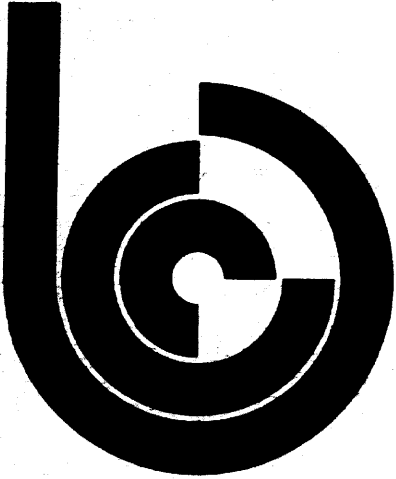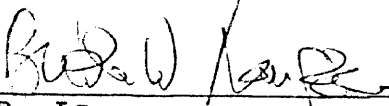BCC

MICROPROCESSOR

MANUAL

BCC

MICROPROCESSOR

MANUAL

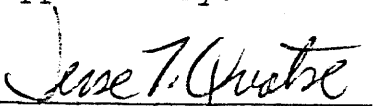50010   1 APRIL 70

Concurred By: Approved By:

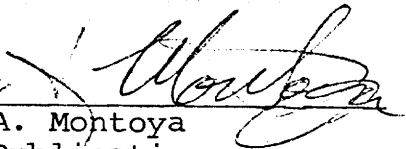B. Lampson   J. T. Quatse
Programming   Vice-President

C. Thacker
Engineering

A. Montoya
Publications

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

INTRODUCTION

The BCC 500 system will accommodate a large number of term-
minals 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
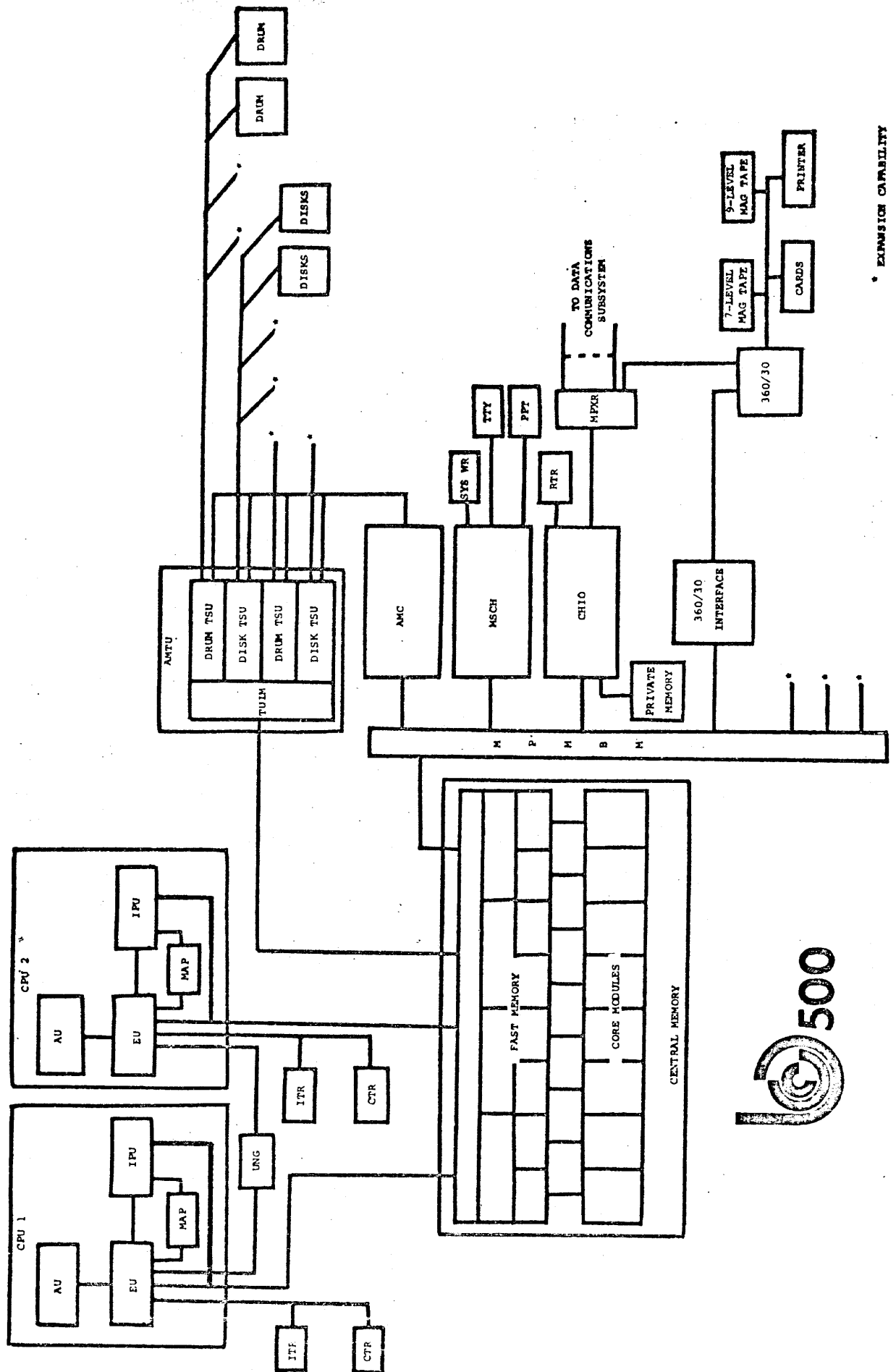
Figure 1. BCC 500, System Block Diagram.

SECTION 1
GENERAL DESCRIPTION

A.   INTRODUCTION.

The purpose of this manual is to provide the individual
with a working knowledge of the Berkeley Computer Corpora-
tion (BCC) Microprocessor. This manual describes the func-
tional characteristics and operation of the BCC Micropro-
cessor.

B.   PURPOSE OF EQUIPMENT.

The BCC Microprocessor is a system-oriented 24-bit
computer synchronous with a 100-nanosecond clock. The BCC
Microprocessor is a controlling element, capable of pro-
viding a wide variety of data processing and control func-
tions, in a system. Special functions may be added as
required to a particular unit, providing the system designer
with a broad spectrun of interface capabilities and
allowing him to optimize the capability of the unit to his
specific application.

C.   BCC 500, SYSTEM, (See Figure 1.)

There are five microprocessors used in the BCC 500
systems.

   1.   The Execution Unit (EU) of Central Processing
        Unit 1 (CPU 1) - Runs user programs.

   2.   The Execution Unit (EU) of Central Processing
        Unit 2 (CPU 2) - Runs user programs.

   3.   Character Input and Output (CHIO) - Buffers data
        between the system and (remote) user terminals.

   4.   Microscheduler (MSCH) - Controls allocation of
        system resources.

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		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

⚠1 Test Purpose only
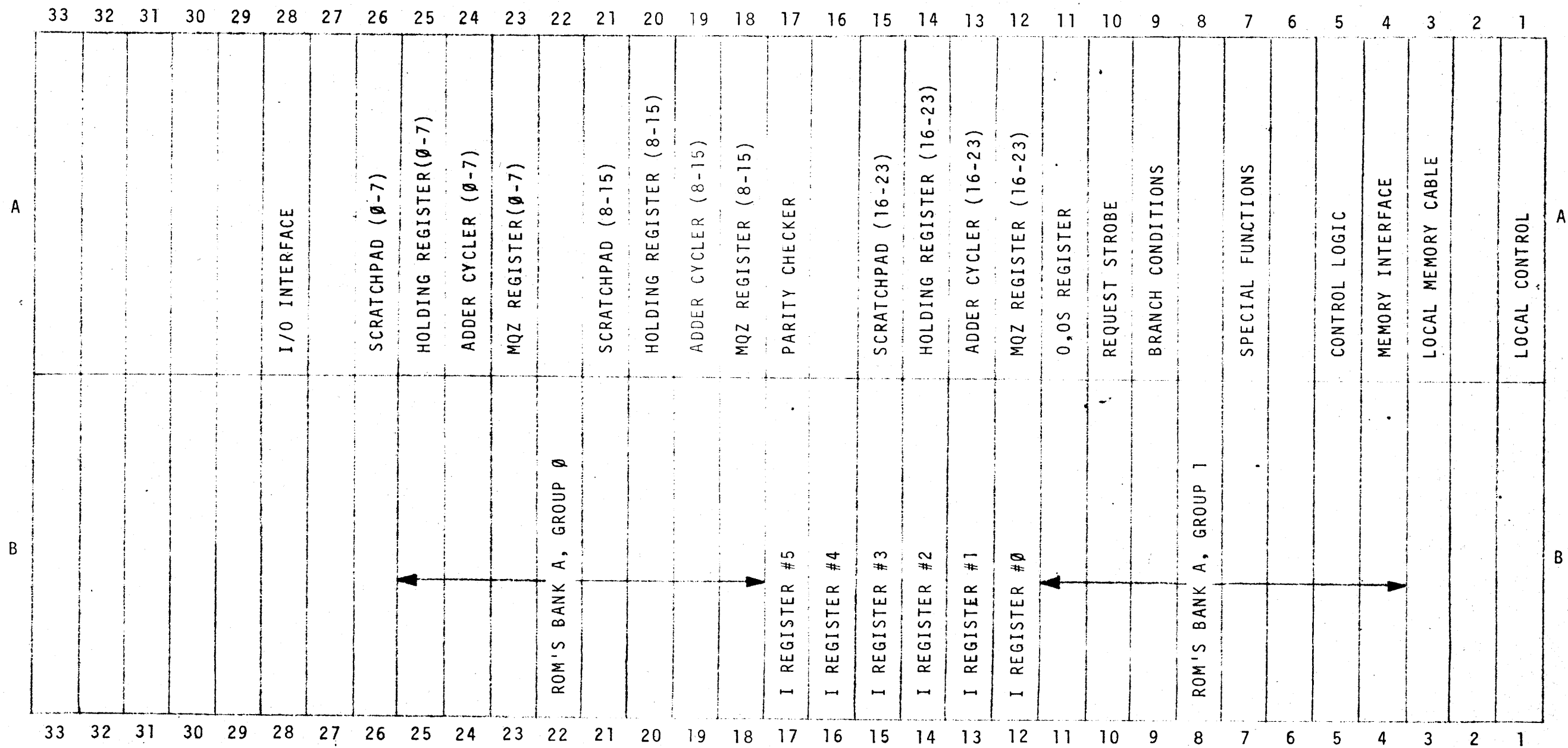
Figure 2. Card Location.

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	AC00251A	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:
        Pilot Power   + 5 vdc / − 5 vdc / +12 vdc
        Operating Power   + 5 vdc / − 5 vdc / +10 vdc
    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 360 model 30 processor
AMC	Auxiliary Memory Control
AMTU	Auxiliary Memory Transfer Unit
BCC	Berkeley Computer Corporation
CHIO	CHaracter Input/Output
CPU	Central Processor(ing) Unit
CTU	Compute Time Register
EOR	Exclusive OR
FM	Fast Memory
I/O	Input-Output
ITR	Interval Timer Register
MAP	Logical memory address to physical memory address MAP
MP	MicroProcessor
MPMBM	MicroProcessor Memory Bus Multiplexer
mpxr	MultiPleXeR
MSCH	MicroSCHeduler
NDRO	Non-Destructive Read-Out
ROM	Read-Only Memory
RTR	Real Time Register
System WR	SYSTEM Warning Registers
TSU	Transfer Sub-Unit
TUIM	Transfer Unit Interface Multiplexer
UNG	Unique Name Generator

SECTION 2

GENERAL THEORY OF OPERATION

SECTION 2
GENERAL THEORY OF OPERATION

A. INTRODUCTION

The BCC 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 Al,

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 accom-

plished by the El-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 Ml-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
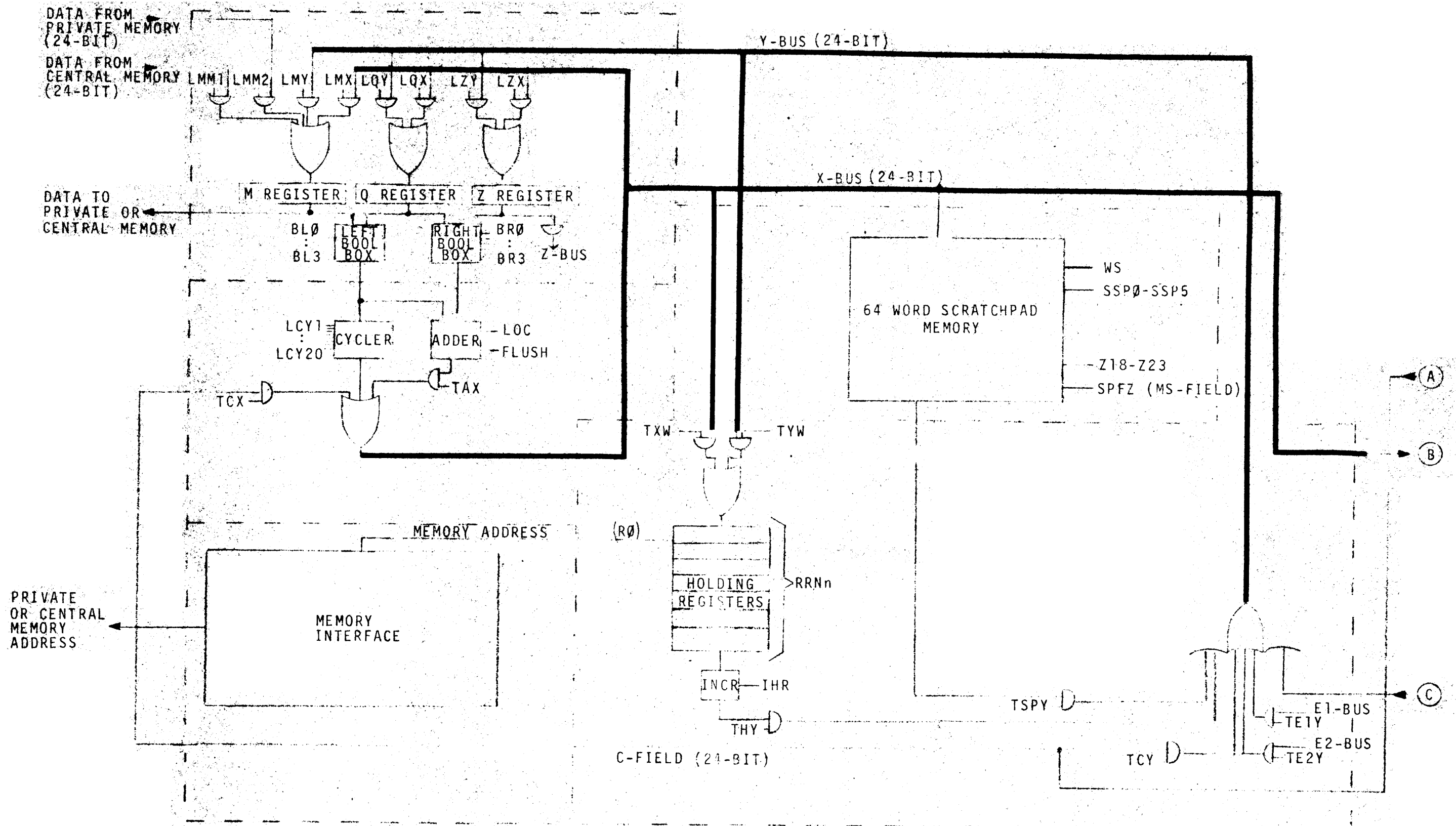
Figure 3a. BCC Microprocessor. (Sheet 1 of 2)
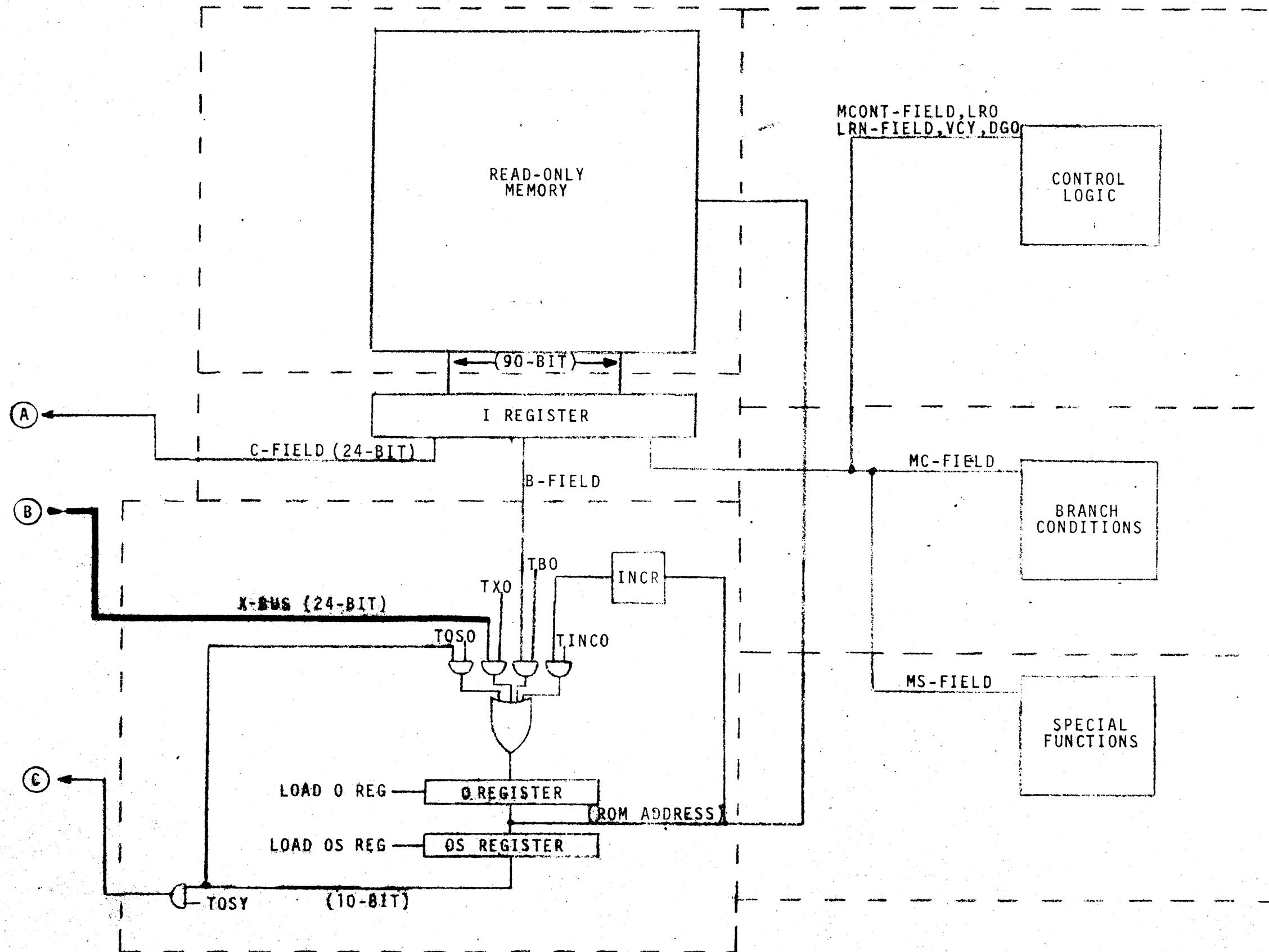
Figure 3b. BCC Microprocessor. (Sheet 2 of 2)

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 ancilliary 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∅ thru R6, 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 MCONT 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, DGO, 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 instruc-

tion.

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

-17-
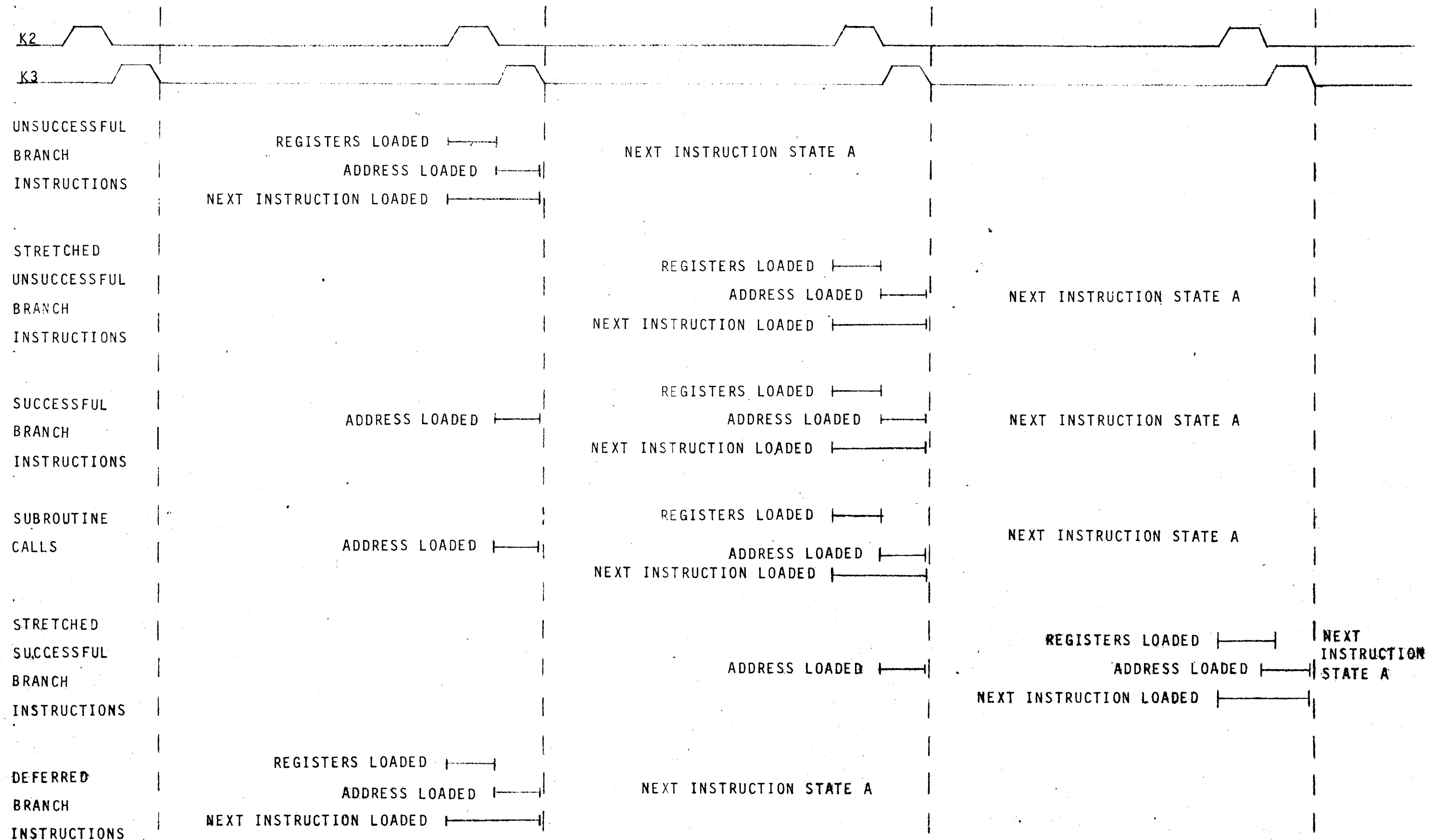
Figure 4. Instruction Cycles

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 propogate 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.

SECTION 3

DETAILED THEORY OF OPERATION

SECTION 3
DETAILED THEORY OF OPERATION

A. CONTROL LOGIC CARD

The control logic card contains the timing, state coun-
ter, and the register clock circuits.

1. Timing.

The control logic card generates the clocks for the
microprocessor from the system clock (K1). The micropro-
cessor 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 indivi-
dual cards. Figure 5 shows the distribution of the clocks.

Table 4. Clock Generation.

NAME	START TAP	STOP TAP	FUNCTION
KIØ	N	M	Clocks state counter flop XXB and XXC.
KI1	O	I	Gated to produce RØCLOCK thru R6CLOCK, MCLOCK, QCLOCK, ZCLOCK, PFL.
KI2			Gated to produce I2CLOCKA and I2CLOCKB.
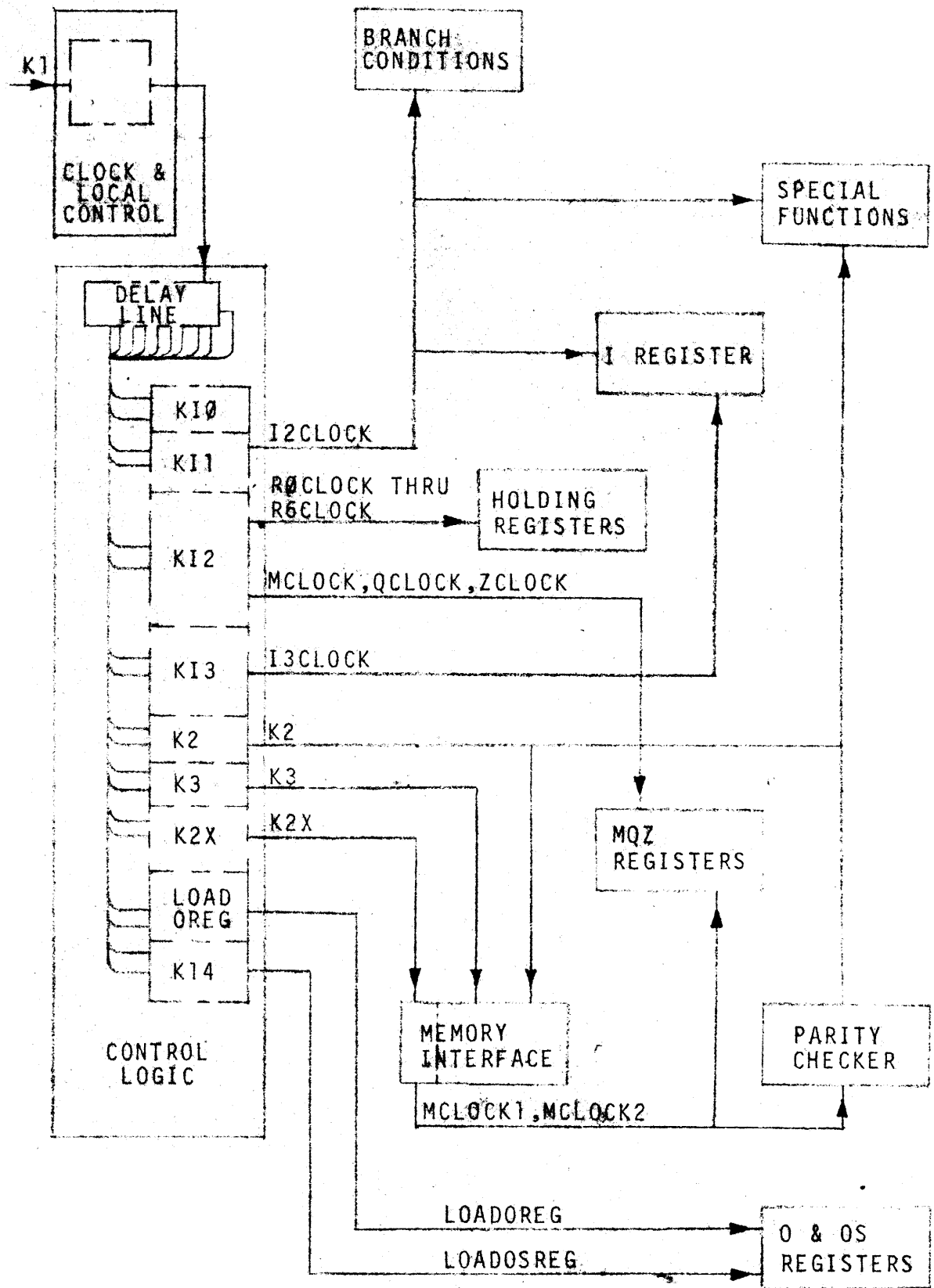
Figure 5. Clock Distribution.

Table 4 . Clock Generation. (cont'd)

NAME	START TAP	STOP TAP	FUNCTION
KI3	J	H	Gated to produce I3CLOCKA and I CLOCKB.
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 $A=XXB'\cdot XXC'$). State B
(state $B=XXB\cdot XXC'$) is entered if the condition specified
in the branch condition field is satisfied and DGO is not
set ($sXXB=XXB'\cdot XXC'\cdot BRANCH\cdot DGO'$ ...), or if the VCY bit in the
instruction is set ($sXXB= ...+XXB'\cdot XXC'\cdot VCY+...$). The
XXB flip-flop will be reset one machine cycle after it
has been set ($rXXB=XXB\cdot...$). State C (state $C=XXB'\cdot XXC$)

	VCY	DGO	BRANCH	NEXT STATE	REGISTER CLOCKS	O REGISTER CLOCK	OS REGISTER CLOCK	O REGISTER DATA SOURCE
STATE A	0	0	0	A	1	1	0	INCREMENTER
	0	0	1	B	0	1	1; if a call	MCONT-FIELD
	0	1	0	A	1	1	0	INCREMENTER
	0	1	1	A	1	1	1; if a call	MCONT-FIELD
	1	0	0	B	0	0	0	-
	1	0	1	B	0	0	0	-
	1	1	0	B	0	0	0	-
	1	1	1	B	0	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	MCONT-FIELD
	1	1	0	A	1	1	0	INCREMENTER
	1	1	1	A	1	1	1; if a call	MCONT-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·VCY·BRANCH·DGO'·...). The XXC flip-flop is reset one machine cycle after it has been set (rXXC=XXC·...). The state counter is clocked by KIØ, and is inhibited during a memory store or fetch by the SC term from the memory interface card (sXXB, rXXB, sXXC, rXXC=...·KIØ·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'·XXC'· VCY'·BRANCH'+...), or if VCY is not set in the instruction and DGO is set in the instruction (RCE= ...+XXB'·XXC'· VCY'·DGO+...). State B is the last cycle of an instruction if VCY is not set in the instruction (RCE = ...+XXB·VCY'+ ...), or if DGO is set in the instruction (RCE= ... +XXB·DGO+ ...). State C is always the last cycle for three cycle instructions (RCE = ...+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 KI1 if the BSELECT 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=KI2·BSELECT'·RCE·SC'). BSELECT is set by a special function and remains set until cleared by a special function.   I2CLOCKB is generated by KI2 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=KI1·BSELECT·RCE·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 KI3 instead of KI2 (I3CLOCKA=KI3· BSELECT'·RCE·SC') (I3CLOCKB=KI3·BSELECT·RCE·SC').

6. <u>RØ thru R6 CLOCKS.</u>

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

7. <u>MCLOCK3, QCLOCK, ZCLOCK.</u>

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-bux 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. RCE·SC'·(LMX+LMY). The QCLOCK is generated by; QCLOCK=KI2·RCE·SC'·(LZX+LZY). The M, Q, and Z registers must be clocked for a loading operation.

8. <u>K2 CLOCK.</u>

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'·(XXB+XXC+VCY')·UK1 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'+...), if VCY is not set

during state B (TINCO= ...+ XXB·VCY' +...), or during

state C (TINCO= ...+XXC). TBO transfers the B field

of the instruction word into the O register during state A

if the MCONT field contains a Ø or 1 (MCONTØ is not set),

the branch condition is satisfied, VCY is not set in the instruction, and the instruction does not interfere with a store (TB0=XXB'·XXC'·MCONT'·BRANCH·SC'+...); or during state B when the branch condition is satisfied, MCONT$\emptyset$ is not set in the instruction, and VCY is set in the instruction (TB0= ...+XXB·BRANCH·MCONT$\emptyset$'·VCY). TOSO transfers the contents of the OS register to the 0 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$\emptyset$·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$\emptyset$·MCONT1'·BRANCH·VCY). TXO will transfer the contents of the contents of the least significant ten bits of the X-bus to the 0 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$\emptyset$· 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$\emptyset$·MCONT1·VCY).

    12.   OS Register Control.

        The contents of the 0 register are clocked into the OS register by the LOAD OSREG clock. The clock is generated in the control logic by the KI4 clock. LOAD

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$\emptyset$' 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$\emptyset$'·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 = ẌXB·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=KI2·state A').

15. Priority Latching.

PFL is generated by the control logic by KI2 clock to load the priority latches for central memory requests from the memory interface. PFL occurs during the last machine cycle of an instruction (RCE' if the special conditions field in the instruction contains an LPF (MSØ thru MS5=22B) and the memory interface is not in the process of doing a fetch or a store (PFL= KI2·RCE·LPF·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 Al in Appendix A for timing a listing and functions of the individual instructions bits). The I register bit is set by RAØn or RAln (Where: n = 0 thru 89.) and the corresponding I2/3CLOCKA (sIn = RAØn·I2/3CLOCKA+RAln· I2/3CLOCKA+ ...); or by RBØn or RBln and the correspon- ding I2/3CLOCKB (sIn =...+RBØn·I2/3CLOCKB+RBln·I2/3CLOCKB). (RAØn is the bit from ROM card Ø thru 7 in the A bank; Raln is the bit from ROM card 8 thru 15 in the A bank; RBØn is a bit from ROM card Ø thru 7 in the B bank; RBln is a bit from ROM card 8 thru 15 in the B bank.
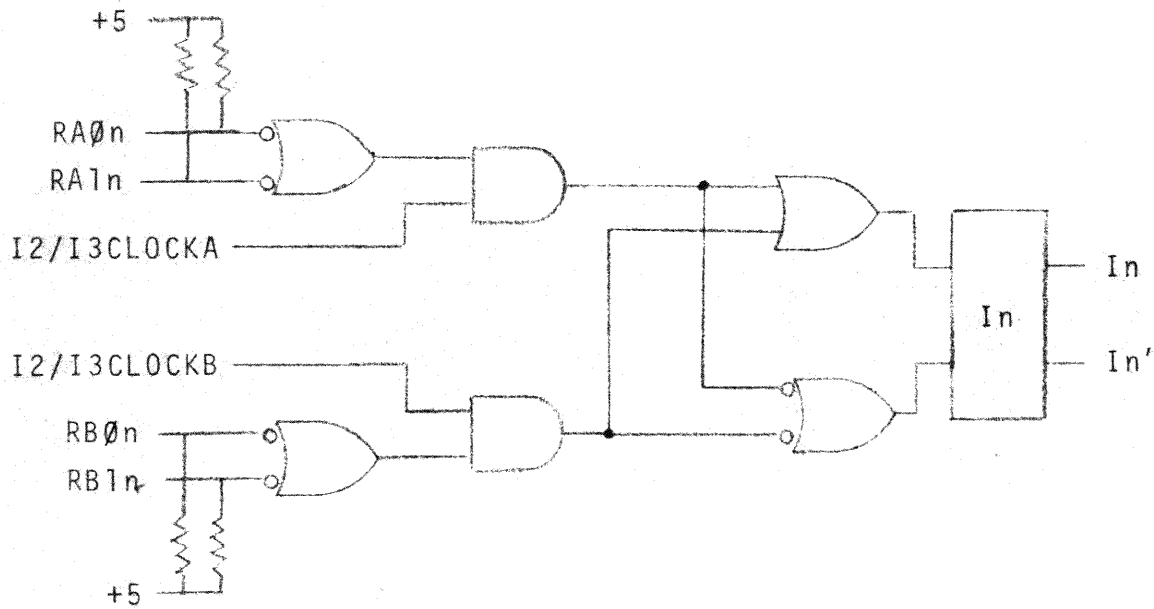
WHERE: n = 0 THRU 89
AND RA0n, RA1n, RB0n, RB1n: SET = GROUND
                                NOT SET = OPEN

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 = RAØn'·RA1n'·I2/3CLOCKA+RAØn·RB1n·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 90-bit words. The contents of the ROM address specified by O register, are loaded into the I register. Both banks of the ROM are addressed by the O 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 O 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 O 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 re-called, the BSELECT flip-flop is set from the X-bus.

3. Address Decodings.

The contents of the O register are decoded by a logic matrix on the O and OS registers card. Bit $\emptyset$ thru 3 of the O register generate the card selected signals, CS$\emptyset$ thru CS7 and CS1$\emptyset$ thru CS17. Bits 4, 5, and 6 of the O register generate the Y-line selected signals, YS$\emptyset$ thru YS7. Bits 7, 8, and 9 of the O register generate the X-line selected signals, XS$\emptyset$ 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 satis-

fied.   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 $\emptyset$ or 1 (sOn = LOAD OREG·Bn

TB0+...) 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·OSn·TOSO...), or by the

LOAD OREG clock and the least significant 10 bits

in the X-bus if the MCONT field of the instruction equals
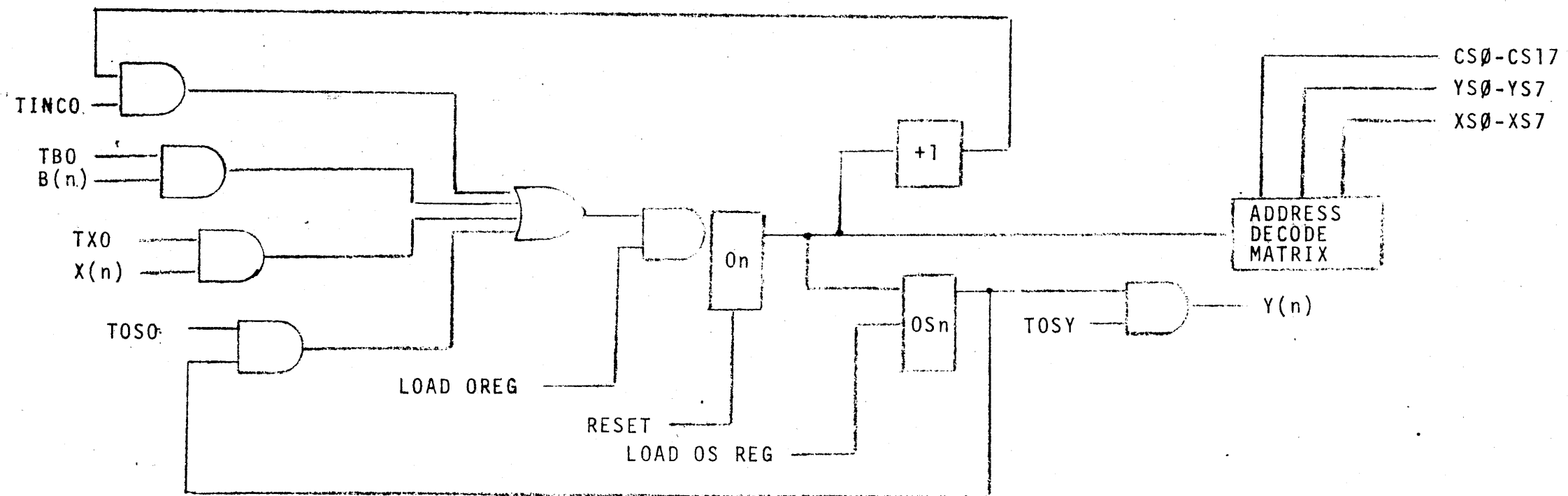
WHERE B,O, & OS:n=0 THRU 9
AND X & Y:n=14 THRU 23

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

3 (sON=...+LOAD OREG·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, $\emptyset\emptyset$
thru $\emptyset 8$, 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 (sOSn=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 $\emptyset$, 1, and 2 in the MC-

field, and is decoded to generate S$\emptyset$ thru S7. The second

group is comprised of bits 3, 4, and 5 in the MC-field,

and is decoded to generate T$\emptyset$ 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
$S\emptyset \cdot T1$	$MC=\emptyset 1$ (will always branch)
$+S\emptyset \cdot T2 \cdot (X(0)' \cdot X(1)' \cdot \ldots \cdot X(23)')$	$(MS=\emptyset 2) \cdot (X=\emptyset)$
$+S\emptyset \cdot T3 \cdot (X(0)+X(1)+\ldots+X(23))$	$(MC=\emptyset 3) \cdot (X\neq\emptyset)$
$+S\emptyset \cdot T4 \cdot X(\emptyset)$	$(MC=\emptyset 4) \cdot (X<\emptyset)$
$+S\emptyset \cdot T5 \cdot X(\emptyset)'$	$(MC=\emptyset 5) \cdot (X\geq\emptyset)$
$+S\emptyset \cdot T6 \cdot X\emptyset' \cdot (X(\emptyset)+X(1)+\ldots+X(23))$	$(MC=\emptyset 6) \cdot (X>\emptyset)$
$+S\emptyset \cdot T7 \cdot Y(\emptyset)'$	$(MC=\emptyset 7) \cdot (Y\geq\emptyset)$
$+S1 \cdot T4 \cdot (X(6) \cdot X(7) \cdot \ldots \cdot X(23))$	$(MS=14) \cdot (X(6)-X(23)= 777777B)$
$+S1 \cdot T5 \cdot (X(6)'+X(7)'+\ldots+X(23)')$	$(MC=15) \cdot (X(6)+X(23)\neq 777777B)$
$+S1 \cdot T6 \cdot Z(\emptyset)'$	$(MS=16) \cdot (Z\geq\emptyset)$
$+S1 \cdot T7 \cdot Z(\emptyset)$	$(MC=17) \cdot (Z<\emptyset)$
$+S2 \cdot T\emptyset$	$MC=2\emptyset$ (will always branch)
$+S2 \cdot T1 \cdot (Y(21)+Y(22)+Y(23))$	$(MC=21) \cdot Y \wedge 7 \neq \emptyset)$
$+S2 \cdot T2 \cdot (BLO' \cdot BL1' \cdot \ldots \cdot BL23')$	$(MC=22) \cdot (BL=\emptyset)$
$+S2 \cdot T3 \cdot (BL\emptyset+BL1+\ldots+BL23)$	$(MS=23) \cdot (BL\neq\emptyset)$
$+S2 \cdot T4 \cdot Y(23)'$	$(MC=24) \cdot (Y(23)=\emptyset)$
$+S2 \cdot T5 \cdot Y(23)$	$(MC=25) \cdot (Y(23)\neq\emptyset)$
$+S2 \cdot T6 \cdot ATLAT1$	$(MC=26) \cdot$ (Attention latch $1=\emptyset$)

Table 5 . BRANCH Logic. (cont'd)

BRANCH=	DESCRIPTION
+S2·T7·RS1'·RS2'	(MC=27)·(RSLAT1=$\emptyset$)·RSLAT2=$\emptyset$)
+S3·T$\emptyset$·PNEX	(MC=3$\emptyset$)·(Protect$\neq$X)
+S3·T1·RS2'	(MC=31)·(RSLAT2=$\emptyset$)
+S3·T2·SPAG'	(MC=32)·(Special flag A=$\emptyset$)
+S3·T3·SPAG	(MC=33)·(Special flag A=$\emptyset$)
+S3·T4·ATLAT2'	(MC=34)·(Attention latch2=$\emptyset$)'
+S3·T5·ATLAT3'	(MC=35)·(Attention latch3=$\emptyset$)'
+S3·T6·ATLAT1	(MC=36)·(Attention latch1$\neq\emptyset$)'
+S4·T$\emptyset$·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$\neq\emptyset$)

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'·K2+

...; sATLAT2=ATLAT2'·K2; sATLAT3=SATLA3'·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·I2CLOCK; sADV2=ADV1·

I2CLOCK). When the ADVANCE switch is pressed, ADV1

is reset (rADV1=ADVNC'·I2CLOCK). One instruction cycle

later, attention latch 1 is set (sATLAT1=ADV1'·ADV2·

ADVNO·I2CLOCK), and ADV2 is reset (rADV2=ADV1'·I2CLOCK).

ADV1 remains reset until the switch is released. (sADV1=

ADVNC·I2CLOCK), and ADV2 remains reset for one instruction

cycle longer (sADV2=ADV1·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 set-

ting or holding the attention latches (rATLAT1=ADV1·

SATLA1·I2CLOCK·(S2·T6)+(S3·T6)+ADV2'·SATLA1·I2CLOCK·

[(S2·T6)+(S3·T6)]+ADVNO·SATLA·I2CLOCK·[(S2·T6)+(S3·T6)];

rATLAT2=SATLA2·S3·T4·K2; rATLAT3=SATLA3·S3·T5·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 $\emptyset$, 1, and 2 in the MS field,
and is decoded to generate U$\emptyset$ thru U7. The second
group is comprised of bits 3, 4, and 5 in the MS field,
and is decoded to generate to V$\emptyset$ thru V7. The logical
product of thrity eight of the sixty four possible
combinations of U$\emptyset$ thru U7, and V$\emptyset$ thru V7 are used
to generate the control terms. The remaining combina-
tions 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 the data from the left bool box to the X-bus... (See Paragraph (3A2 .)	...1 place.	LCY1B = U$\emptyset \cdot$ V1	MS=$\emptyset$1
	...2 places.	LCY2B = U$\emptyset \cdot$ V2	MS=$\emptyset$2
	...3 places.	LCY3B = U$\emptyset \cdot$ V3	MS=$\emptyset$3
	...4 places.	LCY4B = U$\emptyset \cdot$ V4	MS=$\emptyset$4
	...8 places.	LCY8B = U$\emptyset \cdot$ V5	MS=$\emptyset$5
	...12 places.	LCY12B = U$\emptyset \cdot$ V6	MS=$\emptyset$6
	...16 places.	LCY16B = U$\emptyset \cdot$ V7	MS=$\emptyset$7

Table 6 . Special Functions Logic (cont'd)

Function	Equation	Definition of Terms
Left cycle the data from the left bool box to the X-bus (See Paragraph 3A2 .) { ...20 places.	$LCY20B = U1 \cdot V\emptyset$	MC=10
...as specified by Z22 and Z23	$CCFZA = U1 \cdot V1$	MS=11
...as specified by Z19, Z20, and Z21	$CCFZB = U1 \cdot V2$	MS=12
scratchpad address from Z (See Paragraph 3J .)	$SPFZ = U1 \cdot V3$	MS=13
	$sALERT = U1 \cdot V4 \cdot I2CLOCK$	MS=14
	$rALERT = U1 \cdot K2 + V4' \cdot K2$	$MS \neq 14$
Set Parallel Output (See Paragraph 3F1.)	$sPOT\ U1 \cdot V5 \cdot I2CLOCK$	MS=15
Reset Parallel Output (See Paragraph 3F1.)	$rPOT = U1' \cdot K2 + V5' \cdot K2$	$MS \neq 15$
Set Parallel Input (See Paragraph 3F1.)	$sPIN = I2CLOCK \cdot U1 \cdot V6$	MS=16
Reset Parallel Input (See Paragraph 3F1.)	$rPIN = K2 \cdot U1' + K2 \cdot V6'$	$MS \neq 16$
Request Stobe 1	$RSOUT\ 1 = U1 \cdot V7 \cdot BRTEST$	(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.)	$UPXn = U2 \cdot V\emptyset \cdot Xn$ (n = 20, 21, 22, or 23)	(MS=2$\emptyset$)
Clock memory interface to load priority latches for central memory requests	$LPF = U2 \cdot V2$	MS=22
Reset request strobe latch 1	$rRS1 = I2CLOCK \cdot U2 \cdot V3$	MS=23
Reset Central Memory request	$MI\ RESET = U2 \cdot V4$ $+ MR\emptyset$ $+ MR1$	MS=24
Set protect as specified X-bus mask (See Paragraph 3F2.)	$PXn = Us \cdot V5 \cdot Xn$ (n = 20, 21, 22, or 23)	(MS=25)
Reset used by TSU	$RESETTU = U2 \cdot V6$	MS=26
Set special flag	$sSPAG = U3 \cdot V\emptyset \cdot I2CLOCK$	MS=3$\emptyset$
Reset special flag	$rSPAG = U3 \cdot V1 \cdot I2CLOCK$	MS=31
Reset request strobe latch 2	$rRS2 = I2CLOCK \cdot U3 \cdot V2$	MS=32
Request strobe 2	$RSOUT2 = U3 \cdot V3 \cdot BRTEST$	(MS=33) (interval in which the branch condition is tested)
Undefined	$SMS\ 1 = U3 \cdot V4$	MS=34
Release Prestore Store 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 3Ll.)	$rBSELECT = K2 \cdot U6 \cdot V\emptyset \cdot TBO$ $+ Ks \cdot BSS' \cdot TOSO$ $+ K2 \cdot X13' \cdot TXO$	(MS=60)
Select ROM bank B (See Paragraph 3Ll.)	$sBSELECT = K2 \cdot U6 \cdot V1 \cdot TBO$ $+ K2 \cdot BSS \cdot TOSO$ $+ K2 \cdot X13 \cdot TXO$	(MS=61) (Contents of B field transferred to O 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 com-
municate with external devices. (See Paragraph 3N for a
detailed description of the I/O functions). When the micro-
processor 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 de-
vice wishes to send data to a microprocessor (PIN), the

devices send an attention signal, SATLAn, to the micro-
processor. This signal is latched in the microprocessor
and is tested by a branch condition ATLAn=1. The micro-
processor will, via programmed instructions, read a regis-
ter in the external device by sending an alert to the
device. The device will set up a path between the se-
lected retister and the E1-bus or E2-bus. The micro-
processor will then transfer the E1-bus or E2-bus to the
Y-bus; and send a PIN strobe, via programmed instruc-
tion, 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 instruc-
tion 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 re-
solves 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 micropro-

cessor wishes to unprotect, it gates a 4-bit mask to the

X-bus least significant bits, and sends an unprotect sig-

nal 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 micropro-

cessors 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 micro-

processors are to be strobed. It is legal for a micro-

processor to strobe itself. (See Paragraph 3L for a

description of the request strobe.)

G.   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 ($sMn = Xn \cdot LMX \cdot MCLOCK3 + Yn \cdot LMY \cdot MCLOCK3 + \ldots$). (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 ($sMn=...+M1n \cdot LMM1 \cdot MCLOCK + M2n \cdot LMM2 \cdot MCLOCK$). (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 BL$\emptyset$ thru BL3 providing the logical com-
bination of the logical products of M, Q, M', and Q'
(BLn'+BL$\emptyset$·Mn·Qn+BL1'·Mn·Qn'+BL2'Mn'·Qn+BL3'·Mn'·Qn'.
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 out-
put 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=$\emptyset$ thru 23;

And for the left bool box: a-L, b=M

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

Ba=	DEFINITION
Ba$\emptyset$·Ba1'·Ba2'·Ba3'·bn·Qn	(Ba=$\emptyset\emptyset$)·(6 Q)=1
+Ba$\emptyset$'·Ba1'·Ba2'·Ba3·(bn·Qn+bn'·Qn')	(Ba=$\emptyset$1)·(b=Q)=1

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

Ba=	DEFINITION
+Ba0'•Ba1'•Ba2•Ba3'•Qn	(Ba=02)•Q =1
+Ba0'•Ba1'•Ba2•Ba3•(bn'+Qn)	(Ba=03)•(b'+Q)=1
+Ba0'•Ba1•Ba2'•Ba3'•bn	(Ba=04) =1
+Ba0'•Ba1•Ba2'•Ba3•(bn+Qn')	(Ba=05)•(b+Q')=1
+Ba0'•Ba1•Ba2•Ba3'•(bn+Qn)	(Ba=06)•(b+Q)=1
+Ba0'•Ba1•Ba2•Ba3•(bn+bn'+Qn+Qn')	(Ba=07) =1
+Ba0•Ba1'•Ba2'•Ba3'•(bn•bn'•Qn•Qn')	(Ba=10) =0
+Ba0•Ba1'•Ba2'•Ba3'•bn'•Qn'	(Ba=11)•(b'•Q')=1
+Ba0•Ba1'•Ba2•Ba3'•(bn'•Qn)	(Ba=12)•(b'•Q)=1
+Ba0•Ba1'•Ba2•Ba3•bn'	(Ba=13)•b' =1
+Ba0•Ba1•Ba2'•Ba3•bn•Qn'	(Ba=14)•b•Q'=1
+Ba0•Ba1•Ba2'•Ba3•Qn'	(Ba=15)•Q' =1
+Ba0•Ba1•Ba2•Ba3•(bn•Qn'+bn'•Qn)	(Ba=16)•(bn(EOR)Q)=1
+Ba0•Ba1•Ba2•Ba3•(bn'+Qn')	(Ba=17)•(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=$\emptyset$. 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 absorbes 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 haveing no antici-pation, 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 (BLn+BRn=1), or not to propagate a carry (BLn'+BRn'=1),

then the carry-out from the 4-bit section can be de-

termined 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 inputed 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. <u>Anticipated Carry</u>. The anticipatory circuitry

guarantees that a carry out from a group will be stabilized

after the BLn+BRn and BLn'+BRn' 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 BLn' and BRn' at the card input to obtain

a stable carry out from bit $\emptyset$ 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 stata 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$\emptyset$ T1+S2 T$\emptyset$).

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 (LMPAR0OUT=M0 thru M11 contain an even number of high bits; LMPAR1OUT=M12 thru M23 contain an even number of high bits). During a store operation, the parity bits, LMPAR0OUT and LMPAR1OUT, are sent to the local memory with the data. The parity bits to central memory, PAR0OUT and PAR1OUT, 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 M=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 (LMPAR0IN and LMPAR1IN, if the fetch is from local memory; or PAR0IN and PAR1IN, 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.

(sLMPAR0=LMPAR0IN·(MCLOCK1+MCLOCK2);

sLMPAR1=LMPAR1IN·(MCLOCK1+MCLOCK2);

sPAR0=PAR0IN·(MCLOCK1+MCLOCK2);

sPAR1=PAR1IN·(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 (LMPEØ,

LMPE1, CMPEØ or CMPE1) associated with that parity bit

is set one instruction later. LMPARn'·...+LMPARnOUT'·

LMPARn·SLMP·K2; sCMPEn=LMPARnOUT·PARn'·...+LMPARnOUT'·

PARn·SCMP·K2). SLMP is set if the local memory is

clocked (sSLMP=MCLOCK2·K2), and reset at the end of the

next instruction (rSLMP=K2). SCMP is set if the central

memory is clocked (sSCMP=MCLOCK1·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=LMPEØ+LMPE1), and central memory parity error

(CMPE=CMPEØ+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

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

R∅→R∅ (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·BRTEST·TLMP';
rCMPEn=RCMP=K2·BRTEST·TCMP'). The outputs of the parity
error latches are also routed to the system warning
register.

2. <u>Addressing Parity</u>.

The parity checker also generates an odd parity bit

(APAR) for 17 bits of the RØ register, RØ(6) thru RØ(22) inclusive, which comprise the memory address. RØ(6) thru RØ(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 RØ 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 RØ 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·WS). The write strobe is derived from LSPX in the instruction word during state A (sWS= XXB'·XXC'·LSPX·K3; rWS=KI2·STATEA'). The data is read from the selected scratch pad address to the Y-bus by LSPY in the instruction word (Yn=SPn·LSPY).

K. HOLDING REGISTERS. (See Figure 9.)

The holding register cards contain seven 24-bit registers, $R\emptyset$ thru R6; and the gating to the Y-bus. The holding registers are loaded by their respective clocks, $R\emptyset CLOCK$ thru R6CLOCK; and the contents of the X-bus if TWX is set in the instruction word, or Y-bus is TYW is set in the instruction word ($sR\emptyset n = R\emptyset CLOCK \cdot Xn \cdot TXW + R\emptyset CLOCK \cdot Yn \cdot TYW$; $sR1n$ thru $sR6n$, same as $sR\emptyset n$ except that the $R\emptyset CLOCK$ is replaced by the R1CLOCK thru the R6CLOCK respectively). The $R\emptyset CLOCK$ 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 $R\emptyset CLOCK$ which is loaded independent of the LRN field by $LR\emptyset$ 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. ($Yn = \ldots + R\emptyset n \cdot RRN\emptyset ' \cdot RRN1' \cdot RRN2' \cdot THY + \ldots$; for R1 thru R6, $RRN\emptyset$ 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 IHR is set in the instruction word.
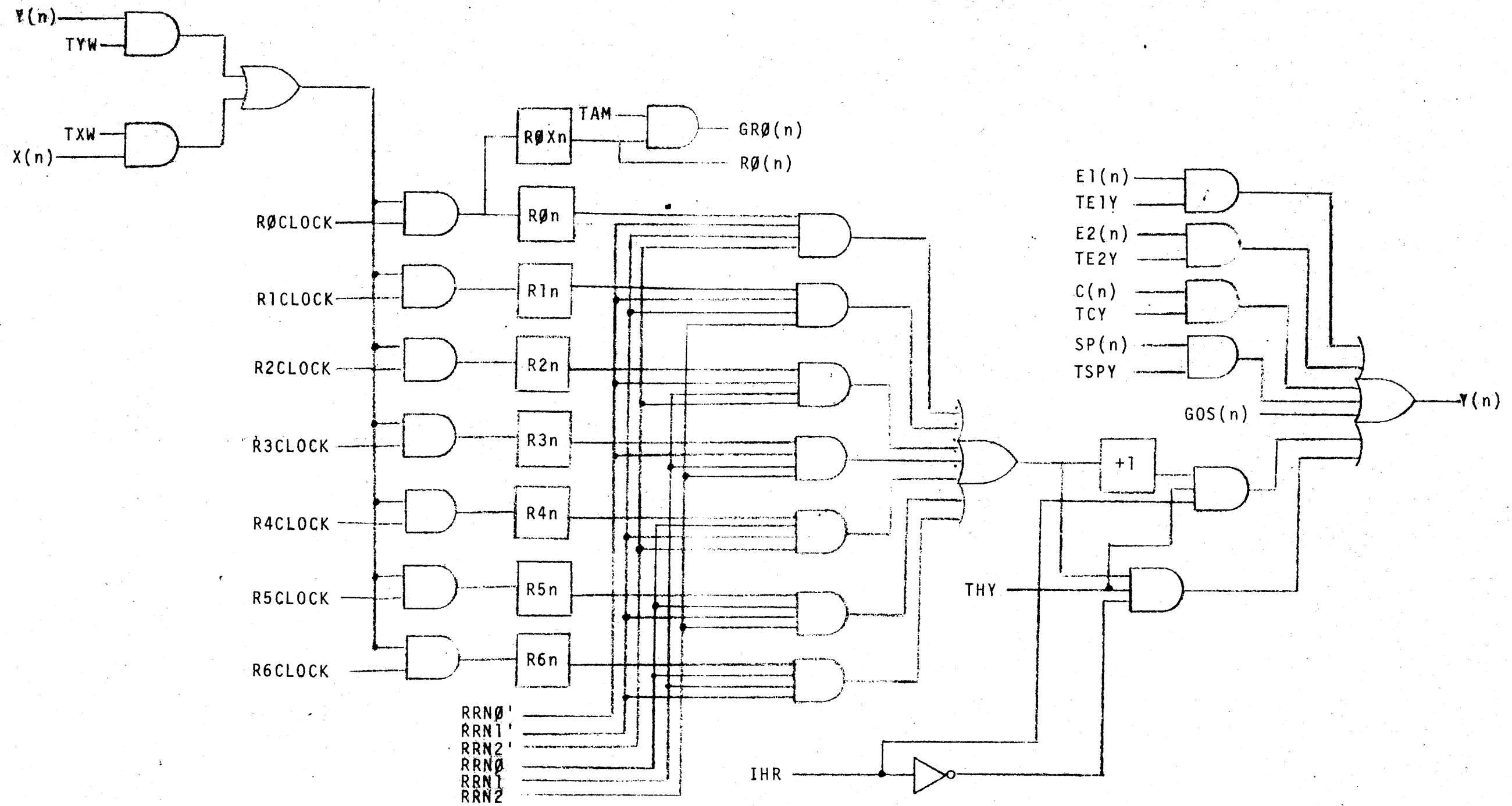
HOLDING REGISTERS, FUNCTIONAL BLOCK DIAGRAM
WHERE: n=0 THRU 23

Figure 9. Holding Registers, Functional Block Diagram.

1. R∅X.

The R∅X register is an additional register, provided on the holding register card that is loaded in parallel to the R∅ register and provides access to the data contained in R∅ by the memory interface via the R∅-bus without the necessity of addressing the R∅ register; the data in R∅ provides the memory address for data transferred between the microprocessor and local or central memory. The data from R∅X is sent to the memory interface independent of the addressing and control associated with R∅ 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, TE1Y, TE2Y, and TCY bits, respectively, in the instruction word ($Yn=...+SPn \cdot TSPY+E1n \cdot TE1Y+E2n \cdot TE2Y+Cn \cdot TCY$).

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 specifie strobe or strobes generated depends on the X-bus bits

that are set. The request strobes from the micropro-
cessor are RS1($\emptyset$) thru RS1(7) and RS2($\emptyset$) 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 cor-
responding X-bus bit (sRS1($\emptyset$)=RSOUT1·X(16); sRS1(1)=
RSOUT1·X(17); ...; sRS1(7)=RSOUT2·X(23)). The re-
quest 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·(K2A+K2); rSTOPA=RESET'·(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

memory and the specific address is specified by the contents of the $R\emptyset$ register. Start memory interface (SMI) is generated when the special functions field of the instruction contains forty through forty-seven (SMIA= $MS\emptyset \cdot MS1' \cdot SM2' + \ldots$), or a sixty-four or sixty-five (SMIA=$\ldots$+ODD; ODD=$U6 \cdot MS3 \cdot MS4'$). The special functions field of the instruction sets the priority latches (CAP$\emptyset$, CAP1, PP) of the request to central memory when a load priority field LPF (MS=22) is specified (LPF= $U2 \cdot V2$). The central and private memories share the request mode latches; F, S, and H. The request mode latches are used by the memory interface logic to establish whether the request is a store (S), a fetch (F), or a hold (H). The F and S terms are used by both the central and private memory. Additionally, combinations of S, F, and H determine fetch and hold, prefetch, store and hold, prestore, and release. The H term is used by the central memory only. A 3-position switch, ST$\emptyset\emptyset\emptyset$3, on the memory interface card enables the selection of the address reference. The setting of the switch determines whether all references are to private memory (PMREF), all references are to central memory (CMREF), or if the memory reference is specified by $R\emptyset(4)$ of the address word.

$$CMREF = R\emptyset' \cdot LR\emptyset'$$
$$+ X4' \cdot LR\emptyset \cdot TXW$$

ST00003 S1B Position A

```
     + Y4'·LRØ·TYW

     + Ø                  STØØØØ3   S1B   Position B

     + 1                  STØØØØ3   S1B   Position C

PMREF= RØ4·LRØ'        ⎫
     + X4·LRØ·TXW       ⎬  STØØØØ3   S1A   Position A
     + Y4·LRØ·TYW      ⎭

     + 1                  STØØØØ3   S1A   Position B

     + Ø                  STØØØØ3   S1A   Position C
```

The memory address is specified by the contents of RØ(6) thru RØ(23). The microprocessor control logic is interlocked by a stop clock signal, SC, from the memory interface. The memory interface runs asynchronous to the microprocessor instruction flow. Some instructions will, if executed while the memory is in operation, cause the loss of data. The microprocessor is allowed to run after a memory request if initiated until an instruction that would interfere with the transfer of data is encountered, the control logic is inhibited by SC until the memory operation is complete. The SC signal is generated during state A of an instruction cycle if:

1. An instruction attempts to restart the memory interface while it is still running (SC= (LMRUN·FDONEd'+CMRUN)·SMIA·STATEA+...).

2. An attempt to read the M register while a fetch is in progress, before the memory has transferred the data to M (SC=(LMRUN·FDONEd'+CMRUN)· (F·S')·BL≠2,7,10,15)·STATEA+...).

3. An attempt to load the M register during a
store or fetch before the memory has trans-
ferred data to or from M(SC=(LMRUN•FDONEd+
CMRUN)•(LMX+LMY)•STATEA+...).

4. An attempt to load RØ (the address register)
before the memory has finished with the address
(SC=(LMRUN•FDONEd+CMRUN)•LRØ•STATEA+...).

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 instruc-
tion 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'•SMIA•SF•K2; rF=SC'•SMIA•SF'•K2) (SF=MS3; SMIA=(SMIA')'+
MSØ•MS1'•MS2'+ODD; ODD=MS3•MS4'•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'•SMIA•SS•K2; rS=SC'•
SMIA•SS'•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'•SMIA•SH•K2; rH=SC'•SMIA•SH'•K2;
SH= MS5). The priority latches provide direct outputs
to the central memory for the determination of the pri-
ority of a memory request. The priority latches (CAPØ,
CAP1, PP) are set when the special function field of

the instruction contains an LPF, MS=22, and the corres-
ponding bit on the X-bus(X(21), X(22), X(23); respect-
ively) is set (sCAPØ=X21·PFL; rCAPØ=X21'·PFL; sCAP1=
X22·PFL; rCAP1=X22'·PFL; sPP=X23·PFL; rPP=X23'·PFL).

2. The Central Memory Request Section.

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

a. Requests To Central Memory Via the MPMBM. A
request to memory generates a start memory interface sig-
nal (SMI), if the memory interface is in an idle state,
state Ø (State Ø=MPREQ'·FCY'·SCY'). SMI sets the
MPREQ flip-flop, if the reference is to central
memory, during the end cycle of the current instruction
(sMPREQ=CMREF·SC'·SMIA·RCE·K2·NOM'). MPREQ set is state
1 of the memory interface cycle (STATE 1=MPREQ·FCY'·SCY');
during state 1 the MPMBM examines the request field of
the microprocessors. The MPMBM latches the highest pri-
ority request, and sends a signal to the microprocessor
selected (OK). The signal OK resets MPREQ (rMPREQ=OK·K2)

VIA MPMBM
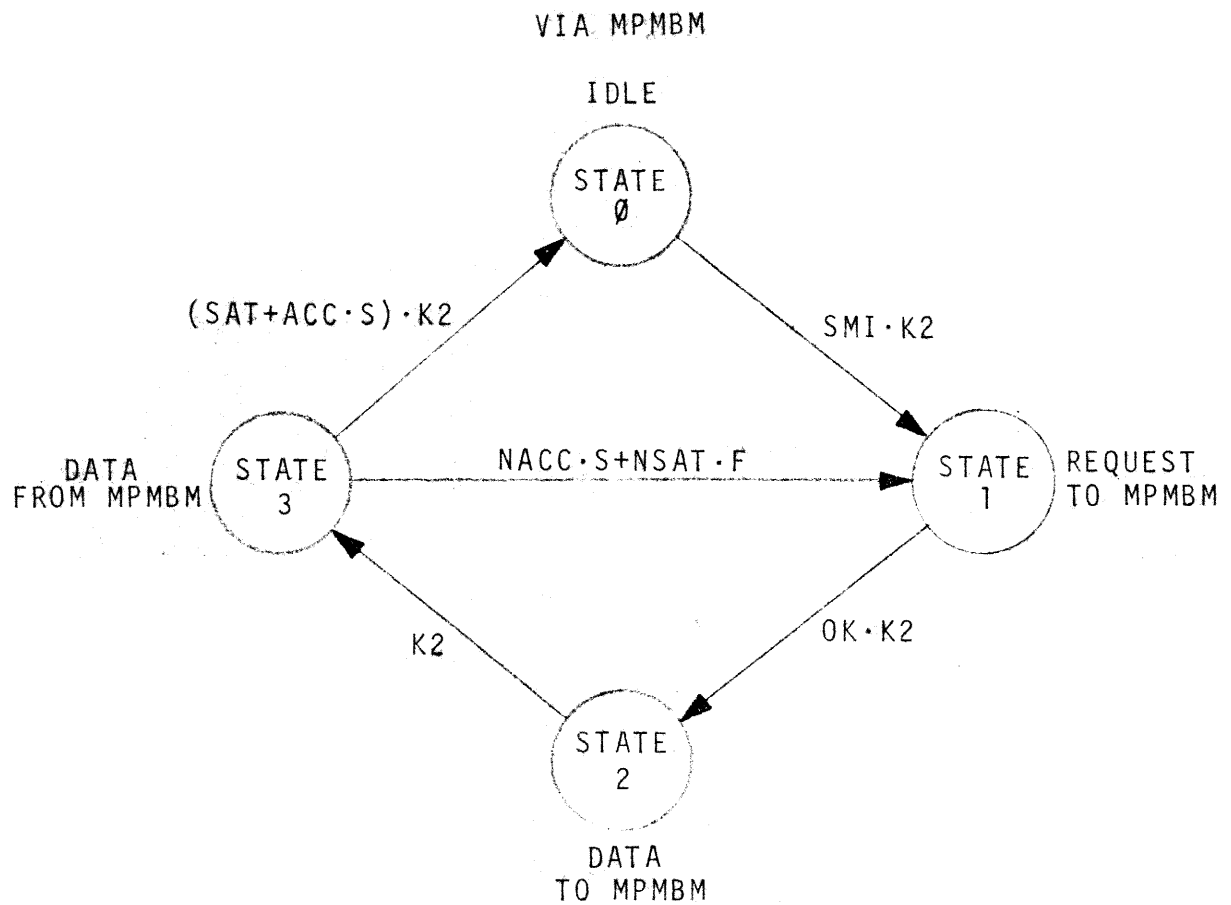
DIRECT TO CENTRAL MEMORY
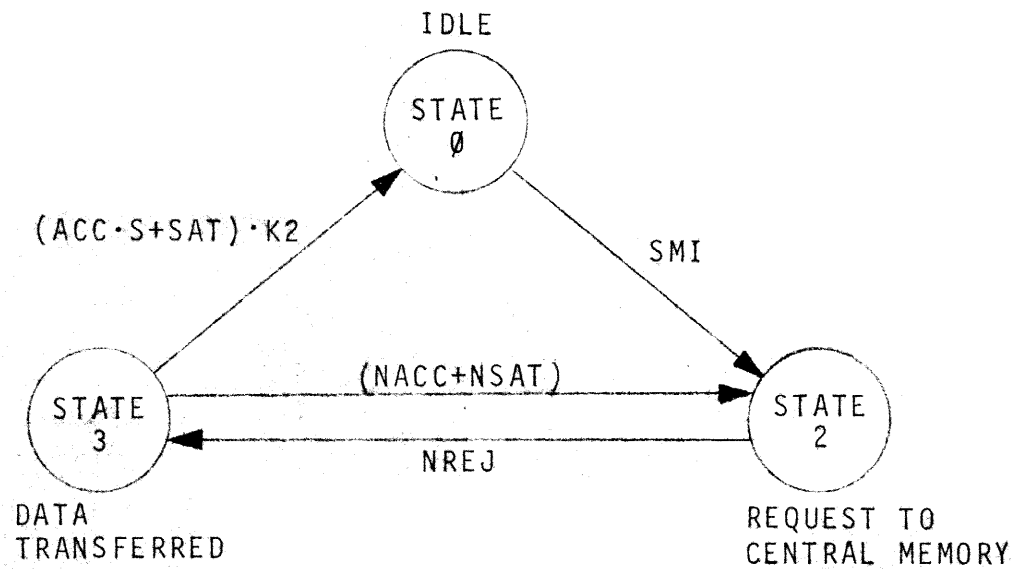
Figure 10. State Counter, Memory Interface.

and sets FCY (sFCY=OK·K2) on the selected microprocessor;

the memory interface of the microprocessor selected

goes to state 2 (STATE 2=MPREQ'·FCY·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 micro-

processor which returns the memory interface to state

1 (sMPREQ=NREJ'·NOM'·FCYd·K2; rFCY=FCYd·NOM'·K2). If

the request is not rejected (NREJ) during state 2, the

microprocessor enters state 3 (STATE 3=MPREQ'·FCY'·SCY;

rFCY=FCYd·NOM'·K2; sSCY=FCYd·NREJ·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'·NACC·SCYd·K2); SCY is reset by K2 (rSCY=SCYd·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 Ø

(rSCY=SCYd·K2). If during state 3 the data is not fetched,

the central memory sends back a signal request not satis-

fied NSAT which sets MPREQ (sMPREQ=NOM'·NSAT·F·S'·SCYd·K2),

SCY is reset by K2 (rSCY=SCYd·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 Ø (rSCY=SCYd·K2).

b. <u>Requests Direct To Central Memory.</u> 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=NOM·CMREF·SMI·RCE·SC'·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 (sSCY=FCYd·NREJ·K2; rFCY=FCYd·
NOM·REJ·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 (sFCY=SCYd·NOM·NACC·K2·F·S'·NOM·
SCYd·NSAT·K2; rSCY=SCYd·K2); otherwise the memory
interface goes to the idle state, state $\emptyset$ (rSCY=SCYd·K2).

c. <u>Priority Requests, Clocks, and Load Signals.</u>
Other signals generated by the central memory request
section are:

HIREQ = PP·MPREQ

LOREQ = PP'·MPREQ

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

LMM1 = (F·S')·SCYd

MCLOCK = (F·S')·SCYd·K2

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

GAM = SMI·CMREF·NOM·RCE·SC'·K2

+SCYd·NOM·NACC·K2   gate address to
                                    Memory

+(F·S')·NOM·NSAT·SCYc·K2

+OK·K2

+FCY

GDM=FCYd·NREJ·K2   gate data to Memory

+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 micropro-

cessor 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;

      (SETSTART P = SF·SS'·PMREF·ODDWORD'·SMIA·RCE·SC'·K2

                  + SF'·SS·PMREF·ODDWORD'·SMIA·RCE·SC'·K2).

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:

$$SUH = S \cdot F' \cdot R\emptyset(23) \cdot SETSTARTPd \quad \text{(delayed by 100nsec)}$$

$$SLH = S \cdot f' \cdot R\emptyset(23) \cdot SETSTARTPd \quad \text{(delayed by 100nsec)}$$

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·FDONE; 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'). FDONE is set 80 nanoseconds later (assuming 100 nanoseconds clock period) and signifies that M and R$\emptyset$ may be modified (sFDONE = DBC·K2; rFDONE = 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.

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=$\emptyset$, 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 condi-
tions are the same as described above. The specifi-
cation 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 out-
put to a device is to execute:

         DEVICE ADDRESS —▷ Z ALERT;

         DATA —▷ Z POT;

This instruction sequence results in the following
signal sequence:
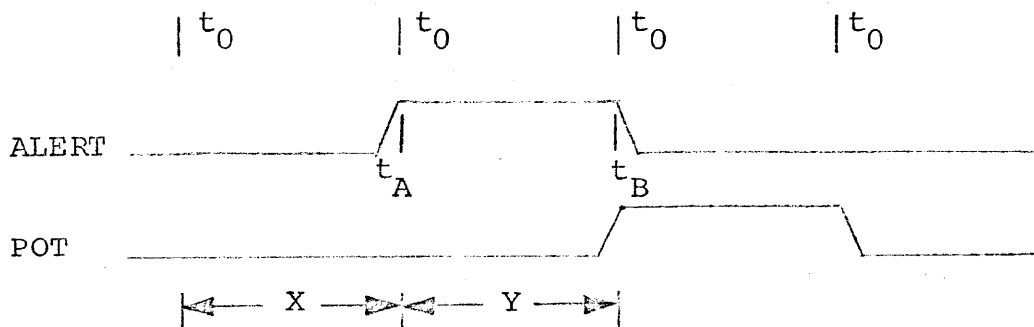
t_0 is the system reference time . It is approximately coincidental with the fall of K2 at the microprocessor. The first instruction above is executed during the interval designated 'X'. The second is executed during 'Y'. The Z register is loaded with the device address at t_A, and with the output data 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:
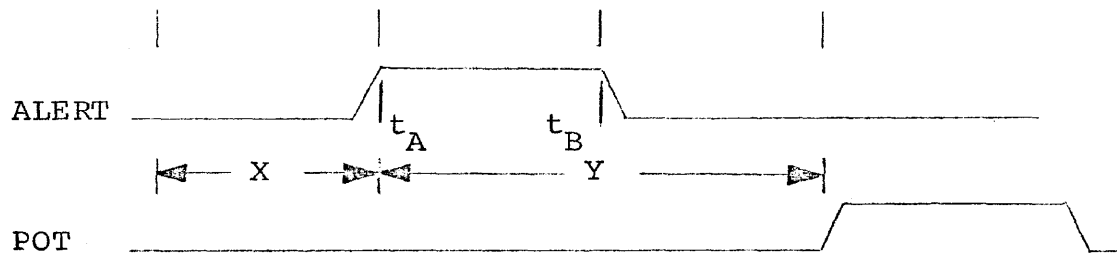
The general rule on timing of the ALERT, POT, and PIN strobes is that they rise at the end of the instruction during which the special condition is executed, and they fall at the end of the first interval during which the special condition does not exist. Thus it is possible to generate arbitrarily long strobes, but this should be avoided.

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

DEVICE ADDRESS ⟶ Z, ALERT

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

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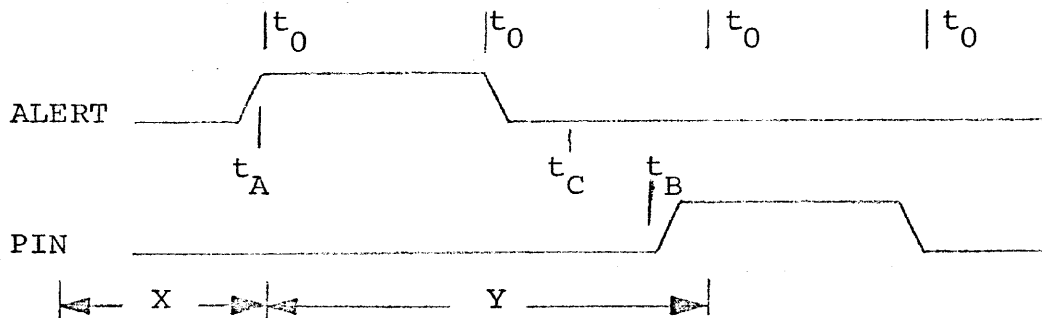
The first instruction is executed during the interval 'X', the second during 'Y'. The device address is loaded into Z at t_A, and the microprocessor loads the data into one of its internal registers at t_B. The external device must have the data stable on the E2-bus no later than $t_C = t_0 + 42$ nsec.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

DEVICE ADDRESS ⟶ 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 $\longrightarrow$ Z;

       X: ALERT;

       Y: DATA $\longrightarrow$ Z;

       Z: POT;

which results in:
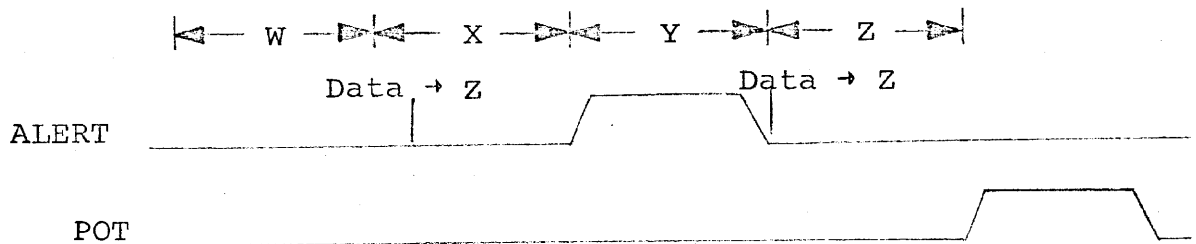
2. INPUT

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

       X: ALERT;

       Y: VCY, NOP

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

which results in:
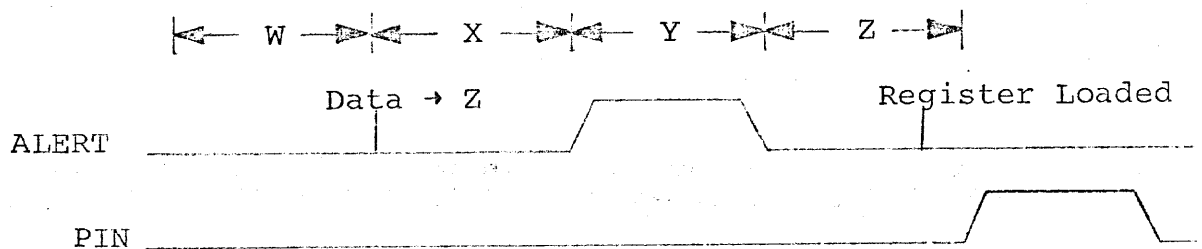
APPENDIX A

Table A1. 90-bit Microinstruction Word.

Signal	Position	Clock	Function
MCØ-MC5	0-5	K2	Branch Condition field (6 bits). (See Table A2).
MCONTØ,1	6,7	K3	Branch control field (2 bits):
			Ø = branch conditionally to the address specified by the contents BØ thru B9.
			1 = branch conditionally to the address specified by the contents of BØ thru B9. 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 X-bus ten lsb.

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

Signal	Position	Clock	Function
BØ, 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		
CØ, C1, C2	18, 19, 20	K3	Constant Field (24-bits).
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.

Table Al. 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.
SSPØ-5	51-56	K2	Selects the contents of scratchpad address $\emptyset-77_{(8)}$
TOSY	57	K3	Transfers the contents of the OS register to Y-bus (bits 14-23)
LRO	58	K3	Load holding register RØ.
LSPX	59	K3	Loads the selected scratchpad address from the X-bus.
MSØ-MS5	60-65	K2	Special functions field. (See Table A3.)

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

Signal	Position	Clock	Function
RRNØ-2	66-68	K2	Holding register select read field (enables selected holding register, RØ thru R6, output).
LRNØ-2	69-71	K3	Holding register select load field (clocks selected holding register, R1 thru R6, 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.
BLØ-BL3	78-81	K2	Left bool box control field (See Table A4).

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

Signal	Position	Clock	Function
BRØ-BR3	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$\emptyset$-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$7	Y$\geq\emptyset$
1$\emptyset$	Y<$\emptyset$
11	R$\emptyset$<$\emptyset$
12	R$\emptyset\geq\emptyset$
13	X$\leq\emptyset$
14	X'$\wedge$777777B=$\emptyset$, (X(6)-X(23)=777777B)
15	X'$\wedge$777777B$\neq\emptyset$, (X(6)-X(23)$\neq$777777B)
16	Z$\geq\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 1=$\emptyset$ △

Table A2. Branch Conditions.(cont'd)

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

⚠ Resets latch.

⚠ 46 thru 77 not decoded

Table A3.   Special Functions.

MS∅-MS5	Function
∅∅	No activity
∅1	LCY1
∅2	LCY2
∅3	LCY3
∅4	LCY4
∅5	LCY8
∅6	LCY12
∅7	LCY16
1∅	LCY2∅
11	LCL Z (CCFZA)
12	LCH Z (CCFZB)
13	SKZ (SPFZ)
14	ALERT
15	POT
16	PIN
17	Request Strobe #1
2∅	Unprotect
21	Unusable
22	LPF
23	Reset Request Strobe Latch #1 ⚠
24	Reset Central Memory Request ⚠
25	Request Protect
26	Reset T.U.
3∅	Set special flag A

Table A3.   Special Functions.(cont'd)

MS∅-MS5	Function
31	Reset special flag A
32	Reset Request Strobe Latch #2
33	Request Strobe #2
34	Undefined
4∅	Release △3
41	Prestore △3
42	Store △3
43	Store & Hold △3
44	Fetch △3
45	Fetch & Hold △3
47	Prefetch △3
6∅	Set Bank B
61	Set Bank A
62	Clear all CPU Maps
64	Fetch △4
65	Fetch & Hold △4
	△5

△1   Occurs at end of instruction

△2   Local and Central Memory

△3   Memory Reference

△4   ODDWORD FETCH

△5   Special functions 27, 35, 36, 37, 46, 63, and 66 thru

77 are not decoded

Table A4. Bool Box Control.

BL∅-BL3	Left Bool Box Output
∅∅	$M \cdot Q$
∅1	$M = Q$
∅2	Q
∅3	$\overline{M}+Q$
∅4	M
∅5	$\overline{M+Q}$
∅6	$M+Q$
∅7	1
1∅	$\emptyset$
11	$\overline{M} \cdot \overline{Q}$
12	$\overline{M} \cdot Q$
13	$\overline{M}$
14	$M \cdot \overline{Q}$
15	$\overline{Q}$
16	$M(EOR)Q$
17	$\overline{M}+\overline{Q}$

BR∅-BR3	Right Bool Box Output
∅∅	$Z \cdot Q$
∅1	$Z = Q$
∅2	Q
∅3	$\overline{Z}+Q$
∅4	Z
∅5	$\overline{Z+Q}$
∅6	$Z+Q$
∅7	1
1∅	$\emptyset$
11	$\overline{Z} \cdot \overline{Q}$
12	$\overline{Z} \cdot Q$
13	$\overline{Z}$
14	$Z \cdot \overline{Q}$
15	$\overline{Q}$
16	$Z(EOR)Q$
17	$\overline{Z}+\overline{Q}$

Table A5.   Summary of Signal Mnemonics

Signal Mnemonic	Page Reference	Signal Mnemonic	Page Reference
A		**B (cont'd)**	
A bank	34	BSELECT	29
ADV1	45	BSS	37
ADV2	45	**C**	
ADVNC	45	C(n)	56
ADVNO	45	CAPØ	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	CPØ	68
B-field	2	CP1	68
BL-field	52	CSØ-CS7	37
BL(n)	52	CS1Ø-CS17	37
BLØ-BL3	54	**D**	
BP	44	DBC	77
BR(n)	52	DGO	25
BR-field	52	**E**	
BRØ-BR3	54		
BRANCH	25	**F**	
BRTEST	33	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
FDONEd	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	LMM1	52
I		LMM2	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	LPF	34
K2A	31	LQX	15
K2X	25	LQY	30
K3	25	LRN	30

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

Signal Mnemonic	Page Reference	Signal Mnemonic	Page Reference
LRNØ	30	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	NREJ	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	68
MS-field	43	PNEX	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		SØ-S7	42
RØ-R6	63	SATLA1-SATLA3	44
RØX	66	SB1-SB3	44
RØCLOCK-R6	30	SC	69
RAØn	34	SCMP	60
RA1n	5	SCY	71
RBØn	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
RRN-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
SPFZ	47	TWX	63
SS	76	TXO	31
SSP	61	TXW	68
STØØØ3	68	TYW	63
Start P	76	**U**	
State A	25	UØ-U7	
State B	25	UK1	31
State C	25	UA	76
STOPA	67	UPX	48
SUH	76	**V**	
T		VØ-V7	46
TØ-T7	42	VCY	25
TRO	31	**W**	
TCMP	61	WS	33
TCY	66	**X**	
TE1Y	66	X(n)	41
TE2Y	66	XSØ-XS7	37
THY	63	XXB	25
TINCO	31	XXC	25
TLMP	61	**Y**	
TOSO	31	Y(n)	66
TSPY	66	YSØ-YS7	37
		Z	
		Z(n)	53