PRELIMINARY INFORMATION PACKAGE

PORT DESCRIPTION

12 November, 1982

CONTENTS

1.0 PORT HARDWARE OVERVIEW

The KC10's I/O architecture assumes that the I/O subsystem is intelligent. The CPU does not handle any of the real-time aspects of doing input or output. Instead, it manipulates queues of commands in memory. The ports read the command queues and perform the operations.

Each of the KC10's ports can be viewed as a special purpose computer. The purpose of each port is to control the physical and data link layers of its interconnect with minimum CPU overhead. The ports process the commands in the queues, relieving the CPU of the details and real time interaction. The ports also handle received unsolicited messages from devices or systems on the interconnects and place the data into queues where the CPU can manipulate it.

Each port is independent and competes with the other ports and the console for use of the TTL I/O BUS. Eight ports can be connected to the TTL I/O BUS in a KC10 mainframe.

A KC10 port can have either three or four modules. The Port Data Mover and Port Processor modules are common to both port types. The NI port's adapter is a single module. The CI port has a Packet Buffer module and a Link module in addition to the processor and data mover.

THIS SECTION WILL EXPAND SOMEWHAT AT A LATER DATE

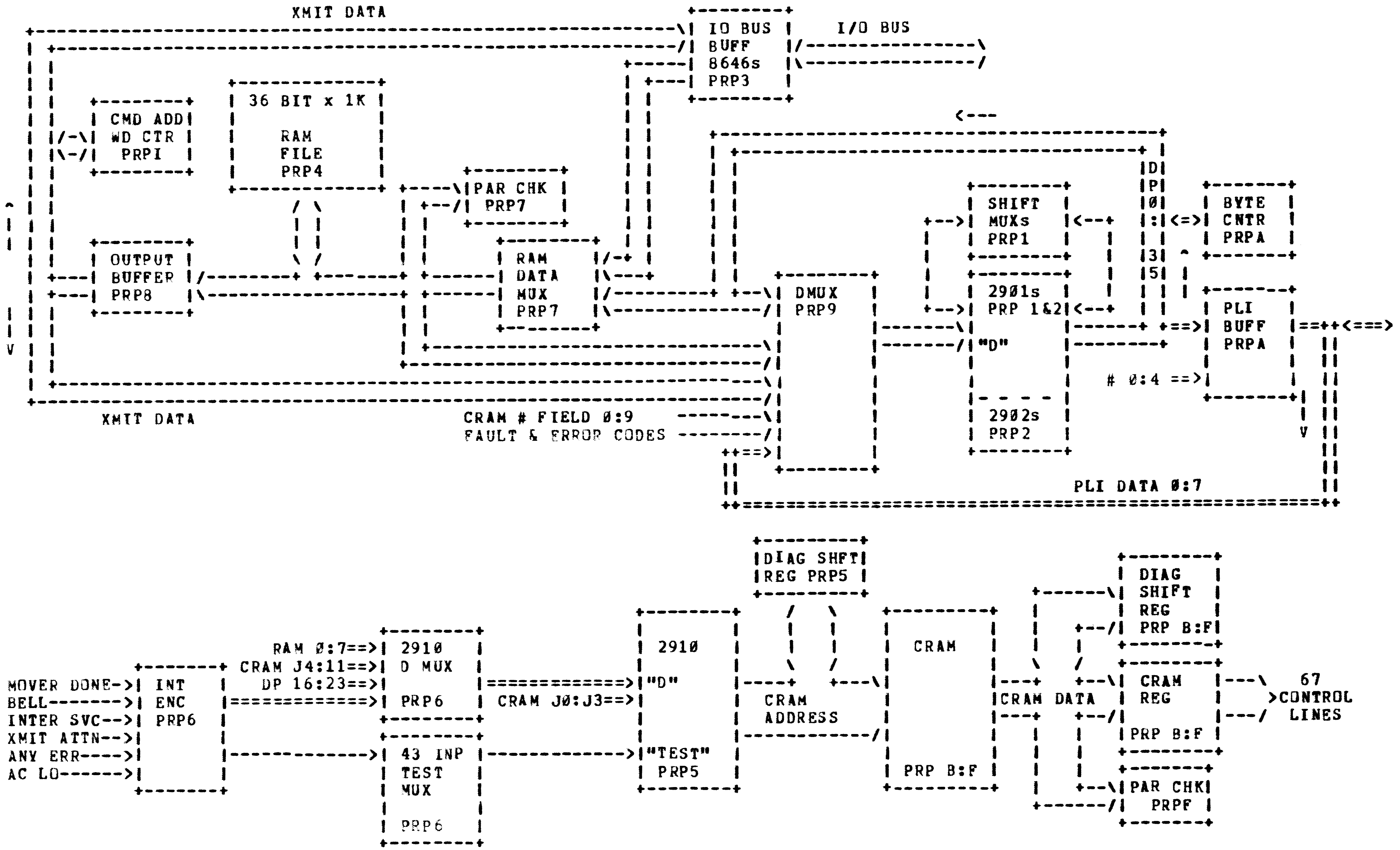
2.0 PORT PROCESSOR DESCRIPTION

The port processor module is a small computer that takes its instructions from command queues in the KC10's main memory. The processor is microcode controlled and has 1K x 36 bits of memory. The processor connects to the I/O bus, and the PLI. It also has several backplane connections to the data mover.

The port processor uses 2901s in its CPU and a 2910 for microprogram sequencing. CRAM (Control Random Access Memory) has 4K words of 68 bits.

2.1 FUNCTIONAL DESCRIPTION

Figure PRP1 is a block diagram of the port processor module. The block diagram will be used, along with other figures and tables, to describe the hardware and its operation. This description will not cover the specific microcode used for each of the adapters. The overall operation of both the CI and NI ports, including microcode, will be covered in later sections.



Port Processor Block Diagram

MRx001

FIGURE PRP1

2.1.1 BLOCK DIAGRAM DESCRIPTION

This section gives an overall description of the port processor module on the block diagram and basic functional levels.

2.1.1.1 I/O BUS DATA PATH

The I/O bus connects to the I/O bus buffer which is made up of 8646s. Receive data from the 8646s goes to the RAM DATA MUX and then into the RAM FILE. Transmit data comes from the RAM FILE through the OUTPUT BUFFER to the 8646s. The 8646s generate and check transmit and receive parity. The actual memory read and memory write functions are discussed later.

2.1.1.2 PLI DATA PATH

PLI output data from the processor board is held in the PLI BUFFER. The PLI BUFFER is fed from processor data path bits 24:31 for data and CRAM number field bits Ø:3 for control. Parity is developed on data path bits 24:31 and is transmitted with the data.

PLI input data is fed directly to the DMUX (Data Bus MUltipleXer) bits Ø:7. DMUX inputs 8:31 come from data path bits Ø:23 when the PLI inputs are selected. This allows successive PLI bytes to be assembled into the most significant 32 bits of the 36-bit word. The DMUX feeds the "D" inputs of the 29Ø1s. Parity is checked on the PLI data lines. PLI parity error is available as a test condition for the 291Ø.

2.1.1.3 29Ø1 INPUTS AND OUTPUTS

The DMUX selects one of four inputs to be sent to the 29Ø1's "D" inputs. See Figure PRP2 for DMUX outputs. The "Y" outputs of the 29Ø1s feed Data Path bits Ø:35 (DP Ø:35). The shift paths for the 29Ø1's RAM shifter and Q REG shifter are controlled by the SHIFT MUXs.

```
MRxØØ2                               29Ø1 D INPUTS FROM DBMUX

    +--------------------+---------+---------+---------+-------+-----------------------------+
    !    NUMBER FIELD    ! # FIELD !PRP FAULT!PRM FAULT!PRM ERR!        NUMBER FIELD         !  DBMUX SEL = Ø
    !       Ø ==> 9      ! Ø ==> 4 !  CODE   !  CODE   ! CODE  !          Ø ===> 9           !
    +--------------------+---------+---------+---------+-------+-----------------------------+
BITS Ø ===============> 9 1Ø ==> 14 15 ==> 18 19 ==> 22 23 : 25 26 =====================> 35
```

PRP FAULT CODE -- The fault code returned to the port processor board in bits 32:35 of the TTL bus when a fault cycle is detected during a processor memory operation. The fault code is latched on PRPG and then is available as an input to the DBMUX.

BIT 32 = 1 -- MBOX error
BIT 32 = Ø and BIT 33 = 1 -- IOBOX Xmit Buff parity error
BIT 32 = Ø and BIT 33 = Ø -- IOBOX Rcvr Buff parity error

PRM FAULT CODE -- This is the same as the PRP fault code except for DATA MOVER memory operations. The code is latched on PRMC.

PRM ERROR CODE -- This is the code generated when any of several errors is detected by the DATA MOVER board. The encoder is on PRMD.

1 = PRM CRAM Parity Error
2 = IO BUS Parity Error (Detected on PRM5)
3 = MEMORY ECC UNCORRECTABLE ERROR
4 = MOVER PAR ERROR
5 = ACKNOWLEDGE ERROR
6 = FAULT CYCLE RECEIVED
7 = PLI PARITY ERROR

```
    +---------------+---------------------------------------------------+------------------+
    ! PLI DATA BITS !                PRP DATA PATH BITS                 !      ZEROS       !  DBMUX SEL = 1
    !    7 ==> Ø    !                    Ø ===> 23                      !                  !
    +---------------+---------------------------------------------------+------------------+
BITS Ø ===========> 7 8 =============================================> 31 32 ==========> 35


    +--------------------------------------------------------------------------------------+
    !                   OUTPUT BUFFER BITS  ØØ ==> 35                                      !  DBMUX SEL = 2
    +--------------------------------------------------------------------------------------+
BITS Ø ==============================================================================> 35


    +--------------------------------------------------------------------------------------+
    !                   RAM FILE BITS ØØ ===> 35                                           !  DBMUX SEL = 3
    +--------------------------------------------------------------------------------------+
BITS Ø ==============================================================================> 35
```

FIGURE PRP2 -- DATA BUS MULTIPLEXER

2.1.1.4 291Ø DATA AND TEST INPUTS

The 291Ø microprogram sequencer gets its four most significant direct input bits directly from CRAM. The eight least significant bits come from the DMUX. See Figure PRP3 for the four possible sets of direct inputs to the 291Ø.

The "TEST" input to the 291Ø comes from multiplexers with a total of 43 possible conditions to test. This is shown as a single 43 input test multiplexer on the block diagram. See Table PRP1 for a breakdown of the 43 inputs.

```
MRxØØ3                     PROCESSOR 291Ø DIRECT INPUTS


     +----------------+-----------------------------------+
     | CRAM JØ => J3 |  CRAM J4 ===> J11                  |  CRAM DIRECT SELECT = Ø
     +----------------+-----------------------------------+
BITS  Ø ===========> 3 4 ===============================> 11
291Ø D11================================================> DØ


     +----------------+-----------------------------------+
     | CRAM JØ => J3 |  RAM FILE Ø ===> 7                 |  CRAM DIRECT SELECT = 1
     +----------------+-----------------------------------+
BITS  Ø ===========> 3 4 ===============================> 11


     +----------------+-----------------------------------+
     | CRAM JØ => J3 |  DATA PATH 16 ===> 23              |  CRAM DIRECT SELECT = 2
     +----------------+-----------------------------------+
BITS  Ø ===========> 3 4 ===============================> 11


     +----------------+--------------------+--------------+
     | CRAM JØ => J3 |  CRAM J4 ==> J8    | INTR CODE 9:11|  CRAM DIRECT SELECT = 3
     +----------------+--------------------+--------------+
BITS  Ø ===========> 3 4 ===============================> 11

INTERRUPT CODES:

        Ø = DATA MOVER DONE
        1 = NOT USED
        2 = BELL
        3 = INTERRUPT SERVICE
        4 = LINK TRANSMIT ATTENTION
        5 = ANY ERROR
        6 = NOT USED
        7 = ACLO

BITS Ø:3 ARE ALWAYS CONTROLLED BY CRAM JØ:J3
BITS 4:11 ARE PUT THROUGH A MUX BEFORE GOING TO THE 291Ø
```

FIGURE PRP3

OCTAL SELECT CODE	TESTED CONDITION
ØØ	ALWAYS FAIL TEST -- TIED TO GROUND
Ø1	BYTE COUNT DONE
Ø2	CI RECEIVER BUFFER B FULL NI RCVR ATTENTION
Ø3	CI RECEIVER BUFFER A FULL NI FREE LIST ERR
Ø4	CI FIRST BUFFER NI CRC ERR
Ø5	CI BUSB BUFFER A
Ø6	CI BUSB BUFFER B NI FREE BUFFER ERROR
Ø7	CI CRC ERROR
1Ø	PORT PROCESSOR MEMORY CYCLE
11	CI ACKNOWLEDGE
12	ADAPTOR BUSY
13	TRANSMITTER BUSY NI LOSS OF CARRIER ERR
14	PLI TRANSMIT ABORT NI SB5
15	PLI CDA (Carrier Detect A)
16	CI CDB (Crrier Detect B) NI XMIT BUFF PAR ERR
17	CI TRANSMIT BUFFER PARITY ERROR
2Ø	CI RECEIVER A ENABLE
21	MOS ECC ERROR
22	WORD COUNT = Ø
23	TRANSMITTER ATTENTION
24	DIAG TEST B
25	NI END OF FRAME
26	I/O BUS PARITY ERROR
27	PLI PARITY ERROR
3Ø	DOORBELL
31	29Ø1 OVERFLOW
32	29Ø1 SIGN
33	PRP RAM FILE PARITY ERROR
34	29Ø1 CARRY OUT
35	MOVER TAG
36	ALWAYS TRUE -- TIED HIGH
37	FAULT CYCLE
4Ø	TIED HIGH
41	SET BELL
42	INTERLOCK OBTAINED
43	PREVIOUS TEST
44	ANY ERROR
45	GRANT SEEN
46	NOT MOVER ERROR
47	ACKNOWLEDGE ERROR
5X	29Ø1 FUNCTION = Ø
6X	INTERRUPT
7X	RECEIVER INTERLOCKED

Port Processor 291Ø TEST Inputs MRxØØ4

TABLE PRP1

2.1.1.5 CRAM DATA PATH

CRAM is loaded via the port scan path under the control of the console. The scan path registers are in parallel with CRAM address and data lines. The scan path can also control random logic in the port. Because the scan path is in parallel with CRAM and 291Ø outputs it can be used to monitor microcode sequences for debugging or diagnostic purposes.

The control lines from CRAM are used throughout the port processor for controlling its operation.

2.1.1.6 PORT PROCESSOR MICROWORD

Figure PRP4 shows the port processor's microword with associated field definitions. A more complete description of the fields is in the PORT HARDWARE SPECIFICATION.

```
00         11   14       23    27    32     37 40   43  45       52 55    60  62    66
 |          |    |        |     |     |      |  |    |   |        |  |     |   |     |
+-----------+--+-----------+---+------+----+----+--+-+-+-+-------+----+---+-+-+-+---+-+-+
| NXT ADDR  |SM| MAGIC #   |DES| SRC/ |PRTA|PRTB|SM|C|I|W| TEST  |CONT|SPC|T|G|P| D |P|M|   PORT PROCESSOR
|           | X|           |   | FUNC | ADD| ADD|HX| | |T| COND  |2910|FUN| |P|S|MIX| |K|   MICROWORD
+-----------+--+-----------+---+------+----+----+--+-+-+-+-------+----+---+-+-+-+---+-+-+
              ||            | |         |  |      ||   |   |    |      | |   |   | |   |
            12 13          24 26       33 36    41 42 44  46   51     56 59 61 63 65  67
```

BITS 00:11 = NEXT ADDRESS FIELD OR "J" FIELD -- Bits J0:J3 of this field are applied directly to the 2910 "D" inputs on the port processor board. Bits J4:J11 can also be applied to the "D" inputs through the DIRECT ADDRESS MUX. J4:J11 are also used to "JAM" a starting address for the mover microcode on the DATA MOVER board.

BITS 12:13 = DIRECT SELECT FIELD OR "SMUX" FIELD -- This two-bit field is used to select the source of DIRECT ADDRESS 4:11 through the direct address mux. See DMX10.FIG

BITS 14:23 = MAGIC NUMBER FIELD -- Used to address the port processor's RAM and for various data constants.

BITS 24:26 = 2901 DESTINATION CONTROL FIELD -- Used to control where data goes in the 2901s.

BITS 27:32 = FUNCTION AND SOURCE FIELD -- Bits 27:29 control the FUNCTION field of the 2901s while bits 30:32 control the SOURCE field of the 2901s.

DEST	RAM B GETS	Q REG GETS	Y OUT GETS
0	HOLD	ALU	ALU
1	HOLD	HOLD	ALU
2	ALU	HOLD	RAM A
3	ALU	HOLD	ALU
4	ALU/2	Q/2	ALU
5	ALU/2	HOLD	ALU
6	ALUx2	Qx2	ALU
7	ALUx2	HOLD	ALU

SRC	R	S
0	RAM A	Q REG
1	RAM A	RAM B
2	0	Q REG
3	0	RAM B
4	0	RAM A
5	D	RAM A
6	D	Q REG
7	D	0

FUNC	
0	R + S
1	S - R
2	R - S
3	R or S
4	R and S
5	-R and S
6	R ex or S
7	R ex nor S

BITS 33:36 = PORT A ADDRESS -- This field is used to address the 2901's internal RAM file through its "A" port. The "A" port can be applied to either side of the ALU or to the Y outputs without going through the ALU.

BITS 37:40 = PORT B ADDRESS -- This field is used to address the 2901's internal RAM file through its "B" port. The "B" port only goes to the "S" side of the ALU. The B port address is also the write address to the RAM file.

MRx010

FIGURE PRP4-1

```
 00           11 14         23  27     32   37   40 43 45        52   55  60  62    66
 |            |  |          |   |      |    |    |  |   |         |    |   |   |     |
+------------+--+----------+---+------+----+----+--+-+-+-+-------+----+---+-+-+-+---+-+-+
| NXT ADDR   |SM|  MAGIC # |DES| SRC/ |PRTA|PRTB|SM|C|I|W| TEST  |CONT|SPC|T|G|P| D |P|M|
|            | X|          |   | FUNC | ADD| ADD|HX| | |T| COND  |2910|FUN| |P|S|MIX| |K|
+------------+--+----------+---+------+----+----+--+-+-+-+-------+----+---+-+-+-+---+-+-+
              ||              |      |    |    ||   |   |     |    |   |   |   | |   |
             12 13           24     26   33   36 41 42 44  46    51  56  59  61 63 65 67
```

BITS 41:42 = ALU SHIFT MUX SEL -- This field controls the shift input mux to select different sources of shifted data.

For shifting LEFT

Sel 00	RAM 35 gets RAM 00	Q35 gets Q00	(36 bit shift loop)
Sel 01	RAM 35 gets carry in	Q35 gets RAM 00	
Sel 10	RAM 35 gets Q00	Q35 gets RAM 00	(72 bit shift loop)
Sel 11	RAM 35 gets zero	Q35 gets zero	

For shifting RIGHT

Sel 00	RAM 00 gets RAM 35	Q00 gets RAM 35	(36 bit shift)
Sel 01	RAM 00 gets DP00	Q00 gets RAM 35	(keep sign)
Sel 10	RAM 00 gets Q35	Q00 gets RAM 35	(72 bit shift)
Sel 11	RAM 00 gets 0	Q00 gets 0	

BIT 43 = CARRY BIT -- This bit controls the external carry in to the least significant 2901.

BIT 44 = INDEX BIT -- If this bit is on, the 5 MSBs of the port processor's RAM address will come from the INDEX reg. This allows addressing of the local memory as 32 sections of 32 locations. If the bit is off, the local RAM is addressed from # field 0:9.

BIT 45 = WRITE BIT -- This bit is used when the port processor writes its local RAM.

BITS 46:51 = TEST CONDITION SELECT -- This field is used to select a condition to be tested by the 2910. See PRTTES.TBL

BITS 52:55 = 2910 CONTROL FIELD -- This field is used to control the 2910. See 10CTL.TBL

BITS 56:59 = SPECIAL FUNCTION -- This field controls various miscellaneous logic apart from the 2901s and the 2910.

BIT 60 = TIME BIT -- Several functions in the port processor require 2 cycles to complete. The T bit must be set for these functions.

BIT 61 = GET PLI - This bit disables DATA MOVER control of the PLI and enables the PORT PROCESSOR control of the PLI.

BIT 62 = SELECT PLI -- This bit selects the PLI and causes a PLI cycle. What actually happens is adapter dependent. The PLI control lines are driven by number field 0:3. The PLI data line drivers are inhibited if #4 is on and enabled if #4 is off.

BITS 63:65 = D MIXER SELECT -- This field selects which inputs get through the DBUS MIXER to the 2901 "D" inputs. See DMIX.FIG

BIT 66 = PARITY -- Used to set overall odd parity for the microword.

BIT 67 = MARK -- Used for debugging. Can be used as a SYNC for scopes or logic analyzers. Does not affect operation.

MRx011

FIGURE PRP4-2

2.1.1.7 RAMFILE READ AND WRITE

The local 1K x 36 +P memory in the port processor is used for local storage and as a buffer in MBOX read and write operations done by the port processor. An expanded block diagram of the port's local memory (also referred to as RAMFILE) is shown in figure PRP5.

```
              +---------+                                               +---------+
              | INDEX   |                                               | MEMORY  +/----- RECEIVER
              |  REG    |                                               |  DATA   +\----- DATA
# FIELD 0:4==>|  PRP4   |===++                                +---------|  MUX    |
LOAD INDEX--->|         |   ||-- ADDRESS 0:4                  | +-------|         +/----- DATA PATH
INDEX REF & ->|         |   ||                                | |       |  PRP7   +\----- 00:35
-PROC MEM CYC +---------+   ||                                | |       +---------+
                            ||                                | |
              +---------+   ||       +-----------------+      | |                  +---------+
# FIELD 5:9==>|         |   ++======>|                 |      | |                  | OUTPUT  |
              | ADDRESS |===========>|   1K X 36 + P   |      \ /                  | BUFFER  +-----\ TO TRANSMITTERS
              |  LATCH  |    5:9     |    RAM FILE     +------+ +-----------------\|         +-----/ AND DPMUX
-PROC MEM CYC>|  PRP4   |  ++=======>|     PRP4        +--------++------+ +-------/|  PRP8   |
              +---------+  ||--0:6   |                 |        ||      | |        |         |
              +---------+  || ++====>|                 |        ||      | |        +---------+
# FIELD 0:6==>| ADDRESS |  || ||     +-----------------+        ||      | +----\ TO DPMUX
              |  LATCH  |==++ ||                            +---++---+  +------/
PROC MEM CYC->|  PRP4   |     ||          FORCE BAD PAR --->| PARITY |
-INDEX REF -->|         |     ||                            | CHECKER|---> PARITY ERR
              +---------+     ||-- ADDRESS 7:9              | / GEN  |
                              ||                            |  PRP7  |
                        +-----++-+                          +--------+
        # FIELD 7:9 ===>| ADDRESS|
                        | COUNTER|
          INC RAM ADD-->|CT PRP4 |
            START MEM-->|LD      |
                        +--------+
```

Port Processor RAMFILE Block Diagram   MRX005

FIGURE PRP5

The ramfile can be addressed by the port processor in two modes. Absolute addressing uses the ten-bit number field of the microcode word to address the ramfile directly through the two address latches. Indexed addressing uses a five-bit index register and the least significant five bits of the number field to address the memory in 32 sections of 32 words. This allows microcode to address sections of the ramfile using a displacement relative to the index register. The index register is loaded using special function 6 and number field bits 0:4. After the index register is loaded, relative addressing can be selected with the index bit of the microword. The microcode can switch freely between indexed and absolute addressing using the index bit.

Write data from the 2901s to the ramfile goes through the memory data mux. This mux is used to select either data from the port processor or data from the I/O BUS into the ramfile. The I/O BUS path is used for KC10 memory data requested by the port processor.

Parity is generated on the data to be written to the ramfile by the parity generator/checker. Bad parity can be forced through the scan path. The word with known-bad parity can then be read to check out the parity circuits.

2.1.2 KC10 MEMORY READ/WRITE

The port processor reads and writes KC10 memory through the IOBOX and the I/O BUS. Either single word or four-word reads and writes can be done.

```
              +--------+                                                +--------+                   +--------+
              | INDEX  |                                                | MEMORY +/------------------|XCEIVERS|
              |  REG   |                                                |  DATA  +\------------------|        |
# FIELD 0:4==>|  PRP4  |===++                                +----------|  MUX   |                   |        |
LOAD INDEX--->|        |   ||-- ADDRESS 0:4                  | +--------|        +/--DATA PATH       |        |
INDEX REF & ->|        |   ||                                | |        |  PRP7  +\--00:35   D CYC-->|        |/----------\
-PROC MEM CYC +--------+   ||                                | |        +--------+          CA CYC-->|        |  I/O BUS
                           ||                                | |                           TAG 0:3==>|        |\----------/
              +--------+   ||        +----------------+      | |                  +--------+         |        |
# FIELD 5:9==>|        |   ++=======>|                |      | |                  | OUTPUT |         |  PRP3  |
              | ADDRESS|============>|   1K X 36 + P  |      \ /                  | BUFFER +--------\|        |
              |  LATCH |     5:9     |     RAM FILE   +------+ +-----------------\|        +--+ +---/|        |
-PROC MEM CYC>|  PRP4  |  ++========>|     PRP4       +--------++------+ +-------/|  PRP8  |  | |    +--++----+
              +--------+  ||--0:6    |                |        ||      | |        |        |  \ /       ||
              +--------+  || ++=====>|                |        ||      | |        +--------+  TO        ||
# FIELD 0:6==>| ADDRESS|  || ||      +----------------+        ||      | +----\ TO DPMUX    DPMUX    +--\/----+
              |  LATCH |==++ ||                            +---++---+  +------/                      |PAR GEN/|---> XMIT PAR
PROC MEM CYC->|  PRP4  |     ||          FORCE BAD PAR --->| PARITY |                                |   CHK  |---> R PAR ERR
-INDEX REF -->|        |     ||                            | CHECKER|---> PARITY ERR                 +--------+
              +--------+     ||-- ADDRESS 7:9              | / GEN  |
                             ||                            |  PRP7  |
                       +-----++-+                          +--------+
       # FIELD 7:9 ===>| ADDRESS|
                       | COUNTER|
         INC RAM ADD-->|CT PRP4 |
           START MEM-->|LD      |
                       +--------+

                                                                                                               MRx007
```

FIGURE PRP6

2.1.2.1 READ

To read KC10 memory the port processor must write a command address word to the RAMFILE with a read command in bits 0:3 and the physical address of the word(s) to be read in bits 12:35. The port processor must then do a special function 3 (START MEMORY) with the number field pointing to the RAMFILE address where the command address word is located. This requests the I/O BUS and latches the address of the command address word into the address latch and the address counter.

When the bus grant signal is received, the command address word is read out of the ramfile and put on the I/O BUS, along with the port's tag and CA cycle, through the output buffer and the tranceivers. The ramfile's address counter is also incremented to point to the location following the command address word.

Two I/O BUS cycles after the command address word is transmitted the port checks for ACK from the IOBOX. ACK tells the port that the command address word was received by the IOBOX with correct parity. If ACK is not received, the port must assume that the IOBOX rejected the word because of bad parity. Parity is generated in the port tranceivers on the 36 data bits plus tag bits 0:3, CA cycle, and D cycle.

The port processor must wait for the data from the memory before using the ramfile. Hardwired logic locks out the port microcode from the ramfile until the read operation is complete. The microcode must abort the memory cycle to access the ramfile before the memory operation is complete. The microcode checks for errors while waiting for the data from memory. ACK ERR is one of the errors which is checked while waiting.

(More complete description of errors will be done when the microcode for error handling is firmed up.)

When an I/O BUS data cycle with the port's tag is detected, the data on the I/O BUS is parity checked in the tranceivers and written into the ramfile at the location pointed to by the address latch and address counter. This will be the location following the command address word. If the read is for four words, the counter is incremented and the next three words will be written into consecutive ramfile locations during the next three bus data cycles.

2.1.2.2 WRITE

To write data to KC10 memory the port processor loads the command address word and data words to be written into consecutive ramfile locations. The port processor then does a special function 3 to request the I/O BUS and latch the ramfile address of the command address word into the address latch and address counter.

When the port processor receives an I/O BUS grant, the command address word is put on the I/O BUS and the address counter is incremented. On the next bus cycle, the first data word is put onto the I/O BUS along with D CYCLE. If a four-word write is being done, the address counter increments on each D CYCLE until the fourth word has been put onto the I/O BUS.

Two I/O BUS cycles after each word is transferred, ACK from the IOBOX is checked. Therefore, five ACKs are expected for a four word write starting two cycles after the command address word is transferred and finishing two cycles after the last data word was transferred.

```
            ----------------
REQ _____|                  |______________________________
                              --
GRANT ___________________|  |______________________________
                                ------------
DATA ________________________|CA| D| D| D| D|_____________
                                      --------------
ACK ______________________________|  |  |  |  |  |________
```

DATA TRANSFER -- ACK RELATIONSHIP MRx013

ERROR HANDLING WILL BE DESCRIBED WHEN IT IS BETTER DEFINED

PROBABLY SOMETIME AFTER THANKSGIVING FOR A FIRST PASS

2.1.3 PLI READ WRITE

The port processor can read the PLI at any time. The data lines of the PLI are tied directly to bits 0:7 of the 2901 DMUX. Whenever the DMUX field is a one, the PLI data is available on the output of the DMUX. The microcode simply has to select the "D" inputs as a source for the 2901 while the DMUX field equals one.

Normally the port processor must put a command on the PLI control lines to cause the adaptor to put data on the PLI data lines. The command is sent, and data is read, during the same microinstruction. The port processor can, however, simply monitor the PLI data lines while the data mover is using the PLI for diagnostic purposes.

Parity is checked on the PLI data lines and PLI PAR ERR is available as a test condition to the 2910. To check the parity on data being read, the microcode must select PLI PAR ERR as a test condition while doing the read command. There is no latch for PLI parity errors.

To write the PLI, the Get PLI and SELECT PLI microword bits must be on. The Get PLI bit disables the data mover's PLI drivers and enables the port processor's PLI control lines. The control lines are driven from number field bits Ø:3. Number field bit 4 and Get PLI control the enables to the port processor's PLI data drivers. Number field bit 4 thus controls the direction of the data lines on the PLI when the port processor has it selected.

The PLI data drivers are connected to bits 24:31 of the port processor's data path. Parity is generated on bits 24:31 of the data path and transmitted along with the data. Bad parity can be forced via the scan path.

2.1.4 DATA MOVER CONTROL

3.0 PORT DATA MOVER DESCRIPTION

The port data mover's function is to move blocks of data between the port adapters and system memory. The data mover can pack or unpack data in any of four modes; core dump, high density, industry compatible, and 7-bit ASCII.

The data mover is a slave of the port processor. The port processor must set up the data mover for doing a data transfer and then start it. Once started, the data mover transfers a block of data between the adapters and system memory without intervention by the port processor. When the data mover is done, it notifies the port processor and waits.

The data mover has 1Kx44 bits of CRAM sequenced by a 2910 microprogram sequencer. It interfaces to the PLI on the adapter side and the TTL I/O BUS on the system memory side. There are also several direct connections to the port processor.

3.1 DATA MOVER FUNCTIONAL DESCRIPTION

3.1.1 DATA MOVER BLOCK DIAGRAM DESCRIPTION

Figure PRM1 is a basic block diagram of the port data mover.

```
                                                 +--------------------------------------+
                                                 | +----------------------------------+ |
                                                 | |   +--------+                     | |
                                                 | +--\| SHIFT  |                     | |
                              +--------+         |     | MIXER  |         +--------+  | |     +--------+
        SHIFT REG PAR 0:7 --->| INPUT  |         +----/| PRM1   |--------\| SHIFTER|  | |     | OUTPUT |--> 36-BIT PAR
                              | PARITY |               |        |         | PRM2   |  | |     | PARITY |
                    +-------->| PRM3   |         +----\| 40 BIT |--------/|        |  | |     | PRM7   |<-- PLI RECD PAR
                    |         +-----+--+         | 0:35|        |         |        |--+ |     +-----^--+
                    |               |            | +--/|        |         |        |    |           |
             +------+-+       +-----V--+         | |   +--------+         |        |--+ |     +-----+--+
      /-----\| TTL BUS|------\| INPUT  |---------+ +---------------------\|        |  | |0:7  | PLI OUT|
    IO BUS   | XCVRS  |       | BUFFER |                        BITS 4:35 |        |  |  ====>| BUFFER |======>++<==>PLI DATA
      \-----/| PRM5   |------/| PRM3   |---------------------------------/|        |  | |     | PRM7   |       ||      (8 BIT)
             | 8646s  |       | WT   RD|<== BUFF MOVER ADD     PLI IN ===>| 72 BIT |  | |     +--------+       ||
             |        |       +-/\-----+                                  |        |  | |                      ||
             |        |         ++=== BUFF MEM ADD                        |        |  | |                      ||
             |        |       +--------+                            +----\|        |  | |                  +---\/---+
             |        |/------| OUTPUT |/---------------------------+ 8:71|        |  | |                  | PLI IN |
             |        |       | BUFFER |      SHIFTER BITS 0:35       +--/|        |  | |     TO SHIFTER<==| BUFFER |
             |        |\------| PRM6   |\---------------------------+ |   +--------+  | |       BITS 64:71 | PRM7   |
             +--------+       | WT   RD|<== BUFF MEM ADD            | +---------------+ |                  +--------+
                              +-/\---^-+                            |                   |
                   ^            ||   +-- 36-BIT PAR                 +-------------------+
                   |            ||- BUFF MOVER ADD                               +--------+                      +----------+
             +-----+--+       +-++-----+                                         | SCAN   |                      |  SCAN    |
             | RD/WRT |------>|ADD CTRS|==> BUFF MEM ADD                         | REG    |                      |  REGS    |
             | CONTROL|       |PRM6    |--> EMPTY                                | PRM8   |                      | PRM9:B   |
             | PRMC   |       |        |--> FULL                                 +--------+                      +----------+
             +--------+       +--------+                                               / \                              / \
                                                                     +--------+        | |                              | |
                                                                     |  2910  |        | |                              \ /
                                                                     |  PRM8  |        \ /               |  CRAM  |-----+ +---\
                                                                 --\ |        |--------+ +--------------\| PRM9:B | 43 CONTROL \
                           +--------+               DIRECT ADD 4:11 >|D 7:0   | Y 0:9    CRAM ADD 2:11   |        |   LINES    /
                 MOVER GO->|0 TEST  |               FROM CRAM    --/ |        |--------+ +--------------/|        |-----+ +---/
                     FULL->|1 MUX   |                          GND==>|D 8:11  |        / \               |        |     | |
                    EMPTY->|2 PRM8  |                                |        |        | |-CRAM ADDR     |        |     \ /
XMIT BUFF RDY orDIAG TEST->|3       |------------------------------->|TEST __ |        | |   4:11        |        |   +--------+
RCVR BUFF B FULL or  EOF ->|4       |                                |     OE |    +--------+            +--------+   | PAR    |
              WORD CT = 0->|5       |                                +------^-+    |  CRAM  |                         | CHK    |
              BC DONE = 0->|6       |                                       |      |ADD BUFF|                         | PRMB   |
                     +3 V->|7       |                                       |----->|  PRM8  |                         +--------+
                           +--------+                     PRPG START MOVER--+      +--------+
                                                                                       / \
                                                                                       | |
                                                                             ----------+ |
                                                                        PRPC CRAM J4:J11 |
                                                                             ------------+
```

Data Mover Basic Block Diagram   MRx014

Figure PRM1

3.1.1.1 I/O BUS DATA PATH

Data from the I/O bus enters the data mover board through the bus tranceivers (PRM5) where it is checked for parity and then sent to the Input Buffer (PRM3). The input buffer is a four-word register file which is called a 4X4 memory on the prints. The input buffer can be read and written simultaneously. Data read from the input buffer goes to the Shift Mixer (PRM1) and the Shifter (PRM2).

Data to the I/O BUS comes from the mover shift register bits Ø:35 to the output buffer. The output buffer is another set of 4X4 memories. When four words have been loaded into the output buffer, a memory request is initiated. Data from the output buffer goes to the tranceivers and then to the I/O BUS.

The input and output buffers are addressed by separate counters (PRM6) for read and write addresses. One counter is controlled by the mover microcode and the other by the read-write logic. The control of the two counters switches between the read-write logic and the microcode depending on the direction of the transfer. A separate counter keeps track of whether the buffer in use is full or empty. The outputs of this third counter are available to the microprogam sequencer as test conditions.

3.1.1.2 PLI DATA PATH

Write data to the PLI comes from bits Ø:7 of the shifter to the PLI output buffer (PRM7). Parity is developed from bits Ø:7 on the shift register output (PRM2) and is gated through the buffer with the data. PLI control data comes from the PLI control field of the data mover microword and is gated through the output buffer with -GET PLI which is the negated signal from the port processor.

Data is read from the PLI into the PLI input mux (PRM7). This mux is used to shift incoming data in core dump mode and to force zeros in industry compatible mode. Parity is checked on the PLI data bits and is generated on the most and least significant 4 bits. The data then goes to shifter bits 64:71 where it is used to assemble 36-bit data words.

3.1.1.3 INTERNAL PARITY

The data mover does not generate and check parity on its internal data paths. Parity is checked as data enters the module and is generated as it leaves. Internally, a sort of parity prediction technique is used.

When data enters the data mover from the PLI, the parity is checked. Either this parity or the parity generated on the most or least significant 4-bits from the PLI is gated into the predictor. When the proper number of bytes for the data packing mode in use have been received, the predicted parity is stored in the output buffer along with the data word. When the word is read out of the output buffer and parity is generated for the I/O BUS, the stored bit is compared to the generated bit. If the bits are not the same, some sort of error occured in the data mover.

When data enters the data mover from the I/O BUS, two parity prediction bits are generated and stored in the input buffer along with the corresponding data word. One bit is 36-bit parity which is generated on the data mover module. The other bit is the exclusive-or of the odd output and parity for bits 32:35. As each byte is transferred over the PLI, its parity bit is also put into the prediction circuit. When an entire word or word pair has been transferred, the aggregate parity is checked. If the predicted parity does not agree with what was transmitted, there was an error in the data mover somewhere between the I/O BUS transceivers and the PLI output. The error can only be detected after the entire input word or words have been transferred over the PLI. (For more detail see section 16.2.14 of the port spec.)

3.1.1.4 SHIFTER DATA PATH

The shifter and shift multiplexer are used to convert data between 36-bit data words and 8-bit bytes. Together with the PLI mux, the shifter data path can convert data back and forth between 8-bit PLI bytes and 36-bit KC1Ø words in four formats.

```
00000000 00111111 11112222 22222233 3333     00000000 00111111 11112222 22222233 3333
01234567 89012345 67890123 45678901 2345     01234567 89012345 67890123 45678901 2345
\______/ \______/ \______/ \______/          \______/ \______/ \______/ \______/
```

INDUSTRY COMPATIBLE: 4 8-bit bytes per word. 4 LSBs in word are forced to 0

```
00000000 00111111 11112222 22222233     3333 00000000 00111111 11112222 22222233     3333
01234567 89012345 67890123 45678901     2345 01234567 89012345 67890123 45678901     2345
\______/ \______/ \______/ \______/ \______/ \______/ \______/ \______/ \______/ \______/
```

CORE DUMP: 5 8-bit bytes per word. 4 MSBs in 5th byte are forced to 0. 4 LSBs of 5th byte contain 4 LSBs of word.

```
00000000 00111111 11112222 22222233 33330000 00000011 11111111 22222222 22333333
01234567 89012345 67890123 45678901 23450123 45678901 23456789 01234567 89012345
\______/ \______/ \______/ \______/ \______/ \______/ \______/ \______/ \______/
```

HIGH DENSITY: 9 8-bit bytes per 2 words

```
0000000 0001111 1111112 2222222 2233333 3 0000000 0001111 1111112 2222222 2233333 3
0123456 7890123 4567890 1234567 8901234 5 0123456 7890123 4567890 1234567 8901234 5
\_____/ \_____/ \_____/ \_____/ \_____/   \_____/ \_____/ \_____/ \_____/ \_____/
```

7-BIT ASCII: 5 7-bit bytes per word. LSB in each word is forced to 0

MRx106

The following discussion uses the data mover shifter bit maps in addition to the basic block diagram.

```
MRxØ19                          DATA MOVER SHIFT DATA PATH BIT MAPS

                                SHIFT REG/MUX

        |Ø =======> SHIFT CLK ØØ-35 ===================> 35|36==========> SHIFT CLK 36-71 ==============> 71|    2 SEGMENT CLOCKS
        +--------------------------------------------------+----+----------------------------------------------+          ØØ-35
        | SHIFT MIX Ø ===> 39                                   | INPUT BUFFER BITS 4 ===> 35                   | SEL = Ø   36-71
        +-------------------------------------------------------+----------------------------------------------+
BITS     Ø ================================================> 39 4Ø ======================================> 71
                                                                                                                 3 SEGMENT SELECTS
        |Ø ==> SHIFT SEL ØØ:15 ==>|16 ====> SHIFT SEL 16:39 ====>|4Ø =========> SHIFT SEL 4Ø:71 ===============>|          ØØ:15
        +----------------------------------------------------------------------------+-------------------------+          16:39
        | SHIFT REG 8 ===========> 71                                                 | BYTE IN 64 ==> 71       | SEL = 1    4Ø:71
        +----------------------------------------------------------------------------+-------------------------+
BITS     Ø ======================================================================> 63 64 ===============> 71


                     PLI INPUT MUX       (DISABLED ON INDUSTRY COMPATIBLE BYTE-5) = ZEROS

        +-----------------------------------------+
        | PLI INPUT 7 ====> Ø                      | SEL = Ø
        +-----------------------------------------+
BITS    64 ================================> 71
        +--------------------+--------------------+
        |PLI INPUT 3 ==> Ø   |      ZEROS          | SEL = 1  (CORE DUMP BYTE 5)
        +--------------------+--------------------+
BITS    64 ============> 67 68 ============> 71


                                        MOVER SHIFTER INPUT MUX OUTPUTS

      +---------------------------------------------------------+--------------+-------------------+
      | INPUT BUFFER BITS Ø ==> 31                               | SHFT REG 68:71 | INP BUFF 32:35  |    SELECT = Ø
      +---------------------------------------------------------+--------------+-------------------+    WRITE CORE DUMP
BITS Ø =====================================================> 31 32 =========> 35 36 ===========> 39    OR HIGH DENSITY Ø:31
                                                                                                        CORE DUMP 32:39

      +------------------------------------------------------------------------+-------------------+
      | SHIFT REG 36 ==> 71                                                      | INP BUFF ØØ:Ø3   |    SELECT = 1
      +------------------------------------------------------------------------+-------------------+    SHIFT LEFT 36
BITS Ø ======================================================================> 35 36 ===========> 39    32:39 HIGH DENSITY WRITE


      +--------------------------------------------------------------------------------------------+
      | SHIFT REG 33:39 | 41 ==> 47      | 49 ==> 55      | 57 ==> 63      | 65 ==> 71     |64|   ZEROS   |    SELECT = 2
      +--------------------------------------------------------------------------------------------+    ASCII READ
BITS Ø ===========> 6 7 =========> 13 14 ========> 2Ø 21 ========> 27 28 =======> 34 35 36 =======> 39


      +--+-----------------------------------------------------------------------------------------+
      | Z| INP BUF BITS Ø:6 | Z|  7 ===> 13    | Z|  14 ===> 2Ø    | Z|  21 ===> 27    |35|  28 ===> 34    |    SELECT = 3
      +--+-----------------------------------------------------------------------------------------+    ASCII WRITE
BITS Ø  1 =============> 7  8 9 ========> 15 16 17 ========> 23 24 25 ========> 31 32 33===========> 39
```

Data in the shift register can be shifted left 8 bits while reading a byte from the PLI into bits 63:71 (shifter select = 1). For high density mode, this occurs 9 times to "build" two 36-bit data words. The left word is sent to the output buffer and the right word is shifted left 36 bits while reading the next byte from the PLI (shift mux sel = 1, shifter sel 0:15 and 16:39 =0, shifter sel 40:71 =1). If there is an open buffer slot, the second word is transferred to the output buffer and the process repeats. If no buffer slot is open, three more PLI bytes will be read into the shifter while waiting. This can be done because the shifter's right and left halves are clocked separately. When a buffer slot opens, the word in the left half of the shifter is transferred to the buffer and five more bytes are read from the PLI to refill the shifter.

In industry compatible mode, four bytes are read from the PLI (SHIFT SELs all = 1) and a fifth byte of zeros is inserted (PLI MUX DISABLED). Four more bytes are then read from the PLI. If the output buffer has an empty slot, the left word is written to it. Another byte of zeros is inserted and four more bytes are read from the PLI. This process repeats until the byte counter equals zero.

Core dump mode resembles industry compatible mode in the way bytes are assembled into KC10 words. Four bytes are read and shifted left. The final four bits come from the least significant four bits of the fifth byte read from the PLI. The PLI input mux shifts these bits four places and zeros the remaining bit positions (PLI MUX SEL = 1). Four more bytes are read, shifting the first assembled word left, along with the new bytes. At this point there is one complete KC10 word in the shifter and most of another. If an output buffer slot is open, the first word is transferred. The operation repeats itself from the point where the fifth byte is read in core dump mode until the transfer is complete.

For ASCII mode reads, five bytes are read from the PLI and shifted left. The five bytes are "squeezed" into a 36 bit data word while reading the first byte of the next word from the PLI (SHIFT SEL 00:15 AND 16:39 = 0, 40:71 = 1, SHIFT MUX SEL = 2). The conversion to packed 7-bit ASCII format is accomplished in the shift multiplexer using select = 2. If a slot is available, the word is written to the output buffer. If no slot is available, three more bytes are read from the PLI into the right half of the shifter. When a buffer slot opens up, the word in the left half of the shifter is written to the buffer and the read operation continues until the transfer is complete.

In high density write operations, the first word of a pair is taken from the input buffer and put into the left half of the shifter using the shift mux (select for bits 0:31 = 0, select for bits 32:39 = 1). The second word of the pair is then placed in the right half of the shifter (bits 0:3 using the shift mux bits

36:39 and bits 4:35 going directly to shifter bits 40:71). After both words of the pair are in the shifter, the PLI will be written nine times from shifter bits 0:7. The remaining data in the shifter shifts left 8-bits after each transfer. This process will be repeated until the transfer is complete. The transfer MUST be a multiple of 2 36-bit data words.

In core dump write operations, bits 0:31 of each word are written to the left half of the shifter through the shift mux (select 0:31 = 0). Bits 32:35 of the data word are written to bits 36:39 of the shifter through the shift mux (select 32-39 = 0). Bits 32:35 of the shifter are fed from bits 68:71.

In industry compatible write operations, bits 0:31 of the data word are sent to the shifter through shift mux bits 0:31 (select 0-31 = 0). The 32 bits are written over the PLI in four 8-bit bytes with the rest of the data word shifting left 8 bits after each word is written.

In ASCII write mode, the 36-bit KC10 word is sent to the shift mux where it is expanded to 40 bits in shifter bits 0:39 (select = 3). A zero bit is appended to the MSB of each character, converting it from 7-bit to 8-bit ASCII. The five characters are transferred over the PLI and the operation repeats until the transfer is complete.

3.1.1.5 MICROCODE SEQUENCING

The data mover uses a 2910 microprogram sequencer chip to control microcode flow. The data mover only has 1K of microcode space, so only 10 bits of the 2910 are used. The 2910's direct inputs are fed directly from the microword's direct address field 2:11.

The test input of the 2910 is driven from an eight input mux. The source for the test is selected by the TEST SEL field of the microword. The conditions that can be tested are as follows:

1. MOVER GO

2. FULL -- The input or output buffer is full.

3. EMPTY -- The input or output buffer is empty.

4. DIAG TEST

5. RCVR BUFFER B FULL

6. WC = Ø

7. BC DONE

8. TIED HIGH -- always qualified.

When the port processor sets up the data mover it first must set up the word and byte counters (discussed later) on the port processor board. When the conditions are ready to start the data mover, the port processor "jams" a microcode address using its JFIELD (J4:J11) and the START MOVER bit. The START MOVER bit is tied to the Output Enable lines of the 291Ø and the CRAM ADDress BUFFer. The signal disables the 291Ø and enables the address in the port processor J field through the buffer and onto the CRAM address lines.

The data mover then executes whatever microcode routine that was forced by the port processor. When it finishes the transfer, the data mover goes back into its idle loop awaiting another command from the port processor.

3.1.1.6 DATA MOVER MICROWORD

The data mover's microcode is 44 bits wide by 1K deep. The word is broken down as follows:

MRxØ21

Data Mover Microword

```
    1                1Ø         14 15      18  2Ø    22           27            32   34  35   37    39   41  42
+--+--------------------+------+----+----+------+--+--+---------+--+----------+--+--+----+--+--+--+--+--+----+--+
|MK| DIRECT ADDRESS     | TEST |SMIX|SMIX| SHIFT|CK|CK|   PLI   |S | 291Ø     |SP|BF|BYTE|LD|P |I |SP|MV| PAR|P |
|  |   J FIELD          | MUX  |Ø:31|32>9| CONT |Ø |36| CONTROL |P | CONTROL  |  |  | 5  | P|  |BC|  | D| SEL|TS|
+--+--------------------+------+----+----+------+--+--+---------+--+----------+--+--+----+--+--+--+--+--+----+--+
 Ø                       11  13      16 17       21  23        26   28       31   33      36    38    4Ø      43
```

BIT Ø	= MARK BIT
BITS 1:1Ø	= DIRECT ADDRESS FIELD TO THE 291Ø "D" INPUTS. ALSO CALLED JUMP FIELD.
BITS 11:13	= TEST INPUT MUX SELECT TO THE 291Ø.
BITS 14:15	= SELECTS FOR BITS Ø:31 OF THE SHIFT MIXER.
BITS 16:17	= SELECTS FOR BITS 32:39 OF THE SHIFT MIXER.
BITS 18:2Ø	= SHIFT CONTROL FIELD FOR THE SHIFTER.
BIT 21	= CLOCK CONTROL FOR BITS Ø:35 OF THE SHIFTER.
BIT 22	= CLOCK CONTROL FOR BITS 36:71 OF THE SHIFTER.
BITS 23:26	= PLI CONTROL BITS. THESE BITS DRIVE THE CONTROL LINES OF THE PLI.
BIT 27	= PLI SELECT.
BITS 28:31	= 291Ø CONTROL
BIT 32	= SPARE BIT A
BIT 33	= COUNT 4X4 MEMORY ADDRESS.
BITS 34:35	= CONTROLS HOW THE FIFTH BYTE OF A 36 BIT WORD IS TREATED. Ø = NO-OP 1 = INDUSTRY COMPATIBLE 2 = CORE DUMP
BIT 36	= LOAD PARITY OUT -- USED TO GENERATE 36 BIT PARITY TO LOAD THE OUTPUT BUFFERS.
BIT 37	= CRAM PARITY BIT
BIT 38	= INCREMENT BYTE COUNT -- USED TO INCREMENT THE BYTE COUNTER FOR THE PLI DATA TRANSFERS.
BIT 39	= SPARE BIT B
BIT 4Ø	= MOVER DIRECTION -- USED TO CONTROL THE PLI OUTPUT DRIVERS. THIS MUST BE A ONE TO ENABLE THE DRIVERS.
BITS 41:42	= USED TO SELECT THE METHOD OF CHECKING RECEIVED DATA PARITY.
BIT 43	= PARITY TEST -- USED TO ENABLE CHECKING OF RECEIVE PARITY.

3.1.1.7 CRAM LOAD PATH

3.1.2 PLI TO MEMORY READ OPERATION

3.1.3 MEMORY TO PLI WRITE OPERATION

4.Ø PLI BUS

5.Ø PACKET BUFFER DESCRIPTION

6.Ø CI LINK DESCRIPTION

7.Ø NI ADAPTER DESCRIPTION

AVAILABLE

8.Ø CI PORT FUNCTIONAL DESCRIPTION

9.Ø NI PORT FUNCTIONAL DESCRIPTION

1Ø.Ø MSCP DISK TRANSFERS

11.Ø TERMINAL AND UNIT RECORD HANDLING
