CI20 HARDWARE PORT SPECIFICATION

(REV 03 - MAY 19, 1983)
BY: ELBERT BLOOM

INDEX

1.0 GENERAL OVERVIEW

1.1 REFERENCE DOCUMENTS

1.2 SPECIFIC GOALS

1.3 SPECIFIC NON-GOALS

1.4 BLOCK DIAGRAM

1.5 MECHANICAL DESCRIPTION

1.6 GENERAL LOGICAL OVERVIEW

1.7 MICROCODE DESCRIPTION

1.8 COOLING REQUIREMENTS

1.9 MTBF

1.10 RAMP FEATURES

1.11 PERFORMANCE

1.12 SUBSYSTEM DOCUMENTATION

1.13 SYSTEM SOFTWARE

1.14 STANDARDS COMPLIANCE

2.0 EBUS INTFC/PORT ALU MODULE

2.1 EBUS CONTROL LOGIC

2.1.1 PI LEVEL 00 INTERRUPTS

2.1.2 PI LEVEL 01 THROUGH 07 INTERRUPTS

2.2 MICROPROC TO EBUS REGISTER (EBUF)

2.3 EBUS CONTROL AND STATUS REGISTER (CSR)

2.3.1 DEFINITION OF THE CONTROL AND STATUS (CSR) REGISTER BITS FOR THE EBUS INTERFACE MODULE

2.4 EBUS TO MICROPROC MUX (EMUX)

2.5 MICROPROC TO EBUS MUX (KMUX)

2.6 EBUS PARITY GENERATOR

2.7 EBUS PARITY CHECKER

2.8 EBUS TRANSCEIVERS

2.9 ARITHMETIC LOGIC UNIT

2.9.1 CONSTANT MUX

3.0 CBUS-PLI INTERFACE MODULE (CMVR)

3.1 CBUS TIMING DESKEW PROCEDURE

3.2 CMVR CONTROL LOGIC

3.3 MICROPROC TO CMVR REGISTER (CBUF)

3.4 DATA FORMATTER and MOVER (MVR/FMTR)

3.5 CBUS INPUT BUFFER

3.6 CBUS OUTPUT BUFFER

3.7 CBUS CONTROL LOGIC

3.8 CBUS OUT PARITY GENERATOR

3.9 CBUS IN PARITY CHECKER

3.10 DATA INPUT MUX (DMUX)

3.11 PLI OUTPUT BUFFER

3.12 PLI INPUT BUFFER

3.13 PLI PARITY OUT GENERATOR

3.14 PLI PARITY IN CHECKER

3.15 PLI CONTROL LOGIC

3.16 PLI SERIAL UP MUX (SUMUX)

3.17 PLI SERIAL DOWN MUX (SDMUX)

3.18 PLI OUTOUT MUX (PMUX)

3.19 CMVR TO MICROPROC MUX (CMUX)

3.20 PARITY PREDICTOR

4.0 PORT MICROPROCESSOR

4.1 CONDITION CODE MUX

4.1.1 CONDITION CODE DEFINITIONS

4.2 MICROSEQUENCER

4.3 RAM ADDRESS REGISTER

4.4 ADDRESS MUX

4.5 LATCH ADDRESS REGISTER (LAR)

4.6 CONTROL STORE RAM

4.7 CONTROL STORE REGISTER

4.7.1 MICROWORD FIELD DEFINITIONS

4.8 JMP MUX

4.9 MICROWORD OUTPUT MUX

4.10 CRAM LOAD BUFFERS

4.11 CRAM PARITY CHECKER

4.12 LOCAL STORAGE RAM

4.13 LOCAL STORAGE ADDRESS REGISTER

4.14 RAM MODE MUX

4.15 COND/SKIP FIELD DECODER

4.16 MICROPROC CONTROL LOGIC

APPENDIX "A" DETAILED BLOCK DIAGRAMS

1.0 GENERAL OVERVIEW

The KL10 CI PORT ADAPTER (CI20) is the unique HARDWARE/FIRMWARE option required in order to interface a KL10 based operating system to the Corporate high speed serial line CI Bus.

With a CI20 installed, and with the necessary software drivers and Port Microcode implemented the KL10 will be capable of communications, through a STAR COUPLER, over the Corporate CI Bus.

The primary functions of the CI20 are:

a) To enable multiple KL10s to be configured in a loosely coupled environment via the CI.

b) To enable KL10s to be configured to HSC50s via the CI.

1.1 REFERENCE DOCUMENTS

The below documents contain details of all the topics necessary for a thorough understanding of the CI20 and may be referenced, should additional information be required:

a) CBUS and EBUS INTERFACES - "RH20 MASSBUS CONTROLLER UNIT DESCRIPTION" (EK-RH20-UD-001).

b) PLI INTERFACE - "PILA HARDWARE SPECIFICATION" by: Shu-Shia Chow

c) AM2901, AM2902, AM2910 - "THE AM2900 FAMILY DATA BOOK".

d) CI MICROCODE ARCHITECTURE - "COMPUTER INTERCONNECT SPECIFICATION" By: D. THOMPSON / J. BUZYNSKI / J. HUTCHISON

e) UNIQUE CI20 MICROCODE ARCHITECTURE - "LCG CI PORT ARCHITECTURE SPECIFICATION" By: Don Dossa

Appendix "A" consists of three detailed block diagrams, one for each of the three PORT modules (EBUS.DRW, PROC.DRW and CBUS.DRW).

DEFINITION OF THE CONTROL AND STATUS (CSR) REGISTER BITS FOR THE EBUS INTERFACE MODULE (section 2.3.1) contains detailed descriptions of the CONTROL and STATUS REGISTER bits.

CONDITION CODE DEFINITIONS (section 4.1.1) contains detailed descriptions of the microprocessor's CONDITION CODE sense inputs.

MICROWORD FIELD DEFINITIONS (section 4.7.1) contains detailed descriptions of the microprocessor's CRAM CONTROL WORD decodes.

Reference should be made to these sections throughout this specification whenever further explanation is required.

1.2 SPECIFIC GOALS

1) FIELD UPGRADABILITY - The CI20 is designed as a FIELD INSTALLABLE upgrade kit for KL10 model Bs.

2) DATA FORMAT MODES - The CI20 supports three data format modes; HIGH DENSITY, CORE DUMP and INDUSTRY COMPATIBLE.

3) CI COMPATIBILITY- The CI20 is designed to meet the corporate CI specification. Therefore, it is able to interface with other CI devices that also meet the corporate CI specification.

4) SMP COMPATIBILITY - CI20s may be installed on SMP systems. However, external memory must be configured in FOUR BUS MODE only. Data Over-runs will frequently occur in a TWO BUS MODE configuration.

1.3 SPECIFIC NON-GOALS

1) No KL10 based device diagnostics are required other than those uniquely required to test the CI20.

2) The CI20 is not supportable by KL10 model "A"s. This is due to additional microcode requirements for the KL10 which may only be implemented on model "B" versions.

3) The CI20 does not support SEVEN BIT ASCII data format mode.

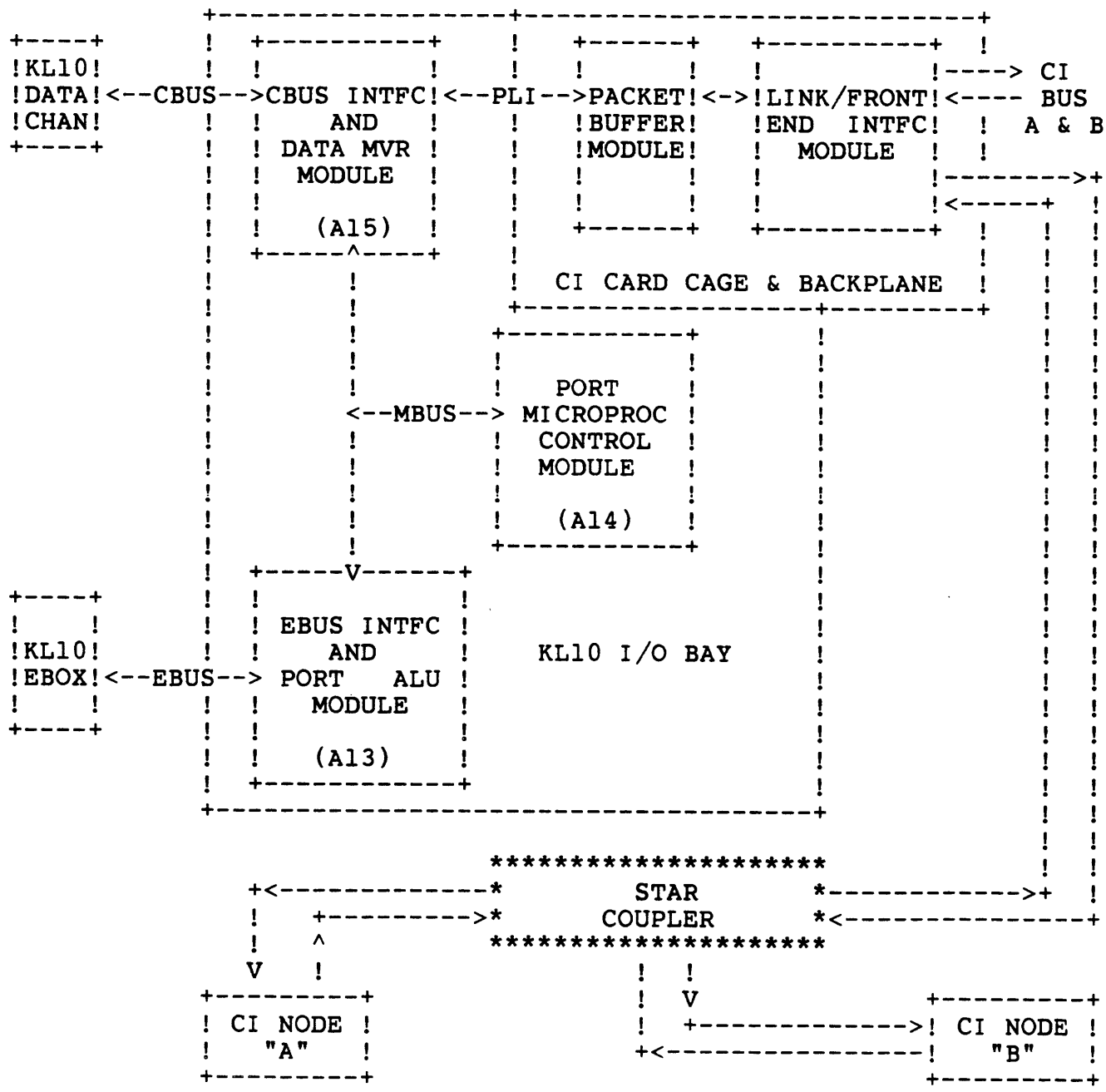
4) Two or more CI20s installed on one KL10 is not supported as a valid configuration.

5) Parity generation/checking is not implemented on the internal data path of the PORT. Instead, parity is checked and re-generated at all external busses to the PORT.

6) The CI20 does not have the ability to be powered down and up separately from the KL10.

1.4 BLOCK DIAGRAM

The CI20 interfaces the KL10 to the CI BUS via the KL10'S DATA CHANNEL and EBUS. The following block diagram illustrates the basic architecture of the CI20.



SIMPLIFIED CI20 PORT/LINK INTERFACE BLOCK DIAGRAM

1.5 MECHANICAL DESCRIPTION

The CI20 consists of an upgrade kit for KL10s. The kit includes:

A) 3 STANDARD HEX MODULES AS FOLLOWS:

1) M3001 EBUS INTFC MODULE

2) M3002 MPROC CONT MODULE

3) M3003 CBUS INTFC MODULE

B) 2 EXTENDED TRI-BOARDS AS FOLLOWS:

1) L0109 PACKET BUFFER MODULE

2) L0100 LINK INTERFACE MODULE

C) 1 +5.0V POWER REGULATOR (H7440)

D) 1 POWER HARNESS

E) 1 FLAT RIBBON CABLE FOR PLI INTFC (BC06R-08)

F) 1 CI CARD CAGE & BACKPLANE

G) 4 INTERNAL CI CABLES

H) 2 BULKHEAD CONNECTORS

I) 1 FLAT RIBBON CABLE FOR MBUS CONNECTOR

J) 1 DUMMY MODULE WHICH PLUGS INTO RH20 POSITION #6
AND ACTS AS A CABLE STRAIN RELIEF

K) 30 AWG WIRE (GREEN)

L) ASSORTED MTG HDWR

M) INSTALLATION PROCEDURES

N) PRINT SET AND MANUALS

P) STANDARD MICROCODE PACKAGE

R) CI20 DIAGNOSTICS

Extender cables for the MBUS are not included as part of the CI20 UPGRADE KIT. however, they are available separately in order to enable manufacturing and field service to troubleshoot the CI20 PORT modules on an extender board.

The EXTERNAL CI CABLES and STAR COUPLERS, which are also required to configure a CI20, are not included in the CI20 kit. They must be ordered separately according to the desired configuration. The following list identifies these components.

SC008-AA — STAR COUPLER control hub which connects CI nodes (single path). Provides electrical isolation between nodes. Consists of:
- 1 8 node STAR COUPLER
- 1 H9642-CA/CB CABINET

SC008-AB — Same as SC008-AA except without cabinet. Used as the control hub of the second path in a redundant path configuration.

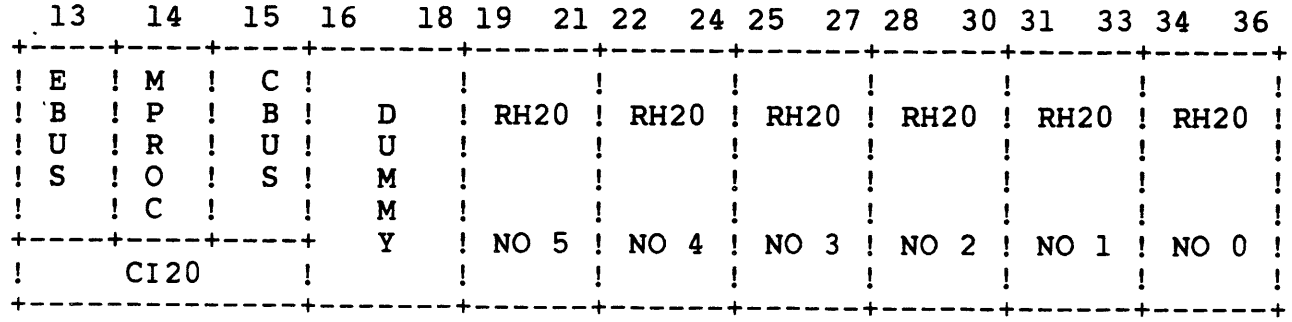
BNCI-10 — EXTERNAL CI CABLE PAIR for single path connection. Consists of 1 pair of 10 meter coaxial cables.

BCNI-20 — Same as BCNI-10 except 20 meter length.

BCNI-45 — Same as BCNI-10 except 45 meter length.

The three STANDARD HEX modules (M3001, M3002 & M3003) comprise the PORT section of the CI20. They are uniquely designed for the CI20. The modules are housed in RH20 position # 7 of the KL10's RH20-DTE20 backplane. RH20 position # 6 must have only the dummy cable strain relief module installed. No RH20 is permitted in slot # 6 when a CI20 is installed. Any KL10 which houses a CI20 will be restricted to a maximum of 6 RH20s and one CI20. Reference the below diagram:

RH20-DTE20 I/O BACK-PLANE SLOTS



The "DUMMY" position is required due to the high current requirements of the CI20. It requires nearly as much current as two RH20s.

Control signals between modules utilize existing backplane wiring for the RH20s which normally occupy these slots. 12 wires must be added to the backplane, however, in order to obtain sufficient interconnects between modules and to bus across five EBUS INTERFACE signals which are are not used by RH20s. 30 AWG wire (green) is used to add these wires to the BACK-PLANE.

The following chart lists the the RH20 BACK-PLANE point to point wire adds:

SIGNAL NAME	FROM	TO
EBUS D11 L	C12F1	B13B1
EBUS D12 L	C12D1	B13B2
EBUS D13 L	C12F2	B13U1
EBUS PI00 L	C12H2	C13N1
EBUS PARITY L	C12E2	C13B1
EBUS PARITY ACTIVE L	C12L1	C13B2
MPR7 MWBUSCTLFLD01 H	C14H2	A15R2
MPR7 MWMGCFLD08 H	C14F1	F15A1
MPR7 MWTIMEFLD H	A14J2	A15E1
CBI1 CLK2 L	C14P1	A15D2
CBI1 CLK4 L	C14K2	A15S2
CBI2 CCCHANERR L	B14J1	B15A1

Once the back-plane for RH20 position # 7 has been modified to house a CI20 an RH20 should never be re-inserted in the position. The RH20 will no longer work properly.

the CI CARD CAGE and BACKPLANE are required for housing the two EXTENDED TRI-BOARDS. For DECsystem-20 style cabinets they are mounted in the CPU BAY of the KL10 next to the memory unit. For DECsystem-10 style cabinets they are mounted in the I/O BAY. Both are uniquely designed for the CI20.

The two EXTENDED TRI-BOARDS (L0100 & L0109) comprise the LINK/FE and PACKET BUFFERS respectively. They are designed by other engineering groups within DEC and are integrated into the CI20 as standard off-the-shelf items.

The BC06R-08 CABLE is used as the PLI INTERFACE CABLE. It connects from a BERG connector on the CBUS INTFC MODULE (M3003) to a BERG connector mounted on the CI CARD CAGE BACK-PLANE. This cable is a standard off-the-shelf item.

The four internal CI CABLES route from the CI CARD CAGE BACK-PLANE to the internal BULKHEAD CONNECTORS.

The two BULKHEAD CONNECTORS are also off-the-shelf items, but require a special mounting plate for installation into the KL10.

The H7440 REGULATOR is inserted in slot 5 of power supply # 2 in the KL10's I/O BAY. It is used to supply +5.0V to the CI CARD CAGE and BACK-PLANE.

The POWER HARNESS is used to route +5.0 volts from the H7440 REGULATOR and -5.2 volts from the KL10's CPU POWER SUPPLY to the CI CARD CAGE and BACK-PLANE.

The flat ribbon cable used as the MBUS CONNECTOR consists of a 50 wire 18 inch long conductor with three 50 pin BERG connectors mounted to it, one at each end and one in the center. It is uniquely designed for the CI20.

Mostly OFF-THE-SHELF technology is used for the portion of the KL10 CI PORT uniquely designed for the CI20. The three PORT modules use a combination of SHOTTKY and LOW POWER SHOTTKY logic.

1.6 GENERAL LOGICAL OVERVIEW

The CI20 consists of three logical sub-groups; a) LINK/FRONT END INTERFACE; b) PACKET BUFFER; c) PORT

This document is intended to describe the PORT hardware function. The LINK/FE and PACKET BUFFER functions are described in their associated specifications. therefore, only a brief description of them will be included here.

Detailed block diagrams of each of the three PORT modules are included in appendix "A" of this specification. They should be continuously referenced as this specification is reviewed.

a) LINK/FRONT END INTERFACE MODULE (L0100) - This module is the standard corporate LINK/FE module. Physically it consists of an EXTENDED TRI PRESS-PIN type module. Its primary function is to interface the PACKET BUFFER MODULE to the serial CI BUS. It is responsible for such functions as:

a. CRC generation/checking
b. Arbitration for information tramsmission
c. Moving information between the buffers and the CI BUS
d. Encoding/decoding of the bit stream
e. Header decoding and recognition
f. Acknowledge generation and transmission
g. the PARALLEL/SERIAL bit stream conversion.

This module uses a combination of Shottky TTL and ECL logic. Its power requirements are as follows:

+5.0 Volts at 8.5 Amps

-5.2 Volts at 8.2 Amps

b) PACKET BUFFER MODULE (L0109) - This module is the standard PACKET BUFFER module used by the HSC50 system. Physically it consists of an EXTENDED TRI PRESS-PIN type module. Logically it consists of four 1K deep by 8-bit wide RAM buffers, along with the necessary control logic for loading and reading the buffers. Its primary function is to act as a temporary storage interface between the LINK/FE and the PORT.

Its power requirements are as follows:

+5.0 Volts at 10.5 Amps

c) PORT - The port functions as an AM2901 based microprocessor which handles the CI PORT protocol in much the same manner as other CI PORTS. Control and status information is passed between the PORT and the KL10 via the KL10's EBUS INTERFACE. DMA data transfers are passed between the PORT and the KL10 via the KL10's CBUS INTERFACE.

The PORT consists of three STANDARD HEX FINE LINE ETCH modules which are inserted into a dedicated low priority RH20 slot in the KL10 RH20-DTE20 BACKPLANE.

They are linked together by a 36-bit tri-state data path called the MBUS. All data is passed between modules via this bus.

The functions of the three modules may be further divided into three logical subgroups as follows: (reference attached block diagrams in appendix "A" of this specification)

1) EBUS INTERFACE/PORT ALU MODULE (M3001)

The EBUS INTERFACE/PORT ALU MODULE acts as a low-speed asynchronous control interface between the KL10 EBOX and the PORT MICROPROCESSOR. It performs all of the functions required for passing data between the EBUS and the MICROPROCESSOR. It also contains a 36 bit CONTROL AND STATUS register which enables the PORT to control and monitor the EBUS operations. Most of the PORT PROTOCOL is processed over the EBUS through this interface.

In addition, this module houses the PORT MICROPROCESSOR ALU. The ALU consists of nine AM2901 bit slice ICs (36 bit data path) and four AM2902 high speed look-ahead carry generators. A CONSTANT MUX is also included, which allows the PORT MICROPROCESSOR to pass pre-assigned constants from the CRAM (CONTROL STORE RAM) to the ALU.

The module's power requirements are as follows:

+5.0 Volts at 8.5 Amps

2) PORT MICROPROCESSOR CONTROL MODULE (M3002)

The PORT MICROPROCESSOR consists of a horizontally programmed bit slice microprocessor controller. It is responsible for control of the CBUS/MOVER and the EBUS/PORT ALU INTERFACES. It performs such functions as data mapping, CI PACKET interpretation and some PACKET BUFFER manipulations. It consists of a 2910 type MICROSEQUENCER, a 4K deep by 60 bit wide CONTROL RAM (CRAM), a 1K deep by 36 bit wide SCRATCH PAD RAM, a CRAM CONTROL REGISTER and other associated control logics.

Once the CRAM is loaded and the MICROPROCESSOR is started the PORT is entirely under control of the microwords which are strobed from the CRAM into the CRAM CONTROL REGISTER at the beginning of each clock cycle.

The module's power requirements are as follows:

+5.0 Volts at 10.0 Amps

3) CBUS-PLI INTERFACE MODULE (M3003)

The CBUS-PLI INTERFACE (CMVR) module acts as a high-speed synchronous DMA data transfer path and data formatter between the PACKET BUFFER and the KL10 CBUS CHANNEL. It uses a 4-bit

parallel by 12-bit serial shift register for the data formatter, which is capable of mapping 8-bit bytes into 36-bit words, and vice versa. It also contains the necessary control logic for performing data transfers between the shift register, the CBUS and the PLI interfaces.

In addition, the module supports a 36-bit read/write data path between the PORT MICROPROCESSOR and the data formatter, and an 8-bit read/write data path between the PORT MICROPROCESSOR and the PLI INTERFACE. This enables the PORT MICROPROCESSOR to directly transfer data to/from the CBUS or the PLI INTERFACES.

The module's power requirements are as follows:

+5.0 Volts at 8.0 Amps

The PORT is controlled by a four phase master clock generator which is located on the CBUS/DATA MOVER MODULE. One microcycle normally requires 270ns (see CMVR CONTROL LOGIC for details).

1.7 MICROCODE DESCRIPTION

The hardware of the three CI20 PORT modules (EBUS, MICROPROC & CBUS INTFCS) supply all of the necessary data paths to enable the PORT to efficiently communicate between the KL10 and the CI. They also supply a uniquely designed, horizontally programmed microprocessor to enable the PORT to control and process the data which passes across these data paths.

The port hardware is incapable of functioning as a CI20 without the PORT MICROCODE. Therefore, the PORT MICROCODE must be considered an integral part of the CI20.

The KL10 must load the PORT MICROCODE into the microprocessor's CONTROL RAM (CRAM) before it starts the PORT. Once the KL10 has loaded the MICROCODE and started the PORT it will be capable of functioning as a CI20.

The primary functions of the PORT MICROCODE are to:

a) Control all data transfers between the PLI INTERFACE and the KL10

b) Process all CI PROTOCOL PACKETS in conformance with the CI ARCHITECTURE SPECIFICATION

c) Provide a FIRMWARE INTERFACE between the CI20 and the KL10's microcode and software operating systems.

d) Provide an adequate ERROR MONITORING/REPORTING interface between the CI20 and the KL10's operating system.

The PORT MICROCODE enables the CI20 to conform to the CI ARCHITECTURE SPECIFICATION.

Recovery of errors that do not compromise the integrity of the CI20 also conforms to the CI ARCHITECTURE SPECIFICATION. The operating system is informed of all errors, either through the RESPONSE QUEUE or the INTERRUPT mechanisms.

The CI20 microcode resides in the following areas:

a) System area of the MONITOR DISK PACK

b) KLAD DISK PACK

Reference the "LCG CI PORT ARCHITECTURE SPECIFICATION" By: Don Dossa for detailed explanations of the microcode functionality.

1.8 COOLING REQUIREMENTS

The present cooling system in the RH20-DTE20 CARD CAGE is adequate for cooling the three PORT MODULES (EBUS, MPROC and CBUS).

Fans are installed in the CI CARD CAGE to assure adequate cooling for the PACKET BUFFER and LINK/FE MODULES.

1.9 MTBF

The CI20 is designed to conform to the following failure rates and repair times.

MTBF = 10,000 Hrs

MTTR = 3/4 Hrs.

MTTD = 1/4 Hrs.

1.10 RAMP FEATURES

Some of the CI20's RAMP FEATURES include:

a) Parity Generation/Checking on all external busses to the PORT

b) CRAM Parity Checking

c) The ability to recover from CRAM PARITY errors with minimal user interruption.

d) A microcode test routine which periodically verifies the data path and control logic.

e) Diagnostic loop-back capability at major data path boundries.

f) Diagnostic ability to generate incorrect parity

g) A SINGLE-CYCLE mode for test and debug

h) Ability to latch the CRAM ADDRESS for diagnosability of failures.

i) A microcode routine which will periodically verify the data integrity of the Local Storage Rams

j) Ability to predict correct parity across the DMA data path between the PLI INTERFACE and the CBUS.

1.11 PERFORMANCE

BANDWIDTH - The CI20, excluding monitor software overhead, provides a realizable data transfer rate capability of approximately 300 pages/sec. This figure, for convenience, may also be translated into other bases such as 0.7 megabytes/sec, 154 kilowords/sec, 3.2 ms/page, etc.

The CI20's realizable bandwidth, excluding monitor software overhead, depends on two factors:

a) THE ACTUAL HARDWARE DMA DATA TRANSFER RATE

The CI20 hardware provides the capability of DMA data transfer rates of 1.6 ms/page (1.4 megabytes/sec)

This bandwidth is highly dependent on the microcode's ability to move the data through the DMA DATA TRANSFER PATH.

The calculation is derived from the following considerations:

1) On the average 11.5 microcycles are required per DMA WORD TRANSFER.

2) Each microcycle requires 270ns execution time

3) One 36 bit KL10 word consists of 4.5 bytes

THEREFORE:

$$BW = \frac{\text{BYTES/WORD}}{\text{TIME X MICROCYCLES}} = \frac{4.5}{270\text{ns X } 11.5} = 1.4 \text{ MEGABYTES/SEC (1.6 MS/PAGE)}$$

b) THE MICROCODE PROTOCOL PROCESSING OVERHEAD

The CI20 protocol processing overhead also takes approximately 1.6 ms/page (measured value).

Therefore, the aggregate realizable bandwidth is simply the sum of the two values derived from (a) and (b) above, or 3.2 ms/page (300 pages/sec).

1.12 SUBSYSTEM DOCUMENTATION

Documentation for the CI20 includes:

a) CI20 HARDWARE MAINTENANCE MANUAL - supplies all information required to enable an CI20 to be easily installed and repaired.

b) KL10 MAINTENANCE GUIDE - updates to reflect necessary information about the CI20.

c) SITE PREPARATION GUIDES

d) ILLUSTRATED PARTS BULLETIN

e) CI20 FIELD PRINT SET

f) MANUFACTURING TEST SPECIFICATIONS

g) TEST & ACCEPTANCE PROCEDURE

h) All KL10 PRINT SET ECO's - if any are required

i) Revision control for KL10 systems will be inclusive of CI20s for any future KL10 ECOs which may be required.

1.13 SYSTEM SOFTWARE

TOPS-20 includes all necessary drivers to support a CI20. These drivers include, but are not necessarily limited to:

a) An MSCP DRIVER

b) An SCA DRIVER

c) A CI20 PORT DRIVER

A software microcode loader for loading the PORT MICROCODE.

Software hooks for accurately reporting CI20 error conditions to SPEAR.

1.14 STANDARDS COMPLIANCE

The CI20 meets DEC-102 TEMP/HUMIDITY standards.

The increase in the current EMI-RFI profile of the KL10 is less than 6 DB average above its current level with a CI20 installed.

2.0 EBUS INTFC/PORT ALU MODULE

The EBUS INTERFACE occupies about 2/3rds of this module and consists primarily of:

a) A CONTROL and STATUS REGISTER (CSR) which is used for passing control and status parameters between the PORT MICROPROCESSOR and the HOST.

b) A data path, via the MBUS, between the PORT MICROPROCESSOR and the EBUS

c) Control logic for loading and starting the PORT'S MICROCODE

d) An EBUS parity generator/checker

e) All necessary control logic for interfacing to and executing the KL10's EBUS protocol, including the "EBUS INTERRUPT" sequence.

f) Diagnostic logic for executing various loop-back and other test functions.

The KL10 accesses the EBUS INTFC by executing DATAO, DATAI, CONO and CONI commands.

The PORT MICROPROCESSOR accesses the EBUS INTFC by executing microprocessor commands. These commands are decoded functions of the MWBUSCTLFLD field and the MWMGCFLD field of the CRAM CONTROL WORD.

The PORT MICROPROCESSOR monitors the EBUS INTFC LOGIC status by sensing CONDITION CODES (see section CONDITION CODE DEFINITIONS for details).

The MICROPROCESSOR ALU is also located on the EBUS INTERFACE MODULE. It consists of:

a) Nine AM2901 type FOUR-BIT BIPOLAR MICROPROCESSOR SLICES (36-bit wide data path) which interfaces to the MBUS.

b) Four AM2902 type HIGH-SPEED LOOK-AHEAD CARRY GENERATORS

c) Five 74LS157 type MULTIPLEXERS (CNST MUX) used by the MICROPROCESSOR CONTROL to input a CONSTANT NUMBER FIELD from the CRAM CONTROL REGISTER into data bits 00 thru 09 and data bits 26 thru 35 of the ALU.

2.1 EBUS CONTROL LOGIC

The EBUS CONTROL LOGIC arbitrates the EBUS protocol, the PORT MICROPROCESSOR protocol for interfacing to the EBUS and the synchronization functions between the two.

The KL10 has full control of the PORT only when the PORT is not in the "MPROC RUN" state (CSR32 reset). In this state the PORT MICROPROCESSOR is not running. The KL10's primary functions are to:

a) load and read verify the PORT'S MICROCODE

b) initially set up the correct CSR REGISTER functions

c) check for error conditions should the PORT halt unexpectedly

Secondary diagnostic functions also exist in this state which enables the KL10 to perform such functions as writing and read/verifying the EBUF, generating bad parity, single cycling, etc.

The KL10 performs these functions by executing CONOs, CONIs, DATAOs, and DATAIs. The PORT's EBUS INTERFACE processes these functions via the normal EBUS protocol.

A brief description of these sequences follows:

a) DATOLOADRAR - If the KL10 executes a DATAO with bit 00 equal to "1" a DATOLOADRAR signal will be generated. This will cause bits EBUS D01-13 to be loaded, via the MBUS, into the PORT'S "RAM ADDRESS REGISTER" located on THE MICROPROCESSOR CONTROL MODULE.

b) DATOLOADMW - If the KL10 executes a DATAO with bit 00 equal to zero a DATOLOADMW signal will be generated. This will cause the 30 least significant bits on the EBUS to be loaded, via the MBUS, into the selected half of the PORT'S "CRAM" location as specified by the "RAM ADDRESS REGISTER".

c) DATIREADMW - If the KL10 executes a DATAI with CSR21=0 a DATIREADMW signal will be generated. This will cause the contents of the selected half of the PORT'S CRAM location, as specified by the contents of the "RAM ADDRESS REGISTER", to be placed on the EBUS.

d) DATIREADLAR - If the KL10 executes a DATAI with CSR21=1 a DATIREADLAR signal will be generated. This will cause the contents of the LAR REGISTER to be placed on EBUS D01-12

e) CONOLOADCSR - If the KL10 executes a CONO a CONOLOADCSR signal will be generated. This will cause the contents of the EBUS to be loaded into all CSR REGISTER bits which are writeable by the KL10.

f) CONIREADCSR - If the KL10 executes a CONI a CONIREADCSR signal will be generated. This will cause the contents of all CSR REGISTER bits which are readable by the KL10 to be placed on the EBUS.

g) TESTLOADEBUF - If the KL10 executes a DATAO with CSR19=1 a TESTLOADEBUF signal will be generated. This will cause EBUS D00-35 to be loaded, via the MBUS, into the EBUF.

h) TESTREADEBUF - If the KL10 executes a DATAI with CSR19=1 a TESTREADEBUF signal will be generated. This will cause the contents of the EBUF to be placed on the EBUS.

When the PORT is in the "MPROC RUN" state (CSR32 set) the KL10 is permitted only to access the CSR REGISTER by executing CONO or CONI commands.

In the "MPROC RUN" state CONO and CONI commands operate in the same manner as described above.

DATAO and DATAI commands executed by the KL10's software when the PORT is RUNNING (CSR32 set) will cause the condition code CCEBUSRQST to be asserted. Since this is an unexpected illegal function the condition will usually be ignored by the PORT. Thus, an EBUS TIMEOUT will occur since "TRANSFER" will never be returned over the EBUS.

When the PORT is RUNNING the PORT MICROPROCESSOR controls the EBUS by generating EBUS INTERRUPTS. There are two types of interrupts generated by the port:

a) Non-Vectored (40 + 2n) software interrupts (IOP WORD = 0)

b) Microcode interrupts in which a non-zero IOP WORD is passed to the KL10's microcode for decode and execution of specific KL10 microcode functions.

A list of the types of interrupts which may be generated by the port are listed below. All of them may be generated by the hardware, even though the PORT MICROCODE does not currently use many of them. Those which are currently used by the PORT are marked by an "*":

* FUNCTION 0 = STANDARD (40 + 2N) INTERRUPT (IOP WORD = 0)

FUNCTION 2 = INCREMENT OR DECREMENT

FUNCTION 3 = DATAO (EXAMINE AND INCREMENT)

* FUNCTION 4 = DATAO (EXAMINE)

* FUNCTION 5 = DATAI (DEPOSIT)

FUNCTION 6 = BYTE TRANSFER (DEPOSIT)

* FUNCTION 7 = DATAO (EXAMINE AND INCREMENT). This is a new function which is being added specifically for the CI20.

2.1.1 PI LEVEL 00 INTERRUPTS

If the EBUS INTERRUPT is a function 2 through 7 interrupt, the PORT MICROPROCESSOR requests the INTERRUPT on PI LEVEL 00 by executing the command "MPEXORDEP" (EXAMINE OR DEPOSIT).

PI LEVEL 00 type INTERRUPTS are always processed by the KL10 as first priority, even when the KL10 has its interrupt enable system turned off. Therefore, the CI20 is capable of executing these interrupts regardless of the state of the KL10's interrupt enable system.

This sequence is executed as follows:

a) The PORT's microcode builds an IOP FUNCTION CONTROL WORD and loads it into the EBUF, via an MPLOADEBUF (LOAD EBUF) command. With the same microword it executes an "MPEXORDEP" (REQUEST EXAMINE OR DEPOSIT) command.

b) The MPEXORDEP command will cause RQST EXAM OR DEP (CSR04) to be set. This will cause the PORT's EBUS INTERFACE to assert the PI REQUEST line PI00 on the EBUS.

c) When the KL10's EBOX recognizes the PI00 REQUEST it responds by asserting

   1) the PORT's channel number on CS04-06 of the EBUS

   2) PI SERVED (4 octal) on the "F" lines of the EBUS

   3) DEMAND, after a sufficient delay

d) The PORT's EBUS INTERFACE responds by asserting the EBUS DATA line which corresponds to its physical device number.

e) The KL10's EBOX, after a sufficient delay, reads the EBUS DATA lines and negates DEMAND

f) The KL10's EBOX then asserts

   1) the PORT's channel number on CS04-06

   2) the PORT's physical number on CS00-03

   3) PI ADR IN (5 octal) on the "F" lines of the EBUS

   4) DEMAND, after a sufficient delay

g) The PORT's EBUS INTERFACE responds by asserting on the EBUS

   1) ACKN

   2) The IOP FUNCTION CONTROL WORD which was previously loaded in the EBUF by the PORT MICROPROCESSOR.

   3) XFER, after a sufficient delay

h) When the KL10's EBOX detects XFER it strobes the data from the EBUS DATA lines and negates DEMAND

i) The trailing edge of DEMAND causes the PORT's EBUS INTERFACE to negate ACKN, XFER and the DATA lines, thus ending the interrupt sequence

The KL10 microcode in turn decodes the IOP FUNCTION CONTROL WORD and executes the appropriate function. If the "IOP FUNCTION CONTROL WORD" specifies that an EBUS CYCLE is required (I.E. EXAMINE or DEPOSIT) the very next EBUS CYCLE following the "IOP FUNCTION CONTROL WORD" read will be addressed to the PORT. This sequence occurs as follows:

a) The KL10's EBOX asserts

   1) a device code of "ZERO" on CS00-06 of the EBUS

   2) DATAO or DATAI (2 or 3 octal) on the "F" lines of the EBUS

   3) Data on the EBUS DATA lines, if the function is a DATAO.

   4) DEMAND, after a sufficient delay

b) The PORT's EBUS INTERFACE responds by asserting ACKN on the EBUS. It also flags the PORT MICROPROCESSOR by asserting the CONDITION CODE, CCEBUSRQST.

   Since the device code on CS00-06 returned by the KL10 is "ZERO", not the PORT'S device code, the PORT does not examine the device code. Instead the PORT assumes that the very next EBUS CYCLE is intended for it and, therefore, takes the appropriate action as soon as it senses DEMAND asserted on the EBUS.

c) When the PORT's microcode detects CCEBUSRQST it responds by executing an MPLOADEBUS or MPREADEBUS command, whichever is applicable.

   If the PORT's microcode executes an MPLOADEBUS it must have previously loaded the EBUF, via an MPLOADEBUF command, with valid data to be transferred to the KL10.

If the PORT's microcode executes an MPREADEBUS the EBUS DATA will be placed on the MBUS. The PORT's microcode must also execute another command on the same microcycle to strobe the data from the MBUS into one of its internal storage media.

d) After a sufficient delay the PORT's EBUS INTERFACE asserts XFER on the EBUS

e) When the KL10's EBOX detects XFER it negates DEMAND

   If the function was an EXAMINE (DATAI) it strobes the data from the EBUS DATA lines

   If the function was a DEPOSIT (DATAO) it de-asserts the data from the EBUS DATA lines

f) The trailing edge of DEMAND causes the PORT's EBUS INTERFACE to negate ACKN, XFER, CCEBUSRQST and the DATA lines (if applicable), thus ending the interrupt sequence.

g) The PORT's MICROCODE must be prepared to respond promptly to the CONDITION CODE, CCEBUSRQST, in order to prevent EBUS TIMEOUTS. The PORT MICROPROCESSOR should not attempt to execute any additional EBUS transfers until it detects the negation of CCEBUSRQST.

2.1.2 PI LEVEL 01 THROUGH 07 INTERRUPTS

If the EBUS INTERRUPT is a function 0 interrupt, the PORT MICROPROCESSOR requests the INTERRUPT on PI LEVEL 01 through 7, depending on the assigned PI LEVEL in CSR33-35, by executing the command "MPRQSTINTR" (REQUEST INTERRUPT). This sequence is executed as follows:

a) The PORT's microcode first checks the CONDITION CODE, "CCINTRACTIVE". If it is asserted the microcode must wait until it is de-asserted before continuing the below sequence.

b) The PORT's microcode then executes an "MPRQSTINTR" command. The MPRQSTINTR command will cause RQST INTERRUPT (CSR05) to be set. This will cause the PORT's EBUS INTERFACE to assert the PI REQUEST line (PI01-07) as specified by PIA00-02 (CSR33-35) on the EBUS.

c) When the KL10's EBOX recognizes the PI01 through 7 REQUEST it responds by asserting

1) the PORT's channel number on CS04-06 of the EBUS

2) PI SERVED (4 octal) on the "F" lines of the EBUS

3) DEMAND, after a sufficient delay

d) The PORT's EBUS INTERFACE responds by asserting the EBUS DATA line which corresponds to its physical device number.

e) The KL10's EBOX, after a sufficient delay, reads the EBUS DATA lines and negates DEMAND

f) The KL10's EBOX then asserts

1) the PORT's channel number on CS04-06

2) the PORT's physical number on CS00-03

3) PI ADR IN (5 octal) on the "F" lines of the EBUS

4) DEMAND, after a sufficient delay

g) The PORT's EBUS INTERFACE responds by asserting on the EBUS

1) ACKN

2) A hardware generated IOP FUNCTION CONTROL WORD of all zeros

3) XFER, after a sufficient delay

h) When the KL10's EBOX detects XFER it strobes the data from the EBUS DATA lines and negates DEMAND

i) The trailing edge of DEMAND causes the PORT's EBUS INTERFACE to negate ACKN, XFER and the DATA lines.

j) When the KL10 microcode decodes the IOP WORD of all zeros it will generated a standard Non-Vectored (40 + 2n) interrupt to the KL10's software.

2.2 MICROPROC TO EBUS REGISTER (EBUF)

The "MICROPROCESSOR TO EBUS REGISTER" (EBUF) is a 36-bit register normally used by the PORT MICROPROCESSOR to pass data from the MBUS (Internal tri-state MICROPROCESSOR BUS) to the EBUS, or to the CSR REGISTER. The PORT MICROPROCESSOR generally loads data into this

register from the MBUS. This data may then be strobed by the next microcycle of the PORT MICROPROCESSOR to either the CSR REGISTER or to the EBUS.

The PORT MICROCODE uses the EBUF for the following two functions:

1) To load the CSR REGISTER. Data is first strobed into the EBUF, then to the CSR REGISTER.

2) To transmit an IOP FUNCTION CONTROL WORD over the EBUS for execution of EXAMINE or DEPOSIT functions on PI LEVEL 00.

The command, MPLOADEBUF (LOAD EBUF), causes the data currently on the MBUS to be loaded into the EBUF at CLK3 time.

A diagnostic loop-back path exists, however, which enables the KL10 to load and read this buffer. This loop-back path is controlled by DIAG TEST EBUF (CSR19). If CSR19 is set and the PORT is not in the "MPROC RUN" state (CSR32 re-set):

a) A DATAO executed by the KL10 will cause the date asserted on the EBUS to be loaded, via the MBUS, into the EBUF

b) A DATAI executed by the KL10 will cause the data in the EBUF to be asserted on the EBUS.

2.3 EBUS CONTROL AND STATUS REGISTER (CSR)

The "EBUS CONTROL and STATUS REGISTER" (CSR) is a 36 bit register which resides in the EBUS INTERFACE.

The KL10 accesses the CSR REGISTER by executing CONO and CONI commands.

The PORT MICROPROCESSOR, when in the "MPROC RUN" state, accesses the CSR REGISTER by executing the following sequence.

a) The PORT MICROPROC loads the EBUF with the desired CSR data, if it is writing to the CSR, by executing an MPLOADEBUF command. On the same microcycle it executes an MPRQSTCSR (Request CSR) command.

b) The PORT MICROPROC then senses for the CONDITION CODE, CCGRNTCSR (CSR GRANTED)

c) Once CCGRNTCSR is valid the PORT MICROPROC may execute either an MPLOADCSR (LOAD CSR) or MPREADCSR (READ CSR) command. If an MPLOADCSR command is executed the contents of the EBUF will be strobed into the CSR REGISTER at CLK3 time.

   If an MPREADCSR command is executed the contents of the CSR REGISTER will be asserted on the MBUS. The PORT MICROPROC must then strobe the MBUS data into one of its internal storage media on the same microcycle.

The CSR REGISTER is read/write interlocked to prevent the PORT and the KL10 from accessing it simultaneously. This is accomplished by the CONDITION CODE, CCGRNTCSR. When the PORT wishes to access the register it executes an MPRQSTCSR command. If the register is available CCGRNTCSR will be asserted by the EBUS INTERFACE LOGIC. If the KL10 is accessing the register at the time via a CONI or a CONO, CCGRNTCSR will not be asserted until the CONI or CONO function is complete.

The PORT MICROPROCESSOR must wait until it senses the asserted state of CCGRNTCSR before it attempts to access the CSR REGISTER.

Likewise, if the PORT MICROPROC is accessing the register when a CONI or a CONO is executed by the KL10 the EBUS INTERFACE LOGIC will cause the command to wait until the PORT has completed its access.

Race conditions between the PORT and the KL10 are prevented by granting access to the KL10 at CLK1 time and granting access to the PORT (asserting CCGRNTCSR) at CLK3 time.

The below chart briefly describes the CRS REGISTER bits. A more thorough description follows:

"*" indicates that the bit is not defined
"R" indicates that the bit is readable
"W" indicates that the bit is writeable
"C" indicates that the bit may be cleared only as a single bit
"S" indicates that the bit may be set only
"H" indicates that the bit is hardware controlled

BIT NO.	BIT DEFINITION	RD/WR KL10	RD/WR PORT
00	PORT PRESENT	R	H
01	DIAG RQST CSR	R	H
02	DIAG CSR CHNG	R/H	H
03		*	*
04	RQST EXAM OR DEP	R/H	R/S
05	RQST INTERRUPT	R/H	R/S
06	CRAM PARITY ERR	R/C	H
07	MBUS ERROR	R	H
08		*	*
09		*	*
10		*	*
11	IDLE	R	R/W
12	DISABLE COMPLETE	R	R/W
13	ENABLE COMPLETE	R	R/W
14		*	*
15	PORT ID CODE 00	R	H
16	PORT ID CODE 01	R	H
17	PORT ID CODE 02	R	H
18	CLEAR PORT	W	*
19	DIAG TEST EBUF	R/W	*
20	DIAG GEN EBUS PE	R/W	*
21	DIAG SEL LAR	R/W	*
22	DIAG SINGLE CYC	R/W	*
23	SPARE	R/W	*
24	EBUS PARITY ERR	H/R/C	R
25	FREE QUEUE ERR	R/C	R/S
26	DATA PATH ERR	R/C	R/S
27	CMD QUEUE AVAIL	R/S	R/C
28	RSP QUEUE AVAIL	R/C	R/S
29		*	*
30	DISABLE	R/S	R/C
31	ENABLE	R/S	R/C
32	MPROC RUN	R/W	R/H
33	PIA 00	R/W	R
34	PIA 01	R/W	R
35	PIA 02	R/W	R

2.3.1 DEFINITION OF THE CONTROL AND STATUS (CSR) REGISTER BITS

BIT #	NAME	DEFINITION
CSR00	PORT PRESENT	INDICATES THAT THE PORT IS PRESENT. READ/WRITE STATUS: KL10 = "R" PORT = "H" THE KL10 ALWAYS READS THIS BIT AS "1" IF THE PORT IS PRESENT (INSTALLED AND POWERED UP). THE PORT ALWAYS READS THIS BIT AS "0"
CSR01	DIAG RQST CSR	DIAGNOSTIC BIT WHICH INDICATES THE STATUS OF "CCRQSTCSR". READ/WRITE STATUS: KL10 = "R" PORT = "H" SET BY: 1) THE PORT MICROPROC REQUESTING ACCESS TO THE CSR (ASSERTING MPRQSTCSR). CLEARED BY: 1) THE PORT MICROPROC READING THE CSR (ASSERTING MPREADCSR). 2) THE PORT MICROPROC WRITING THE CSR (ASSERTING MPLOADCSR). 3) THE KL10 SETTING CLEAR (CSR18) 4) A GENERAL EBUS RESET.

CSR02 DIAG CSR CHNG

DIAGNOSTIC BIT WHICH INDICATES THAT THE CONTENTS OF THE CSR REGISTER HAS CHANGED SINCE THE LAST TIME IT WAS READ BY THE PORT MICROPROCESSOR.

READ/WRITE STATUS: KL10 = "R/H"
PORT = "H"

SET BY:

1) THE KL10 WRITING THE CSR (EXECUTING A CONO FUNCTION)

2) DETECTION OF AN "EBUS PARITY ERROR" (CSR24 SETTING).

CLEARED BY:

1) THE PORT MICROPROC READING THE CSR (ASSERTING MPREADCSR)

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR03 UNUSED

THIS BIT IS NOT USED BY EITHER THE PORT MICROPROC OR THE KL10.

READ/WRITE STATUS: KL10 = "*"
PORT = "*"

CSR04 RQST EXAM OR DEP

USED BY THE PORT MICROPROCESSOR TO REQUEST AN EBUS INTERRUPT FUNCTION ON PI LEVEL 00 (EXAMINE OR DEPOSIT FUNCTION). A "PI LEVEL 00 INTERRUPT" WILL BE IMMEDIATELY GENERATED WHEN THIS BIT IS SET.

READ/WRITE STATUS: KL10 = "R/H"
PORT = "R/S"

SET BY:

1) THE PORT MICROPROC REQUESTING AN EBUS EXAMINE OR DEPOSIT INTERRUPT ON PI LEVEL 00 (ASSERTING MPEXORDEP)

CLEARED BY:

1) SUCCESSFUL COMPLETION OF THE EXAMINE OR DEPOSIT SEQUENCE.

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR05 RQST INTERRUPT

USED BY THE PORT MICROPROCESSOR TO REQUEST AN EBUS NON-VECTORED (40 + 2n) INTERRUPT ON PI LEVELS 01 THRU 07. A "PI LEVEL 01 THRU 07 INTERRUPT" WILL BE IMMEDIATELY GENERATED WHEN THIS BIT IS SET.

READ/WRITE STATUS: KL10 = "R/H"
PORT = "R/S"

SET BY:

1) THE PORT MICROPROC REQUESTING AN EBUS INTERRUPT ON PI LEVELS 01 THRU 07 (ASSERTING MPRQSTINTR)

2) CRAM PAR ERR (CSR06) SETTING

3) MBUS ERR (CSR07) SETTING

CLEARED BY:

1) SUCCESSFUL COMPLETION OF THE INTERRUPT SEQUENCE.

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR06 CRAM PAR ERR

INDICATES THAT A CONTROL RAM PARITY ERROR HAS BEEN DETECTED.

IF THIS BIT IS SET THE PORT MICROPROC WILL BE HALTED IMMEDIATELY AND RQST INTERRUPT (CSR05) WILL BE SET. A HARDWARE NON-VECTORED (40 + 2n) INTERRUPT WILL BE FORCE GENERATED.

THE PORT MICROPROCESSOR CANNOT BE RE-STARTED (CSR32 SET) UNTIL THIS BIT IS CLEARED.

OCCASIONALLY A "CRAM PARITY ERROR" MAY BE INTENTIONALLY FORCED IN ORDER TO HALT THE PORT MICROPROC AT A SPECIFIC LOCATION (BREAK POINT).

READ/WRITE STATUS: KL10 = "R/C"
PORT = "H"

SET BY:

1) THE DETECTION OF A CONTROL RAM PARITY ERROR

CLEARED BY:

1) THE KL10 STORING A "1" IN CSR24 (EBUS PARITY ERR)

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR07 MBUS ERR

INDICATES THAT MORE THAN ONE MBUS DRIVER HAS BEEN TURNED ON AT THE SAME TIME.

IF THIS BIT IS SET THE PORT MICROPROC WILL BE HALTED IMMEDIATELY AND RQST INTERRUPT (CSR05) WILL BE SET. A HARDWARE NON-VECTORED (40 + 2n) INTERRUPT WILL BE FORCE GENERATED.

THE PORT MICROPROCESSOR CANNOT BE RE-STARTED (CSR32 SET) UNTIL THIS BIT IS CLEARED.

READ/WRITE STATUS: KL10 = "R"
PORT = "H"

SET BY:

1) THE DETECTION OF MORE THAN ONE MBUS DRIVER BEING TURNED ON AT THE SAME TIME

CLEARED BY:

1) THE KL10 SETTING CLEAR (CSR18)

2) A GENERAL EBUS RESET.

CSR08 UNUSED

THIS BIT IS NOT USED BY EITHER THE PORT MICROPROC OR THE KL10.

READ/WRITE STATUS: KL10 = "*"
PORT = "*"

CSR09 UNUSED

THIS BIT IS NOT USED BY EITHER THE PORT MICROPROC OR THE KL10.

READ/WRITE STATUS: KL10 = "*"
PORT = "*"

CSR10 UNUSED

THIS BIT IS NOT USED BY EITHER THE PORT MICROPROC OR THE KL10.

READ/WRITE STATUS: KL10 = "*"
PORT = "*"

CSR11 IDLE

THIS IS A MICROCODE-SOFTWARE DEFINED BIT USED BY THE PORT TO INFORM THE KL10 OPERATING SYSTEM THAT THE PORT MICROPROC IS IN THE MICROCODE "IDLE" LOOP. THE PORT MICROCODE SETS THIS BIT EACH TIME IT ENTERS THE "IDLE" LOOP AND CLEARS THE BIT EACH TIME IT LEAVES THE "IDLE" LOOP.

READ/WRITE STATUS: KL10 = "R"
PORT = "R/W"

SET BY:

1) THE PORT STORING A "1" IN THE BIT.

CLEARED BY:

1) THE PORT STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR12 DISABLE COMPLETE

THIS IS A MICROCODE-SOFTWARE DEFINED BIT USED BY THE PORT TO INFORM THE KL10 OPERATING SYSTEM THAT THE PORT MICROPROC HAS PLACED ITSELF IN THE "DISABLED" STATE.

READ/WRITE STATUS: KL10 = "R"
PORT = "R/W"

SET BY:

1) THE PORT STORING A "1" IN THE BIT.

CLEARED BY:

1) THE PORT STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR13 ENABLE COMPLETE

THIS IS A MICROCODE-SOFTWARE DEFINED BIT USED BY THE PORT TO INFORM THE KL10 OPERATING SYSTEM THAT THE PORT MICROPROC HAS PLACED ITSELF IN THE "ENABLED" STATE.

READ/WRITE STATUS: KL10 = "R"
PORT = "R/W"

SET BY:

1) THE PORT STORING A "1" IN THE BIT.

CLEARED BY:

1) THE PORT STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR14 UNUSED

THIS BIT IS NOT USED BY EITHER THE PORT MICROPROC OR THE KL10.

READ/WRITE STATUS: KL10 = "*"
PORT = "*"

CSR15 PORT ID CODE 00

THIS BIT REPRESENTS BIT "00" OF THE THREE BIT "PORT IDENT CODE" FIELD.

READ/WRITE STATUS: KL10 = "R"
PORT = "H"

THE KL10 ALWAYS READS THIS BIT AS "0" IF THE PORT IS PRESENT (INSTALLED AND POWERED UP).

THE PORT ALWAYS READS THIS BIT AS "0"

CSR16 PORT ID CODE 01

THIS BIT REPRESENTS BIT "01" OF THE THREE BIT "PORT IDENT CODE" FIELD.

READ/WRITE STATUS: KL10 = "R"
PORT = "H"

THE KL10 ALWAYS READS THIS BIT AS "1" IF THE PORT IS PRESENT (INSTALLED AND POWERED UP).

THE PORT ALWAYS READS THIS BIT AS "0"

CSR17 PORT ID CODE 02

THIS BIT REPRESENTS BIT "02" OF THE THREE BIT "PORT IDENT CODE" FIELD.

READ/WRITE STATUS: KL10 = "R"
PORT = "H"

THE KL10 ALWAYS READS THIS BIT AS "1" IF THE PORT IS PRESENT (INSTALLED AND POWERED UP).

THE PORT ALWAYS READS THIS BIT AS "0"

CSR18 CLEAR PORT

THIS BIT, WHEN SET BY THE KL10, CAUSES THE PORT TO BE RESET. THE MICROPROC IS HALTED AND ALL PERTINENT REGISTERS AND CONTROL LOGIC IS PLACED IN A RESET STATE.

READ/WRITE STATUS: KL10 = "W"
PORT = "*"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) CLEARS ITSELF AFTER IT HAS COMPLETED ITS RESET FUNCTION

CSR19 DIAG TEST EBUF

DIAGNOSTIC BIT WHICH ENABLES THE KL10 TO PERFORM A LOOPBACK FUNCTION OF THE EBUS INTERFACE BY LOADING AND READING THE EBUF.

IF THE PORT IS NOT RUNNING (CSR32 IS RESET) AND CSR19 IS SET:

1) A "DATAO" FROM THE KL10 WILL CAUSE THE DATA ON THE EBUS TO BE LOADED INTO THE EBUF.

2) A "DATAI" FROM THE KL10 WILL CAUSE THE DATA IN THE EBUF TO BE PLACED ON THE EBUS.

READ/WRITE STATUS: KL10 = "R/W"
PORT = "*"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR20 DIAG GEN EBUS PE

DIAGNOSTIC BIT WHICH ENABLES THE KL10 TO TEST THE EBUS PARITY CHECKER BY FORCING IT TO DECODE AN EBUS PARITY ERROR.

When this bit is set EBUS PAR ERR (CSR24) will be set on the same CONO, assuming that EBUS PARITY was actually correct.

READ/WRITE STATUS: KL10 = "R/W"
PORT = "*"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR21 DIAG SEL LAR

DIAGNOSTIC BIT WHICH ENABLES THE KL10 TO READ THE "LAR" REGISTER.

IF THIS BIT IS SET, THE PORT IS NOT RUNNING (CSR32 RESET) AND "DIAG TEST EBUF" (CSR19) IS RESET A DATAI EXECUTED BY THE KL10 WILL CAUSE THE LATCH ADDRESS REG (LAR) TO BE ASSERTED ON EBUS BITS D01-D12. ALL OTHER EBUS BITS ARE UNDEFINED.

READ/WRITE STATUS: KL10 = "R/W"
PORT = "*"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR22 DIAG SINGLE CYC

DIAGNOSTIC BIT WHICH ENABLES THE PORT MICROPROC TO BE SINGLE CYCLED.

IF THIS BIT IS SET AND THE KL10 SETS MPROC RUN (CSR32) THE PORT MICROPROC WILL EXECUTE ONE MICROCYCLE AND HALT. MPROC RUN WILL BE CLEARED WHEN THE MICROPROC HALTS.

THE CURRENT ADDRESS TO BE EXECUTED IS FETCHED FROM THE "RAR" REGISTER.

THE NEXT MICROPROC ADDRESS TO BE EXECUTED IS STORED IN THE "LAR" REGISTER AT THE COMPLETION OF THE MICROCYCLE. THE KL10 MUST READ THIS ADDRESS AND LOAD IT INTO THE "RAR" REGISTER BEFORE EXECUTING THE NEXT SINGLE CYCLE.

NOTE: THIS BIT MUST BE RESET IN ORDER FOR THE KL10 TO CORRECTLY READ AND WRITE THE CRAM

READ/WRITE STATUS: KL10 = "R/W"
PORT = "*"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR23 SPARE

RESERVED FOR FUTURE SOFTWARE USE.

READ/WRITE STATUS: KL10 = "R/W"
PORT = "*"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR24 EBUS PARITY ERR

WHEN READ BY THE KL10 THIS BIT INDICATES THAT AN EBUS PARITY ERROR HAS BEEN DETECTED.

WHEN WRITTEN AS A "1" BY THE KL10 THIS BIT WILL CLEAR ITSELF AND "CRAM PARITY ERR" (CSR06).

READ/WRITE STATUS: KL10 = "H/R/C"
PORT = "R"

SET BY:

THE DETECTION OF AN EBUS PARITY ERROR WHILE THE PORT IS READING DATA FROM THE EBUS.

CLEARED BY:

1) THE KL10 STORING A "1" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR25 FREE QUEUE ERR

THIS IS A MICROCODE-SOFTWARE DEFINED BIT USED BY THE PORT TO INFORM THE KL10 OPERATING SYSTEM THAT THERE ARE NO FREE QUEUE ENTRIES AVAILABLE ON THE FREE QUEUE. THE STATE OF THE BIT HAS NO HARDWARE FUNCTION.

READ/WRITE STATUS: KL10 = "R/C"
PORT = "R/S"

SET BY:

1) THE PORT STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "1" IN THE BIT.

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR26 DATA PATH ERR

THIS IS A MICROCODE-SOFTWARE DEFINED BIT USED BY THE PORT TO REPORT TO THE KL10 OPERATING SYSTEM THAT IT HAS DETECTED AN ERROR IN THE "DMA DATA PATH", INCLUDING THE MOVER/FORMATTER. THE STATE OF THE BIT HAS NO HARDWARE FUNCTION.

READ/WRITE STATUS: KL10 = "R/C"
PORT = "R/S"

SET BY:

1) THE PORT STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "1" IN THE BIT.

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR27 CMD QUEUE AVAIL

THIS IS A MICROCODE-SOFTWARE DEFINED BIT USED BY THE KL10 OPERATING SYSTEM TO INFORM THE PORT THAT IT HAS PLACED A COMMAND QUEUE ENTRY ON A PREVIOUSLY EMPTY COMMAND QUEUE. THE STATE OF THE BIT HAS NO HARDWARE FUNCTION.

READ/WRITE STATUS: KL10 = "R/S"
PORT = "R/C"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE PORT STORING A "1" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR28 RESP QUEUE AVAIL

THIS IS A MICROCODE-SOFTWARE DEFINED BIT USED BY THE PORT TO INFORM THE KL10 OPERATING SYSTEM THAT IT HAS PLACED A RESPONSE QUEUE ENTRY ON A PREVIOUSLY EMPTY RESPONSE QUEUE. THE STATE OF THE BIT HAS NO HARDWARE FUNCTION.

READ/WRITE STATUS: KL10 = "R/C"
PORT = "R/S"

SET BY:

1) THE PORT STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "1" IN THE BIT.

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR29 UNUSED

THIS BIT IS NOT USED BY EITHER THE PORT MICROPROC OR THE KL10.

READ/WRITE STATUS: KL10 = "*"
PORT = "*"

CSR30 DISABLE

THIS IS A MICROCODE-SOFTWARE DEFINED BIT USED BY THE KL10 OPERATING SYSTEM TO INFORM THE PORT TO PLACE ITSELF IN THE "DISABLED" STATE (SET CSR12). THE STATE OF THE BIT HAS NO HARDWARE FUNCTION.

READ/WRITE STATUS: KL10 = "R/S"
PORT = "R/C"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE PORT STORING A "1" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR31 ENABLE

THIS IS A MICROCODE-SOFTWARE DEFINED BIT USED BY THE KL10 OPERATING SYSTEM TO INFORM THE PORT TO PLACE ITSELF IN THE "ENABLED" STATE (SET CSR13). THE STATE OF THE BIT HAS NO HARDWARE FUNCTION.

READ/WRITE STATUS: KL10 = "R/S"
PORT = "R/C"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE PORT STORING A "1" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR32 MPROC RUN

THIS BIT, WHEN SET BY THE KL10, ENABLES THE PORT MICROPROCESSOR CLOCKS. THE PORT WILL START CYCLING AT THE ADDRESS CONTAINED IN THE "RAM ADDRESS REGISTER" (RAR). THE NEXT AND SUBSEQUENT ADDRESSES WILL BE FETCHED FROM THE "Y" OUTPUTS OF THE AM2910 SEQUENCER.

WHEN RESET FOR ANY REASON THE BIT CAUSES THE CRAM CONTROL REGISTER TO BE RESET, THUS PREVENTING ANY FURTHER PORT ACTIVITY AS A RESULT OF THE LAST MICROWORD.

READ/WRITE STATUS: KL10 = "R/W"
PORT = "R/H"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) AFTER EACH MICROWORD CYCLE IF "DIAG SINGLE CYC" (CSR22) IS SET.

4) CRAM PAR ERR (CSR06) OR MBUS ERR (CSR07) SETTING.

5) A GENERAL EBUS RESET.

CSR33 PIA00

THIS BIT REPRESENTS BIT "0" OF THE THREE BIT PHYSICAL INTERRUPT ASSIGNMENT FIELD (PI LEVEL 01 THRU 07) OF THE KL10'S EBUS.

READ/WRITE STATUS: KL10 = "R/W"
PORT = "R"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR34 PIA01

THIS BIT REPRESENTS BIT "1" OF THE THREE BIT PHYSICAL INTERRUPT ASSIGNMENT FIELD (PI LEVEL 01 THRU 07) OF THE KL10'S EBUS.

READ/WRITE STATUS: KL10 = "R/W"
PORT = "R"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

CSR35 PIA02

THIS BIT REPRESENTS BIT "2" OF THE THREE BIT PHYSICAL INTERRUPT ASSIGNMENT FIELD (PI LEVEL 01 THRU 07) OF THE KL10'S EBUS.

READ/WRITE STATUS: KL10 = "R/W"
PORT = "R"

SET BY:

1) THE KL10 STORING A "1" IN THE BIT.

CLEARED BY:

1) THE KL10 STORING A "0" IN THE BIT

2) THE KL10 SETTING CLEAR (CSR18)

3) A GENERAL EBUS RESET.

2.4 EBUS TO MICROPROC MUX (EMUX)

The EMUX is a two input by 36-bit wide multiplexer which takes data from either the EBUS, or the CSR REGISTER and passes it to the MBUS. The PORT MICROPROCESSOR may then strobe the data into one of its own storage medias.

When the PORT is in the "MPROC RUN" state (CSR32 set) this mux is normally enabled by the microprocessor commands, MPREADEBUS (READ EBUS) or MPREADCSR (READ CSR).

MPREADEBUS causes the contents of the EBUS to be passed to the MBUS. The PORT MICROPROC may strobe this data to one of its internal storage media.

MPREADCSR causes the contents of the CSR REGISTER to be passed to the MBUS. The PORT MICROPROC may strobe this data to one of its internal storage media.

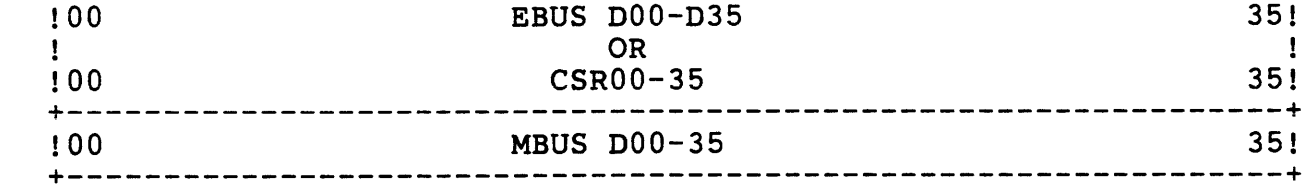
When the PORT is not in the "MPROC RUN" state the KL10 may enable the MUX by executing DATAO commands in order to:

a) write the CRAM

b) write the RAR

c) write the EBUF via the diagnostic loop-back path.

The below diagram illustrates the bit mapping of the EMUX.



2.5 MICROPROC TO EBUS MUX (KMUX)

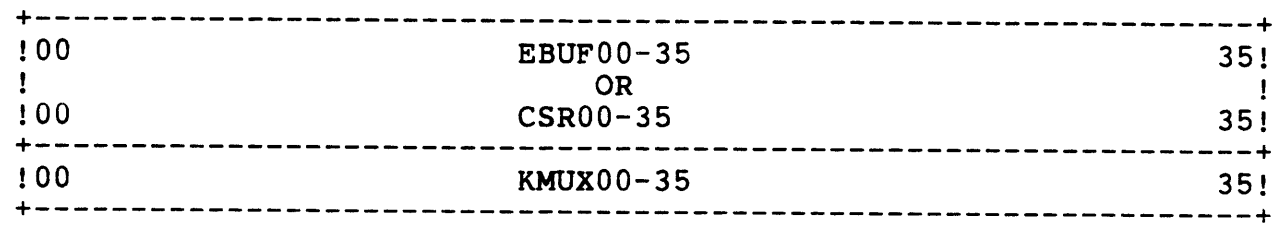
The KMUX is a two input by 36-bit wide multiplexer which takes data from either the EBUF, or the CSR REGISTER and passes it to the EBUS. The KL10 may then read the data from the EBUS.

When the PORT is not in the "MPROC RUN" state (CSR32 reset) the KL10 may enable the MUX by executing DATAI or CONI COMMANDS in order to (a) read the CRAM, (b) read the LAR, (c) read the EBUF, or (d) read the CSR.

When the PORT is in the "MPROC RUN" state (CSR32 set) the KL10 may enable the MUX by executing CONI COMMANDS in order to read the CSR, or by executing a "PI ADR IN" command during an interrupt sequence in order to read the IOP word.

The PORT MICROPROCESSOR may enable the MUX by executing the microprocessor command, MPLOADEBUS (LOAD EBUS). This command will cause the data currently asserted on the MBUS to be passed to the EBUS.

The below diagram illustrates the bit mapping of the KMUX.



2.6 EBUS PARITY GENERATOR

The EBUS PARITY GENERATOR generates odd parity for every 36-bit data word which the PORT passes to the EBUS.

The signals EBUS PARITY and EBUS PARITY ACTIVE are inhibited from being asserted on the EBUS during the transmission of an IOP FUNCTION CONTROL WORD. This is because the KL10 architecture does not permit parity checking on an IOP WORD.

2.7 EBUS PARITY CHECKER

The EBUS PARITY CHECKER normally checks every 36-bit data word which the PORT reads from the EBUS for odd parity. If parity is incorrect the EBUS PARITY ERROR bit (CSR24) in the CSR REGISTER will be set.

If CSR20 (DIAG GEN EBUS PE) is set, however, the EBUS PARITY CHECKER will check for even parity. This will cause EBUS PARITY ERROR (CSR24) to be set on the same CONO command. This feature is useful for diagnostic purposes.

2.8 EBUS TRANSCEIVERS

The EBUS Transceivers consist of OPEN-COLLECTOR 8838 type transceivers, as are currently used by other devices which interface to the EBUS.

2.9 ARITHMETIC LOGIC UNIT

The PORT MICROPROCESSOR ALU is also located on the EBUS INTFC/PORT ALU module. Therefore, even though it is logically part of the PORT MICROPROCESSOR, it will be discussed here.

The ALU may be sub-divided into:

1) Nine AM2901 type FOUR-BIT BIPOLAR MICROPROCESSOR SLICES

2) Four AM2902 type HIGH-SPEED LOOK-AHEAD CARRY GENERATORS

3) Five 74LS157 type MULTIPLEXERS used by the MICROPROCESSOR CONTROL to input a CONSTANT NUMBER FIELD into the ALU

The nine AM2901s and four AM2902s are configured in a standard parallel manner so as to form a 36-bit wide word with high speed carry look-ahead capability.

The "Y" outputs and "D" inputs of the ALU are connected directly to the MBUS, except for bits 00-09 and 26-35 of the "D" inputs. These 20 bits are connected through a two input multiplexer, the CNST MUX, to the 2901s' "D" inputs (see CONSTANT MUX for details).

The clock input to the ALU is the CPUCLOCK (CLK4 gated by MPROC RUN).

For shift operations the ALU will always shift 0s into either the MSB or the LSB, depending on the direction of the shift.

The PORT MICROPROCESSOR controls the ALU by executing the following microprocessor commands:

1) MWSORCEFLD<00-02> (ALU SOURCE INPUT FIELD (I2-I0)) - This field is the SOURCE INPUT FIELD (I2-I0) of the AM2901 ALU. Its function is defined by "THE AM2900 FAMILY DATA BOOK" as follows:

OCTAL CODE	R	S
MWSORCEFLD<00:02> = 0	A	Q
MWSORCEFLD<00:02> = 1	A	B
MWSORCEFLD<00:02> = 2	Z	Q
MWSORCEFLD<00:02> = 3	Z	B
MWSORCEFLD<00:02> = 4	Z	A
MWSORCEFLD<00:02> = 5	D	A
MWSORCEFLD<00:02> = 6	D	Q
MWSORCEFLD<00:02> = 7	D	Z

2) MWFUNCTFLD<00-02> (ALU FUNCTION FIELD (I5-I3)) - This field is the FUNCTION INPUT FIELD (I5-I3) of the AM2901 ALU. Its function is defined by "THE AM2900 FAMILY DATA BOOK" as follows:

OCTAL CODE	SYMBOL
MWFUNCTFLD<00:02> = 0	R + S
MWFUNCTFLD<00:02> = 1	S - R
MWFUNCTFLD<00:02> = 2	R - S
MWFUNCTFLD<00:02> = 3	R or S
MWFUNCTFLD<00:02> = 4	R and S
MWFUNCTFLD<00:02> = 5	$\overline{R}$ and S
MWFUNCTFLD<00:02> = 6	R xor S
MWFUNCTFLD<00:02> = 7	$\overline{R \text{ xor } S}$

3) MWDESTFLD<00-02> (ALU DESTINATION OUTPUT FIELD (I8-I6)) - This field is the DESTINATION FIELD (I8-I6) inputs to the AM2901 ALU. Its function is defined by "THE AM2900 FAMILY DATA BOOK" as follows:

OCTAL CODE	MNEMONIC
MWDESTFLD<00:02> = 0	QREG
MWDESTFLD<00:02> = 1	NOP
MWDESTFLD<00:02> = 2	RAMA
MWDESTFLD<00:02> = 3	RAMF
MWDESTFLD<00:02> = 4	RAMQD
MWDESTFLD<00:02> = 5	RAMD
MWDESTFLD<00:02> = 6	RAMQU
MWDESTFLD<00:02> = 7	RAMU

4) MWPORTAFLD<00-03> (ALU PORT "A" (A3-A0) ADDRESS FIELD) - This field is the PORT "A" (A3-A0) ADDRESS FIELD inputs to the AM2901 ALU (see "THE AM2900 FAMILY DATA BOOK")

5) MWPORTBFLD<00-03> (ALU PORT "B" (B3-B0) ADDRESS FIELD) - This field is the PORT "B" (B3-B0) ADDRESS FIELD inputs to the AM2901 ALU (see "THE AM2900 FAMILY DATA BOOK")

6) MWCARRY (MICROWORD CARRY INPUT BIT TO ALU) - This bit is the carry input to the least significant bit of the AM2901 ALU.

MWCARRY = 0 Carry "ZERO" into the LSB of the ALU
MWCARRY = 1 Carry "ONE" into the LSB of the ALU

The PORT MICROPROCESSOR monitors the ALU status by sensing the following CONDITION CODES:

1) CCFEQL0 - This CONDITION CODE indicates that the result of the last ALU operation produced all "ZEROS".

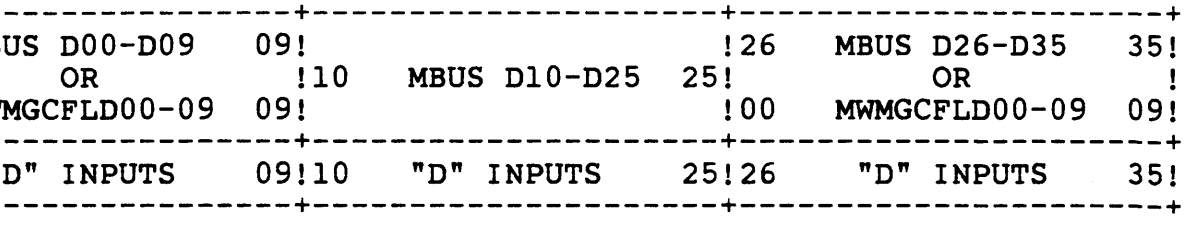
2) CCMBSIGN - This CONDITION CODE indicates that the SIGN (MSB or bit 00) was set as the result of the last ALU operation.

2.9.1 CONSTANT MUX

The CNST MUX is a two input multiplexer which allows either MBUS D00-D09 and D26-D35, or MWMGCFLD00-09 of the MICROWORD to be loaded into the ten MSBs and the ten LSBs of the "D" inputs to the MICROPROC ALU (AM2901s). This enables the MICROPROCESSOR to load a CONSTANT NUMBER VALUE into the ALU's 10 most significant and 10 least significant bits. MBUS D10-D25 are always loaded into the corresponding "D" inputs.

The PORT loads the MWMGCFLD field of the CRAM CONTROL WORD, instead of the MBUS DATA bits, into the ALU's "D" inputs by executing the microprocessor command, SELCNSTFLD (SELECT CONSTANT FIELD) (setting the MWSKIPFLD to 24 or 34).

The below diagram illustrates the bit mapping of this MUX.



3.0 CBUS/DATA MOVER INTFC MODULE (CMVR)

The CBUS/DATA MOVER INTFC MODULE (CMVR) consists primarily of:

a) A data path between the KL10's CBUS and the PACKET BUFFER'S PLI INTERFACE, which may be further sub-divided into:

   1) CBUS INTERFACE BUFFERS, CBUS PARITY GEN/CHKRS and associated CBUS CONTROL LOGIC

   2) PLI INTERFACE BUFFERS, PLI PAR GEN/CHKR and associated PLI CONTROL LOGIC

   3) A DATA FORMATTER and MOVER which resides between the CBUS and the PLI INTERFACE, and has the function of mapping 8-bit PLI bytes into 36-bit KL10 words, and vice versa.

b) A data path between the CBUS/DATA MOVER INTFC MODULE (CMVR) and the PORT MICROPROCESSOR which enables the MICROPROCESSOR to:

   1) Load or read the DATA FORMATTER and MOVER (MVR/FMTR)

   2) Load or read the PACKET BUFFERS via the PLI INTERFACE

C) CMVR CONTROL LOGIC which decodes and executes the commands specified by the MICRPPROCESSOR CONTROLLER'S microword.

The PORT MICROPROCESSOR accesses the CMVR MODULE by executing microprocessor commands. These commands are decoded functions of the MWBUSCTLFLD field and the MWMGCFLD field of the CRAM CONTROL WORD.

The PORT MICROPROCESSOR monitors the CMVR CONTROL LOGICS status by sensing CONDITION CODES.

3.1 CBUS TIMING DESKEW PROCEDURE

The following deskew procedure should be performed before CBUS TRANSFERS are attempted.

a) Equipment required - Tektronics 475 or equivalent (100 MHz) scope with identical probes and short ground clips.

b) Objectives - To deskew the CI20 to the MBOX clock that produces channel time zero, CHT0.

c) Notes - Recheck skew whenever the CBUS cable is replaced.

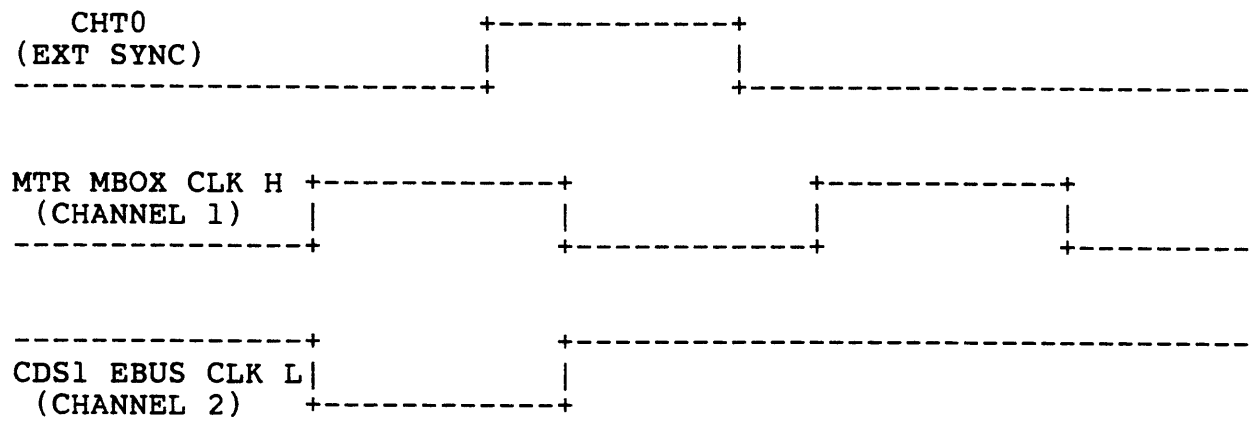
d) Adjustment procedure -

1) Attach a probe (either ext, sync or channel 3) to CHT0 H, 4B09K1

2) Sync positive external

3) Attach channel 1 probe to MTR MBOX CLK H, 4D33P1

4) Push TRIGGER VIEW and verify that the MBOX CLK that occurs just prior to CHT0 can be seen on the scope. See diagram below:



5) Attach channel 2 probe to CDS1 EBUS CLK L, 2A15F1

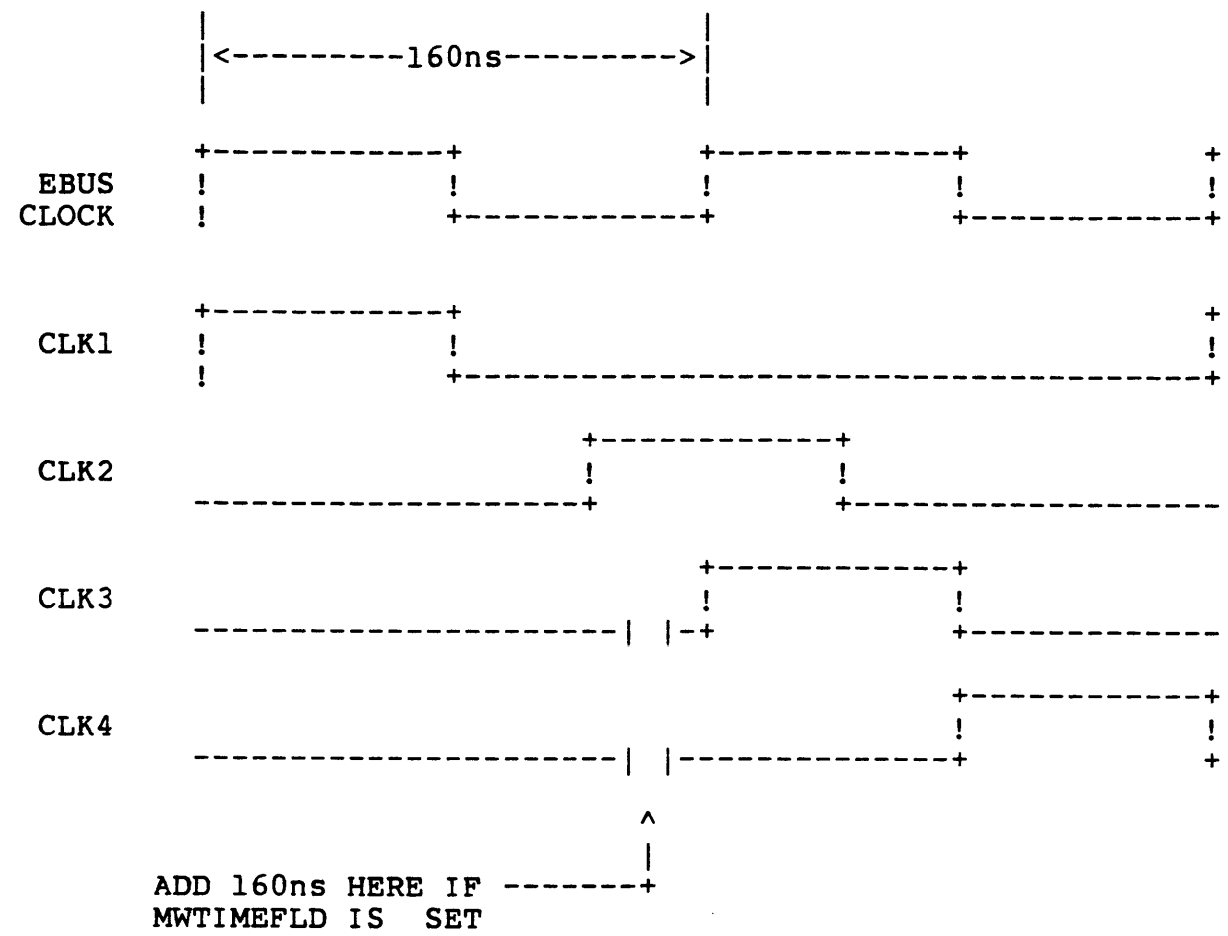
6) Align clock pulse on channel 2 with the MBOX clock that occurs approximately 10 nanoseconds before CHT0.

3.2 CMVR CONTROL LOGIC

The CMVR CONTROL LOGIC decodes and executes the commands specified by the MICRPPROCESSOR CONTROLLER'S microword.

It also generates and distributes the PORT clocks (CLK1, CLK2, CLK3 and CLK4) to all three PORT modules.

The timing for the clocks is derived from the KL10's EBUS CLOCK. The below diagram illustrates the timing relationship of the clocks when the KL10's EBUS CLOCK is operating at the normal 160ns cycle time.



CLK1 normally strobes the next MICROWORD into the PORT'S MICROWORD CONTROL REGISTER. It also has several other timing functions, depending on the specific operation being executed.

CLK2,CLK3 & CLK4 are generally used by the various control logics to execute the function which is specified by the MICROWORD.

All four clocks are gated by control logic on the PORT MICROPROC CONTROL module. The gated clocks are known as RUNCLK1, RUNCLK2, RUNCLK3 and RUNCLK4 respectively. CLK4 also generates CPUCLOCK, which is used as the clock input to the AM2901 ALU. The gating of these clocks allows the PORT to be started, stopped and single cycled in an orderly manner.

A microword bit, MWTIMEFLD (TIME EXTENTION FIELD), may be set on any specific microcycle. Setting this bit will cause CLK3 and CLK4 to occur 135ns later than normal for that microcycle, thus lengthening the microcycle from 270ns to 405ns. This feature allows more execution time for the microcycle, thus offering an easy solution for timing problems which may arise due to insufficient microcycle execution time on any specific micro-instruction.

3.3 MICROPROC TO CMVR REGISTER (CBUF)

The "MICROPROCESSOR TO CMVR BUFFER" (CBUF) is a 36-bit buffer normally used by the PORT MICROPROCESSOR to pass data from the MBUS (Internal tri-state MICROPROCESSOR BUS) to the CBUS/DATA MOVER INTFC MODULE (CMVR).

The CBUF acts only as an isolation buffer to the tri-state MBUS and is logically transparent to the PORT MICROPROCESSOR.

3.4 DATA FORMATTER and MOVER (MVR/FMTR)

The MVR/FMTR consists of a series of parallel/serial shift registers and their associated control. It may be:

a) Parallel loaded and read as a 36-bit register from the PORT MICROPROCESSOR.

b) Parallel loaded and read as a 36-bit register from the CBUS.

c) Parallel read as an 8-bit register by the PLI INTERFACE.

d) Serially loaded and left shifted (from LSB to MSB), four or eight bits at a time, from the PLI INTERFACE. In this mode data is not wrapped around, but is shifted out of and lost from the MSB.

e) Serially loaded and right shifted (from MSB to LSB), four or eight bits at a time, from the PLI INTERFACE. In this mode the data may be wrapped around from LSB to MSB. Additional shifts may be used to re-align the data in the desired format.

Four bit nibbles from the PLI INTERFACE are shifted up from the bottom or down from the top of the MVR/FMTR. Two shifts load one 8-bit byte. Logic hardware enables the PORT MICROPROCESSOR to execute one shift or two shifts on a single microcycle. By loading four bit nibbles from the PLI INTFC and shifting either up or down 36-bit KL10 words may be formed and parallel transferred to the CBUS.

If the PLI INTERFACE is loaded from the bottom of the MVR/FMTR and shifted up the data is not wrapped around, but is shifted out and lost from the 4 MSBs of the MVR/FMTR.

If the PLI INTERFACE is loaded from the top and shifted down the data may be wrapped around such that the 4 LSBs of the MVR/FMTR feed back into the 4 MSBs of the MVR/FMTR. In this manner data may be shifted indefinitely around the MVR/FMTR and re-aligned in the desired manner.

The PORT MICROPROCESSOR has the ability to select either the four MSBs or the four LSBs of the PLI byte to input first to the MVR/FMTR.

36-bit KL10 words may be loaded into the MVR/FMTR from the CBUS. Once a word is loaded the PORT MICROPROC can shift it, either up or down, into the PLI OUT REG for transfer to the PLI INTERFACE.

The associated commands which are used to control the MVR/FMTR are as follows:

1) MPSHFTFMTR8 (SHIFT FORMATTER BY 8 BITS) causes the contents of the MVR/FMTR to be shifted eight bits to the left or right, depending on the state of the command "MPSHIFTRIGHT".

2) MPSHFTFMTR4A (SHIFT FORMATTER BY 4 BITS) causes the contents of the MVR/FMTR to be shifted four bits to the left or right, depending on the state of the command "MPSHIFTRIGHT".

3) MPSHFTFMTR4B (SHIFT FORMATTER BY 4 BITS) causes the contents of the MVR/FMTR to be shifted four bits to the left or right, depending on the state of the command "MPSHIFTRIGHT".

4) MPCBUFTOFMTR (CBUF TO FORMATTER) causes the data which has been previously stored in the CBUF to be loaded into the MVR/FMTR REGISTER

5) MPPLINTOFMTR (PLI INPUT BUFFER TO FORMATTER) causes the 8-bit data byte which is currently stored in the PLI INPUT BUFFER to be shifted into the serial input lines of the MVR/FMTR, four bits at a time. This command must be executed in conjunction with MPSHFTFMTR4A or MPSHFTFMTR8. If MPSHFTFMTR4A is executed, then only four bits from the PLI INPUT BUFFER will be shifted into the

serial input lines of the MVR/FMTR. If MPSHFTFMTR8 is executed, then all 8 bits from the PLI INPUT BUFFER will be shifted into the serial input lines. Either the four MSBs or the four LSBs are shifted in first, depending on the state of the command, MPRHTNIBFIRST. If an MPSHFTFMTR4A is executed, then only the first four bits specified by the state of MPRHTNIBFIRST will be shifted into the serial input lines. If MPSHIFTRIGHT is asserted the 4-bit nibbles will be shifted into the four MSBs of the MVR/FMTR and shifted right. Otherwise, they will be shifted into the four LSBs of the MVR/FMTR and shifted left.

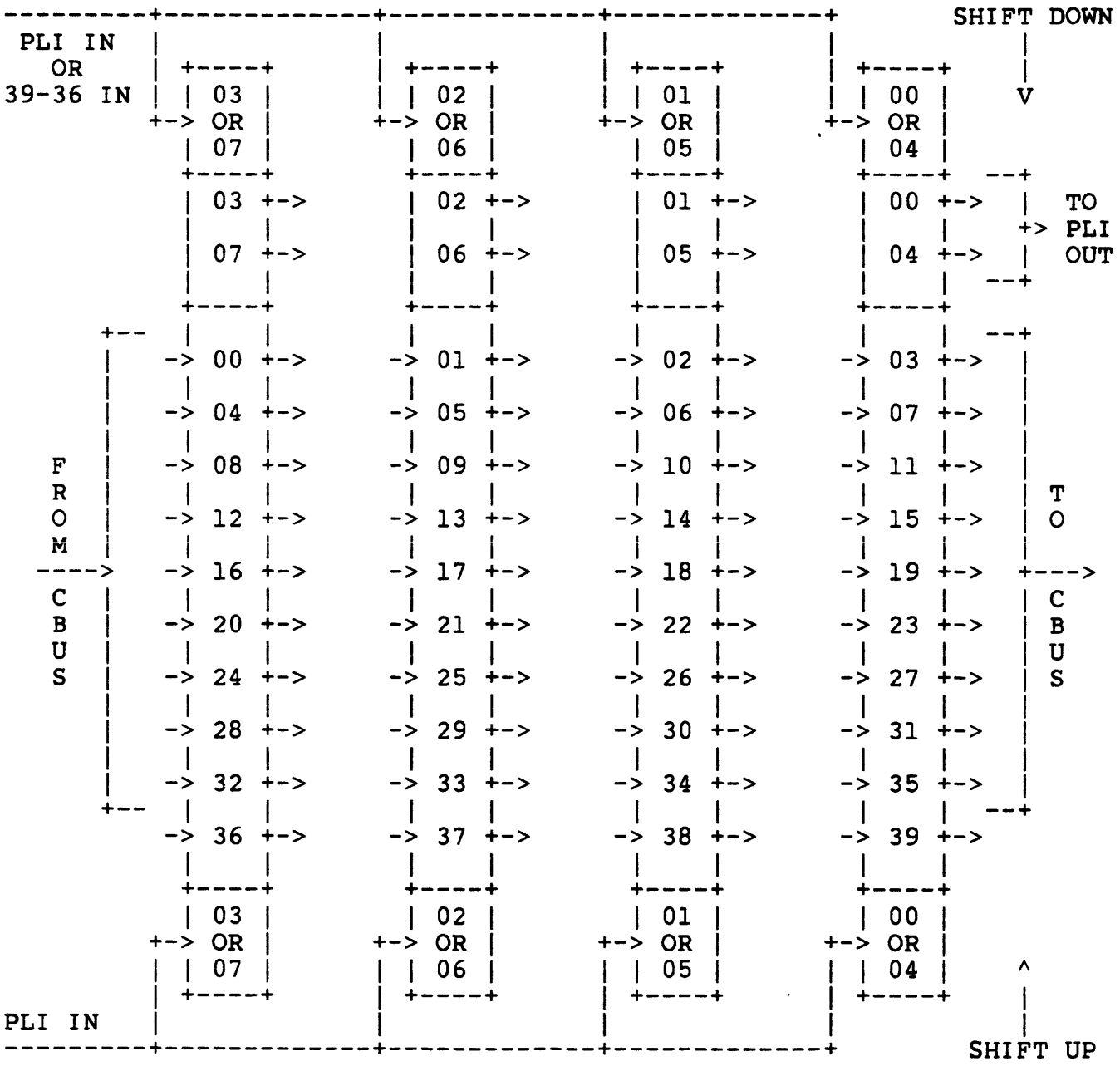
6) MPFMTRTOPLOUT (FORMATTER TO PLI OUTPUT BUFFER) causes the 8-bit data byte which is currently stored in the PLI OUTPUT REG of the MVR/FMTR to be loaded into the PLI OUTPUT BUFFER.

7) MPSHIFTRIGHT (SHIFT RIGHT) when asserted, either the currently selected nibble in the PLI INPUT BUFFER, or the four LSBs (MVROUT36-39) are shifted into the four MSB serial input lines of the MVR/FMTR and shifted right. The actual data shifted into the MVR/FMTR depends on the state of MPPLINTOFMTR. If MPPLINTOFMTR is asserted, then the currently selected 4-bit nibble in the PLI INPUT BUFFER will be shifted into the four MSBs of the MVR/FMTR and shifted right. If MPPLINTOFMTR is de-asserted, then MVROUT36-39 will be shifted into the four MSBs of the MVR/FMTR and shifted right. Two 4-bit shifts may be executed during one microcycle, thus enabling an 8-bit byte to be right shifted during the same microcycle (see MPPLINTOFMTR for more details).

8) MPRHTNIBFIRST (RIGHT NIBBLE FIRST) when asserted, the four LSBs of the PLI INPUT BUFFER are shifted into the serial input lines of the MVR/FMTR first. When de-aserted, the four MSBs of the PLI INPUT BUFFER are shifted into the serial input lines of the MVR/FMTR first.

9) MPZEROLFTNIB (ZEROLEFTNIBBLE) when asserted, causes the four MSBs from the PLI OUTPUT REG of the MVR/FMTR to be forced to "zeros" before they are loaded into the PLI OUTPUT BUFFER.

The below diagram represents the the MVR/FMTR's basic principle of operation.

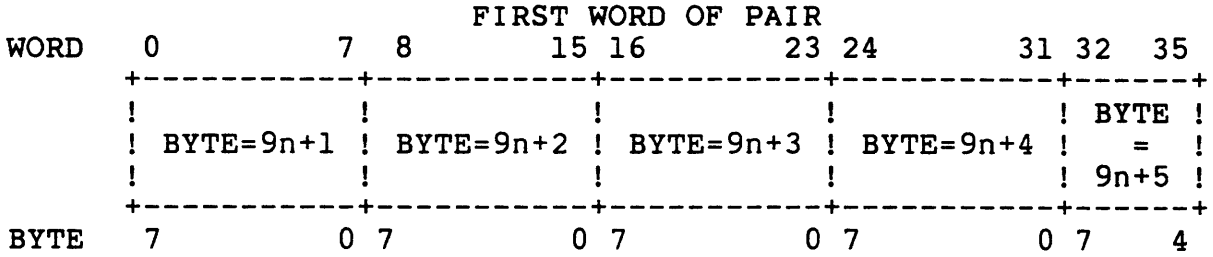


The MVR/FMTR is capable of supporting three different data formats as follows:

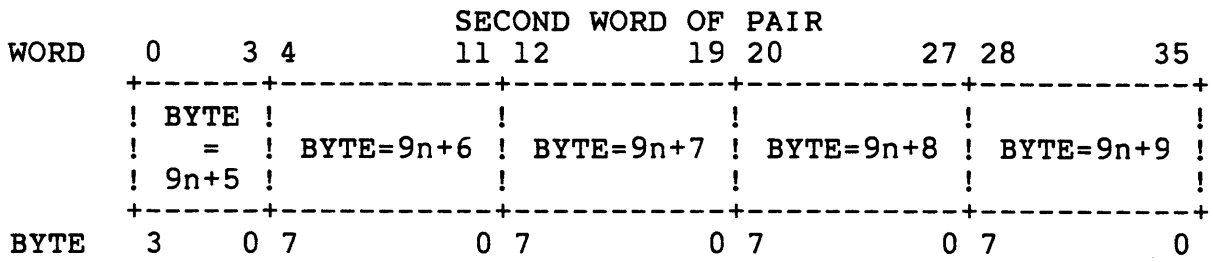
a) HIGH DENSITY - 4 1/2 8-bit bytes per 36-bit word as follows:

KL10 WORD PAIR

FIRST WORD OF PAIR



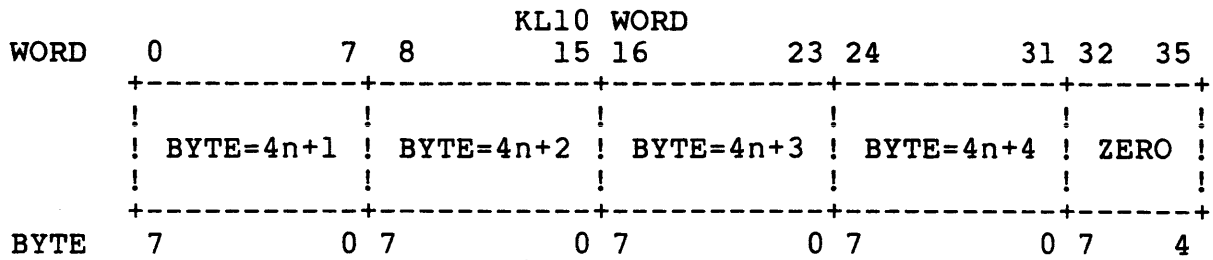
SECOND WORD OF PAIR



n = Number of complete 36-bit KL10 word pairs processed since the start of byte transfers.

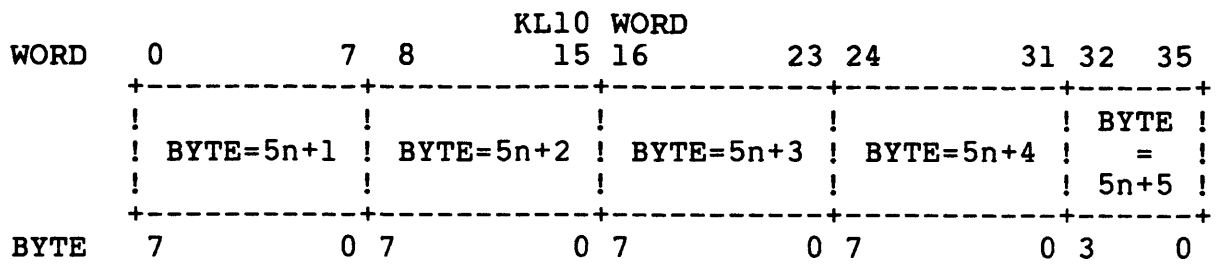
b) INDUSTRY COMPATIBLE - 4 8-bit bytes per 36-bit word (bits 32-35 of the word are ZERO stuffed as follows:

KL10 WORD



n = Number of complete 36-bit KL10 words processed since the start of byte transfers.

c) CORE DUMP - 5 8-bit bytes per 36-bit word (four bits of every 5th byte are discarded as follows:



n = Number of complete 36-bit KL10 words processed since the start of byte transfers.

These data formats are implemented via different microcode subroutines which reside in the MICROPROCESSOR CONTROLLER's CRAM.

3.5 CBUS INPUT BUFFER

The CBUS INPUT BUFFER is a latched 38-bit (36 data bits + 2 parity bits) register which is used to pass data from the CBUS to the MVR/FMTR.

The CBUS may load the register whenever the MVR/FMTR is not reading it. Likewise, the MVR/FMTR may read the register whenever the CBUS is not loading it.

The contents of the CBUS INPUT BUFFER is normally strobed into the MVR/FMTR by the microprocessor command, MPCBINTOFMTR (CBUS INPUT BUFFER TO FORMATTER), at CLK2 time if the CONDITION CODE, CCCBUSAVAIL (CBUS AVAILABLE) was asserted on the previous microcycle.

3.6 CBUS OUTPUT BUFFER

The CBUS OUTPUT BUFFER is a latched 38-bit (36 data bits + two parity bits) register which is used to pass data from the MVR/FMTR to the CBUS.

The CBUS may read the register whenever the MVR/FMTR is not loading it. Likewise, the MVR/FMTR may load the register whenever the CBUS is not reading it.

Odd parity for each half of the word is generated and loaded into the register whenever data is passed to it.

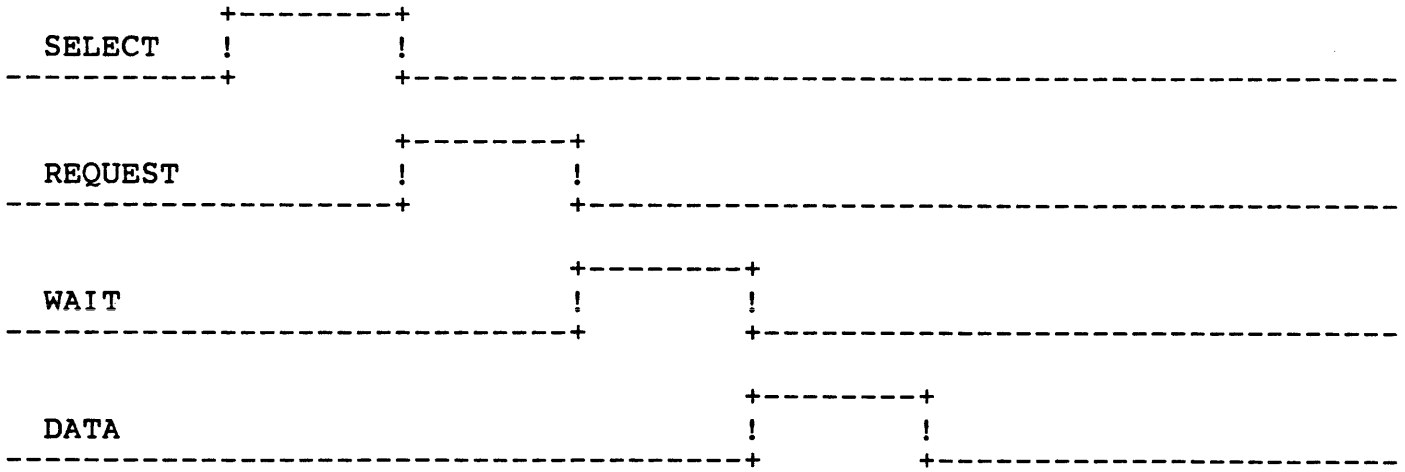
The CBUS OUTPUT BUFFER is normally loaded from the MVR/FMTR by the microprocessor command, MPFMTRTOCBOUT (FORMATTER TO CBUS OUTPUT BUFFER), at CLK2 time if the CONDITION CODE, CCCBUSAVAIL was asserted on the previous microcycle.

3.7 CBUS CONTROL LOGIC

The CBUS CONTROL LOGIC arbitrates the CBUS protocol, the PORT MICROPROCESSOR protocol for starting and stopping the CBUS, and the synchronization functions between the CBUS and the MVR/FMTR. The clock timing for the PORT is also generated by this logic.

The CBUS is a synchronous high speed time division multiplexed tri-state data bus. Each device on the CBUS has its own unique time slot.

There are four cycles to a CBUS data transfer as shown by the below diagram.



a) SELECT - The CBUS SELECT line for the PORT is asserted by the KL10's DATA CHANNEL when the PORT's time slot occurs. The CBUS CONTROL LOGIC senses when its SELECT line is active.

b) REQUEST - The CBUS REQUEST line is asserted by the PORT during this cycle if it has detected its SELECT line and is ready to make a data transfer. Otherwise, this CBUS data transfer cycle is ignored by the PORT.

c) WAIT - This is a dummy cycle for the PORT. No CBUS function is executed.

d) DATA - Data is placed on the CBUS data lines by either the KL10 or the PORT (depending on the direction of transfer) during this cycle. Otherwise, it is ignored by the PORT.

The PORT microcode starts the CBUS DATA CHANNEL by executing the command, MPSTARTCBUS (START CBUS). If the transfer is to the KL10's memory the PORT microcode also executes the command MPWRITEMEM (WRITE TO KL10 MEMORY). These commands are latched by the CMVR MODULE for execution by the PORT CBUS CONTROL when its CBUS slot is detected.

When the PORT's CBUS CONTROL LOGIC detects its CBUS SELECT line is asserted and the CBUS READY line is negated it starts the CHANNEL by asserting CBUS START and CBUS RESET during the subsequent DATA cycle. It then clears the corresponding latches previously set by the PORT's microcode. It also asserts CBUS CTOM at this time if the transfer is to KL10 memory (the PORT MICROPROC executed an MPWRITEMEM command). It does not clear this latch, however, until the data transfer is complete.

When the CHANNEL is ready to transfer data over the CBUS, it asserts CBUS READY during the PORT's DATA cycle.

After receiving CBUS READY, the PORT CBUS CONTROL asserts CBUS REQUEST during its REQUEST cycle whenever it requires a data word from the CHANNEL (device write), or whenever it requires that the CHANNEL accept a data word (device read). The words are asserted on the CBUS DATA lines during the PORT's DATA cycle following it corresponding REQUEST cycle.

The PORT will be READY to transfer data across the CBUS whenever its CBUS INPUT BUFFER is empty, or whenever its CBUS OUTPUT BUFFER is full.

The CBUS INPUT BUFFER is emptied (transferred to the MVR/FMTR) by the PORT microcode executing an MPCBINTOFMTR command when it senses the CONDITION CODE, CCCBUSAVAIL and is prepared to accept data from the CBUS.

The CBUS OUTPUT BUFFER is loaded (the MVR/FMTR contents transferred to it) by the PORT microcode executing an MPFMTRTOCBOUT command when it senses the CONDITION CODE, CCCBUSAVAIL and has data available for transfer to the CBUS.

When the CHANNEL places the last word on the CBUS during a device write operation, it asserts CBUS LAST WORD. In response, the PORT CBUS CONTROL asserts CONDITION CODE, CCCBLSTWD.

When CCCBLSTWD is detected by the PORT's microcode. The microcode responds by executing the command MPSTOPCBUS (STOP CBUS). This will cause the PORT CBUS CONTROL to assert CBUS DONE during its next DATA cycle. No more data requests will be made during subsequent REQUEST cycles.

CBUS DONE causes the CHANNEL to terminate the operation. CBUS READY is negated when the CHANNEL is prepared to begin another data transfer.

The microcode may also execute the command MPSTORECBUS (STORE CBUS STATUS INFORMATION) on the same microcycle that it executes an MPSTOPCBUS command. This will cause the PORT CBUS CONTROL to also assert CBUS STORE on the next DATA cycle. This forces the channel status to be stored in the channel's assigned reset and status logout area.

NOTE: The command MPSTORECBUS should never be asserted unless MPSTOPCBUS is also asserted during the same microcycle.

Also, the microcode may execute the command MPSTOPCBUS and MPSTORECBUS when it has transferred all data over the CBUS during a device read operation, or when it detects a transfer error (CONDITION CODES CCCBUSPARERR, CCCMVRPARCHK, CCCHANERR or CCPLIPARERR set) during a read or write. This will also cause the PORT CBUS CONTROL to assert CBUS DONE and CBUS STORE. No more data requests will be made during subsequent REQUEST cycles.

3.8 CBUS OUT PARITY GENERATOR

The CBUS OUT PARITY GENERATOR generates odd parity for each 18-bit half word which the MVR/FMTR passes to the CBUS OUTPUT BUFFER. The two PARITY bits for a complete 36-bit word are latched into the buffer for output to the CBUS.

3.9 CBUS IN PARITY CHECKER

The CBUS IN PARITY CHECKER checks each 18-bit half word which the MVR/FMTR reads from the 36-bit CBUS INPUT BUFFER for odd parity. If parity is incorrect CCCBUSPARRERR (CBUS PARITY ERROR) is generated and passed to the MICROPROCESSOR CONTROLLER's CC MUX. This CONDITION CODE remains latched until the MICROPROCESSOR CONTROLLER clears it.

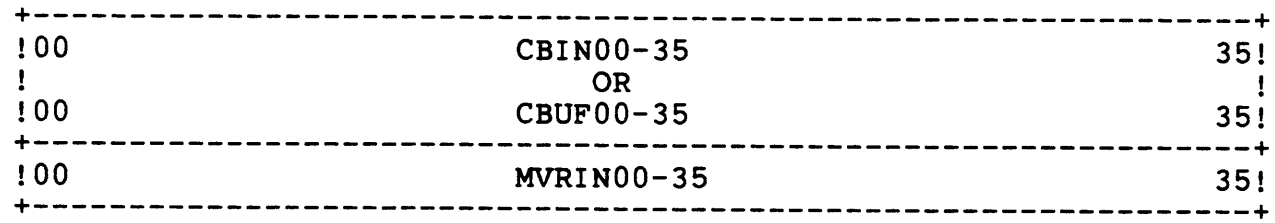
When the PORT MICROPROCCESSOR senses that the condition code, CCCBUSPARERR (CBUS PARITY ERROR), is set it should set CSR26 (DATA PATH ERR) in the CSR REGISTER.

3.10 DATA INPUT MUX (DMUX)

The DMUX is a two input by 36-bit wide multiplexer which takes data from either the CBUS INPUT BUFFER, or the CBUF and passes it to the MVR/FMTR as a 36-bit word.

Its select path is controlled by the PORT MICROPROC's command, MPCBUFTOFMTR (CBUF TO FORMATTER). When this command is executed the CBUF path will be enabled. For all other conditions the CBIN path is enabled.

The below diagram illustrates the bit mapping of this MUX.



3.11 PLI OUTPUT BUFFER

The PLI OUTPUT BUFFER is a latched 9-bit (8 data bits + one parity bit) register which is used to pass data from the MVR/FMTR to the PLI BUS.

The register is loaded by the PORT, either from the PLI OUTPUT BYTE of the MVR/FMTR or the 8 least significant bits of the MBUS whenever it has an 8-bit byte assembled and ready for transfer to the PLI BUS. Odd parity is generated and loaded into a holding flip flop when the register is loaded.

The commands used to load this register by the PORT MICROPROC are MPCBUFTOPLOUT (CBUF TO PLI OUTPUT BUFFER) and MPFMTRTOPLOUT (FORMATTER TO PLI OUTPUT BUFFER) . The register is loaded at CLK4 and CLK2 times respectively of these commands.

Once the register is loaded the PORT may enable the tri-state outputs of the register via an MPXMITPLI (TRANSMIT PLI) command, thus placing the data on the PLI BUS.

3.12 PLI INPUT BUFFER

The PLI INPUT BUFFER is a latched 9-bit (8 data bits + one parity bit) register which is used to pass data from the PLI BUS to the MVR/FMTR.

The register is loaded from the PLI BUS by CLK4, via an MPRECVPLI (RECEIVE PLI) command, whenever an 8-bit byte is present for input from the PLI BUS. Parity from the PLI BUS is loaded into a holding flip flop each time the register is loaded.

Once the register is loaded the PORT may transfer the data to the MVR/FMTR (four bits at a time) with the commands "MPPLINTOFMTR", "MPLFTNIBFIRST", "MPSHIFTRIGHT", etc.

The PORT MICROPROCESSOR may also transfer the data as an 8-bit byte to the 8 least significant bits of the MBUS for transfer to one of the MICROPROCESSOR CONTROLLER's internal storage media. This is accomplished by the command, MPPLINTOCBUF (PLI INPUT BUFFER TO CBUF).

3.13 PLI PARITY OUT GENERATOR

The PLI PARITY OUT GENERATOR normally generates odd parity for every 8-bit data byte which the PORT passes to the PLI OUTPUT BUFFER. The PARITY bit is latched into a holding flip flop for output to the PLI BUS.

For diagnostic purposes the command, MPTESTPLIPAR (TEST PLI PARITY GENERATOR), will force the PLI PARITY GENERATOR to generate even parity.

3.14 PLI PARITY IN CHECKER

The PLI PARITY IN CHECKER checks every 8-bit data byte which the PORT reads from the PLI INPUT BUFFER for odd parity. If parity is incorrect CCPLIPARERR (PLI PARITY ERROR) is generated and passed to the MICROPROCESSOR CONTROLLER's CC MUX. CCPLIPARERR remains latched until the MOCROPROCESSOR CONTROLLER clears it.

When the PORT MICROPROCCESSOR senses that the condition code, CCPLIPARERR, is set it should set CSR26 (DATA PATH ERR) in the CSR REGISTER.

3.15 PLI CONTROL LOGIC

The PLI CONTROL LOGIC arbitrates the PLI INTERFACE protocol, the PORT MICROPROCESSOR protocol for accessing the PLI INTERFACE, and the synchronization functions between the PLI INTERFACE and the CMVR.

The PLI LINK CONTROL LINES, PLI SELECT and PLI CLOCK are also generated by this logic.

PLI CLOCK is always generated at CLK3 time on every microcycle. The remaining commands which are used to control the PLI CONTROL LOGIC are generated by the PORT MICROPROCESSOR as follows:

1) PLI LINK CONTROL 0 passes PLI LINK CONTROL 0 to the PLI BUS

2) PLI LINK CONTROL 1 passes PLI LINK CONTROL 1 to the PLI BUS

3) PLI LINK CONTROL 2 passes PLI LINK CONTROL 2 to the PLI BUS

4) PLI LINK CONTROL 3 passes PLI LINK CONTROL 3 to the PLI BUS

5) MPSELECTPLI (SELECT PLI) asserts the PLI SELECT line.

The PLI INTERFACE signals which are supported are as follows:

a) PLI DATA <7:0>
b) PLI SELECT
c) PLI RCVR BUF A FULL
d) PLI RCVR BUF B FULL
e) PLI XMTR ATTN
f) PLI LINK CONTROL <3:0>
g) PLI TRAN DATA PAR
h) PLI RECV DATA PAR
i) PLI CLOCK
j) PLI INITIALIZE

The following PLI INTERFACE signals are not supported. The same status information which is normally passed by these lines may be obtained via the PLI DATA <7:0> lines when the PLI LINK CONTROL lines are set to the corresponding selection code:

a) PLI RECEIVER STATUS
b) PLI TRANSMIT STATUS

A detailed description of the PLI INTERFACE protocol will not be included in this specification. This information may be obtained from the "PILA HARDWARE SPECIFICATION".

3.16 PLI SERIAL UP MUX (SUMUX)

The SUMUX is a two input multiplexer which takes 4-bit nibbles from either the four LSBs, or the four MSBs of the 8-bit bytes stored in the PLI INPUT BUFFER and passes them to the four LSB serial input lines of the MVR/FMTR. The nibbles can then be left shifted to form 36-bit words.

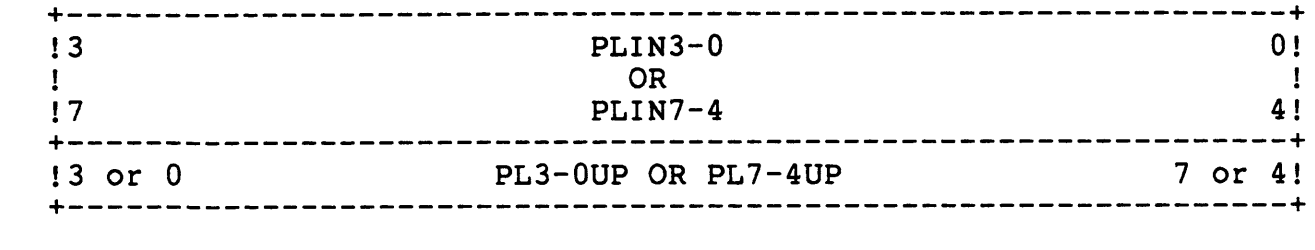
The 4-bit nibbles from the SUMUX are also passed to the SDMUX (see section PLI SERIAL DOWN MUX for details).

The four MSBs of the MVR/FMTR are shifted out and lost on each left shift command.

Two 4-bit nibbles may be processed per microcycle, thus enabling one byte from the PLI INPUT BUFFER to be input to the MVR/FMTR on one microcycle.

The associated commands which control the SUMUX are MPPLINTOFMTR (enables the SUMUX) and MPRHTNIBFIRST (selects the four LSBs of the PLI INPUT BUFFER as the SUMUX inputs). The functions of these commands are described in detail elsewhere in this specification.

The below diagram illustrates the bit mapping of this MUX.



3.17 PLI SERIAL DOWN MUX (SDMUX)

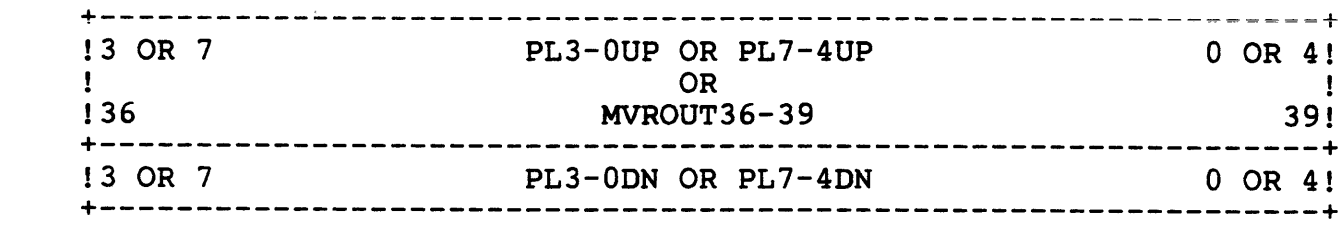
The SDMUX is a two input multiplexer which takes 4-bit nibbles from either the SUMUX or the four LSBs of the MVR/FMTR parallel outputs (MVR0UT36-39) and passes them to the four MSB serial input lines of the MVR/FMTR. The nibbles can then be right shifted to form 36-bit words.

By shifting MVROUT36-39 back into the four MSBs of the MVR/FMTR data may be wrapped around indefinitely on right shift commands. This feature enables repetitive information to be retained and shifted to any desired position within the MVR/FMTR.

Two 4-bit nibbles may be processed per microcycle, thus enabling one byte from the PLI INPUT BUFFER to be input to the MVR/FMTR on one microcycle.

The associated commands which control the SDMUX are MPPLINTOFMTR (selects the output of the SUMUX for input to the SDMUX) and MPSHIFTRIGHT (enables the SDMUX). The functions of these commands are described in detail elsewhere in this specification.

The below diagram illustrates the bit mapping of this MUX.



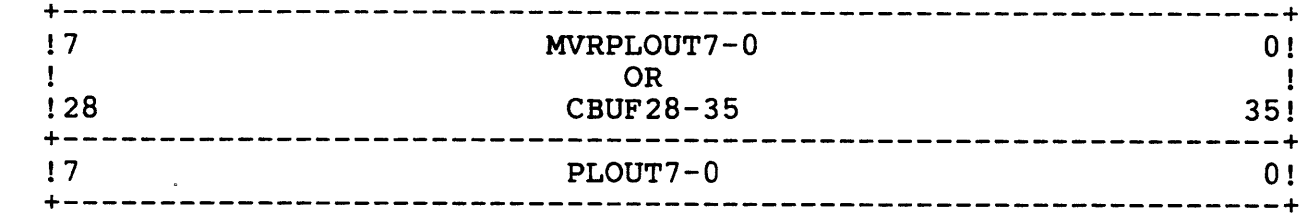
3.18 PLI OUTOUT MUX (PMUX)

The PLI OUTPUT MUX (PMUX) is a two input by 8-bit wide multiplexer which takes data from either the MVR/FMTR's 8-bit PLI OUTPUT path, or the eight least significant bits of the CBUF and passes it to the PLI OUTPUT BUFFER as 8-bit bytes.

The the CBUF input leg of the PMUX is selected by the command, MPCBUFTOPLOUT (CBUF TO PLI OUTPUT BUFFER). When this command is executed by the PORT MICROPROCESSOR, MBUS D28-D35 are passed to the PLI OUTPUT BUFFER, via the CBUF. For all other conditions the PLOUT OUTPUT BUFFER of the MVR/FMTR is passed to the PLI OUTPUT BUFFER.

The output of the four MSBs of the PMUX may be forced to "zeros", regardless of the contents of the MVR/FMTR's PLI OUTPUT path. This function is executed by the command, MPZEROLFTNIB (ZERO LEFT NIBBLE). This feature is useful for CORE DUMP mode, where the four MSBs of each fifth byte must be "zero".

The below diagram illustrates the bit mapping of this MUX.



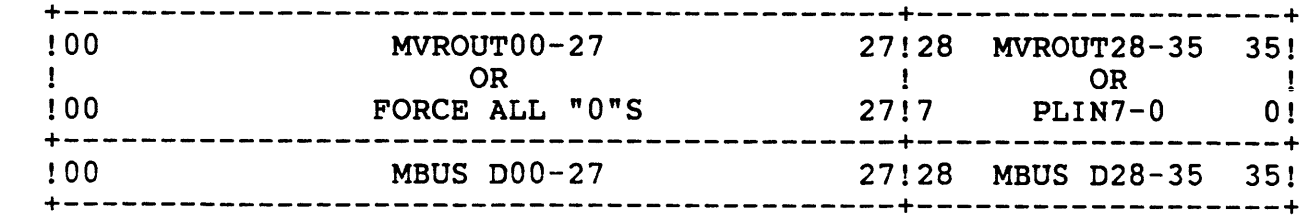
3.19 CMVR TO MICROPROC MUX (CMUX)

The CMVR TO MICROPROC MUX (CMUX) is a two input by 36-bit wide multiplexer which interfaces to the PORT'S internal MBUS. It enables the PORT MICROPROCESSOR to read the following registers.

1) MVR/FMTR - as a 36-bit word register. The respective MICROPROC command is MPENACMUX (enables the CMUX).

2) PACKET BUFFERS, via the PLI INPUT BUFFER, as 8-bit bytes. The PLI INPUT BUFFER is input to the 8 least significant bits of the CMUX. The 28 most significant bits are forced to "0". The respective MICROPROC commands are MPENACMUX (enables the CMUX) and MPPLINTOCMUX (selects the PLI INPUT BUFFER as input to the 8 LSBs of the CMUX).

The below diagram illustrates the bit mapping of this MUX.



3.20 PARITY PREDICTOR

It is not possible to propagate correct parity through the MVR/FMTR since it consists of a series of serial shift registers which may change the data format in several different ways.

The PARITY PREDICTOR is a circuit which enables the PORT MICROPROCESSOR to verify data integrity through the MVR/FMTR by a combination of hardware and microcode functions which predict correct parity.

The hardware portion of the PARITY PREDICTOR consists of a J-K FLIP FLOP, two 4-bit parity checkers and some related control logic. The J-K FLIP FLOP is toggled each time a parity bit (1) is detected at either the CBUS or the PLI INTERFACE, regardless of the direction of data transfer.

The output of the J-K FLIP FLOP is monitored by the MICROPROCESSOR as the condition code, CCMVRPARCHK.

Several MICROPROCESSOR COMMANDS are used to control the PARITY PREDICTOR. Most of them are also simultaneously used for other functions and, therefore, are transparent to the microcode. Two unique COMMANDS are required, however, in order to properly control the PARITY PREDICTOR. They are as follows:

1) MPINDSTCOMP (INDUSTRY COMPATIBLE MODE) sets the PARITY PREDICTOR to operate correctly in the INDUSTRY COMPATIBLE MODE. It enables the PARITY PREDICTOR to calculate correct parity for CBUS D32-D35, which are not passed through the MVR/FMTR in INDUSTRY COMPATIBLE MODE. The value of this parity is then subtracted from the value of the parity for the entire CBUS WORD. This command must be executed in INDUSTRY COMPATIBLE MODE when transferring data from the CBUS to the PLI INTERFACE. Otherwise, the results of the PARITY PREDICTOR will not be correct.

2) MPCLRPARCHK (CLEAR PARITY CHECK) clears CCMVRPARCHK

It is possible to develop different microcode algorithms which can accurately predict what CCMVRPARCHK should equal, based on the number of parity bits detected at the CBUS and PLI INTERFACES during data transfers. These microcode algorithms are slightly different for each of the three data format modes.

If CCMVRPARCHK does not equal the correct state whenever it is checked by the microcode, then an error has most likely occurred during a data transfer across the MVR/FMTR.

When the PORT MICROPROCCESSOR senses that the condition code, CCMVRPARCHK (DATA MOVER PARITY CHECK), does not equal the correct state it should set CSR26 (DATA PATH ERR) in the CSR REGISTER.

There are six different microcode algorithms required to accurately predict correct parity for all three data format modes. They sense for the following conditions:

a) HIGH DENSITY (CBUS to PLI INTERFACE) - CCMVRPARCHK will always be toggled an odd number of times for every two word transfer from the CBUS to the PLI INTERFACE

b) HIGH DENSITY (PLI INTERFACE to CBUS) - CCMVRPARCHK will always be toggled an odd number of times for every two word transfer from the PLI INTERFACE to the CBUS.

c) INDUSTRY COMPATIBLE (CBUS to PLI INTERFACE) - CCMVRPARCHK will always be toggled an even number of times for every one word transfer from the CBUS to the PLI INTERFACE. The command, MPINDSTCOMP, must be executed for every transfer in order to guarantee correct results.

d) INDUSTRY COMPATIBLE (PLI INTERFACE to CBUS) - CCMVRPARCHK will always be toggled an even number of times for every one word transfer from the PLI INTERFACE to the CBUS.

e) CORE DUMP (CBUS to PLI INTERFACE) - CCMVRPARCHK will always be toggled an even number of times for every two word transfer from the CBUS to the PLI INTERFACE.

f) CORE DUMP (PLI INTERFACE to CBUS) - CCMVRPARCHK will always be toggled an even number of times for every two word transfer from the PLI INTERFACE to the CBUS.

4.0 PORT MICROPROCESSOR

The PORT MICROPROCESSOR consists of:

a) The MICROPROCESSOR ALU which is physically located on the EBUS INTERFACE MODULE.

   The ALU is discussed in detail in sections "ARITHMETIC LOGIC UNIT" and "CONSTANT MUX". Therefore, it will not be further discussed here.

b) A MICROPROCESSOR-CONTROLLER which may be sub-divided into:

   1) One AM2910 MICROSEQUENCER, along with the necessary control input and output functions.

   2) A 4K bit deep by 60 bit wide CONTROL STORE RAM, along with a load and read/verify path from the MBUS.

   3) A 60 bit wide MICROCODE CONTROL REGISTER which is used to latch the current microword being executed.

c) A local storage memory 1K-word deep by 36-bits wide which interfaces to the MBUS and may be read or written by the MICROPROCESSOR.

d) MICROPROC CONTROL LOGIC which is used to control all of the various timing functions of the PORT MICROPROCESSOR.

4.1 CONDITION CODE MUX

The CC MUX is a 16 input by one-bit wide multiplexer which inputs the 16 CONDITION CODES from the various control logics of the PORT into the CC input line of the AM2910 MICROSEQUENCER.

The PORT MICROPROCESSOR selects one of the sixteen possible CONDITION CODES by executing the specific code in the MWSKIPFLD<01-04> field which causes the desired CONDITION CODE to be passed through the CC MUX to the input of the AM2910 SEQUENCER's CC line. On the same microcycle it enables the CC input line by executing the command, MWCCENA.

The CC MUX decodes only MWSKIPFLD <01-04>. MWSKIPFLD00 is not decoded and may be equal to either "0" or "1". The decoded function selects one of the 16 CONDITION CODES from the CC MUX for input to the AM2910. See the following section for details.

4.1.1 CONDITION CODE DEFINITIONS

The following table describes the function of each of the CONDITION CODES which are input into the AM2910 MICROSEQUENCER from the CC MUX.

There are five CONDITION CODES which originate from the EBUS CONTROL LOGIC and are used by the PORT MICROPROCESSOR to determine the state of the EBUS. They are as follows:

1) MWSKIPFLD = 01 Or 21 - Select CCGRNTCSR (CSR REGISTER GRANTED). Informs the PORT MICROPROC that it may access the CSR REGISTER. It will be asserted whenever the MICROPROC CONTROLLER requests access of the CSR and the KL10 is not simultaneously accessing it.

2) MWSKIPFLD = 10 or 30 - Select CCEBUSRQST (EBUS REQUEST). If the PORT is in the "MPROC RUN" state (CSR32 set) this CONDITION CODE will be asserted whenever the EBUS executes a DATAO or DATAI. The PORT MICROPROCESSOR examines the CONDITION CODE and determines the correct action to implement.

3) MWSKIPFLD = 03 or 23 - Select CCCSRCHNG (CSR REGISTER CHANGED). This CONDITION CODE will be asserted whenever the KL10 writes the CSR via a CONO, or an EBUS PARITY ERROR is detected. It will be de-asserted whenever the PORT MICROPROC reads the CSR (MPREADCSR asserted). The function of the CONDITION CODE is to inform the PORT MICROPROC whenever the KL10 has changed the contents of the CSR.

4) MWSKIPFLD = 04 or 24 - Select CCEBPARERR (EBUS PARITY ERROR). This CONDITION CODE will be asserted whenever an EBUS PARITY ERROR is detected.

5) MWSKIPFLD = 11 or 31 - Select CCINTRACTIVE (INTERRUPT ACTIVE). This CONDITION CODE indicates that an interrupt request on PI LEVEL 01 thru 07 which was previously executed by the PORT MICROCODE is still pending process by the KL10.

There are nine CONDITION CODES which originate on the CBUS/DATA MOVER INTFC MODULE (CMVR) and are used by the PORT MICROPROCESSOR to determine the state of the CMVR. They are as follows:

1) MWWSKIPFLD = 06 or 26 - Select CCRCVRBUFBFUL (RECEIVER BUFFER "B" FULL). This CONDITION CODE originates from the PLI INTERFACE and will be asserted whenever RECEIVE BUFFER "B" from the PACKET BUFFER MODULE is loaded with a CI PACKET.

2) MWSKIPFLD = 05 or 25 - Select CCRCVRBUFAFUL (RECEIVER BUFFER "A" FULL). This CONDITION CODE originates from the PLI INTERFACE and will be asserted whenever RECEIVE BUFFER "A" from the PACKET BUFFER MODULE is loaded with a CI PACKET.

3) MWSKIPFLD = 07 or 27 - Select CCXMTRATTN (TRANSMITTER ATTENTION). This CONDITION CODE originates from the PLI INTERFACE and will be asserted whenever the TRANSMIT BUFFER of the PACKET BUFFER MODULE requires attention (see the PLI SPEC for details).

4) MWSKIPFLD = 14 or 34 - Select CCCBUSPARERR (CBUS PARITY ERROR). This CONDITION CODE will be asserted whenever a parity error is detected in a word read from the CBUS. The CONDITION CODE is latched until it is cleared by a command from the MICROPROC CONTROLLER.

5) MWSKIPFLD = 13 or 33 - Select CCMVRPARCHK (MOVER PARITY CHECK). This CONDITION CODE will be toggled whenever a "1" parity bit is sensed from either the CBUS, or the PLI INTERFACE during DMA DATA TRANSFERS between these busses. By comparing the actual value of this CONDITION CODE with a predicted value it is possible for the MICROPROCESSOR to determine whether or not a parity error occurred during data transfers through the MVR/FMTR.

6) MWSKIPFLD = 15 or 35 - Select CCPLIPARERR (PLI PARITY ERROR). This CONDITION CODE will be asserted whenever a parity error is detected in a word read from the PLI INTFC. The CONDITION CODE is latched until it is cleared by a command from the MICROPROC CONTROLLER.

7) MWSKIPFLD = 16 or 36 - Select CCCHANERR (CBUS CHANNEL ERROR). This CONDITION CODE will be asserted whenever the CBUS ERROR signal is asserted on the CBUS INTERFACE. The CONDITION CODE is latched until it is cleared by a command from the MICROPROC CONTROLLER.

8) MWSKIPFLD = 17 or 37 - Select CCCBLSTWD (CBUS LAST WORD). This CONDITION CODE will be asserted whenever the CBUS LAST WORD signal is asserted on the CBUS INTERFACE. The CONDITION CODE is latched until it is cleared by a command from the MICROPROC CONTROLLER.

9) MWSKIPFLD = 00 or 20 - Select CCCBUSAVAIL (CBUS AVAILABLE). This CONDITION CODE will be asserted whenever:

   a) the CBUS INPUT BUFFER is available to receive a word from the CBUS

   b) the CBUS OUTPUT BUFFER is available to be loaded with a word for transfer to the CBUS

   c) the CBUS is not currently active (no data transfers occurring)

There are two CONDITION CODES which originate from the MICROPROCESSOR ALU and are used by the PORT MICROPROCESSOR to determine the state of the last arithmetic operation which transpired. They are as follows:

1) MWSKIPFLD = 02 or 22 - Select CCFEQ0 (ALU FUNCTION = 0). This CONDITION CODE indicates that the result of the last ALU operation produced all "ZEROS".

2) MWSKIPFLD = 12 or 32 - Select CCMBSIGN (ALU SIGN BIT SET). This CONDITION CODE indicates that the SIGN (MSB or bit 00) was set as the result of the last ALU operation.

4.2 MICROSEQUENCER

The MICROSEQUENCER is an AM2910 MICROPROCESSOR SEQUENCE CONTROLLER which is used to select the address of the next MICROWORD to be executed. The PORT MICROPROCESSOR controls the AM2910 from its CRAM CONTROL WORD by executing the following commands via the MWCTRLFLD<00-03> (MICROSEQR CONTROL INPUT FIELD <I0-I3>) - This field is the INSTRUCTION INPUT FIELD (I0-I3) for the AM2910 MICROSEQUENCER. Its function is defined by "THE AM2900 FAMILY DATA BOOK" as follows:

OCTAL CODE	MNEMONIC
MWCTRLFLD<00:03> = 0	JZ
MWCTRLFLD<00:03> = 1	CJS
MWCTRLFLD<00:03> = 2	JMAP
MWCTRLFLD<00:03> = 3	CJP
MWCTRLFLD<00:03> = 4	PUSH
MWCTRLFLD<00:03> = 5	JSRP
MWCTRLFLD<00:03> = 6	CJV
MWCTRLFLD<00:03> = 7	JRP
MWCTRLFLD<00:03> = 10	RFCT
MWCTRLFLD<00:03> = 11	RPCT
MWCTRLFLD<00:03> = 12	CRTN
MWCTRLFLD<00:03> = 13	CJPP
MWCTRLFLD<00:03> = 14	LDCT
MWCTRLFLD<00:03> = 15	LOOP
MWCTRLFLD<00:03> = 16	CONT
MWCTRLFLD<00:03> = 17	TWB

The MWCCENA (CONDITION CODE ENABLE) field is also used to control the CONDITION CODE ENABLE bit to the AM2910 MICROSEQUENCER (see "THE AM2900 FAMILY DATA BOOK"). It is decoded as follows:

MWCCENA = 0 - Condition always met
MWCCENA = 1 - Condition met only if input from CC MUX is true

The SEQUENCER's implementation in this design is standard and is clearly illustrated by the detailed block diagram. Therefore, its operation will not be discussed in detail (see "THE AM2900 FAMILY DATA BOOK" for specific details).

NOTE: The first microword executed by the PORT MICROCODE upon initial start-up (the KL10 setting CSR32) must be a non conditional jump. This guarantees that the AM2910's MICROPROGRAM COUNTER REGISTER is correctly loaded on the first executed instruction.

4.3 RAM ADDRESS REGISTER

The RAM ADDRESS REGISTER is used to load and read/verify the contents of the CONTROL STORE RAM (CRAM) when the PORT is not running (CSR32 reset).

It selects the next address to be loaded into, or read from the CRAM. It is a 13-bit register which is loaded from MBUS D01-D13 whenever a DATOLOADRAR is executed from the EBUS INTERFACE (DATAO with EBUS D00 equal to 1).

The CRAM is a 60-bit wide word which must be loaded and read/verified across the 36-bit wide MBUS. Therefore, it is loaded and read as 30-bit half words at a time. The LSB of the RAM ADDRESS REGISTER selects the half word to be accessed. The half word selection is made as follows:

RAR12 = 0 - Access the least significant half

RAR12 = 1 - Access the most significant half

The register cannot function as an up/down counter. Therefore, it must be re-loaded each time the HOST (KL10) wants to access another location in the CONTROL STORE RAM.

In order to load each 30-bit half word of the CRAM the KL10 must execute two commands; DATOLOADRAR and DATOLOADMW respectively

In order to read each 30-bit half word of the CRAM the KL10 must also execute two commands; DATOLOADRAR and DATIREADMW respectively.

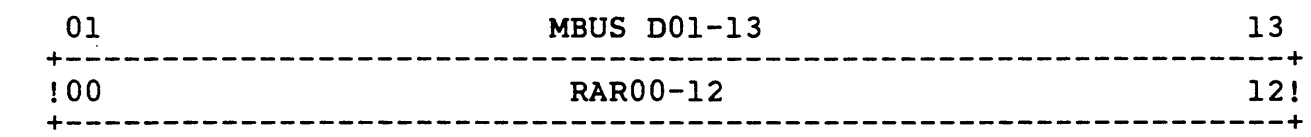
The RAR is also used to hold the starting CRAM address. The first CRAM address is always fetched from this register when the PORT MICROPROCESSOR is initially started (CSR32 set).

When the PORT is being operated in SINGLE CYCLE mode (CSR22 set) the KL10 must re-load the RAR with the next address to be executed at the end of each single cycle. The next address is contained in the LAR (see section LATCH ADDRESS REGISTER).

See section "EBUS CONTROL LOGIC" of this specification for details of the above commands.

The register can only be loaded by the HOST (KL10). It cannot be read.

The below diagram illustrates the bit mapping of the RAR REGISTER.



4.4 ADDRESS MUX

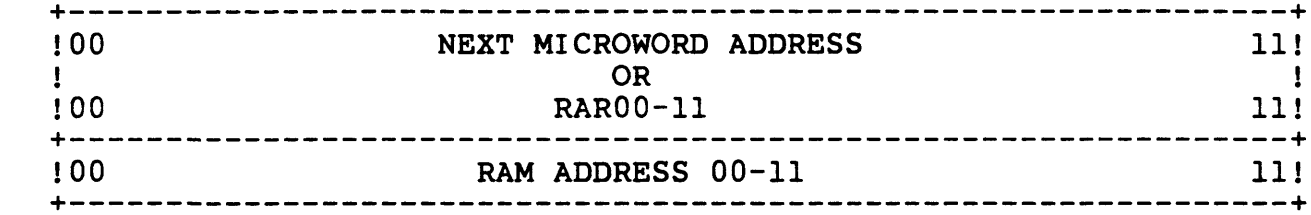
The ADDR MUX is a two input by 12-bit wide multiplexer which allows either the AM2910 MICROSEQUENCER, or the 12 most significant bits of the RAM ADDRESS REGISTER to select the next CRAM address.

In the "MPROC RUN" state (CSR32 set) the next address is normally fetched from the AM2910's "Y" outputs.

The one exception to the above rule is for the first address. The first address to be executed when "MPROC RUN" (CSR32) is initially set is always fetched from the RAR.

When the PORT is not running (CSR32 reset) the starting address is always fetched from the 12 most significant bits of the RAM ADDRESS REGISTER.

The below diagram illustrates the bit mapping of THE ADDR MUX.



Note that RAR12 is not passed through the MUX, but instead is directly routed to the MICROPROC CONTROL LOGIC for half word selection decoding.

4.5 LATCH ADDRESS REGISTER

The LATCH ADDRESS REGISTER (LAR) is a diagnostic tool. Its function is to latch the address of the MICROWORD CRAM location on every microcycle.

The KL10 may read the LAR over the EBUS by setting "DIAG SEL LAR" (CSR21) and executing a DATAI when the PORT is not running.

If the PORT MICROPROCESSOR halts the LAR may contain either the last CRAM address executed, or the next CRAM address to be executed, depending on the reason for the halt. The actual address which is latched into the LAR is determined by the state of "DIAG SINGLE CYCLE" (CSR22) as follows:

a) If the PORT MICROPROCESSOR is running (CSR32 set) and is not being single cycled ("DIAG SINGLE CYCLE" not set) the LAR will contain the address of the last MICROWORD which was executed, should the PORT MICROPROCESSOR halt for any reason.

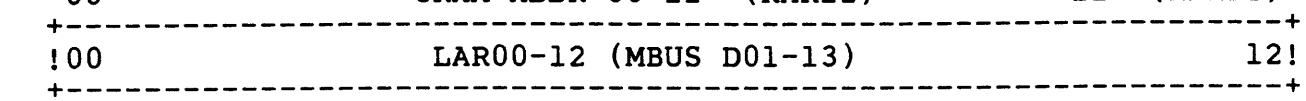
b) If the PORT MICROPROCESSOR is being run in the "SINGLE CYCLE" state ("DIAG SINGLE CYCLE set) it will automatically halt at the completion of each microcycle. The LAR will contain the address of the next MICROWORD which is to be executed. The KL10 must read this address and load it back into the "RAM ADDRESS REGISTER" (RAR) before executing the next single cycle (setting CSR32 again). This will enable the PORT MICROPROCESSOR to execute the next microcycle when it is re-started.

Reading the LAR will cause the current data in the EBUF to be destroyed. Therefore, the KL10 should take the precaution to read the EBUF before it reads the LAR in order to preserve any valid data which may be stored there. Once it reads the LAR it should then restore the preserved data to the EBUF.

The LAR is a 13-bit register. The 12 MSBs are loaded from CRAM ADDR 00-11. The LSB is directly loaded with the value of RAR12. The load function occurs at CLK1 time of every microcycle when the PORT is not in the "SINGLE CYCLE" state, or CLK 4 time of every microcycle when the PORT is in the "SINGLE CYCLE" state.

The LAR's outputs are mapped to MBUS D01-13 respectively. All other MBUS bits are undefined during a read LAR function.

The below diagram illustrates the bit mapping of the LAR



4.6 CONTROL STORE RAM

The CONTROL STORE RAM (CRAM) IS A 4K-word deep by 60-bit wide tri-stated input/output 55 ns access time ram memory. Its function is to store the PORT MICROPROCESSOR'S microcode.

The CRAM is initially loaded and read/verified, 1/2 word at a time, from the 30 least significant bits of the MBUS when the PORT is not running (CSR32 not set).

It requires four EBUS transfers to load one CRAM location. The sequence of these transfers is as follows:

a) Load the RAM ADDRESS REGISTER (RAR) with the right half address of the desired CRAM location. This is done by executing a DATAO with EBUS D00 equal to 1 and EBUS D01 - EBUS D13 equal to the desired right half address (DATOLOADRAR). The remaining EBUS bits are undefined.

b) Load the right half of the CRAM with the desired CRAM data. This is done by executing a DATAO with EBUS D00 equal to 0 and EBUS D06 - EBUS D36 equal to the desired right half data (DATOLOADMW). The remaining EBUS bits are undefined.

c) Load the RAM ADDRESS REGISTER (RAR) with the left half address of the desired CRAM location. This is done by executing a DATAO with EBUS D00 equal to 1 and EBUS D01 - EBUS D12 equal to the desired left half address (DATOLOADRAR). The remaining EBUS bits are undefined.

d) Load the left half of the CRAM with the desired CRAM data. This is done by executing a DATAO with EBUS D00 equal to 0 and EBUS D06 - EBUS D36 equal to the desired left half data (DATOLOADMW). The remaining EBUS bits are undefined.

At RUNCLK1 (CLK1) time of every microcycle, when the PORT MICROPROCESSOR is running, the location currently being addressed by the AM2910 is normally strobed into the 60-bit wide CONTROL STORE REGISTER for execution.

During initial MICROPROCESSOR start-up, or during "SINGLE CYCLE" mode, however, the first CRAM address is always fetched from the RAM ADDRESS REGISTER instead of the AM2910.

NOTE: If a CRAM PARITY or an MBUS ERROR occurs while the microprocessor is running data integrity is no longer guaranteed for any CRAM location. The entire microcode should be re-loaded before the PORT MICROPROCESSOR is re-started.

4.7 CONTROL STORE REGISTER

The CONTROL STORE REGISTER is a 60-bit wide register which is loaded with the current MICROWORD being addressed from the CONTROL RAM at RUNCLK1 (CLK1) time of every PORT MICROPROCESSOR cycle. RUNCLK2 (CLK2), RUNCLK3 (CLK3) and RUNCLK4 (CLK4) then execute the functions specified by the MICROWORD (see the following section for individual bit definitions).

NOTE: RUNCLK1, RUNCLK2, RUNCLK3 and RUNCLK4 are gated outputs of CLK1, CLK2, CLK3 and CLK4 respectively. They are active only when the MICROPROCESSOR is in the "MPROC RUN" state (CSR32 set).

4.7.1 MICROWORD FIELD DEFINITIONS

The following table defines the individual field definitions of the PORT MICROPROCESSOR'S MICROWORD:

a) CRAM00:11 = MWJMPFLD<00-11> (MICROSEQR JUMP ADDRESS) - Bits 00-03 of this field are directly input to bits D00-D03 of the AM2910 SEQUENCER. Bits 04-11 of this field are indirectly input, through the JMP MUX, to bits D04-D11 of the AM2910 SEQUENCER. When the MWSKIPFLD is not equal to 05 (SELMBUSFLD) the MWJMPFLD field will be used as the 12-bit JUMP ADDRESS by the "D" inputs of the AM2910. When the MWSKIPFLD is equal to 05 (SELMBUSFLD), then bits 00-03 of the MWJMPFLD field will constitute the four MSBs and bits MBUS D16-D23 will constitute the eight LSBs of the AM2910s JUMP ADDRESS.

b) CRAM12 = MWPAR (MICROWORD PARITY BIT) - This is the parity bit of the MICROWORD. The value of this bit should be chosen for each MICROWORD such that the word contains odd parity. MICROWORDS with even parity will cause a CRAM PARITY ERROR to be generated. Parity is not automatically assigned by the hardware. Correct parity must be loaded into each CRAM location by the KL10 when the CRAM is initially loaded.

c) CRAM13 = MWOUTPUTENA (ALU DATA OUTPUT ENABLE BIT) - This bit is the OUTPUT ENABLE bit to the AM2901 ALU tri-state "Y" outputs. When this bit equals "1" the "Y" outputs of the AM2901 ALU will be asserted on the MBUS (see "THE AM2900 FAMILY DATA BOOK").

d) CRAM14:23 = MWMGCFLD<00-09> (MAGIC NUMBER FIELD) - This FIELD is used, in conjunction with other MICROWORD CONTROL FIELDS, to control the EBUS and the CMVR INTERFACES, to input constants into the internal AM2901 ALU RAM, and to select the address of the LOCAL RAM STORAGE MEMORY (see MWBUSCTLFLD, MWSKIPFLD and MWRAMODE for more details):

e) CRAM24:26 = MWSORCEFLD<00-02> (ALU SOURCE INPUT FIELD (I2-I0)) - This field is the SOURCE INPUT FIELD (I2-I0) of the AM2901 ALU. Its function is defined by "THE AM2900 FAMILY DATA BOOK" as follows:

OCTAL CODE	R	S
MWSORCEFLD<00:02> = 0	A	Q
MWSORCEFLD<00:02> = 1	A	B
MWSORCEFLD<00:02> = 2	Z	Q
MWSORCEFLD<00:02> = 3	Z	B
MWSORCEFLD<00:02> = 4	Z	A
MWSORCEFLD<00:02> = 5	D	A
MWSORCEFLD<00:02> = 6	D	Q
MWSORCEFLD<00:02> = 7	D	Z

f) CRAM27:29 = MWFUNCTFLD<00-02> (ALU FUNCTION FIELD (I5-I3)) - This field is the FUNCTION INPUT FIELD (I5-I3) of the AM2901 ALU. Its function is defined by "THE AM2900 FAMILY DATA BOOK" as follows:

OCTAL CODE	SYMBOL
MWFUNCTFLD<00:02> = 0	R + S
MWFUNCTFLD<00:02> = 1	S - R
MWFUNCTFLD<00:02> = 2	R - S
MWFUNCTFLD<00:02> = 3	R or S
MWFUNCTFLD<00:02> = 4	R or S

MWFUNCTFLD<00:02> = 5	R or S
MWFUNCTFLD<00:02> = 6	R xor S
MWFUNCTFLD<00:02> = 7	$\overline{\text{R xor S}}$

g) CRAM30:32 = MWDESTFLD<00-02> (ALU DESTINATION OUTPUT FIELD (I8-I6)) - This field is the DESTINATION FIELD (I8-I6) inputs to the AM2901 ALU. Its function is defined by "THE AM2900 FAMILY DATA BOOK" as follows:

OCTAL CODE	MNEMONIC
MWDESTFLD<00:02> = 0	QREG
MWDESTFLD<00:02> = 1	NOP
MWDESTFLD<00:02> = 2	RAMA
MWDESTFLD<00:02> = 3	RAMF
MWDESTFLD<00:02> = 4	RAMQD
MWDESTFLD<00:02> = 5	RAMD
MWDESTFLD<00:02> = 6	RAMQU
MWDESTFLD<00:02> = 7	RAMU

h) CRAM33 = MWCCENA (CONDITION CODE ENABLE BIT) - This bit is the CONDITION CODE ENABLE bit to the AM2910 MICROSEQUENCER (see "THE AM2900 FAMILY DATA BOOK").

MWCCENA = 0 - Condition always met
MWCCENA = 1 - Condition met only if input from CC MUX is true

i) CRAM34 = MWRAMODE (MICROWORD LOCAL STORAGE RAM MODE BIT) - This bit controls whether LOCAL or GLOBAL addressing will be used to address the LOCAL STORAGE RAM.

When MWRAMODE = 0 (GLOBAL ADDRESSING) MWMGCFLD<00:09> are used as the full address selection value for the LOCAL RAM STORAGE MEMORY.

When MWRAMODE = 1 (LOCAL ADDRESSING) MWMGCFLD<05:09> are used as the 5 low order address selection bits for the LOCAL RAM STORAGE MEMORY. SADREG<00:04> are used as the 5 high order address selection bits for the LOCAL RAM STORAGE MEMORY.

j) CRAM35:38 = MWPORTAFLD<00-03> (ALU PORT "A" (A3-A0) ADDRESS FIELD) - This field is the PORT "A" (A3-A0) ADDRESS FIELD inputs to the AM2901 ALU (see "THE AM2900 FAMILY DATA BOOK")

k) CRAM39:42 = MWPORTBFLD<00-03> (ALU PORT "B" (B3-B0) ADDRESS FIELD) - This field is the PORT "B" (B3-B0) ADDRESS FIELD inputs to the AM2901 ALU (see "THE AM2900 FAMILY DATA BOOK")

1) CRAM43:47 = MWSKIPFLD<00-04> (MICROWORD SKIP FIELD) - This field is decoded by the MICROPROC COND SKIP FIELD DECODER and the CONDITION CODE MUX (CC MUX)

The CC MUX decodes only MWSKIPFLD <01-04>. MWSKIPFLD00 is not decoded and may be equal to either "0" or "1". The decoded function selects one of the 16 CONDITION CODES from the CC MUX for input to the AM2910 as follows:

MWSKIPFLD = 00 or 20 - Select CCCBUSAVAIL

MWSKIPFLD = 01 0r 21 - Select CCGRNTCSR

MWSKIPFLD = 02 or 22 - Select CCFEQL0

MWSKIPFLD = 03 or 23 - Select CCCSRCHNG

MWSKIPFLD = 04 or 24 - Select CCEBPARERR

MWSKIPFLD = 05 or 25 - Select CCRCVRBUFAFUL

MWSKIPFLD = 06 or 26 - Select CCRCVRBUFBFUL

MWSKIPFLD = 07 or 27 - Select CCXMTRATTN

MWSKIPFLD = 10 or 30 - Select CCEBUSRQST

MWSKIPFLD = 11 or 31 - Select CCINTRACTIVE

MWSKIPFLD = 12 or 32 - Select CCMBSIGN

MWSKIPFLD = 13 or 33 - Select CCMVRPARCHK

MWSKIPFLD = 14 or 34 - Select CCCBUSPARERR

MWSKIPFLD = 15 or 35 - Select CCPLIPARERR

MWSKIPFLD = 16 or 36 - Select CCCHANERR

MWSKIPFLD = 17 or 37 - Select CCCBLSTWD

The MICROPROC COND SKIP FIELD DECODER decodes only MWSKIPFLD <00, 02, 03, 04>. MWSKIPFLD01 is not decoded and may be equal to either "0" or "1". These functions will be executed whenever the MWSKIPFLD is equal to the following values:

MWSKIPFLD = 20 or 30 (LOADSADREG). Causes the LOCAL STORAGE ADDRESS REGISTER to be loaded by the contents of the MWMGCFLD <05-09>.

MWSKIPFLD = 21 or 31 (SELMBUSFLD). Causes the JMP MUX to pass the contents of MBUS D16-D23 to the corresponding "D" inputs of the AM2910 MICROSEQUENCER. MWJMPFLD00-03 are still passed to the four most significant bits of the "D" inputs.

MWSKIPFLD = 22 or 32 (RDLOCALMEM). Causes the contents of the LOCAL RAM STORAGE MEMORY location currently being addressed to be placed on the MBUS.

MWSKIPFLD = 23 or 33 (LDLOCALMEM). Causes the contents of the MBUS to be loaded into the LOCAL RAM STORAGE MEMORY location currently being addressed.

MWSKIPFLD = 24 or 34 (SELCNSTFLD). Causes the CNST MUX to pass the contents of MWMGCFLD00-09 to the 10 least significant and the 10 most significant bits of the AM2901 ALU "D" inputs.

m) CRAM48:50 = MWBUSCTLFLD<00-02> (MICROPROCESSOR BUS CONTROL FIELD) - This field is used by the MICROPROCESSOR, in conjunction with the MWMGCFLD field, to control the various functions of the EBUS and the CMVR INTERFACES. The field is decoded as follows:

When MWBUSCTLFLD = 0 (no function)

When MWBUSCTLFLD = 1 (SELECT PLI FIELD)

MWMGCFLD00-01 = No function

MWMGCFLD02 = PLI LINK CONTROL 0. Passes PLI LINK CONTROL 0 to the PLI BUS

MWMGCFLD03 = PLI LINK CONTROL 1. Passes PLI LINK CONTROL 1 to the PLI BUS

MWMGCFLD04 = PLI LINK CONTROL 2. Passes PLI LINK CONTROL 2 to the PLI BUS

MWMGCFLD05 = PLI LINK CONTROL 3. Passes PLI LINK CONTROL 3 to the PLI BUS

MWMGCFLD06 = MPSELECTPLI. Asserts the PLI SELECT line.

MWMGCFLD07 = MPXMITPLI. Enables the tri-state outputs of the PLI OUTPUT BUFFER onto the PLI BUS.

MWMGCFLD08 = MPRECVPLI. Loads the contents of the PLI BUS into the PLI INPUT BUFFER.

MWMGCFLD09 = No function

When MWBUSCTLFLD=2 (SELECT MBUS FIELD)

MWMGCFLD00-01 = No function

MWMGCFLD02 = MPENACMUX. Causes the tri-state outputs of the CMUX to be enabled onto the MBUS.

MWMGCFLD03 = MPPLINTOCMUX. Enables the PLI input path to the CMUX, which allows the PLI INPUT BUFFER to be asserted on the 8 LSBs of the MBUS.

MWMGCFLD04 = MPCBUFTOPLOUT. Loads the PLI OUTPUT BUFFER with the contents of the 8 least significant bits of the CBUF

MWMGCFLD05 = MPCLRCCCODE. Causes all of the CONDITION CODE status bits on the CMVR to be cleared, except CCMVRPARCHK.

MWMGCFLD06 = MPCLRPARCHK. Causes the condition code, CCMVRPARCHK to be cleared.

MWMGCFLD07-08 = No function

MWMGCFLD09 = MPTESTPLIPAR. Causes the "PLI PAR OUT GENERATOR" to generate even (bad) parity.

When MWBUSCTLFLD=3 (SELECT FMTR FIELD)

MWMGCFLD00-01 = No function

MWMGCFLD02 = MPSHFTFMTR8. Causes the contents of the MVR/FMTR to be shifted eight bits to the left or right, depending on the state of the command "MPSHIFTRIGHT".

MWMGCFLD03 = MPSHFTFMTR4A. Causes the contents of the MVR/FMTR to be shifted four bits to the left or right, depending on the state of the command "MPSHIFTRIGHT".

MWMGCFLD04 = MPCBUFTOFMTR. Causes the data which has been previously stored in the CBUF to be loaded into the MVR/FMTR REGISTER

MWMGCFLD05 = MPPLINTOFMTR. Causes the 8-bit data byte which is currently stored in the PLI INPUT BUFFER to be shifted into the serial input lines of the MVR/FMTR, four bits at a time. This command must be executed in conjunction with MPSHFTFMTR4A or MPSHFTFMTR8. If MPSHFTFMTR4A is executed, then only four bits from the PLI INPUT BUFFER will be shifted into the serial input lines of the MVR/FMTR. If MPSHFTFMTR8 is executed, then all 8 bits from the PLI INPUT BUFFER will be shifted into the serial input lines. Either the four MSBs or the four LSBs are shifted in first, depending on the state of the command, MPRHTNIBFIRST. If an MPSHFTFMTR4A is executed, then only the first four bits specified by the state of MPRHTNIBFIRST will be shifted into the serial input lines. If MPSHIFTRIGHT is asserted the 4-bit nibbles will be shifted into the four MSBs of the MVR/FMTR and shifted right. Otherwise, they will be shifted into the four LSBs of the MVR/FMTR and shifted left.

MWMGCFLD06 = MPFMTRTOPLOUT. Causes the 8-bit data byte which is currently stored in the PLI OUTPUT REG of the MVR/FMTR to be loaded into the PLI OUTPUT BUFFER.

MWMGCFLD07 = MPSHIFTRIGHT. When asserted, either the currently selected nibble in the PLI INPUT BUFFER, or the four LSBs (MVROUT36-39) are shifted into the four MSB serial input lines of the MVR/FMTR and shifted right. The actual data shifted into the MVR/FMTR depends on the state of MPPLINTOFMTR. If MPPLINTOFMTR is asserted, then the currently selected 4-bit nibble in the PLI INPUT BUFFER will be shifted into the four MSBS of the MVR/FMTR and shifted right. If MPPLINTOFMTR is de-asserted, then MVROUT36-39 will be shifted into the four MSBS of the MVR/FMTR and shifted right. Two 4-bit shifts may be executed during one microcycle, thus enabling an 8-bit byte to be right shifted during one microcycle (see MPPLINTOFMTR for more details).

MWMGCFLD08 - MPRHTNIBFIRST. When asserted, the four LSBs of the PLI INPUT BUFFER are shifted into the serial input lines of the MVR/FMTR first. When de-aserted, the four MSBs of the PLI INPUT BUFFER are shifted into the serial input lines of the MVR/FMTR first.

MWMGCFLD09 = MPZEROLFTNIB. When asserted, causes the four MSBs from the PLI OUTPUT REG of the MVR/FMTR to be forced to "zeros" before they are loaded into the PLI OUTPUT BUFFER.

When MWBUSCTLFLD=4 (SELECT CBUS FIELD)

MWMGCFLD00-01 = No function

MWMGCFLD02 = MPSTARTCBUS. When executed causes CBUS START and CBUS RESET to be asserted on the CBUS at the proper time during the next CBUS SELECT cycle.

MWMGCFLD03 = MPSTOPCBUS. When executed causes CBUS DONE to be asserted on the CBUS at the proper time during the next CBUS SELECT cycle.

MWMGCFLD04 = MPSTORECBUS. When executed causes CBUS STORE to be asserted on the CBUS at the proper time during the next CBUS SELECT cycle. This command should only be executed simutaneously with MPSTOPCBUS.

MWMGCFLD05 = MPWRITEMEM. When executed causes CBUS CTOM to be asserted on the CBUS at the proper time during the next CBUS SELECT cycle. This command should only be executed simutaneously with MPSTARTCBUS when a transfer to KL10 memory is to be implemented.

MWMGCFLD06 = MPINDSTCOMP. Enables the PARITY PREDICTOR to predict correct parity in INDUSTRY COMPATIBLE MODE. This command must be executed in INDUSTRY COMPATIBLE MODE when transferring data from the CBUS to the PLI INTERFACE. Otherwise, the results of the PARITY PREDICTOR will not be correct.

MWMGCFLD07 = MPCBINTOFMTR. Causes the contents of the CBUS INPUT to be loaded into the MVR/FMTR.

MWMGCFLD08 = MPFMTRTOCBOUT. Causes the contents of the MVR/FMTR to be loaded into the CBUS OUTPUT BUFFER.

MWMGCFLD09 = MPSHFTFMTR4B. Causes the contents of the MVR/FMTR to be shifted four bits to the left.

When MWBUSCTLFLD=5 (SELECT EBUS FIELD)

MWMGCFLD00-01 = No function.

MWMGCFLD02 = MPLOADCSR. Loads CSR00-CSR17 with the contents of the EBUF (EBUF00-EBUF17)

MWMGCFLD03 = MPREADCSR. Places the contents of the CSR REGISTER (CSR00- CSR35) on the MBUS

MWMGCFLD04 = MPRQSTCSR. Requests access to the CSR. Access will be granted if the KL10 is not currently accessing the register

MWMGCFLD05 = MPLOADEBUS. Causes the contents of the EBUF to be asserted on the EBUS

MWMGCFLD06 = MPREADEBUS. Causes the contents of the EBUS to be asserted on the MBUS

MWMGCFLD07 = MPLOADEBUF. Causes the data which the PORT MICROPROC places on the MBUS to be loaded into the EBUF.

MWMGCFLD08 = MPRQSTINTR. Causes the PORT to request an EBUS INTERRUPT (function 00 - 03) on PI LEVEL 01 THRU 07. The interrupt function is determined by an "IOPF FUNCTION CONTROL WORD" which has been previously built and loaded into the EBUF by the PORT MICROPROC. The IOP WORD is passed to the EBUS by the MICROPROC when requested (CCEBUSRQST being asserted)

MWMGCFLD09 = MPEXORDEP. Causes the PORT to request an EBUS EXAMINE or DEPOSIT INTERRUPT (function 04 - 07) on PI LEVEL 00. The interrupt function is determined by an "IOPF FUNCTION CONTROL WORD" which has been previously built and loaded into the EBUF by the PORT MICROPROC. The IOP WORD is passed to the EBUS by the MICROPROC when requested (CCEBUSRQST being asserted)

When MWBUSCTLFLD = 6-7 (no function)

n) CRAM51 = MWCARRY (MICROWORD CARRY INPUT BIT TO ALU) - This bit is the carry input to the least significant bit of the AM2901 ALU.

MWCARRY = 0 Carry "ZERO" into the LSB of the ALU

MWCARRY = 1 Carry "ONE" into the LSB of the ALU

o) CRAM52:55 = MWCTRLFLD<00-03> (MICROSEQR CONTROL INPUT FIELD (I0-I3)) - This field is the INSTRUCTION INPUT FIELD (I0-I3) for the AM2910 MICROSEQUENCER. Its function is defined by "THE AM2900 FAMILY DATA BOOK" as follows:

OCTAL CODE	MNEMONIC
MWCTRLFLD<00:03> = 0	JZ
MWCTRLFLD<00:03> = 1	CJS
MWCTRLFLD<00:03> = 2	JMAP
MWCTRLFLD<00:03> = 3	CJP
MWCTRLFLD<00:03> = 4	PUSH
MWCTRLFLD<00:03> = 5	JSRP
MWCTRLFLD<00:03> = 6	CJV
MWCTRLFLD<00:03> = 7	JRP
MWCTRLFLD<00:03> = 10	RFCT
MWCTRLFLD<00:03> = 11	RPCT
MWCTRLFLD<00:03> = 12	CRTN
MWCTRLFLD<00:03> = 13	CJPP
MWCTRLFLD<00:03> = 14	LDCT
MWCTRLFLD<00:03> = 15	LOOP
MWCTRLFLD<00:03> = 16	CONT
MWCTRLFLD<00:03> = 17	TWB

p) CRAM56 = MWTIMEFLD (MICROWORD TIME FIELD BIT) - When this bit is asserted the current micro-instruction will be allowed 1 1/2 times the normal execution time to complete. This feature is useful where insufficient time may exist for a specific micro-sequence to successfully complete during the normal microcycle execution time.

NOTE: Currently, the only known condition for which the "TIME FIELD" bit must be set is when executing an arithmetic operation on a location from LOCAL STORAGE and simultaneously testing for CCFEQL0 (see below example).

MWCCENA = 1, MWSKIPFLD = 02 or 22 (Testing for CCFEQL0)
MWSORCEFLD<00:02> = 5, 6, or 7 ("D" input selected)
MWFUNCTFLD<00:02> = 0, 1, or 2 (Arithmetic function)

q) CRAM57 = MWSPARE00 (MICROWORD SPARE BIT 00) - No function

r) CRAM58 = MWSPARE01 (MICROWORD SPARE BIT 01) - No function

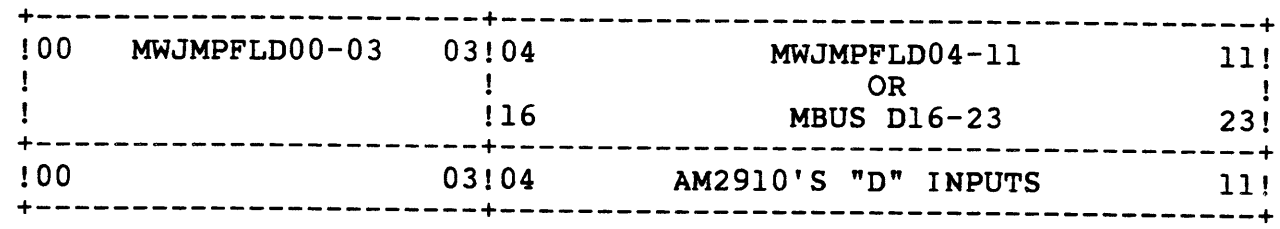
s) CRAM59 = MWMARKBIT (MICROWORD MARK BIT) - This bit has no microcode function. It is used for hardware/microcode debug purposes only. The operator may set the bit in any specific microword and then use it as a sync point for an oscilloscope, thus enabling him to sync on the execution of a specific microword. The bit is not checked for CRAM PARITY. Therefore, it may be set to any value without affecting the contents of the remaining microword.

4.8 JMP MUX

The JMP MUX is a two input by 8-bit wide multiplexer which allows either the 8 least significant bits of the microword's MWJMPFLD field, or MBUS D16-D23 to be loaded into the AM2910's corresponding 8 least significant "D" input bits. The 4 most significant bits of the MWJMPFLD field are always passed to the corresponding four most significant bits of the "D" inputs.

The MBUS is selected when MWSKIPFLD = 21 or 31. For all other conditions the MWJMPFLD field is selected.

The below diagram illustrates the bit mapping of this MUX.



4.9 MICROWORD OUTPUT MUX

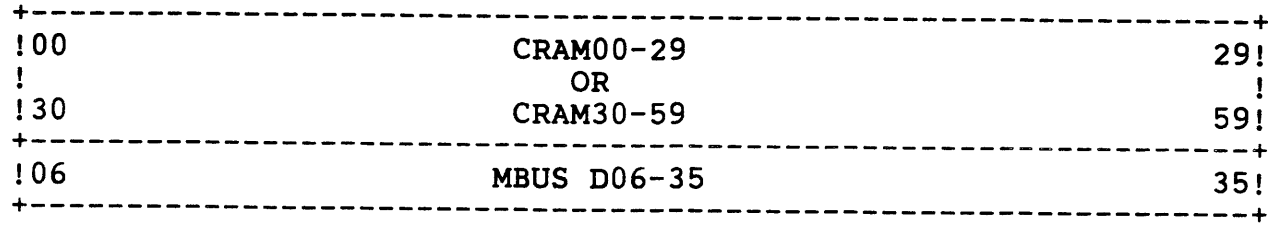
The MW OUT MUX is a two input by 30-bit wide multiplexer which enables either the right half, or the left half of the MICROWORD which is addressed by the RAM ADDRESS REGISTER onto the 30 least significant bits of the MBUS during a CRAM read/verify function. The 6 most significant bits of the MBUS are undefined.

RAR12 selects which half of the MICROWORD is to be read as follows:

RAR12=0 - Read right half of MICROWORD

RAR12=1 - Read left half of MICROWORD

The below diagram illustrates the bit mapping of the MW OUT MUX.



4.10 CRAM LOAD BUFFERS

There are two CRAM LOAD BUFFERS, LEFT CRAM LOAD BUFF and RIGHT CRAM LOAD BUFF, which are used to load the CRAM when the PORT is not running (CSR32 reset).

Each buffer is a 30-bits wide tri-state buffer which inputs the 30 least significant bits from the EBUS, via the MBUS, to the "DATA" lines of the CRAM.

EBUS D00 must be equal to "ZERO" whenever the CRAM is being written. EBUS D01 - EBUS D05 are undefined.

LEFT CRAM LOAD BUFF interfaces to the 30 least significant bits of the CRAM.

RIGHT CRAM LOAD BUFF interfaces to the 30 most significant bits of the CRAM.

The tri-state outputs of one of the buffers are enabled by the state of RAR12 from the RAM ADDRESS REGISTER as follows:

RAR12=0 - Enable RIGHT CRAM LOAD BUFF

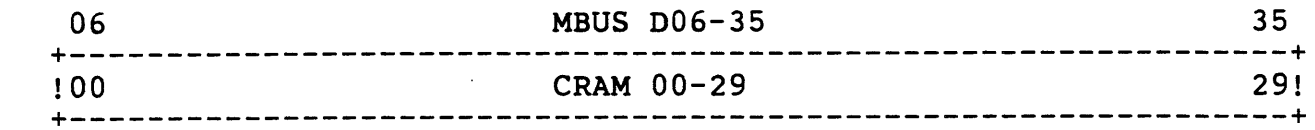
RAR12=1 - Enable LEFT CRAM LOAD BUFF

The CRAM LOAD BUFFERS are pass through buffers only. The data is not latched.

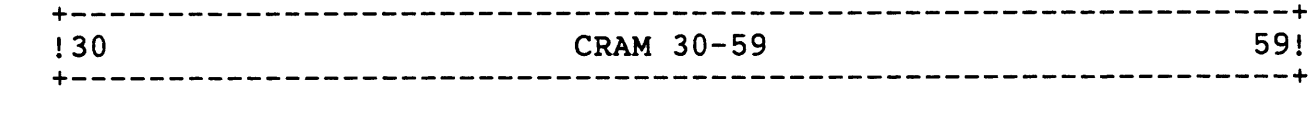
These BUFFERS may only be enabled when the PORT is not running (CSR32 reset).

The below diagram illustrates the bit mapping of these BUFFERS.

LEFT CRAM LOAD BUFF



RIGHT CRAM LOAD BUFF



4.11 CRAM PARITY CHECKER

The CRAM PARITY CHECKER checks the 59 most significant bits of the 60 bit wide MICROWORD which the PORT MICROPROCESSOR strobes into the MICROWORD CONTROL REGISTER for odd parity. If parity is incorrect the CRAM PARITY ERROR and the RQST INTERRUPT bits are set in the CSR REGISTER of the EBUS MODULE, and the PORT MICROPROCESSOR is halted. The least significant bit (MWMARKBIT) is not checked for CRAM PARTIY.

The PARITY bit is not automatically generated when the CRAM is loaded. Therefore, it is the responsibility of the microcoder to assign correct parity to each microword.

Occasionally a "CRAM PARITY ERROR" may be intentionally forced in order to halt the PORT MICROPROCESSOR at a specific location (Break Point). For this case it may be cleared by executing a CONO with EBUS D24 equal to "ONE". The PORT MICROPROCESSOR may then be re-started by setting "MPROC RUN" (CSR32).

4.12 LOCAL STORAGE RAM

The LOCAL STORAGE RAM is a 1K-word deep by 36-bit wide tri-state ram memory with a 55 ns access time. Its "DATA PATH" is directly connected to the MBUS. The PORT MICROPROCESSOR may either load or read this RAM memory as a local data storage memory.

The PORT MICROPROCESSOR loads the LOCAL STORAGE RAM by setting the MWSKIPFLD field of the MICROWORD to either 23 or 33 and asserting the desired data on the MBUS. The data will be strobed into the RAM at the address specified by the RAM MODE MUX at CLK4 time.

The PORT MICROPROCESSOR reads the LOCAL STORAGE RAM by setting the MWSKIPFLD field of the MICROWORD to either 22 or 32 and strobing the data from the MBUS into another storage media. The data will be read from the RAM at the address specified by the RAM MODE MUX.

The LOCAL STORAGE RAM may be addressed in either a GLOBAL or a PAGE addressing mode, depending on the state of the MWRAMODE bit of the MICROWORD (see LOCAL STORAGE ADDRESS REGISTER, RAM MODE MUX and MICROWORD bit definitions for more details).

4.13 LOCAL STORAGE ADDRESS REGISTER

The LOCAL STORAGE ADDRESS REGISTER (SADREG) is a 5-bit wide register which is used to address the five MSBs of the LOCAL STORAGE RAM in the PAGE addressing mode. It is initially loaded by MWMGCFLD<00-04> of the MICROWORD.

The PORT MICROPROCESSOR loads the SADREG by setting the MWSKIPFLD field of the MICROWORD to either 20 or 30 and setting MWMGCFLD<00-04> of the MICROWORD to the desired address. The value of MWMGCFLD<00-04> will be strobed into the SADREG at RUNCLK4 (CLK4) time.

The primary function of the SADREG is to enable the LOCAL STORAGE RAM to be indexed into 32 separate PAGES, each consisting of 32 words. By re-loading the SADREG different PAGES may be accessed. The 5 low order address line for the LOCAL STORAGE RAM are always directly selected by MWMGCFLD<05-09> of the MICROWORD. Therefore, any word within a PAGE may be accessed without re-loading the SADREG.

4.14 RAM MODE MUX

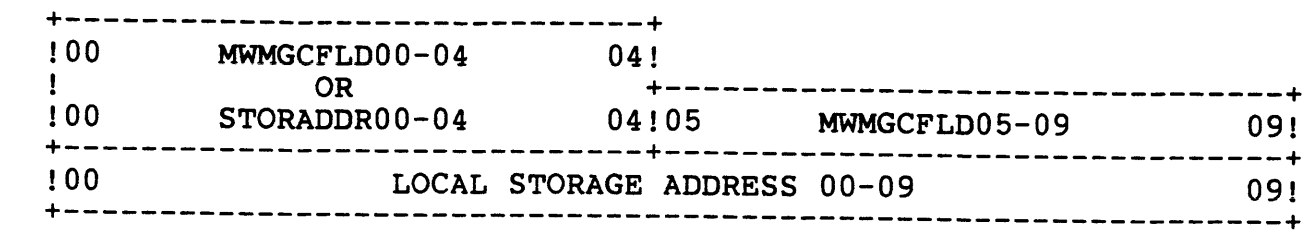
The RAM MODE MUX is a two input by 5-bit wide multiplexer which allows either MWMGCFLD<00-04> of the MICROWORD, or the contents of the LOCAL STORAGE ADDRESS REGISTER (SADREG) to select the five MSBs of the next LOCAL STORAGE RAM address. The input chosen depends on the state of the MWRAMODE bit of the MICROWORD.

When MWRAMODE equals "ZERO" the contents of MWMGCFLD<00-04> is passed as the five MSBs of the next LOCAL STORAGE RAM address. This mode functions as a GLOBAL addressing mode. Any location in the LOCAL STORAGE RAM may be directly addressed by the MWMGCFLD field of the MICROWORD.

When MWRAMODE equals "ONE" the contents of the SADREG will be passed as the fIVE MSBs (current PAGE) of the next LOCAL STORAGE RAM address. This mode functions as a PAGE addressing mode. Only one of 32 locations may be directly accessed by the MWMGCFLD field. In order to address locations in a different PAGE the contents of the SADREG must be changed.

The five LSBs of the LOCAL STORAGE RAM address are always selected by MWMGCFLD<05-09>, regardless of the state of MWRAMODE.

The below diagram illustrates the bit mapping of the RAM MODE MUX



4.15 COND/SKIP FIELD DECODER

The COND/SKIP FIELD DECODER decodes the MWSKIPFLD field of the MICROWORD to determine which one of the functions controlled by this field is to be executed. All of these functions are used by the PORT MICROPROCESSOR to execute some function internal to the MICROPROCESSOR'S operation. A list of these functions follows (see section MICROWORD FIELD DEFINITIONS for additional details):

MWSKIPFLD = 00 or 20 - Select CCCBUSAVAIL
01 0r 21 - Select CCGRNTCSR
02 or 22 - Select CCFEQ0
03 or 23 - Select CCCSRCHNG
04 or 24 - Select CCEBPARERR
05 or 25 - Select CCRCVRBUFAFUL
06 or 26 - Select CCRCVRBUFBFUL
07 or 27 - Select CCXMTRATTN
10 or 30 - Select CCEBUSRQST
11 or 31 - Select CCINTRACTIVE
12 or 32 - Select CCMBSIGN
13 or 33 - Select CCMVRPARCHK
14 or 34 - Select CCCBUSPARERR
15 or 35 - Select CCPLIPARERR
16 or 36 - Select CCCHANERR
17 or 37 - Select CCCBLSTWD
20 or 30 - LOADSADREG
21 or 31 - SELMBUSFLD
22 or 32 - RDLOCALMEM
23 or 33 - LDLOCALMEM
24 or 34 - SELCNSTFLD

4.16 MICROPROC CONTROL LOGIC

The MICROPROCESSOR CONTROL LOGIC controls all of the timing functions of the PORT MICROPROCESSOR. It contains the necessary control logic to:

a) write the RAM ADDRESS REGISTER (RAR)

b) write and read/verify the CONTROL STORE RAM (CRAM)

c) read the LATCH ADDRESS REGISTER (LAR)

d) Generate RUNCLK1, RUNCLK2, RUNCLK3 and RUNCLK4 from CLK1, CLK2, CLK3 and CLK4 respectively

e) start and stop the PORT MICROPROCESSOR in an orderly manner

APPENDIX "A"