each microprocessor has two latches that are set by the other microprocessors in the system. These latches may be tested by a branch condition. A unit can also selectively set the latches in all the other microprocessors by gating the contents of the X-bus to the request strobe lines. The X-bus will contain an 8-bit mask which will determine which of the other microprocessors are to be strobed. It is legal for a microprocessor to strobe itself. (See Paragraph 3L for a description of the request strobe.)

### C. MQZ REGISTERS CARD.

The MQZ registers card is comprised of three 24-bit

registers (M register, Q register, Z register), and two boolean function circuits (left bool box, right bool box) associated with the registers. The M, Q, and Z registers are loaded from X-bus or the Y-bus. The M register is also loaded from the local or central memory via the M2-bus or the M1-bus, respectively. The output of the right bool box is the logical combination, as specified by the BR field, of the contents of the Z and Q registers (See Table A4 in Appendix A for a listing of the boolean functions). The output of the left bool box is the logical combination, as specified by the BL field, of the contents of the M and Q registers (See Table A4 in Appendix A for a listing of the boolean functions). The outputs of both the bool boxes go to the adder/cycler.

#### 1. M Register.

The M register is loaded from the X-bus or the Y-bus by LMX or LMY, respectively, in the instruction word and the M register clock ( $sMn=Xn \cdot LMX \cdot MCLOCK3 + Yn \cdot LMY \cdot MCLOCK3 + \dots$ ). (See Paragraph 3A7 for a description of the MCLOCK3.) The M register is also loaded from the central memory or the local memory by LMM1 and LMM2, respectively, from the memory interface and MCLOCK1 or MCLOCK2, respectively, which are logically summed with the MCLOCK3 from the control logic to generate MCLOCK ( $MCLOCK=MCLOCK1 \cdot MCLOCK2 \cdot MCLOCK3$ ); data from



central memory is transferred via the M1-bus, and data from local memory via the M2-bus ( $sMn = \dots + M1n \cdot LMM1 \cdot MCLOCK + M2n \cdot LMM2 \cdot MCLOCK$ ). (See Paragraph 3M for a description of LMM1 and LMM2.) (See Paragraph 3M for a description of MCLOCK1 and MCLOCK2.) The data loaded in the M register goes to the left bool box and the M-bus. The data on the M-bus is gated to the memory by signals from the memory interface. (See Paragraph 3A7 for a description of the memory interface, and the transfer of data from the M-bus to memory.)

#### 2. Q Register.

The Q register is loaded from the X-bus or the Y-bus by LQX or LQY, respectively, in the instruction word and the Q register clock ( $sQn = Xn \cdot LQX \cdot QCLOCK + Yn \cdot LQY \cdot QCLOCK$ ). (See Paragraph 3A7 for a description of QCLOCK). The data in the Q register goes to the left bool box and the right bool box.

#### 3. Z Register.

The Z register is loaded from the X-bus or the Y-bus by LZX or LZY, respectively, in the instruction word and the Z register clock ( $sZn = Xn \cdot LZX \cdot ZCLOCK + Yn \cdot LZY \cdot ZCLOCK$ ). (See Paragraph 3A7 for a description of ZCLOCK.) The data in the Z register goes to the right bool box and to the Z-bus.

#### 4. Bool Boxes.

The left bool box and the right bool box are

logic matrixes that provide one of 16 possible logical combinations of two variables. The output corresponds to the boolean functions specified by the BL field and the BR field of the instruction. The left bool box combines the M register and Q register outputs under control of BL0 thru BL3 providing the logical combination of the logical products of M, Q, M', and Q' ( $BL0 \cdot M \cdot Q + BL1 \cdot M \cdot Q' + BL2 \cdot M' \cdot Q + BL3 \cdot M' \cdot Q'$ ). Note that this is inverted.) The resultant output of the left bool box is the logical sum of the products to the adder/cycler where it is inverted to provide one of the 16 functions specified by the count in the BL field. (Table 7 is a list of the logic equations for the bool boxes and a description of their functions.) The output of the left bool box goes to the adder and to the cycler of the adder/cycler card.

Table 7. Left (Right) Bool Box Logic.

Where:  $n=0$  thru 23;

And for the left bool box:  $a=M, b=Q$

And for the right bool box:  $a=R, b=Z$

Ba=	DEFINITION
$Ba0 \cdot Ba1' \cdot Ba2' \cdot Ba3' \cdot bn \cdot Qn$	$(Ba=0) \cdot (b=Q)=1$
$+Ba0', Ba1', Ba2', Ba3' \cdot (bn \cdot Qn + bn' \cdot Qn')$	$(Ba=1) \cdot (b=Q)=1$

Table 7 , Left (Right) Bool Box Logic (cont'd)

Ba=	DEFINITION
$+Ba\bar{b}' \cdot Ba1' \cdot Ba2 \cdot Ba3' \cdot Qn$	$(Ba=\bar{b}2) \cdot Q = 1$
$+Ba\bar{b}' \cdot Ba1' \cdot Ba2 \cdot Ba3 \cdot (bn' + Qn)$	$(Ba=\bar{b}3) \cdot (b' + Q) = 1$
$+Ba\bar{b}' \cdot Ba1 \cdot Ba2' \cdot Ba3' \cdot bn$	$(Ba=\bar{b}4) = 1$
$+Ba\bar{b}' \cdot Ba1 \cdot Ba2' \cdot Ba3 \cdot (bn + Qn')$	$(Ba=\bar{b}5) \cdot (b + Q') = 1$
$+Ba\bar{b}' \cdot Ba1 \cdot Ba2 \cdot Ba3' \cdot (bn + Qn)$	$(Ba=\bar{b}6) \cdot (b + Q) = 1$
$+Ba\bar{b}' \cdot Ba1 \cdot Ba2 \cdot Ba3 \cdot (bn + bn' + Qn + Qn')$	$(Ba=\bar{b}7) = 1$
$+Ba\bar{b}' \cdot Ba1' \cdot Ba2' \cdot Ba3' \cdot (bn \cdot bn' \cdot Qn \cdot Qn')$	$(Ba=1\bar{b}) = \bar{b}$
$+Ba\bar{b}' \cdot Ba1' \cdot Ba2' \cdot Ba3' \cdot bn' \cdot Qn'$	$(Ba=11) \cdot (b' \cdot Q') = 1$
$+Ba\bar{b}' \cdot Ba1' \cdot Ba2 \cdot Ba3' \cdot (bn' \cdot Qn)$	$(Ba=12) \cdot (b' \cdot Q) = 1$
$+Ba\bar{b}' \cdot Ba1' \cdot Ba2 \cdot Ba3 \cdot bn'$	$(Ba=13) \cdot b' = 1$
$+Ba\bar{b}' \cdot Ba1 \cdot Ba2' \cdot Ba3 \cdot bn \cdot Qn'$	$(Ba=14) \cdot b \cdot Q' = 1$
$+Ba\bar{b}' \cdot Ba1 \cdot Ba2' \cdot Ba3 \cdot Qn'$	$(Ba=15) \cdot Q' = 1$
$+Ba\bar{b}' \cdot Ba1 \cdot Ba2 \cdot Ba3 \cdot (bn \cdot Qn' + bn' \cdot Qn)$	$(Ba=16) \cdot (bn \text{ (EOR) } Q) = 1$
$+Ba\bar{b}' \cdot Ba1 \cdot Ba2 \cdot Ba3 \cdot (bn' + Qn')$	$(Ba=17) \cdot (b' + Q') = 1$

## H. ADDER/CYCLER CARD.

The adder/cycler is comprised of three physically identical cards. The input data to the adder/cycler is from the left bool box (BLn) and the right bool box (BRn). The adder uses both inputs, and the cycler uses only the left bool box data.

1. Adder.

The adder is comprised of six 4-bit sections with carry acceleration over each section. A consideration in the adder is the time required to do the add, the delay around the loop from the M, Q, and Z registers, through the adder, onto the X-bus, and back into the M, Q, and Z registers, which takes more than one clock period. Data must be stable during the clock (K2) which loads M, Q, or Z registers, therefore VCY must be set during adds.

a. Carry Portion. The carry portion of a single stage of the adder does one of three things with respect to carries; generates a carry-out independent of the carry-in, absorbs a carry-in, or propagates a carry-in across itself. A carry-out is generated, independent of the carry-in when;  $BLn + BRn = 1$ . A carry-in is absorbed when;  $BLn + BRn = \bar{b}$ . A carry-in is propagated when;  $BLn$  (EOR)  $BRn = 1$ , and the input carry  $(Cn-1) = 1$ . The adder is divided into six 4-bit sections to provide carry acceleration over each 4-bit section. If a given stage either generates or absorbs a carry, the output carry state from that stage is fully determined independent of the carry-in for that stage. Within a 4-bit section having no anticipation, the output carry from the high order bit is fully determined 4 logic levels after the input signals for

all the bits have stabilized. The longest time required for a carry to arrive at a given stage is 4 logic levels from the time the carry arrives at the low order bit, assuming that the low order 3 bits propagate the carry, and the 4th bit absorbs the carry: if all four bits of a group are in the correct state to propagate a carry- in ( $BL_n + BR_n = 1$ ), or not to propagate a carry ( $BL_n' + BR_n' = 1$ ), then the carry-out from the 4-bit section can be determined in two levels rather than 4. The PK signal indicates that a carry is not generated by the section, is not propagated across the section. The PK' signal is used as one of the carry inputs to the next section, and is also used to force the no carry output of the high order bit of the group to 1. In a similar fashion, the PC signal indicates that a carry is input to the section, and is propagated across all 4 bits. PC' is sent to the next section as part of the no-carry input term, and also forces the carry output of the high order bit of the group to 1.

b. Anticipated Carry. The anticipatory circuitry guarantees that a carry out from a group will be stabilized after the  $BL_n + BR_n$  and  $BL_n' + BR_n'$  signals for the group have stabilized, or after the carry signals from the next less significant group are stable at the input of the group; whichever is greater. The total time required after the arrival of  $BL_n'$  and  $BR_n'$  at the card input to obtain

a stable carry out from bit 0 is approximately 16 levels. From the time the carries have stabilized, additional time is required to stabilize the sum on the X-bus.

c. FLUSH. FLUSH is a signal derived from the branch conditions card. FLUSH inhibits the carry circuit if an add is not required and a zero-shift of data thru the adder is required. More than one cycle is required to do an add. If VCY is not set in the instruction and the branch condition is not satisfied, the instruction will use only state A. FLUSH' inhibits the adder carry if VCY is not set, or if the branch conditions field of the instruction specifies an always branch (call). ( $FLUSH' = VCY' + S_1 T_1 + S_2 T_2'$ ).

## 2. Cycler.

The cycler does left cycles only. The 8 inputs to the gates for each bit are connected to the left bool box signals for the bits that correspond to 1, 2, 3, 4, 8, 12, 16 and 20 bits to the right of the bit, and are gated by the cycle count, LCY1A thru LCY20A or LCY1B thru LCY20B. The cycle count is taken from the special function field, LCY1B thru LCY20B, generated on the special function card or from the Z register, LCY1A thru LCY20A are generated on the control logic card.

## I. PARITY CHECKER.

The parity checker generates the parity bits for

data contained in the M register. The contents of the M register are examined as two fields, M0 thru M11 and M12 thru M23. Each field is decoded by identical logic matrices. If the field contains an even number of high bits, then the parity bit is high (LMPAROUT=M0 thru M11 contain an even number of high bits; LMPAROUT=M12 thru M23 contain an even number of high bits). During a store operation, the parity bits, LMPAROUT and LMPAROUT, are sent to the local memory with the data. The parity bits to central memory, PAROUT and PAROUT, are the local memory parity bits AND-gated with GDM (GDM is always high) which is generated by the memory interface during a store (PARnOUT=GDM·LMPARnOUT; where; n=0 for bits M0 thru M11 parity; and n=1 for bits M12 thru M23 parity). Data resides in the Microprocessor M register for at least 1 cycle before being gated to central memory, or being strobed into the private memory data register, allowing the parity generator to become stable before the output is used.

#### 1. Parity Register.

During a fetch operation the two parity bits from memory (LMPARIN and LMPARIN, if the fetch is from local memory; or PARIN and PARIN, if the fetch is from central memory) are stored in a parity register at the same time that the M register is loaded with the data.

$$sLMPARn = LMPARn \cdot (MCLOCK1 + MCLOCK2);$$

$$sLMPAR1 = LMPAR1 \cdot (MCLOCK1 + MCLOCK2);$$

$$sPARn = PARn \cdot (MCLOCK1 + MCLOCK2);$$

$sPAR1 = PAR1 \cdot (MCLOCK1 + MCLOCK2)$ ). The contents of the M register are examined for even parity and compared with the contents of the corresponding bits in the parity register. If the parity bit of the corresponding M register field is not the same as that stored in the parity register, the parity error flip-flop (LMPE0, LMPE1, CMPE0 or CMPE1) associated with that parity bit is set one instruction later.  $LMPARN' \dots + LMPARNOUT'$ ,  $LMPARN \cdot SLMP \cdot K2$ ;  $sCMPEn = LMPARNOUT \cdot PARN' \dots + LMPARNOUT'$ ,  $PARN \cdot SCMP \cdot K2$ ). SLMP is set if the local memory is clocked ( $sSLMP = MCLOCK2 \cdot K2$ ), and reset at the end of the next instruction ( $rSLMP = K2$ ). SCMP is set if the central memory is clocked ( $sSCMP = MCLOCK1 \cdot K2$ ), and reset at the end of the next instruction ( $rSCMP = K2$ ). The status of the parity error flip-flops (latches) is tested by two branch conditions; local memory parity error (LMPE=LMPE0+LMPE1), and central memory parity error (CMPE=CMPE0+CMPE1).

#### NOTE

Care must be taken when using the branch conditions on the parity error latches, since the latches are not set on an error until 1 cycle after the memory has stopped. An instruction sequence;

FETCH

50010

GO TO ZILCH IF PARITY ERROR,

checks the parity of transfers which occurred prior to the fetch, since the memory is still running when the test is done. To check the parity of an isolated reference, the following works:

FETCH

$R0 \rightarrow R0$  (or some other instruction which will stop execution until the memory stops.)

NOP (the parity error latch will be set at the end of this instruction.)

GO TO ZILCH IF BAD PARITY.

The parity error latches are reset at the end of every branch instruction if the branch condition is not a memory parity error check ( $RLMPE=RLMP=K2 \cdot BRTEST \cdot TLMP$ ;  $RCMPEn=RCMP=K2 \cdot BRTEST \cdot TCMP$ ). The outputs of the parity error latches are also routed to the system warning register.

## 2. Addressing Parity.

The parity checker also generates an odd parity bit

50010

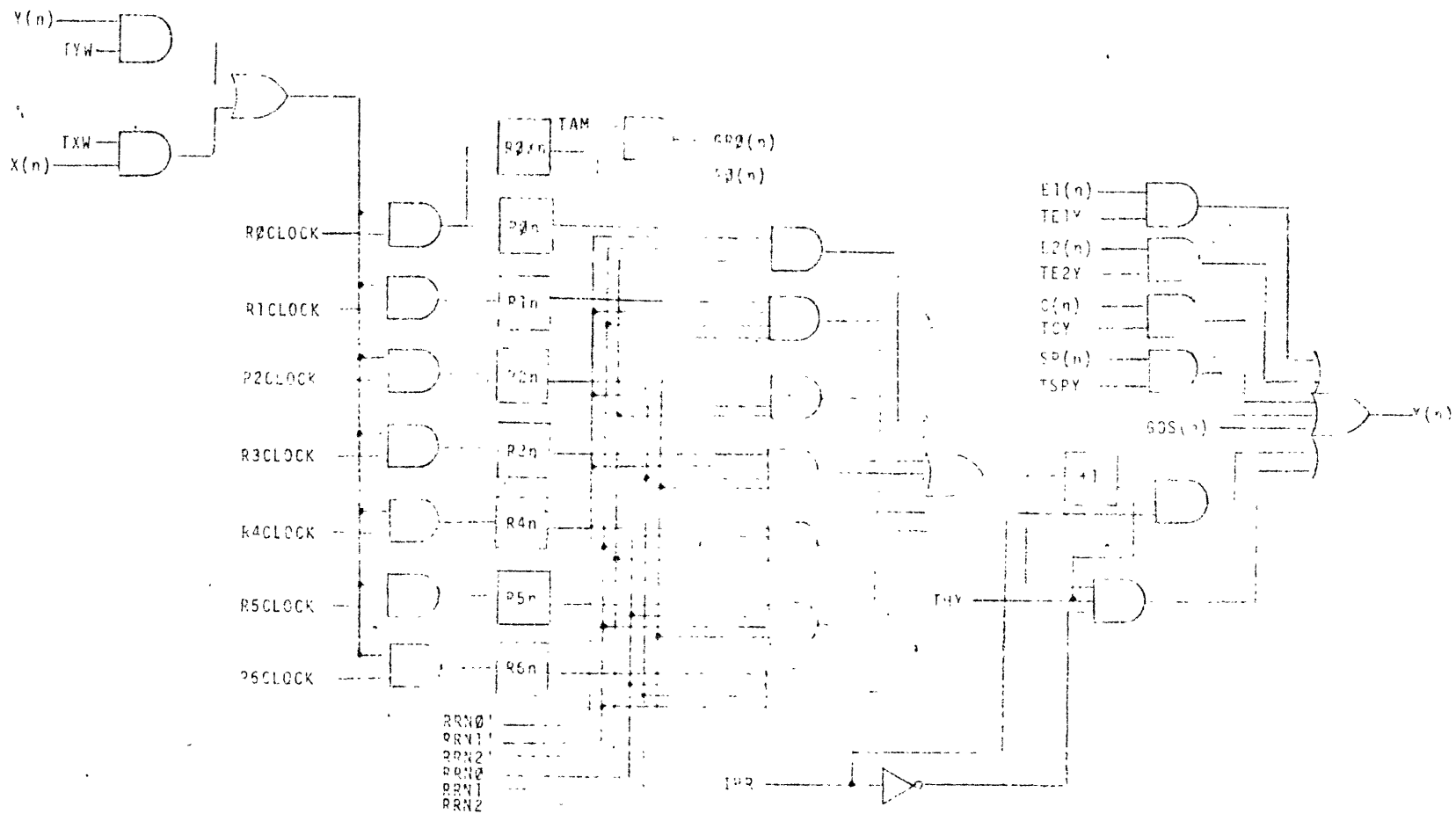
(APAR) for 17 bits of the  $R0$  register,  $R0(6)$  thru  $R0(22)$  inclusive, which comprise the memory address.  $R0(6)$  thru  $R0(22)$  are decoded by a logic matrix so that, if an odd number of bits is high, then APAR will be high. In most cases, the address is in  $R0$  for 1 cycle before being sent to the memory (while the microprocessor is making a request to the MPMBM); in the case of the CPU, this is not necessarily the case. Address parity (APAR) will be stable 60 nanoseconds after the data in  $R0$  is stable. This implies that in the worse case, address parity will arrive at the memory 60 nanoseconds after the address and the request field arrive.

## J. SCRATCHPAD.

The scratch pad is a non-destructive read-out (ndro) memory comprised of sixty-four 24-bit registers. The scratch pad address is decoded from the SSP field of the instruction; or the least significant 6 bits of the Z register, if specified by the special functions (SPFZ) field. The scratch pad card contains 64 words. The data is loaded into the selected scratch pad address from the X-bus by a write strobe (WS) from the control logic ( $SPn=Xn \cdot WS$ ). The write strobe is derived from LSPX in the instruction word during state A ( $SWS=XXB \cdot XXC \cdot LSPX \cdot K3$ ;  $rWS=R12 \cdot STATEA$ ). The data is read from the selected scratch pad address to the Y-bus by LSPY in the instruction word ( $Yn=SPn \cdot LSPY$ ).

K. HOLDING REGISTERS. (See Figure 9.)

The holding register cards contain seven 24-bit registers; R0 thru R6; and the gating to the Y-bus. The holding registers are loaded by their respective clocks; R0CLOCK thru R6CLOCK; and the contents of the X-bus if TXW is set in the instruction word; or Y-bus if TYW is set in the instruction word ( $sR0n = R0CLOCK \cdot Xn \cdot TXW + R0CLOCK \cdot Yn \cdot TYW$ ;  $sR1n$  thru  $sR6n$ , same as  $sR0n$  except that the R0CLOCK is replaced by the R1CLOCK thru the R6CLOCK respectively). The R0CLOCK thru R6CLOCK are derived from the K12 clock, LRO and LRN field of the instruction, in the control logic during an end cycle of an instruction if the instruction would interfere with a store or fetch operation. The specific register clock generated is determined by the contents of the LRN field in the instruction with the exception of the R0CLOCK which is loaded independent of the LRN field by LR0 in the instruction word. (See paragraph 3A6 for a description of the holding register clocks.) The holding registers are read-out to the Y-bus individually as specified by the contents of the RRN field of the instruction if THY is set in the instruction. ( $Yn = \dots + R0n \cdot RRN0' \cdot RRN1' \cdot RRN2' \cdot THY + \dots$ ; for R1 thru R6, RRN0 thru RRN2 are set to correspond to the register number). The register read-out without changing the contents of the register, however, the data transferred to the Y-bus is incremented if INR is set in the instruction word.



HOLDING REGISTERS, FUNCTIONAL BLOCK DIAGRAM  
 WHEEL: n=2 T4PU 23

Figure 9. Holding Registers, Functional Block Diagram.

1. R0X.

The R0X register is an additional register, provided on the holding register card that is loaded in parallel to the R0 register and provides access to the data contained in R0 by the memory interface via the R0-bus without the necessity of addressing the R0 register; the data in R0 provides the memory address for data transferred between the microprocessor and local or central memory. The data from R0X is sent to the memory interface independent of the addressing and control associated with R0 thru R6.

2. Y-bus Gating.

The gating of data from the scratch pad memory, E1-bus, E2-bus, and C-field of the instruction to the Y-bus is provided on the holding register card by TSPY, TELY, TE2Y, and TCY bits, respectively, in the instruction word ( $Y_n = \dots + S_{Pn} \cdot TSPY + E_{1n} \cdot TELY + E_{2n} \cdot TE2Y + C_n \cdot TCY$ ).

L. REQUEST STROBE CARD.

Each microprocessor has two latches that are set by other microprocessors in the system. The latch may be tested by a branch condition. The special functions card generates the signals to set the latches specified by the contents of the 8 least significant bits of the X-bus. A request strobe is generated by the request strobe card when the special functions, MS, field contains a 17 (RSOUT1) or a 33 (RSOUT2); the specific strobe or strobes generated depends on the X-bus bits

that are set. The request strobes from the microprocessor are RS1(0) thru RS1(7) and RS2(0) thru RS2(7). The request strobe card contains a 16-bit storage register for the request strobes. Eight bits of the storage register are set by the request strobe 1 signal from the special functions card and the corresponding X-bus bit ( $sRS1(0) = RSOUT1 \cdot X(16)$ ;  $sRS1(1) = RSOUT1 \cdot X(17)$ ; ...;  $sRS1(7) = RSOUT2 \cdot X(23)$ ). The request strobe storage register is reset by K2; therefore, and request strobe bit is reset one cycle after it was set (RSOUT1 and RSOUT2 are set only during the interval that a branch condition is being tested, slightly less than one machine cycle). Other signals generated by the request strobe card are RSIN1, RSIN2, and STOPA. RSIN1 is jumpered to RS1(3) or RS2(4) as required. STOPA is a system reset derived from the microprocessor RESET; STOPA is synchronized with the K2 and K2A clocks ( $sSTOPA = RESET \cdot (K2A + K2)$ ;  $rSTOPA = RESET \cdot (K2A + K2)$ ).

M. MEMORY INTERFACE.

The memory interface provides the timing and control for data requests to both central and private memory. The memory interface is divided into three sections; the request field latches section, the private memory request section, and the central memory request. Requests to memory are made via the special functions field of the instruction word. Reference to central or private





3. An attempt to load the M register during a store or fetch before the memory has transferred data to or from M ( $SC = (LMRUN \cdot PDONED + CMRUN) \cdot (LMX + LMY) \cdot STATEA + \dots$ ).
4. An attempt to load  $R\bar{0}$  (the address register) before the memory has finished with the address ( $SC = (LMRUN \cdot PDONED + CMRUN) \cdot LR\bar{0} \cdot STATEA + \dots$ ).

#### 1. Request Mode and Priority Latches.

The request mode latches (F,S,H,) are loaded by the least significant three bits of the MS field in the instruction word by the start memory interface signal (SMI). The F request mode latch indicates that the instruction is a fetch operation, and is set by the K2 clock when the MS field contains a count of 45, 46, 47, 64, or 65. ( $sF = SC' \cdot SMIA \cdot SF \cdot K2$ ;  $rF = SC' \cdot SMIA \cdot SF' \cdot K2$ ) ( $SF = MS3$ ;  $SMIA = (SMIA')'$ ) +  $MS\bar{0} \cdot MS1' \cdot MS2' + ODD$ ;  $ODD = MS3 \cdot MS4' \cdot U6$ ). The S request mode latch indicates that the instruction is a store operation; and is set by the K2 clock when the MS field contains a count of 42, 43, or 47. ( $sS = SC' \cdot SMIA \cdot SS \cdot K2$ ;  $rS = SC' \cdot SMIA \cdot SS' \cdot K2$ ;  $SS = MS4$ ). The H request mode latch indicates that the instruction is a hold operation; and is set by the K2 clock when the MS field contains a count of 41, 43, 35, or 47. ( $sH = SC' \cdot SMIA \cdot SH \cdot K2$ ;  $rH = SC' \cdot SMIA \cdot SH' \cdot K2$ ;  $SH = MS5$ ). The priority latches provide direct outputs to the central memory for the determination of the priority of a memory request. The priority latches ( $CAP\bar{0}$ ,  $CAP1$ ,  $PP$ ) are set when the special function field of

the instruction contains an LPF,  $MS=22$ , and the corresponding bit on the X-bus ( $X(21)$ ,  $X(22)$ ,  $X(23)$ ; respectively) is set ( $sCAP\bar{0} = X21 \cdot PFL$ ;  $rCAP\bar{0} = X21' \cdot PFL$ ;  $sCAP1 = X22 \cdot PFL$ ;  $rCAP1 = X22' \cdot PFL$ ;  $sPP = X23 \cdot PFL$ ;  $rPP = X23' \cdot PFL$ ).

#### 2. The Central Memory Request Section.

The central memory request section consists of a state counter (MPREQ, FCY, and SCY). Requests to central memory via the MPMEM require four states, state  $\bar{0}$  through state 3. Requests that go directly to central memory (eg. CPU1 and CPU2), require three states, state  $\bar{0}$ , state 2, and state 3. The signal NOM (NO MPMEM) differentiates between microprocessors if the unit is going directly to the central memory (NOM is high) or the MPMEM (NOM is low). (See Figure 10.)

a. Requests To Central Memory Via the MPMEM. A request to memory generates a start memory interface signal (SMI), if the memory interface is in an idle state, state  $\bar{0}$  (State  $\bar{0} = MPREQ' \cdot FCY' \cdot SCY'$ ). SMI sets the MPREQ flip-flop, if the reference is to central memory, during the end cycle of the current instruction ( $sMPREQ = CMRUN \cdot SC' \cdot SMIA \cdot RCE \cdot K2 \cdot NOM'$ ). MPREQ set is state 1 of the memory interface cycle (STATE 1 =  $MPREQ \cdot FCY' \cdot SCY'$ ); during state 1 the MPMEM examines the request field of the microprocessors. The MPMEM latches the highest priority request, and sends a signal to the microprocessor selected (OK). The signal OK resets MPREQ ( $rMPREQ = OK \cdot K2$ ).

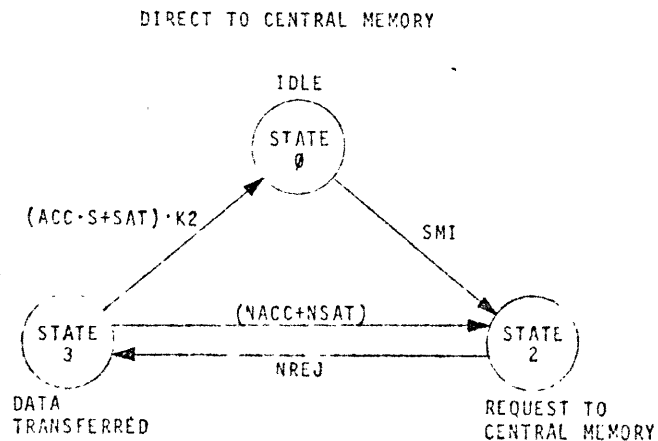
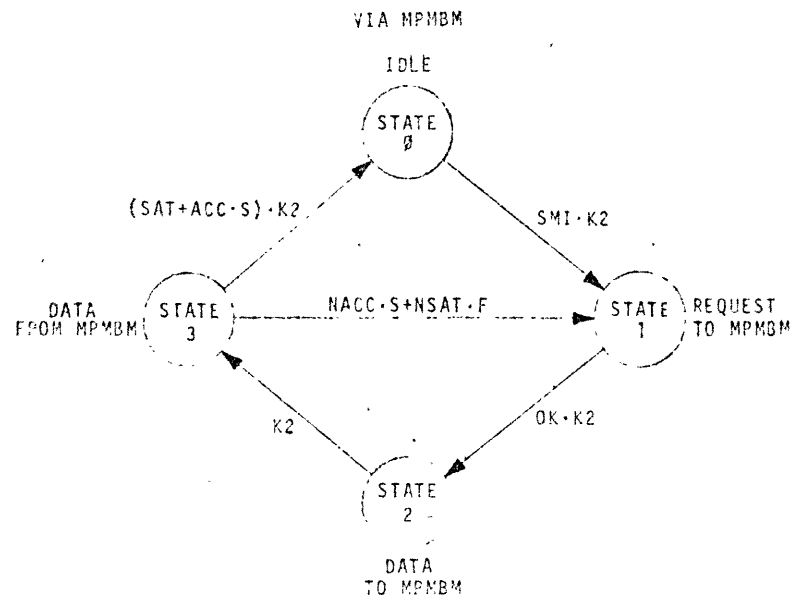


Figure 10. State Counter, Memory Interface.

and sets FCY ( $sFCY = OK \cdot K2$ ) on the selected microprocessor; the memory interface of the microprocessor selected goes to state 2 ( $STATE\ 2 = MPREQ' \cdot FCY \cdot SCY'$ ), any other microprocessors awaiting an OK remain in state 1. In state 2 the microprocessor sends the store data to the MPMBM. If the memory cannot process the data at this time, it will send a rejected signal ( $NREJ'$ ) to the microprocessor which returns the memory interface to state 1 ( $sMPREQ = NREJ' \cdot NOM' \cdot FCYd \cdot K2$ ;  $rFCY = FCYd \cdot NOM' \cdot K2$ ). If the request is not rejected ( $NREJ$ ) during state 2, the microprocessor enters state 3 ( $STATE\ 3 = MPREQ' \cdot FCY' \cdot SCY$ ;  $rFCY = FCYd \cdot NOM' \cdot K2$ ;  $sSCY = FCYd \cdot NREJ \cdot K2$ ). During a fetch, the M register is loaded from the memory bus via the MPMBM during state 3. If, during state 3, the data is not stored, the central memory sends back a signal during state 3 not accepted NACC which sets MPREQ ( $sMPREQ = NOM' \cdot NACC \cdot SCYd \cdot K2$ ); SCY is reset by K2 ( $rSCY = SCYd \cdot K2$ ), and the microprocessor is returned to state 1 where it repeats the cycle until the data is accepted; when the data is accepted, the microprocessor returns to state 0 ( $rSCY = SCYd \cdot K2$ ). If during state 3 the data is not fetched, the central memory sends back a signal request not satisfied NSAT which sets MPREQ ( $sMPREQ = NOM' \cdot NSAT \cdot F \cdot S' \cdot SCYd \cdot K2$ ), SCY is reset by K2 ( $rSCY = SCYd \cdot K2$ ), and the microprocessor is returned to state 1 where it repeats the cycle until the data is fetched; when the fetch is satisfied, the microprocessor returns to state 0 ( $rSCY = SCYd \cdot K2$ ).

50010

b. Requests Direct To Central Memory. Requests to central memory that do not go through the MPMBM, do not use MPREQ. Therefore state 1 is bypassed. When NOM is high, MPREQ is not set by SMI, FCY is set instead ( $SFCY = \text{NOM} \cdot \text{CMREF} \cdot \text{SMI} \cdot \text{RCE} \cdot \text{SC}' \cdot \text{K2}$ ). During state 2 the request is sent to central memory. If the memory request is not rejected (NREJ), the memory interface enters state 3 ( $s\text{SCY} = \text{FCYd} \cdot \text{NREJ} \cdot \text{K2}$ ;  $r\text{FCY} = \text{FCYd} \cdot \text{NOM} \cdot \text{REJ} \cdot \text{K2}$ ); otherwise the memory interface remains in state 2. Data is transferred during state 3. If a store is not accepted (NACC) or if a fetch is not satisfied (NSAT) during state 3, the memory interface returns to state 2 ( $s\text{FCY} = \text{SCYd} \cdot \text{NOM} \cdot \text{NACC} \cdot \text{K2} \cdot \text{F} \cdot \text{S}' \cdot \text{NOM} \cdot \text{SCYd} \cdot \text{NSAT} \cdot \text{K2}$ ;  $r\text{SCY} = \text{SCYd} \cdot \text{K2}$ ); otherwise the memory interface goes to the idle state, state 0 ( $r\text{SCY} = \text{SCYd} \cdot \text{K2}$ ).

c. Priority Requests, Clocks, and Load Signals.

Other signals generated by the central memory request section are:

$$\text{HIREQ} = \text{PP} \cdot \text{MPREQ}$$

$$\text{LOREQ} = \text{PP}' \cdot \text{MPREQ}$$

These signals make requests to the MPMBM during state 1 at low and high port priority respectively.

$$\text{LMMI} = (\text{F} \cdot \text{S}') \cdot \text{SCYd}$$

$$\text{MCLOCK} = (\text{F} \cdot \text{S}') \cdot \text{SCYd} \cdot \text{K2}$$

These signals are the clock and load signals for the M register

$$\text{GAM} = \text{SMI} \cdot \text{CMREF} \cdot \text{NOM} \cdot \text{RCE} \cdot \text{SC}' \cdot \text{K2}$$

50010

$$+\text{SCYd} \cdot \text{NOM} \cdot \text{NACC} \cdot \text{K2}$$

gate address to  
Memory

$$+(\text{F} \cdot \text{S}') \cdot \text{NOM} \cdot \text{NSAT} \cdot \text{SCYc} \cdot \text{K2}$$

$$+\text{OK} \cdot \text{K2}$$

$$+\text{FCY}$$

$$\text{GDM} = \text{FCYd} \cdot \text{NREJ} \cdot \text{K2}$$

gate data to Memory

$$+\text{SCY}$$

GDM and GAM signals are used on the M, Q, Z register card to gate the M register contents to the MPMBM when the request is a store, GDM and GAM; or a fetch, GAM.

3. The Private Memory Control Section.

The private memory control section provides timing and control signals for one 16K module of a microprocessor core memory. Data from the local memory will be loaded into the M register approximately 500 nanoseconds after the SMI special condition is executed. The start pulse (SET START P) for the private memory is generated from a latch-delay line circuit which makes an 80 nanoseconds pulse. This pulse is generated only if the request is a fetch or store;

$$\begin{aligned} (\text{SETSTART P} = & \text{SF} \cdot \text{SS}' \cdot \text{PMREF} \cdot \text{ODDWORD}' \cdot \text{SMIA} \cdot \text{RCE} \cdot \text{SC}' \cdot \text{K2} \\ & + \text{SF}' \cdot \text{SS} \cdot \text{PMREF} \cdot \text{ODDWORD}' \cdot \text{SMIA} \cdot \text{RCE} \cdot \text{SC}' \cdot \text{K2}). \end{aligned}$$

The 'store' strobes for the private memory select the even or odd half-doubleword in core as the destination for stored data, and cause the store to occur. They are delayed from the start pulse by 100 nanoseconds regardless of system clock rate:

50010

SUM = S.F'·R<sub>3</sub>(23)·SETSTARTPd (delayed by 100nsec)  
SLM = S.f'·R<sub>3</sub>(23)·SETSTARTPd (delayed by 100nsec)

A number of flip-flops are used to synchronize the private memory with the system clock. The LMRUN flip-flop is logically equivalent to UA (Unit Available) from the module, but is synchronized with the system clock (sLMRUN = STARTPd; rLMRUN = MDONE·K2+RESET). MDONE synchronizes UA with the system clock, and ensures that the memory will not be started for at least 100 nanoseconds after UA rises (sMDONE = UA·K2·PDONE; rMDONE = LMRUN'). Eighty nanoseconds after MDONE rises, (assuming a 100 nanoseconds clock period) LMRUN is reset, which resets everything else. To synchronize data transfers from core, the RDA (Read Data Available) signal from the core module is used. RDA is a pulse, shorter than 100 nanoseconds, and therefore must be latched (sRDAP = RDA; rRDAP = LMRUN'). The DBC (Data Back from Core) flip-flop synchronizes RDAP with the system clock (sDBC = K3·RDAP; rDBC = LMRUN'). PDONE is set 80 nanoseconds later (assuming 100 nanoseconds clock period) and signifies that M and R<sub>3</sub> may be modified (sPDONE = DBC·K2; rPDONE = LMRUN').

#### N. I/O INTERFACE

The I/O Interface card contains drivers and receivers for the following signals:

1. 24 bits of output data from the Z register.
2. 24 bits of input data to the E2 bus.
3. Attention latch 1, 2, and 3 inputs to the microprocessor branch logic.

50010

4. ALERT, POT, and PIN, and TUCLR outputs from the microprocessor special function card.
5. The global clock, which may be regenerated and used by external devices.

SN74H40N IC's are used for both drivers and receivers. The receiver inputs are terminated to +5V. All busses connected to external devices are low true. The drivers and receivers are assumed capable of operating with up to 12 TTL loads in addition to the loading introduced by interconnecting cables. The inputs of the E2 receivers are sources of approximately 20 ma. each. In the AMC, the TSU's are the only device that is connected to the I/O system. At the edge of the cage, the I/O busses are connected to two NE00058 connectors. These connectors are used to connect external devices such as the CHIO multiplexer via ribbon cables.

#### 1. ATTENTION.

The ATTENTION' signals are requests for service generated by external devices. These signals set latches in the microprocessor which may be tested and reset by branch conditions. Ideally, ATTENTION' is a 100 nanoseconds pulse overlapping  $t_0$ . The branch tests on the attention latches cause a branch on the latch not set. The latch is reset during the interval in which the branch test is done (STATE A is VCY=0, STATE B if VCY=1) unless the external device is attempting to simultaneously set

the latch, in which case it is set. Attention latch #1 also has a test for the latch set, but the reset conditions are the same as described above. The specification for a particular device should be consulted to determine the attention latch to which it is connected (if any), as well as the correct POT/PIN sequence to use.

## 2. TUCLR.

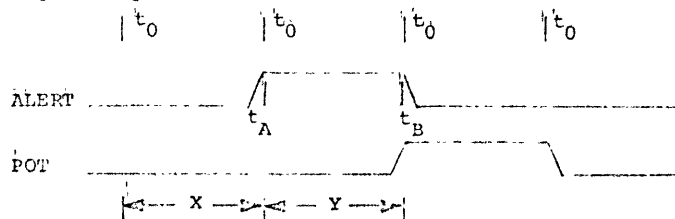
The signal TUCLR is a dc reset signal used by the TSU. It is set by the special condition RESET T.U. Unlike POT, PIN, and ALERT, it exists during the instruction in which the special condition is set, rather than the interval after the execution of the instruction. This is possible since the effects of TUCLR on the external device are assumed to be time-independent.

## 3. Timing and Programming.

The normal instruction sequence for doing output to a device is to execute:

```
DEVICE ADDRESS -> Z ALERT;
DATA -> Z POT;
```

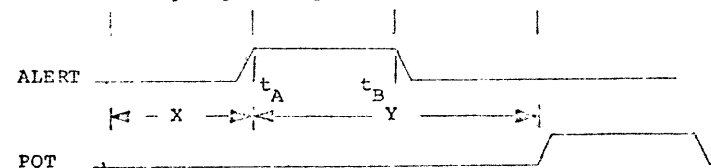
This instruction sequence results in the following signal sequence:



$t_0$  is the system reference time. It is approximately coincidental with the fall of K2 at the microprocessor. The first instruction above is executed during the interval designated 'X'. The second is executed during 'Y'. The Z register is loaded with the device address at  $t_A$ , and with the output data at  $t_B$ . If the external device requires more than one machine cycle to set up the data path, or if the external register is loaded with the POT strobe (rather than POT.K2), the following sequence may be executed:

```
D.A -> Z ALERT;
DATA -> Z, VCY, POT;
```

The resulting signal sequence is:



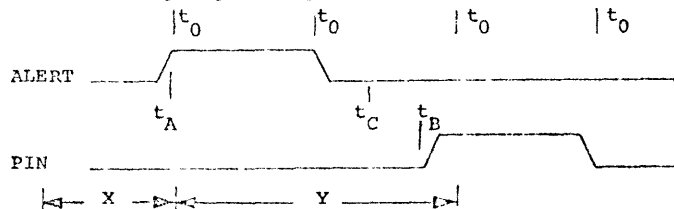
The general rule on timing of the ALERT, POT, and PIN strobes is that they rise at the end of the instruction during which the special condition is executed, and they fall at the end of the first interval during which the special condition does not exist. Thus it is possible to generate arbitrarily long strobes, but this should be avoided.

The normal sequence for doing input from a device is to execute:

DEVICE ADDRESS  $\rightarrow$  Z, ALERT

PIN, TE2Y, VCY; or PIN, TE2Y, UNCONDITIONAL BRANCH

The resulting signal sequence is:



The first instruction is executed during the interval 'X', the second during 'Y'. The device address is loaded into Z at  $t_A$ , and the microprocessor loads the data into one of its internal registers at  $t_B$ . The external device must have the data stable on the E2-bus no later than  $t_C = t_0 + 42$  nsec. If this timing is not usable due to limitations in the external device, the following sequence may be used:

DEVICE ADDRESS  $\rightarrow$  Z, ALERT;

NOP or NOP, VCY;

PIN, TE2Y;

This allows an extra one or two machine cycles between the presentation of the device address and the utilization of the returned data. The Bryant TSU's require a different POT/PIN sequence than most other I/O devices. The restriction on the TSU's is that output data

will remain stable on the Z-bus for 200 nanoseconds before ALERT or POT falls, and that input data will be allowed two machine cycles to stabilize before the fall of PIN. The instructions required to obtain this sequence are:

#### 1. OUTPUT

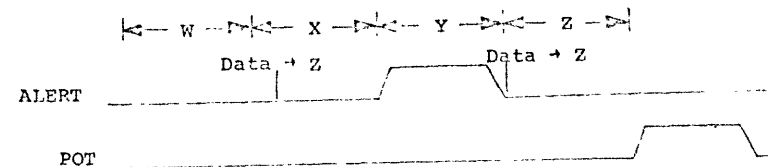
W: D.A.  $\rightarrow$  Z;

X: ALERT;

Y: DATA  $\rightarrow$  Z;

Z: POT;

which results in:



#### 2. INPUT

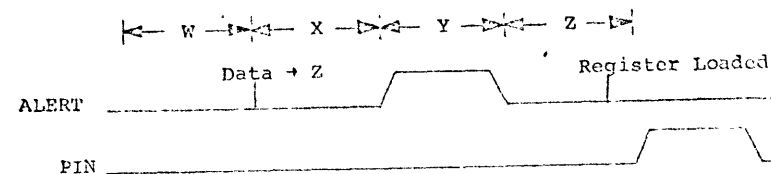
W: D.A.  $\rightarrow$  Z;

X: ALERT;

Y: VCY, NOP

Z: TE2Y, Y  $\rightarrow$  REGISTER; PIN

which results in:



50010

APPENDIX A

-A1-

50010

Table A1. 90-bit Microinstruction Word.

Signal	Position	Clock	Function
MCS <sup>0</sup> -MCS <sup>5</sup>	0-5	K2	Branch Condition field (6 bits). (See Table A2).
MCONTC <sup>0,1</sup>	6,7	K3	Branch control field (2 bits): 0 = branch conditionally to the address specified by the contents B <sub>0</sub> thru B <sub>9</sub> . 1 = branch conditionally to the address specified by the contents of B <sub>0</sub> thru B <sub>9</sub> . Store the contents of the O Register (return address) in the OS Register. 2 = branch conditionally to the address specified by the contents of the OS Register. 3 = branch conditionally to the address specified by the contents of the λ-bus ten lsb.

-A2-



50010

Table A1. 90-bit Microinstruction Word. (cont'd)

Signal	Position	Clock	Function
B0; B1, B2	8; 9; 10,	K3	Branch address field (10-bits).
B3; B4, B5	11, 12; 13		
B6, B7, B8	14, 15, 16		
B9	17		Constant Field (24-bits).
C0, C1, C2	18, 19, 20	K3	
C3, C4, C5	21, 22, 23		
C6, C7, C8	24, 25, 26		
C9, C10,	27, 28, 29		
C11, C12,	30, 31, 32		
C13, C14,	33, 34, 35		
C15, C16,	36, 37, 38		
C17, C18,	39, 40, 41		
C19, C20,			
C21, C22, .			
C23			
IHR	42	K2	Increment holding register.
TCX	43	K3	Transfer the contents of C-field to the X-bus.
TCY	44	K3	Transfer the contents of C-field to Y-bus.
TSPY	45	K3	Transfer the contents of the selected scratchpad address to the Y-bus.

50010

Table A1. 90-bit Microinstruction Word. (cont'd)

Signal	Position	Clock	Function
THY	46	K3	Transfer the contents of the holding register to the Y-bus.
TXW	47	K3	Transfer the data in the X-bus to the holding register.
TYW	48	K3	Transfer the data in the Y-bus to the holding register
TAX	49	K3	Transfer adder output to the X-bus
LOC	50	K2	Low order carry to adder.
SSP0-5	51-56	K2	Selects the contents of scratchpad address 0-77 (3)
TOSY	57	K3	Transfers the contents of the OS register to Y-bus (bits 14-23)
LRO	58	K3	Load holding register R0.
LSPX	59	K3	Loads the selected scratchpad address from the X-bus.
MSG-MS5	60-65	K2	Special functions field. (See Table A3.)

Table A1. 90-bit Microinstruction Word. (cont'd)

Signal	Position	Clock	Function
RRN $\bar{0}$ -2	66-68	K2	Holding register select read field (enables selected holding register, R $\bar{0}$ thru R $\bar{6}$ , output).
LRN $\bar{0}$ -2	69-71	K3	Holding register select load field (clocks selected holding register, R1 thru R $\bar{6}$ , input).
LMX	72	K3	Loads the M register from the X-bus.
LMY	73	K3	Loads the M register from the Y-bus.
LQX	74	K3	Loads the Q register from the X-bus.
LQY	75	K3	Loads the Q register from the Y-bus.
LZX	76	K3	Loads the Z register from the X-bus.
LZY	77	K3	Loads the Z register from the Y-bus.
BL $\bar{0}$ -BL $\bar{3}$	78-81	K2	Left bool box control field (See Table A4).

Table A1. 90-bit Microinstruction Word. (cont'd)

Signal	Position	Clock	Function
BR $\bar{0}$ -BR $\bar{3}$	82-85	K2	Right bool box control field (See Table A4).
VCY'	86	K3	State counter set-term.
DGO	87	K3	State counter set-term.
TE1Y	89	K3	Transfer data from the E-1 Bus to the Y-bus.
TE2Y	90	K3	Transfer data from the E-2 bus to the Y-bus.

Table A2. Branch Conditions.

MC $\beta$ -MC5	Branch Condition
$\emptyset\emptyset$	Never branch
$\emptyset 1$	Always branch
$\emptyset 2$	$X=\emptyset$
$\emptyset 3$	$X\neq\emptyset$
$\emptyset 4$	$X<\emptyset$
$\emptyset 5$	$X\geq\emptyset$
$\emptyset 6$	$X>\emptyset$
$\emptyset 7$	$Y>\emptyset$
$1\emptyset$	$Y<\emptyset$
$11$	$R\emptyset<\emptyset$
$12$	$R\emptyset>\emptyset$
$13$	$X<\emptyset$
$14$	$X'\wedge 777777B=\emptyset, (X(6)-X(23)=777777B)$
$15$	$X'\wedge 777777B\neq\emptyset, (X(6)-X(23)\neq 777777B)$
$16$	$Z>\emptyset$
$17$	$Z<\emptyset$
$2\emptyset$	Always Branch
$21$	$Y\wedge 7\neq\emptyset, (Y(23)\vee Y(22)\vee Y(21)=1)$
$22$	$BL=\emptyset$
$23$	$BL\neq\emptyset$
$24$	$Y(23)=\emptyset$
$25$	$Y(23)\neq\emptyset$
$26$	Attention latch $i=\emptyset$ $\Delta$

Table A2. Branch Conditions.(cont'd)

MC $\beta$ -MC5	Branch Condition
$27$	$(RSLAT1=\emptyset)\wedge(RSLAT2=\emptyset)$
$3\emptyset$	Protect $\neq X$
$31$	$RSLAT2=\emptyset$
$32$	Special flag $A=\emptyset$
$33$	Special flag $A\neq\emptyset$
$34$	Attention latch2= $\emptyset$ $\Delta$
$35$	Attention latch3= $\emptyset$ $\Delta$
$36$	Attention latch1 $\neq\emptyset$ $\Delta$
$37$	Not decoded
$4\emptyset$	Undefined
$41$	Undefined
$42$	Local memory parity error=1 $\Delta$
$43$	Undefined
$44$	Central memory parity error=1 $\Delta$
$45$	Breakpoint $\neq\emptyset$
$46-77$	$\Delta$

 $\Delta$  Resets latch. $\Delta$  46 thru 77 not decoded

Table A3. Special Functions.

MSG-MS5	Function
00	No activity
01	LCY1
02	LCY2
03	LCY3
04	LCY4
05	LCY8
06	LCY12
07	LCY16
10	LCY20
11	LCL Z (CCFZA)
12	LCH Z (CCFZB)
13	SKZ (SPFZ)
14	ALERT
15	POT
16	PIN
17	Request Strobe #1
20	Unprotect
21	Unusable
22	LPF
23	Reset Request Strobe Latch #1 $\triangle$
24	Reset Central Memory Request $\triangle$
25	Request Protect
26	Reset T.U.
30	Set special flag A

Table A3. Special Functions.(cont'd)

MSG-MS5	Function
31	Reset special flag A
32	Reset Request Strobe Latch #2
33	Request Strobe #2
34	Undefined
40	Release $\triangle$
41	Prestore $\triangle$
42	Store $\triangle$
43	Store & Hold $\triangle$
44	Fetch $\triangle$
45	Fetch & Hold $\triangle$
47	Prefetch $\triangle$
60	Set Bank B
61	Set Bank A
62	Clear all CPU Maps
64	Fetch $\triangle$
65	Fetch & Hold $\triangle$
	$\triangle$

 $\triangle$  Occurs at end of instruction $\triangle$  Local and Central Memory $\triangle$  Memory Reference $\triangle$  ODDWORD FETCH $\triangle$  Special functions 27, 35, 36, 37, 46, 63, and 66 thru 77 are not decoded

Table A4. Bool Box Control.

BL $\phi$ -BL3	Left Bool Box Output	BR $\phi$ -BR3	Right Bool Box Output
$\phi\phi$	M $\cdot$ Q	$\phi\phi$	Z $\cdot$ Q
$\phi 1$	M=Q	$\phi 1$	Z=Q
$\phi 2$	Q	$\phi 2$	Q
$\phi 3$	$\bar{M}+Q$	$\phi 3$	$\bar{Z}+Q$
$\phi 4$	M	$\phi 4$	Z
$\phi 5$	$\bar{M}+Q$	$\phi 5$	$\bar{Z}+Q$
$\phi 6$	M+Q	$\phi 6$	Z+Q
$\phi 7$	1	$\phi 7$	1
$1\phi$	$\phi$	$1\phi$	$\phi$
11	$\bar{M}\cdot\bar{Q}$	11	$\bar{Z}\cdot\bar{Q}$
12	$\bar{N}\cdot Q$	12	$\bar{Z}\cdot Q$
13	$\bar{N}$	13	$\bar{Z}$
14	M $\cdot\bar{Q}$	14	Z $\cdot\bar{Q}$
15	$\bar{Q}$	15	$\bar{Q}$
16	M(EOR)Q	16	Z(EOR)Q
17	$\bar{M}+\bar{Q}$	17	$\bar{Z}+\bar{Q}$

Table A5. Summary of Signal Mnemonics

Signal Mnemonic	Page Reference	Signal Mnemonic	Page Reference
		B (cont'd)	
A		BSELECT	29
A bank	34	BSS	37
ADV1	45		
ADV2	45	C	
ADVNC	45	C(n)	56
ADVNO	45	CAP $\phi$	70
ALERT	77	CAP1	70
APAR	62	CCFZA	47
ATLAT1	43	CCFZB	47
ATLAT2	44	CMAP	49
ATLAT3	44	CMPE	44
ATTENTION	77	CMREF	68
B		CMRUN	70
B bank	34	CP $\phi$	68
B-field	2	CPI	68
BL-field	52	CS $\phi$ -CS7	37
BL(n)	52	CS1 $\phi$ -CS17	37
BL $\phi$ -BL3	54		
BP	44	D	
BR(n)	52	DBC	77
BR-field	52	DGO	25
BR $\phi$ -BR3	54		
BRANCH	25	E	
BR $\phi$ -ST	33	F	
		F	68

Table A5. Summary of Signal Mnemonics (cont'd)

Signal Mnemonic	Page Reference	Signal Mnemonic	Page Reference
F (cont'd)		K(cont'd)	
FCY	71	KIØ	23
FDONLD	69	KI1	29
FLUSH	58	KI2	23
G		KI3	9
GAM	75	KI4	10
GDM	57	L	
H		LCY(n)A	58
H	68	LCY(n)B	46
HIREQ	74	LMML	52
I		LMN2	52
I(n)	34	LMPAR(n)IN	59
12CLOCK	29	LMPAR(n)OUT	59
13CLOCK	29	LMPE	44
12CLOCKA	29	LMRUN	70
12/3CLOCKA	34	LMX	30
12CLOCKB	29	LMY	30
IHR	63	LOAD OREG	25
K		LOAD OSREG	32
K1	23	LOREQ	75
K2	25	LPP	34
K2A	31	IQX	15
K2X	25	IQY	30
K3	25	LRN	30

Table A5. Summary of Signal Mnemonics (cont'd)

Signal Mnemonic	Page Reference	Signal Mnemonic	Page Reference
LRNØ		M (cont'd)	
LRN1	30	MSØ-MS5	34
LRN2	30	N	
LRO	30	NACC	73
RSPX	33	NOM	71
LSPY	62	NOP	61
LZQ	30	NREF	73
LZY	25	NSAT	73
M		O	
M(n)	54	O(n)	38
M1-bus	53	O-register	38
M2-bus	53	ODD	68
MC-field	31	ODDWORD	75
MCLOCK	30	OK	75
MCLOCK1	30	OS(n)	41
MCLOCK2	30	OS-register	41
MCLOCK3	52	P	
MCONT-field	31	PAR(n)OUT	59
MCONTØ	31	Pc	57
MCONT1	32	PFL	34
MDONE	76	PIN	47
MI RESET	48	PK	57
MPREQ	71	PMREF	66
MS-field	43	PSEX	44

Table A5. Summary of Signal Mnemonics (cont'd)

Signal Mnemonic	Page Reference	Signal Mnemonic	Page Reference
P (cont'd)		R (cont'd)	
POT	47	RSIN1	67
PP	68	RSIN2	67
PX	48	RSLAT1-RSLAT2	44
Q		RSOUT(n)	47
Q(n)	54	S	
QCLOCK	30	S	69
R		S $\phi$ -S7	42
R $\phi$ -R6	63	SATLAI-SATLA3	44
R $\phi$ X	66	SB1-SB3	44
R $\phi$ CLOCK-R6	30	SC	69
RA $\phi$ n	34	SCMP	60
RA1n	5	SCY	71
RB $\phi$ n	34	SELECTA	36
RB1n	34	SELECTB	36
RCE	28	SET START P	75
RCMP	61	SF	75
RDA	76	SLH	76
RDAP	76	SLMP	60
RESET	48	SMI	68
RESETTU	48	SMIA	68
RLMP	61	SMS	48
RPN-field	30	SP(n)	66
RS1-RS2	44	SPAG	44

Table A5. Summary of Signal Mnemonics (cont'd)

Signal Mnemonic	Page Reference	Signal Mnemonic	Page Reference
S (cont'd)		T (cont'd)	
Special Flag A	44	TUCLR	77
SPPZ	47	TWX	63
SS	76	TXC	31
SSP	61	TXW	68
ST $\phi$ $\phi$ $\phi$ $\phi$ 3	68	TYW	63
Start P	76	U	
State A	25	U $\phi$ -U7	
State B	25	U $\phi$ 1	31
State C	25	UA	76
STOPA	67	UPX	48
SUH	76	V	
T		V $\phi$ -V7	46
T $\phi$ -T7	42	VCY	25
T $\phi$ O	31	W	
TCMP	61	WS	33
TCY	66	X	
TE1Y	66	X(n)	41
TE2Y	66	X $\phi$ -XS7	37
THY	63	XNB	25
TINCO	31	XXC	25
TIMP	61	Y	
TOSO	31	Y(n)	66
TS1Y	66	Y $\phi$ -YS7	37
		Z	
		Z(n)	53