C3800 Troubleshooting Refresher Course

CONVEX COMPUTER CORPORATION

C3800
Troubleshooting
Refresher
Course

R. Opholz

CONVEX Education Center

May 12, 1993

Convex Computer Corporation

Table of Contents

Page

Section One - Power Troubleshooting

Section Two - NCU/SPU Troubleshooting

Section Three - Crossbar Interface

Section Four - NIA Troubleshooting

Section Five - Hard_logger Interpretation

Section Six - Connectivity Troubleshooting

Appendix A - I/O Cabinet Cabling

Appendix B - CPU Cabinet Cabling

Appendix C - Central Cabinet Cabling

List of Figures

Section One

C3800 Series

Power Troubleshooting

1.0 Power Rules

Rule: BPC processor receives 12 volt power from the input harmonizer and is disabled by turning the Bay breaker to the off position. When the Bay Breaker is turned to the on position, 12 volts is automatically supplied to the BPC.

Rule: PPC processor receives 12 volt power from the output of the BPS's and is disabled by turning off the BPS breaker which supplies power to that power pallet.

Assumption: The bay power supply breakers and the bay breaker will normally be in the on position, the key switch in the enabled position, and the SPU controlling what boards are receiving power. NOTE: When the SPU asserts it's reset to an individual bay, the BPC is still receiving 12 volts (the processor is held in reset) and the PPC's are also receiving 12 volts (they too are held in reset.) What is not available to the pallets is the 300 volts which was turned off as part of going into reset. To physically remove all power to a PPC requires the BPS to be physically off. To physically remove all power from a BPC requires the bay breaker to be physically off.

Rule: Turning on the BPS breakers and the Bay breakers before placing the Central Cabinet key switch in the enabled position only provides 12V to the BPC's and PPC's. No 300 volts out of any BPS is available to any power pallet.

Rule: The BPC firmware has the ability to automatically recognize when a board is installed or removed from the system. The BPC firmware reports any unsolicited changes in bay configuration to the SPU workstation.

Rule: A CPU (SP/VP pair) uses a shared BPS and must be treated as a single board for power operations. Specifically, an NSP can be plugged into a slot, but power can not be supplied to that board until its NVP is added (and vice-versa.) Likewise, an NSP can not be powered down and removed from the system without the NVP also being powered down.

Rule: The CU and XBAR use a shared BPS and are also treated as a single board for powerup and powerdown conditions.

Rule: The IA8 and CCUPPL share a BPS and are treated as a single board for powerup and powerdown conditions.

Rule: The CCUPPR does not share a BPS. Anytime power is removed from the IA8, power must also be removed from this unit. If at any time, the system is receiving ccu boards for installation in the right CCU backplane, a CCUPPR and BPS must also be ordered if no other boards were previously installed in this side.

Rule: A memory board in not functional unless all 32 NMC are plugged in.

2.0 Power Unit Specifications

2.1 Input Power Harmonizer (Figure 1-2)

2.1.1 Input Specification

Input Voltage:	200-220VAC 3-phase wye +/-10% - four wires plus ground. (Neutral used only for surge suppression devices and line indicators)
Input Frequence:	47-63Hz
Input Power:	0-23.4 KVA
Input Current:	75 Amps per phase maximum

2.1.2 Output Specification

Same as the input except that the power has been filtered by the IPH line filter and neutral inductor.

2.2 Bay Power Unit (Figure 1-3a, 3b)

2.2.1 Input Specifications

Output of the IPH

2.2.3 Output Specifications

Same as the input (distributed to the BPS's)

2.3 Bay Power Controller Enclosure (BPCE) (Figure 1-4)

2.3.1 Input Specifications

BPU to BPCE - 200-220VAC 3 phase wye (with safety ground) +/-10% 47-63Hz
at 4.0 amps/phase nominal.

BPC Board to BPCE
- Transformer Primary 200-220VAC 1phase
- Fan Contactor DC Coil Power (24VDC)

2.4 Bay Power Supply (Figure 1-5)

2.4.1 Input Specifications

Same as IPH Output

2.4.2 Output Specifications

AC Output - Same as AC input but switched and filtered.

300VDC
12 VDC
9 VDC

2.5 Power Pallet

2.5.1 Input Voltages

300VDC
12 VDC
9 VDC

2.5.2 Output Voltages - All power pallets are designed around DC:DC converters (power bricks). These converters generate the +5VDC, -5VDC, -2VDC, and -4.5VDC voltages required to operate the logic cards.

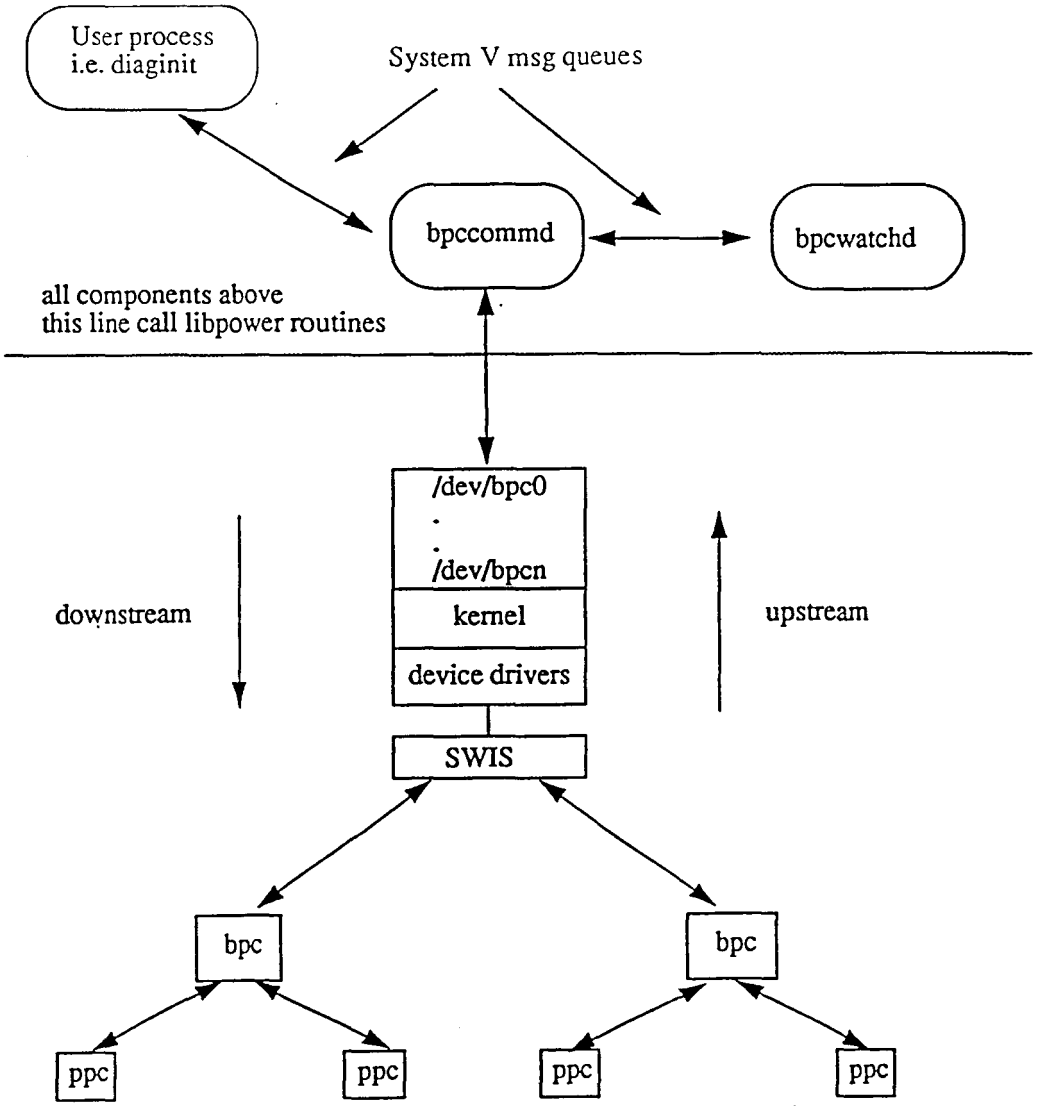


Figure 1-1
Power System Components

Any messages originating from the SPU will cause a response from the bpc or ppc and this response is termed a solicited message. The bpc and ppc also send exceptional conditions and failures which are referred as unsolicited messages. The most common form of these messages are deadman messages which indicate that a bpc channel is still alive and well. The bpcwatchd monitors these deadman messages and signals an error if it has not heard from a channel in a predetermined amount of time.

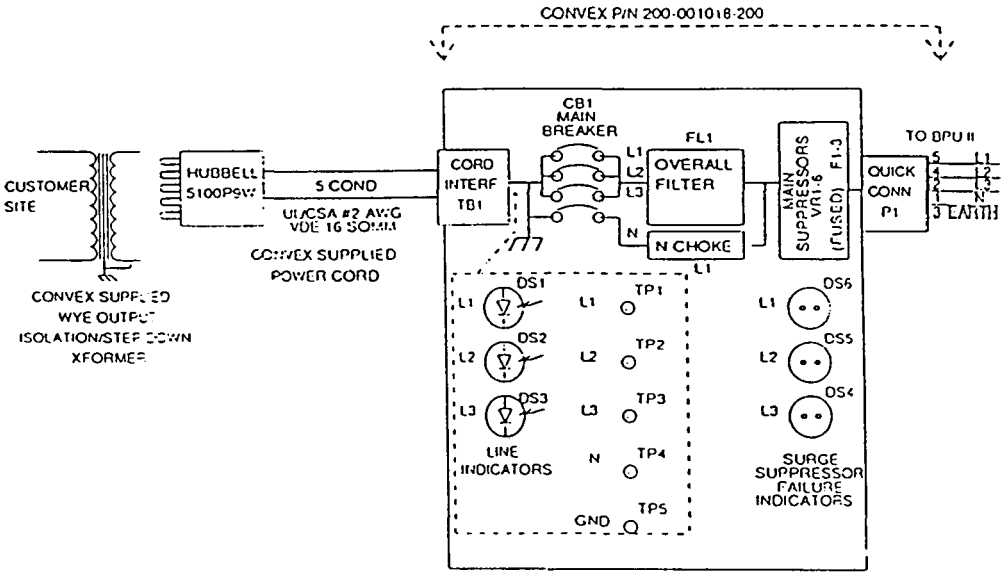


Figure 1-2
Input Power Harmonizer

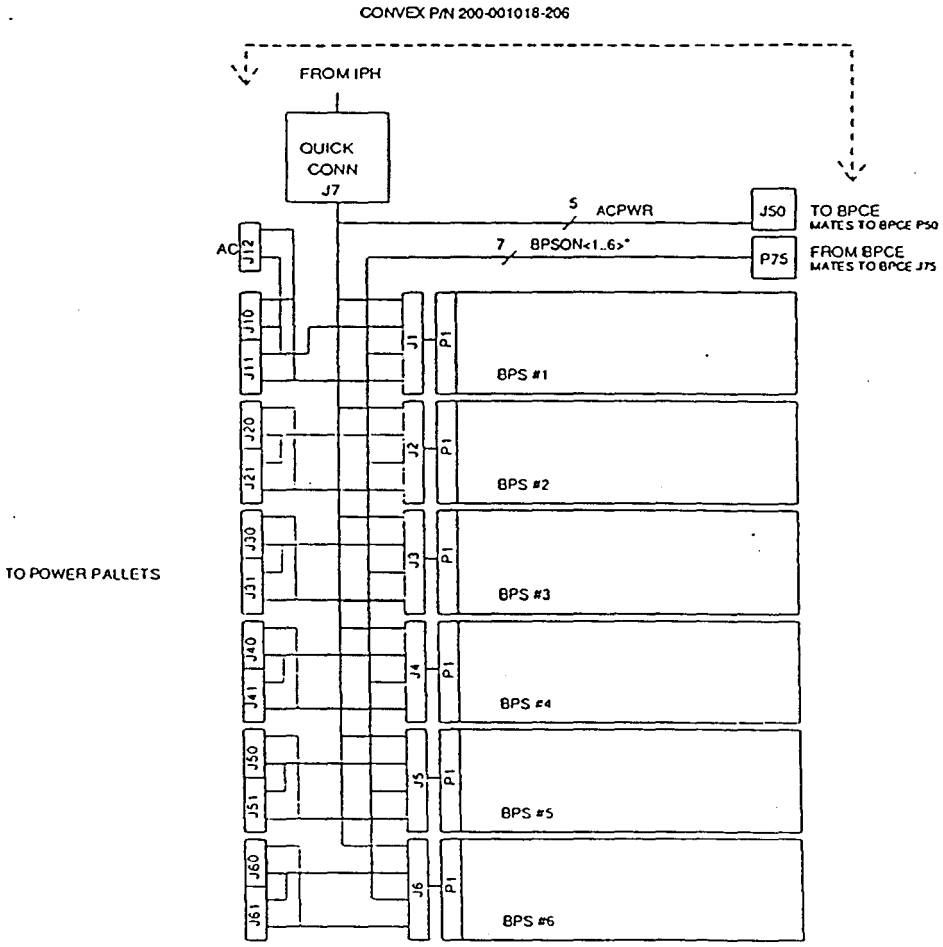


Figure 1-3a
Bay Power Unit Interface

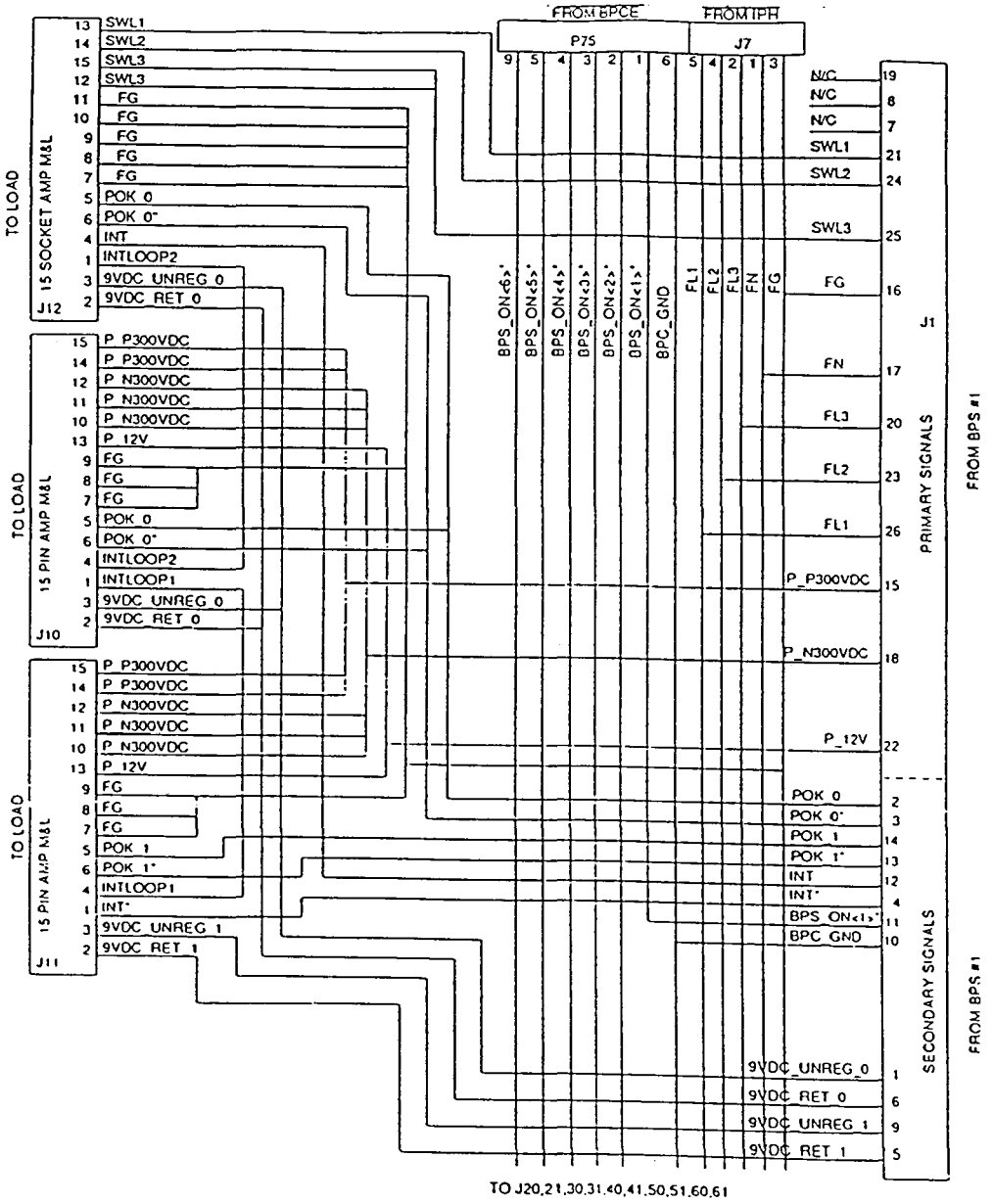


Figure 1-3b
BPU Load Connections

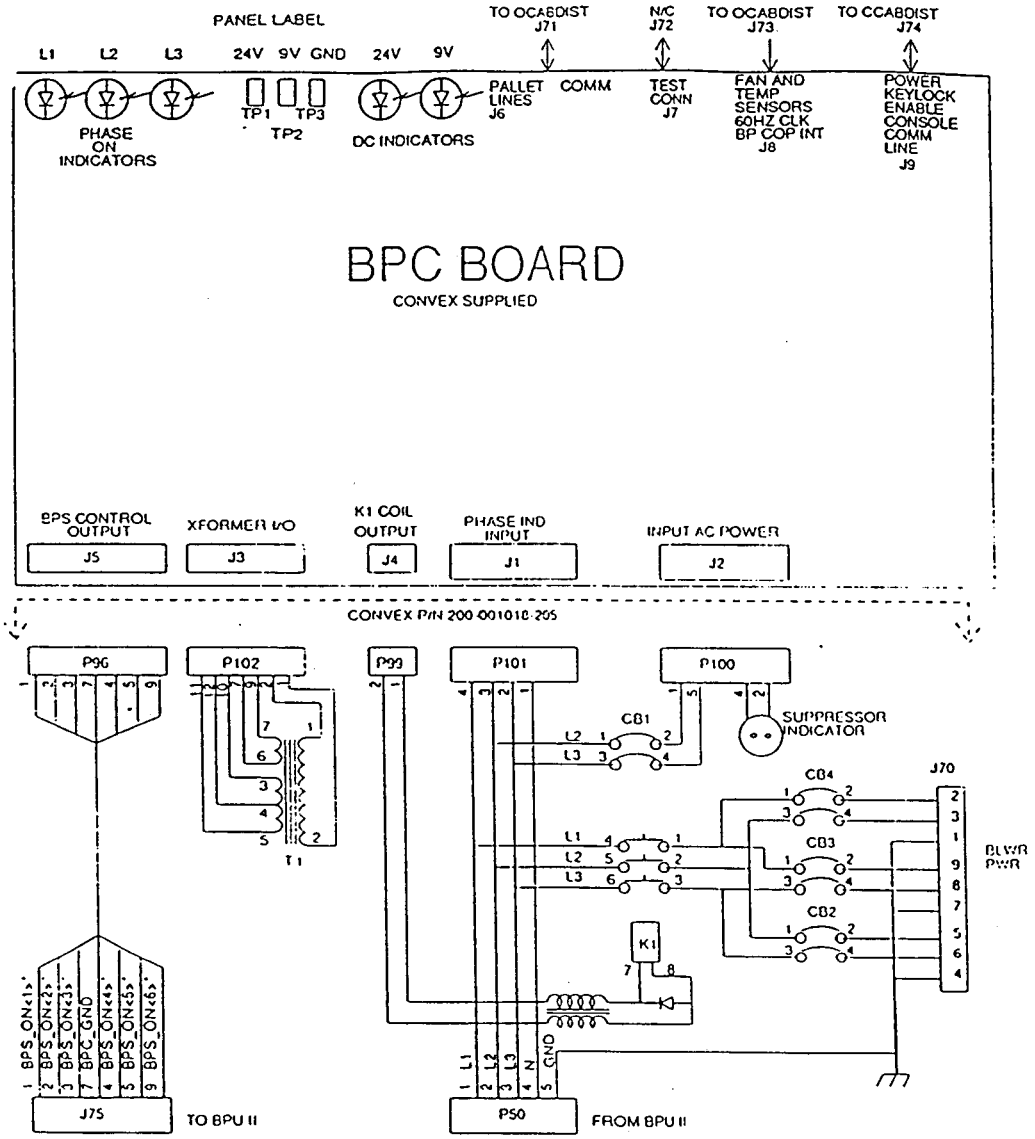


Figure 1-4
Bay Power Controller Enclosure (BPCE)

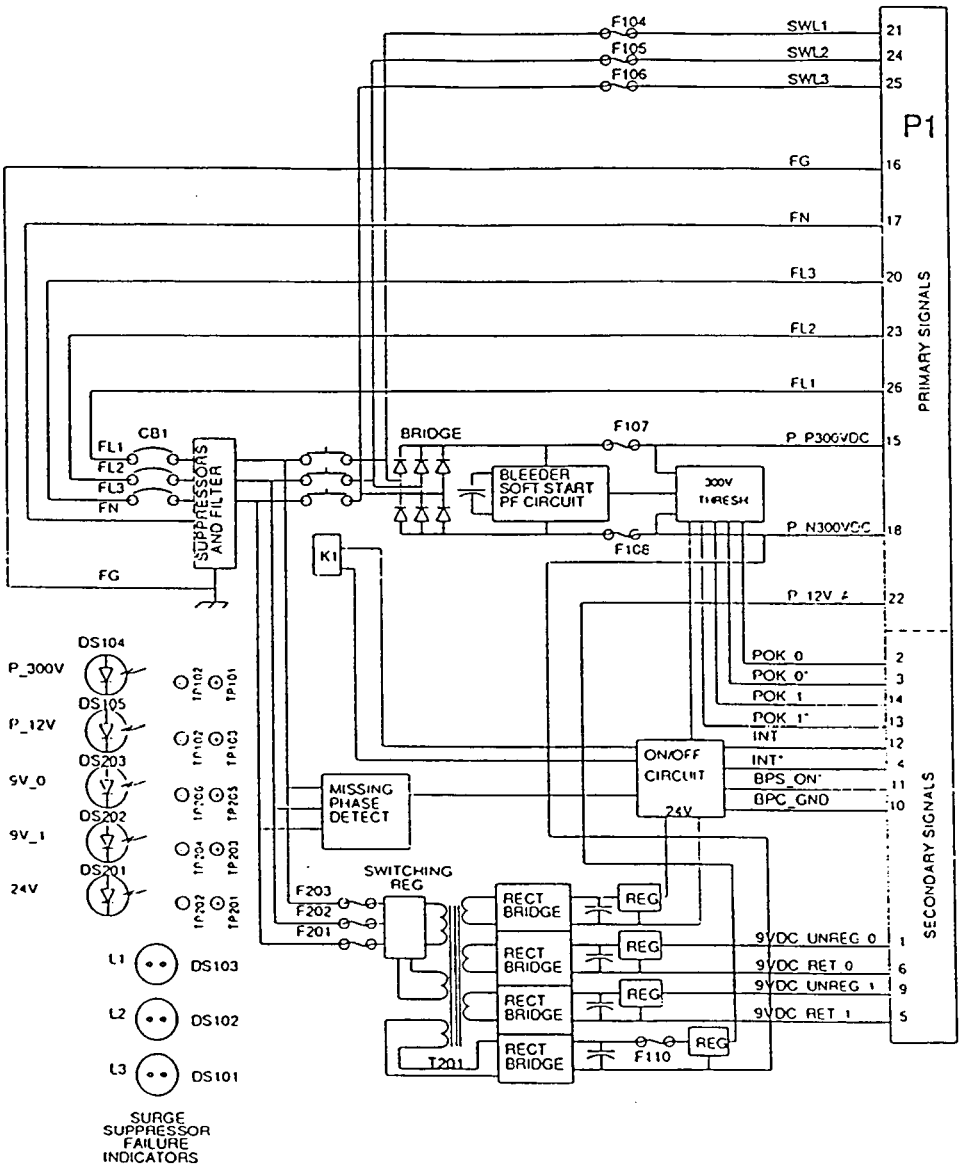


Figure 1-5
Bay Power Supply (BPS)

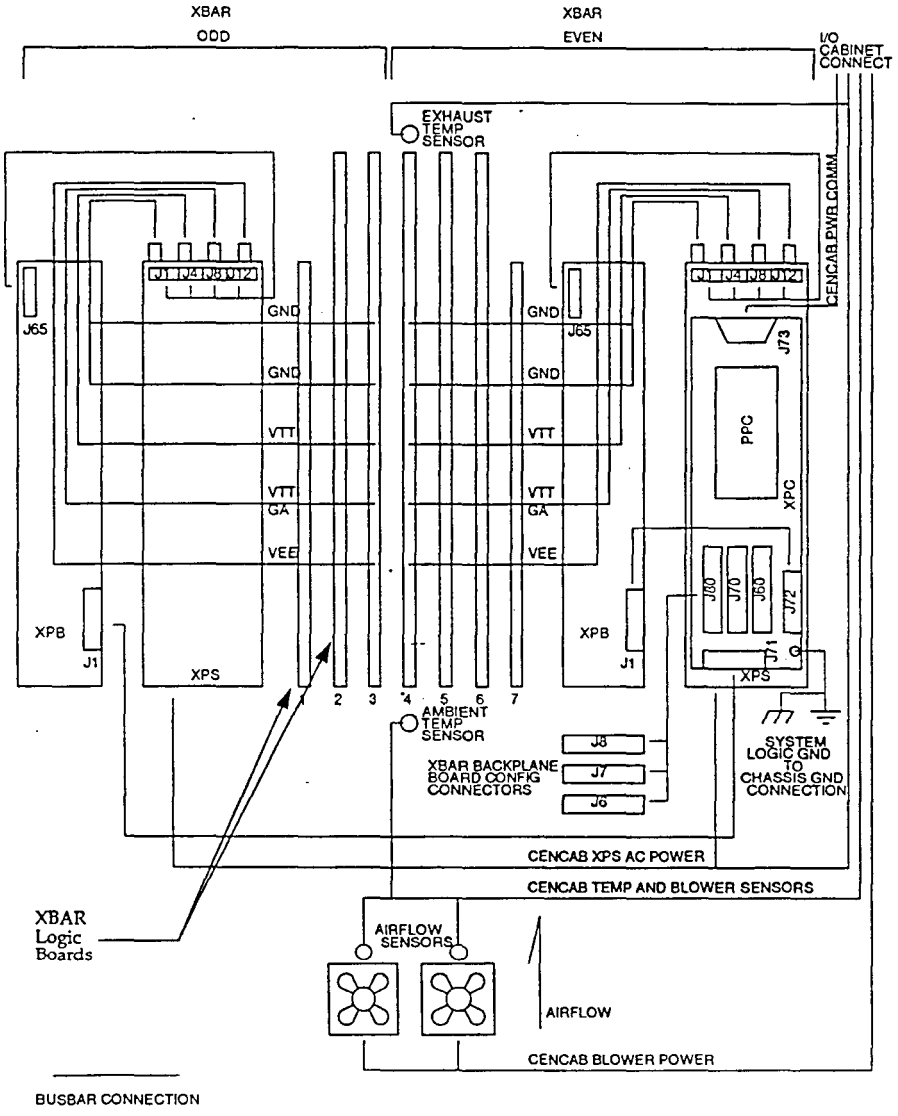


Figure 1-6
XBAR Power Distribution

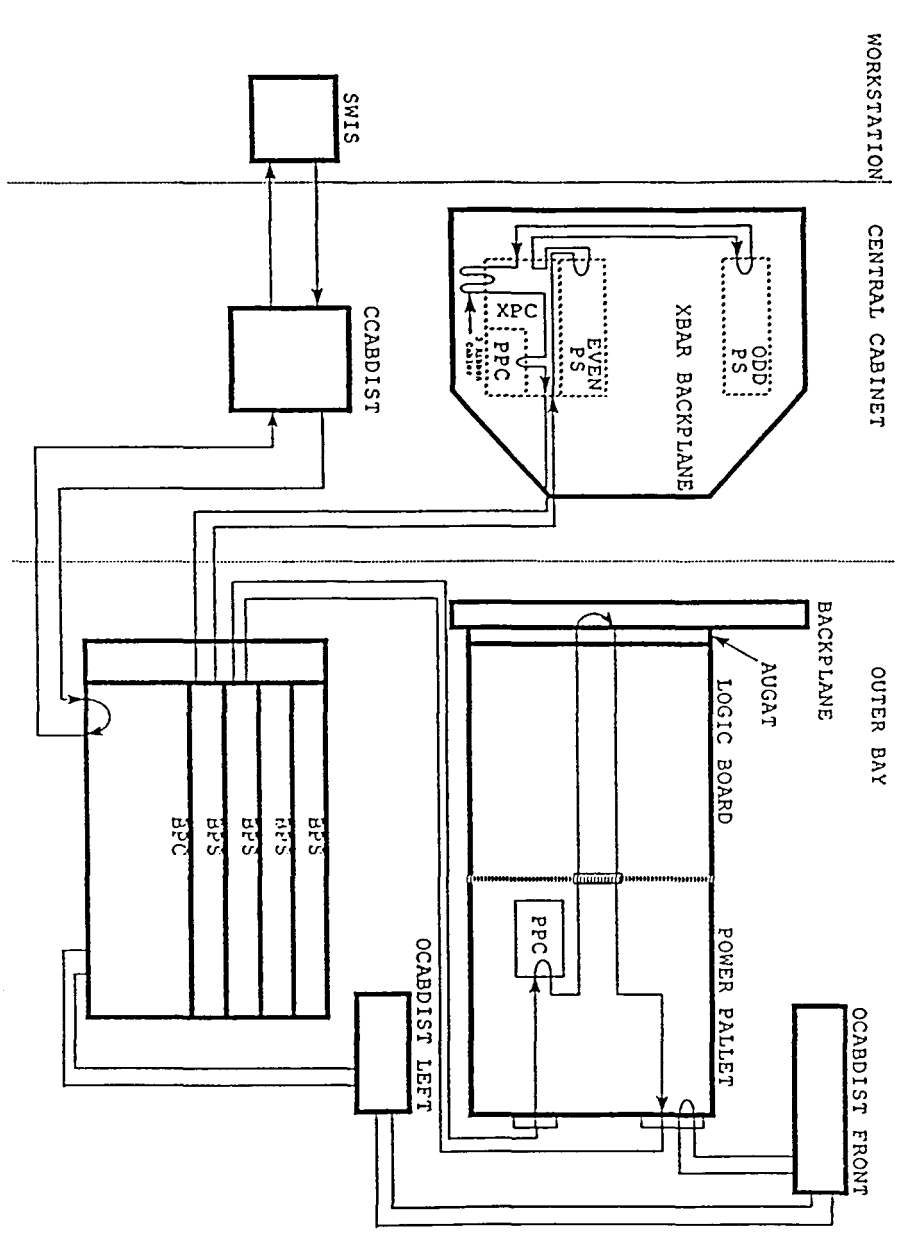


Figure 1-7
System Interlock Circuitry

3.0 Power Tools

3.0.1 spu4000

The class 1 subtests of spu4000 are designed to verify the operation of the SWIS board. These tests must run with the C3800 system powered down.

Subtest 210: SWIS Registers Test

This subtest is a data storage test of registers located on the SWIS. The objective of the test is to verify that these status and control registers can be written and read properly. The patterns used are designed to dedtect shorts, opens, and stuck-at's. Included in this testing is a call to open the printer port and an ioctl call to initialize the uart's and the bpccommd's shadow registers. The registers tested are: (1) Parity Error Force Register; (2) Miscellaneous Control/Status Register; (3) Interrupt Enable Register; and (4) Interrupt Force Register.

```
spu> dump_swis
    swis_bsrc_msb:      16#00
    swis_force_par_err: 16#00 (1)
    swis_misc:          16#09 (2)
    swis_int_level_ctl: 16#90
    swis_int_ena:       16#1f (3)
    swis_int_stat:      16#00
    swis_force_int:     16#00 (4)
     key_switch:        16#17
spu>
```

Subtest 211: SWIS Interrupt Test

The objective of this subtest is to verify operation of the interrupt circuitry on the SWIS.

Subtest 212: SWIS Error Detection Test

The objective of this subtest is to exercise the error detection circuitry of the SWIS and verify its proper operation. For the SWIS, this means verifying that it can correctly detect an invalid address and generate a bus error.

Subtest 213: SWIS UART Register Read/Write Test

The objective of this test is to verify that the Spu can access the registers used to communicate with the 8 channel UART of the SWIS.

Subtest 214: SWIS UART Loopback Test

This is the confidence test of the UART on the SWIS (channels 6 and 7).

3.0.2 display_log

This utility prints messages contained in the specified log file for the specified time period. These messages are printed in sequential order, from oldest to latest, to standard output.

The -(n) option is used to specify the percentage of messages logged to print. This is a floating point number and denotes a percentage of days from the current time. For example, a value of "-1.0" specifies all messages logged within the past day, a value of "-2.0" specifies all messages logged within the past two days and a value of "-.5" specifies all messages logged within the past 12 hours. The default is all messages. NOTE: This percentage of messages logged applies ONLY to normal operations. If the system has been rebooted numerous times within a 24 hour period, or, if the system has crashed as the result of a power problem and 70 + harderrors were reported, all messages recorded during this time frame may fill or overflow the message buffer.

When possible, this utility should be used for all environmental related problems occurring within the system.

3.0.3 pwr_util

Pwr_util is a menu-driven utility that allows the user to directly access many of the elements of the power system and the Service Processor Unit (SPU) to C3800 Series communications system.

The following are examples of utilizing pwr_util to observe power and temperature conditions in a system.

spu> pwr_util

```
 a) Bpc chk interlock   b) Reset uart      c) Bpc reset       d) Init bpc
 e) Offline bpc         f) F/W revision    g) F/W Download    h) Bay Config chk
 i) Cop Read            j) Cop Write       k) Bay Power chk   l) Power On
 m) Power Down          n) Send Status     o) Set voltage     p) Set temp
 r) Busses off          t) Transparent
 q) Quit                ?) Print Menu

Enter command: n
Select uart channel [0-4]: 4
Select ppc id [0-8]
  where 8 indicates bay controller only: 8

+++>
<Mon May 10 15:13:00 1993>   pwr_util(2013):../bay_checks.c:111
Env Info (DiagER236): Bay Power Controller status message

PPC Status for BPC number  04
  bay pp  fw  prt bkpln bkpln plt bd bd bd bd bd bd bd bd bay bps bus bus
  cnf cnf rev  id slot  type  typ 0  1  2  3  4  5  6  7  pwr num  OK  on
0|00  00  020a 08  05    0a    1d 69 00 00 00 00 00 00 00  00  01   00  00
1|ff  ff  ffff ff  ff    ff    ff ff ff ff ff ff ff ff ff  f3  ff   ff  ff
2|00  00  020a 08  07    08    04 04 00 00 00 00 00 00 00  00  01   00  00
3|00  00  020a 08  08    08    05 05 00 00 00 00 00 00 00  00  00   00  00
4|ff  ff  ffff ff  ff    ff    ff ff ff ff ff ff ff ff ff  f3  ff   ff  ff
5|ff  ff  ffff ff  ff    ff    ff ff ff ff ff ff ff ff ff  f3  ff   ff  ff
6|00  00  020a 09  06    01    09 09 0b 0a 09 08 09 0a 0b  00  00   00  00
7|ff  ff  ffff ff  ff    ff    ff ff ff ff ff ff ff ff ff  f3  ff   ff  ff

Temperature Status
  f/w rev temp_stat temp_0 temp_1 temp_2 temp_3 temp_4 temp_5 temp_6 temp_7
    0311      00       ff     ff     aa     a3     a9     a9     ff     93
in_wrm_set in_hot_set out_wrm_set out_hot_set BPC_err fan0 fan1 fan2
    92         7e         5b          4e         00     00   00   00
****
+++>  Send status had a return code of 0 which is good status

 a) Bpc chk interlock   b) Reset uart      c) Bpc reset       d) Init bpc
 e) Offline bpc         f) F/W revision    g) F/W Download    h) Bay Config chk
 i) Cop Read            j) Cop Write       k) Bay Power chk   l) Power On
 m) Power Down          n) Send Status     o) Set voltage     p) Set temp
 r) Busses off          t) Transparent
 q) Quit                ?) Print Menu

Enter command: n
Select uart channel [0-4]: 4
Select ppc id [0-8]
  where 8 indicates bay controller only: 0

+++>
<Mon May 10 15:20:23 1993>   pwr_util(2026):../pwr_util.c:231
SW Info (DiagIN238): Power Pallet Controller status message
```

```
PPC Status
  fill  fw    pp  open err err  plt bd bd bd bd bd bd bd bd slot warm hot bus
  byte state fail ilck grp code id  0  1  2  3  4  5  6  7   id  flag flg trim
  00    02    00   00   f1  1f  1d  04 00 00 00 00 00 00 00  85  00   00   00

  bus_on brick brick brick brick tmp0 tmp1 tmp2 tmp3 tmp4 tmp5 tmp6 tmp7 A/D
  error  exist type  fuses input snsr snsr snsr snsr snsr snsr snsr snsr over
  00     2b03  0000  0000  0000  3f   3f   3f   41   41   00   00   00   00

  hous A/D0 A/D1 A/D2 A/D3 A/D4 A/D5 A/D6 A/D7
  volt cnvt cnvt cnvt cnvt cnvt cnvt cnvt cnvt
  00   f802 0810 ffff ffff f796 fffe 0869 fffe
****
+++>  Send status had a return code of 0 which is good status
```

3.0.4 powermon

Powermon formats the commands necessary to request local environment status from the appropriate BPC or PPC. Xpowermon functions similarly, but it uses the X-Window system rather than Maryland Windows to display the data. Command line arguments for powermon are entered on the screen for xpowermon.

The following is an example of a powermon screen.

```
spu> powermon bay4

                                                          Tue May 11 06:43:03 199
                 ------------------ PPC Status --------------------
  confg  f/w  pt backplane plt brd brd brd brd brd brd brd brd bay bps bus bus
  by pp  rev  id slot type typ -0- -1- -2- -3- -4- -5- -6- -7- pwr num OK  on
0|00 00  2.10 08  05  CCUL CCU ERR                              00  01  00  00
1|   ** ***** **
2|00 00  2.10 08  07  IO_B IA  IA                               00  01  00  00
3|00 00  2.10 08  08  IO_B CU  CU                               00  00  00  00
4|   ** ***** **
5|   ** ***** **
6|00 00  2.10 09  06  XBAR XPB XPB XRT XS1 XS0 XCL XS0 XS1 XRT 00  00  00  00
7|   ** ***** **
```

```
          ------------------ Temperature Status --------------------
          --- Status ---       --- Outlet(C) ---       --- Inlet(C) ---
f/w rev   Sensor   BdErr         warm        hot         warm        hot
 3.17       00      00           45          50          30          35
Temp Sensors:      1        2        3        4        5        6        7        8
   Status:        OK       OK       OK       OK       OK       OK       OK       OK
   Temp(C):       19       16       15       14       14       15       10       10
Fan Sensors:       0        1        2
    Status:       OK       OK       OK
     Value:       00       00       00


spu> powermon ccu32

                                                        Tue May 11 06:52:40 199
PalletID=CCU     BoardID=ERR     PortSlotID=0x85
PrimPower=OK     Interlock=OK    HouseKpg=OK
Trap=0xf11f      VoltAdj=0

              ------ Logic -----        -------- Pallet ------------
Temp Sensors:      1        2        3        4        3        2        1        0
   Status:        OK       OK       OK       OK       OK       OK       OK       OK
   Temp(C):      -17      -17      -17       18       18       17       17       17

ADC Chan:          M4       M3       M2       M1       X1       X2      Vcc      -5V
   Trimmed:        No       No       No       No       No       No      N/A      N/A
   Overrange:      No       No       No       No       No       No       No       No
   Voltage:    -0.00    +5.26    -0.00    -5.26    -0.00    -0.00    +5.04    -5.00

Bricks/Fuses: 300v  Inp  M1  M2  M3  M4  S1  S2  S3  S4  S5  S6  S7  S8  S9  SA
   Exist:                 *       *       *   *                               *   *
   Inputs:               OK      OK      OK  OK                              OK  OK
   Type:                 5V      5V      5V  5V                              5V  5V
   Fuses:        OK   OK  OK      OK      OK  OK                              OK  OK
```

4.0 SWIS Error Processing

There are two types of error detecting internal to the SWIS; hardware and software. Hardware errors are errors that require logging the BSRC. Software errors are detected through polling status registers internal to the OCTART (UART).

Hardware errors are limited to problems in the SBus transaction. These include:

Invalid Address - The swis was given address during an SBus transfer that was not mapped to any device or function.

Invalid Transfer Size - The swis received an SBus extended transfer request.

Invalid Transfer Request - The swis received an Sbus transfer request.

Hardware errors result in the process making the access receiving a bus error signal (SIGBUS). If the process doesn't have a bus error handler, it will be terminated by the kernel.

```
spu> dump_swis
    swis_bsrc_msb:      16#00
    swis_force_par_err: 16#00
    swis_misc:          16#0c
    swis_int_level_ctl: 16#90
    swis_int_ena:       16#1f
    swis_int_stat:      16#00
    swis_force_int:     16#00
    key_switch:         16#17

swis_bsrc_msb
VALUE: 16#00
| 15  | 14  | 13  | 12  | # 11  | 10  | 9  | 8  | # 7  | 6  | 5  | 4  | # 3  | 2  | 1  | 0
|                                                  | # Inv | Inv | Inv |
|          unused                                  | #Addr | Size | Req |
```

Software errors are reported by the kernel in the console window. These errors are considered non-fatal, so the kernel attempts to sync back up to the BPC and continue operation. The OCTART registers are located at board offset 0x300 to 0x3ff.

spu4000 (st210-214) test the swis board and cable. NOTE: Running these tests while the system is powered up, will cause a system crash.

5.0 Crossbar Power Supplies

There are two crossbar power supplies in the C3800 (one even and one odd side supply). The even side power supply also supplies power to the XCL. The power pallet controller for both power supplies is located on the even side power board.

As the XCL is supplied by the even side power supply, this supply is under a greater load than the odd side supply. Because of this, it may be possible to move a weak supply from the even side to the odd side until a replacement can be obtained. It is also possible to isolate a defective even side supply by removing the XCL and powering the XBAR up using the k and l options of pwr_util. If the crossbar will power up with the XCL removed, it is a candidate to be replaced.

The following is a listing of the Xbar even and odd power busses.

Odd busses are called X2, M2, and M4, and are powered from the ODD XBAR supply.

Even busses are called X1, M1, and M3, and are powered from the EVEN XBAR supply.

When the Xbar fails to power, a good tool to use is "pwr_util". Using the n, 4, and 6 commands will display environmental and power on status and may help isolate the problem.

6.0 Power Problem Isolation

It is important to understand that power problems can play a part in many hard crashes and not necessarily be very obvious. In many cases the component that reports the crash is not the one experiencing the power failure.

Quite often the failure will be lost in the noise, but upon closer examination a power problem will be found, either by examining the "display_log", or by examining the error more closely. When this happens the error will be reported, quite often, as a PPC failure. This PPC failure will be followed by a number, as

```
#BPC: 0   PPC: 4              Warning: 300 volts removed from pallet
#BPC config error code:  00   PWR config error code:   19
```

This failure indicates a problem with the power pallet on board number 4 in bay 0. To determine the correct location of the failure it is necessary to count across bay 0 to the correct board. It is important to understand the correct sequence and the fact that slots 0 and 7 will, likely, be empty. In this case the failure will be with NVP 1.

```
0   1   2   3   4   5   6   7

C   N   N   N   N   N   N   C
C   M   S   V   V   S   M   C
U   B   P   P   P   P   B   U
P                           P
P                           P
```

In addition, the BPC status displayed by pwr_util can be used to isolate the failed module. In the case below, again the failure is identified as slot 4. Notice that slots 0 and 7 indicate no activity.

```
--> Bay Power Update Returned 00 ppc mask for bpc uart 0
SW Info (DiagIN236): Bay Power Controller status message

PPC Status for BPC number  00
   bay pp  fw  prt bkpln bkpln plt bd bd bd bd bd bd bd bd bay bps bus bus
   cnf cnf rev  id slot  type  typ 0  1  2  3  4  5  6  7  pwr num  OK  on
 0|ff  ff  ffff ff  ff    ff    ff ff ff ff ff ff ff ff ff  f3  ff   ff  ff
 1|00  00  0205 00  06    02    01 01 00 00 00 00 00 00 00  00  00   00  00
 2|00  00  0205 00  07    02    02 02 00 00 00 00 00 00 00  00  01   00  00
 3|00  00  0205 00  08    02    06 06 00 00 00 00 00 00 00  00  01   00  00
 4|00  00  0205 01  09    03    06 06 ff ff ff ff ff ff ff  19  02   01  ff
 5|00  00  0205 01  0a    03    02 02 00 00 00 00 00 00 00  00  04   00  00
 6|00  00  0205 01  0b    03    01 01 00 00 00 00 00 00 00  00  05   00  00
 7|ff  ff  ffff ff  ff    ff    ff ff ff ff ff ff ff ff ff  f3  ff   ff  ff
```

It is quite common for the board reporting a hard error to have nothing at all to do with the failure. So hard errors should be examined very closely to determine their actual meaning.

The process of troubleshooting a power problem can be an easy one if the tools are properly utilized. The following is intended to be a guide in the isolation of the source of power failures.

```
spu> pwr_util

 a) Bpc chk interlock   b) Reset uart     c) Bpc reset      d) Init bpc
 e) Offline bpc         f) F/W revision   g) F/W Download   h) Bay Config chk
 i) Cop Read            j) Cop Write      k) Bay Power chk  l) Power On
 m) Power Down          n) Send Status    o) Set voltage    p) Set temp
 r) Busses off          t) Transparent
 q) Quit             ?) Print Menu

Enter command: n
Select uart channel [0-4]: 0
Select ppc id [0-8]
 where 8 indicates bay controller only: 2

+++>
<Tue Jun 30 08:02:55 1992>   pwr_util:../pwr_util.c:230
SW Info (DiagIN238): Power Pallet Controller status message

PPC Status
fill  fw    pp  open err err  plt bd bd bd bd bd bd bd bd slot warm hot bus
byte state fail ilck grp code id  0  1  2  3  4  5  6  7   id  flag flg trim
00    02    00   00   f1   1f  02  02 00 00 00 00 00 00 00  07  00   00   00

bus_on brick brick brick brick tmp0 tmp1 tmp2 tmp3 tmp4 tmp5 tmp6 tmp7 A/D
error  exist type  fuses input snsr snsr snsr snsr snsr snsr snsr snsr over
00     3fff  3bdb  0000  0000  57   58   58   56   54   5b   5e   5a   00

hous A/D0 A/D1 A/D2 A/D3 A/D4 A/D5 A/D6 A/D7
volt cnvt cnvt cnvt cnvt cnvt cnvt cnvt cnvt
00   f7fe 0813 ffff f7ad fcb8 fce2 fffe f8ca
****
```

Below is a list of definitions for entries in the example above:

1) brick exist - A "1" is set for every brick installed on the pallet. A different bit is used to represent each brick installed. The below list is used for all brick and fuse status. Where S10 is slave supply 10 and M1 is master supply 1.

    S10=0001 S9=0002 S8=0004 S7=0008 S6=0010 S5=0020 S4=0040 S3=0080

    S2=0100 S1=0200 M4=0400 M3=0800 M2=1000 M1=2000

        example: 3fff indicates that all supplies are installed.

2) brick input - Is the opposite of brick exists, as a "1" indicates that the individual supply is turned off. This is a great aid in determining which power busses are defective.

3) brick fuses - A "1" indicates which fuses are open. In addition to the list of supplies above, this entry has two additional bits.

        FUSED300V=4000   INPUT300V=8000

4) brick type - This entry will indicate the type of bricks installed, where a "1" indicates a 2V brick and a "0" indicates a 5V brick.

The A/Dx fields are a 2's complement hex equivavlent representation of a DC voltage level. To convert to a DC voltage the 2's complement hex value converted to decimal and multiplied by 5/2048. It is important to understand that values of the 0xxx is positive and fxxx is negative.

Example: A/D1 = 0813 converts to 2067 decimal * 5/2048 =5VDC
A/D0 = f7fe converts to 0802 hex and 2050 decimal * 5/2048=-5VDC

Those values that are a 0000 would generally indicate a shorted input. The master supply would be suspect here. A value of ffff would mean that this bus does not exist on this board.

The table below indicates the bus covered by each A/D circuit on a board by board basis.

BOARD	A/D0	A/D1	A/D2	A/D3	A/D4	A/D5	A/D6	A/D7
NMB	PPCM5V	PPCVCC	GND	GND	VTTGA	VTT	VCC	VEE
NSP	PPCM5V	PPCVCC	GND	VEE10K	VTTGA	VTT	GND	VEE
NVP	PPCM5V	PPCVCC	GND	GND	VTTGA	VTT	0	VEE
NIA	PPCM5V	PPCVCC	GND	VEE10K	VTTGA	VTT	VCC	VEE
NCU	PPCM5V	PPCVCC	GND	VEE10K	VTTGA	VTT	VCC	VEE
XBAR	PPCM5V	PPCVCC	VEE	VEE	VTTGA	VTTGA	VTT	VTT
CCU	PPCM5V	PPCVCC	GND	GND	M5V	VTT	VCC	VEE

Where: M5V = -5VDC
VCC = 5VDC
VTT = -2VDC
VEE = -4.5VDC
GND = 0VDC
VEE10K= -5.2VDC (STRAM's)
VTTGA = -2VDC

The temperature sensors located on the boards (tmp0-7) indicate the fahrenheit temperatue in 10mv increments. The conversion is as follows (or you can utilize powermon which will give the temperatures in C):

adc counts * (2.5/.01*255)
2.5/.01*255 = .98
adc count *.98
convert hex value of adc count to decimal and multiply by .98.

example: tmp0=57 0x57 converts to 87 dec.
87*.98 = 85.26 degrees F

The other fields can be identified by use of the C38XX Power System Firmware Description.

With this document and the Firmware Description, it should be relatively easy to identify and correct Power Pallet related failures.

7.0 Examples of Failures as Reported in the Event Log (display_log)

7.0.1 Example of a BPC Failure

```
+++>
<Mon Apr 26 10:22:39 1993>   diaginit(765):../config_update.c:477
Config Error (DiagER375): Invalid pair of pallets sharing bay power supply

@BPC=3   @PPC=1              Warning: 300 volts removed from pallet
BPC config error code:  00   PWR config error code:   f1
@BPS=00
****

+++>
<Mon Apr 26 10:22:40 1993>   diaginit(765):../config_update.c:477
Config Error (DiagER375): Invalid pair of pallets sharing bay power supply

@BPC=3   @PPC=2              Warning: 300 volts removed from pallet
BPC config error code:  00   PWR config error code:   f1
@BPS=00
****

+++>
<Mon Apr 26 10:22:40 1993>   diaginit(765):../config_update.c:477
Config Error (DiagER375): Invalid pair of pallets sharing bay power supply

@BPC=3   @PPC=3              Warning: 300 volts removed from pallet
BPC config error code:  00   PWR config error code:   f1
@BPS=00
****
--> Bay Power Update Returned 70 ppc mask for bpc uart 3

*******************************************************************************
** In the above example, all of the logic boards in bay failed to powerup. **
** The BPC was replaced but the failure continued to exist.  At this point **
** inspection of the connectors (side of bpc) detected a defective wire.   **
*******************************************************************************
```

7.0.2 Example of a BPS Fairlure

```
Env Error (DiagER350): message received by SPU

BPC: 4              BAY:  4
PPC: 6              SLOT: UNKNOWN       TARGET: UNKNOWN
PPC MSG:     MESSAGE ERROR
invalid byte count on PPC message recieved

msgid:81 length:08 source:06 nwi_uart:04 misc0:00 misc1:01 group:80 error:04
@BPC= 4             @BAY=  4            @MSGID= 81   @GROUP= 80  @ERROR= 04
@PPC= 6             @SLOT= UNKNOWN      @TARGET= UNKNOWN
****
```

```
<Fri Feb 19 11:23:09 1993>   /diag/bin/bpcwatchd(247):../bpcwatchd.c:164
Env Error (DiagER350): message received by SPU

BPC: 4              BAY:  4
PPC: UNKNOWN        SLOT: UNKNOWN       TARGET: bay4
BPC MSG:     DEADMAN TIMEOUT on PPC channel(s):
3

msgid:82 length:08 source:08 nwi_uart:04 misc0:00 misc1:00 group:77 error:08
@BPC= 4             @BAY=  4            @MSGID= 82   @GROUP= 77  @ERROR= 08
@PPC= UNKNOWN       @SLOT= UNKNOWN      @TARGET= bay4
****

+++>
<Fri Feb 19 11:23:11 1993>   /diag/bin/bpcwatchd(247):../bpcwatchd.c:164
Env Error (DiagER350): message received by SPU

BPC: 4              BAY:  4
PPC: UNKNOWN        SLOT: UNKNOWN       TARGET: bay4
BPC MSG:     DEADMAN TIMEOUT on PPC channel(s):
6

msgid:82 length:08 source:08 nwi_uart:04 misc0:00 misc1:00 group:77 error:40
@BPC= 4             @BAY=  4            @MSGID= 82   @GROUP= 77  @ERROR= 40
@PPC= UNKNOWN       @SLOT= UNKNOWN      @TARGET= bay4
****

*******************************************************************************
** As the NCU (ppc 3) and XBAR (ppc 6) timed out and the NIA and CCUPP con-**
** tinued with no problems, the NCU/XBAR BPS was replaced.  This did fix   **
** the problem.                                                            **
*******************************************************************************
```

7.0.3 Example of a Power Pallet Failure

```
 +++>
 <Mon Sep 14 11:39:35 1992>   diaginit:../config_update.c:1025
 Config Error (DiagER420): Busses not turned on--existing error -- check busses_
on fld

 @BPC=4              @PPC=2
 bay_config field:       00              ppc_config field:        00
 power_check field:      00
 busses_OK field:        f0              busses_on field:         02
 ****

 +++>
 <Mon Sep 14 11:39:35 1992>   diaginit:../config_update.c:1294
 Config Error (DiagER416): Unrecognized code in busses_on -- hit default case!

 @BPC=4              @PPC=2
 bay_config field:       00              ppc_config field:        00
 power_check field:      00
 busses_OK field:        f0              busses_on field:         02
```

```
****
***** FAILED POWERING IA8 *****
```

7.0.4 Examples of Temperature Failures

```
<Fri Aug 21 09:18:41 1992>   /diag/bin/bpcwatchd:../bpcwatchd.c:169
Voltage Trim (DiagER350): message received by SPU

BPC: 4              BAY:  4
PPC: 6              SLOT: 6              TARGET: xbar
PPC MSG:     Bus voltage trim on ADC channels:
2(X2)
BUS: XVEE   Msg hex:020402090011fffff85ff85f
msgid:81 length:12 source:06 nwi_uart:04 misc0:80 misc1:0e group:02 error:04
@BPC= 4             @BAY=  4             @MSGID= 81   @GROUP= 02  @ERROR= 04
@PPC= 6             @SLOT= 6             @TARGET= xbar
****

+++>
<Fri Aug 21 09:19:21 1992>   /diag/bin/bpcwatchd:../bpcwatchd.c:164
Env Error (DiagER350): message received by SPU

BPC: 4              BAY:  4
PPC: UNKNOWN        SLOT: UNKNOWN        TARGET: bay4
BPC MSG:     REPORTED WARM BPC TEMPERATURE ON SENSORS:
MainCC Outlt

msgid:82 length:08 source:08 nwi_uart:04 misc0:00 misc1:00 group:81 error:20
@BPC= 4             @BAY=  4             @MSGID= 82   @GROUP= 81  @ERROR= 20
@PPC= UNKNOWN       @SLOT= UNKNOWN       @TARGET= bay4
****

*******************************************************************************
** Found to be the temperature sensor in the bay4 fan assembly.              **
*******************************************************************************

+++>
<Thu Apr  8 19:16:30 1993>   /diag/bin/bpcwatchd(242):../bpcwatchd.c:169
Env Error (DiagER350): message received by SPU

BPC: 4              BAY:  4
PPC: 6              SLOT: UNKNOWN        TARGET: UNKNOWN
PPC MSG:     Temperature error on sensors:
4

msgid:81 length:08 source:06 nwi_uart:04 misc0:80 misc1:04 group:f2 error:10
@BPC= 4             @BAY=  4             @MSGID= 81   @GROUP= f2  @ERROR= 10
@PPC= 6             @SLOT= UNKNOWN       @TARGET= UNKNOWN
****

+++>
<Thu Apr  8 19:16:37 1993>   /diag/bin/bpcwatchd(242):../bpcwatchd.c:169
Env Error (DiagER350): message received by SPU
```

```
BPC: 4              BAY:  4
PPC: 6              SLOT: UNKNOWN       TARGET: UNKNOWN
PPC MSG:     Warm temps on sensors:
4,

msgid:81 length:08 source:06 nwi_uart:04 misc0:80 misc1:04 group:01 error:10
@BPC= 4             @BAY=  4            @MSGID= 81   @GROUP= 01  @ERROR= 10
@PPC= 6             @SLOT= UNKNOWN      @TARGET= UNKNOWN
****

+++>
<Thu Apr  8 19:22:06 1993>   /diag/bin/bpcwatchd(242):../bpcwatchd.c:169
Env Error (DiagER350): message received by SPU

BPC: 4              BAY:  4
PPC: 6              SLOT: UNKNOWN       TARGET: UNKNOWN
PPC MSG:     Unexpected Power off message
no error

msgid:81 length:08 source:06 nwi_uart:04 misc0:80 misc1:04 group:00 error:00
@BPC= 4             @BAY=  4            @MSGID= 81   @GROUP= 00  @ERROR= 00
@PPC= 6             @SLOT= UNKNOWN      @TARGET= UNKNOWN
****


** Powermon would help solve this problem **

spu> powermon xbar

                                                        Tue May 11 10:06:31 199
PalletID=XPB    BoardID=XPB     PortSlotID=0x91
PrimPower=OK    Interlock=OK    HouseKpg=OK
Trap=0          VoltAdj=0

                 ----- Logic -----       ------------ Pallet ------------
Temp Sensors:     1       2       3       4       3       2       1       0
   Status:       OK      OK      OK      OK      OK      OK      OK      OK
   Temp(C):      20      21      25      45      25      26      23      17

ADC Chan:         M4      M3      M2      M1      X1      X2     Vcc     -5V
   Trimmed:       No      No      No      No      No      No     N/A     N/A
   Overrange:     No      No      No      No      No      No      No      No
   Voltage:   -1.95   -1.95   -2.05   -2.06   -4.51   -4.50   +5.03   -4.98

Bricks/Fuses: 300v  Inp  M1  M2  M3  M4  S1  S2  S3  S4  S5  S6  S7  S8  S9  SA
   Exist:
   Inputs:
   Type:
   Fuses:        OK   OK

*******************************************************************************
** Sensor 4 is on the XCL -- The xcl temperature caused the BPC to shutdown **
** In this case, the xbar fan door was not properly closed                   **
*******************************************************************************
```

7.0.5 Example of a Fan Failure

```
<Tue Apr 20 17:32:24 1993>   /diag/bin/bpcwatchd(246):../bpcwatchd.c:164
Env Error (DiagER350): message received by SPU

BPC: 4              BAY:  UNKNOWN
PPC: UNKNOWN        SLOT: UNKNOWN        TARGET: UNKNOWN
BPC MSG:     blower operation
Rear outer fan error

msgid:82 length:14 source:08 nwi_uart:04 misc0:00 misc1:00 group:7b error:01
@BPC= 4             @BAY=  UNKNOWN       @MSGID= 82   @GROUP= 7b  @ERROR= 01
@PPC= UNKNOWN       @SLOT= UNKNOWN       @TARGET= UNKNOWN
****


spu> powermon bay4


                  ------------------ PPC Status --------------------
  confg  f/w  pt backplane plt brd brd brd brd brd brd brd brd bay bps bus bus
  by pp  rev  id slot type typ -0- -1- -2- -3- -4- -5- -6- -7- pwr num OK  on
0|00 00  2.10 08  05  CCUL CCU ERR                              00  01  00  00
1|   ** ***** **
2|00 00  2.10 08  07  IO_B IA  IA                               00  01  00  00
3|00 00  2.10 08  08  IO_B CU  CU                               00  00  00  00
4|   ** ***** **
5|   ** ***** **
6|00 00  2.10 09  06  XBAR XPB XPB XRT XS1 XS0 XCL XS0 XS1 XRT 00  00  00  00
7|   ** ***** **
           ------------------ Temperature Status --------------------
          --- Status ---        --- Outlet(C) ---        --- Inlet(C) ---
f/w rev   Sensor   BdErr         warm         hot          warm         hot
 3.17       f2      00           45           50           30           35
Temp Sensors:      1        2        3        4        5        6        7        8
   Status:        OK       OK       OK       OK    WARM     WARM       OK       OK
   Temp(C):       35       32       32       32       31       32       10       10
Fan Sensors:       0        1        2
    Status:       OK       OK       OK
     Value:       00       00       00
```

Sensor not installed or broken.

7.0.6 Example of spu4000 detecting a memory board that failed to power.

```
<Mon Nov 16 08:53:29 1992>   /diag/test/spu/spu4000(585):st_713.c:283
Subtest Fail (DiagER334): Spu_4000 Error: Interrupts Test Failure

Incorrect Interrupt received
Interrupt Status:                              00001001
Expected:                                      00000001
****
Wrong interrupt was received.
        CU Int Status: 1002.
```

```
Expected 2 (System Interrupt 9)

+++>
<Mon Nov 16 08:53:31 1992>   /diag/test/spu/spu4000(585):st_713.c:283
Subtest Fail (DiagER334): Spu_4000 Error: Interrupts Test Failure

Incorrect Interrupt received
Interrupt Status:                                   00001002
Expected:                                           00000002
****
Wrong interrupt was received.
        CU Int Status: 1004.
Expected 4 (System Interrupt A)

+++>


**************************************************************************
**This (st713) is not a connectivity test.  It is a functional test of the **
**interrupt system.  When you get a failure, do a get on the "ncu_int_stat"**
**register.  In this case the 1xxx bit indicates a Memory board soft error.**
**************************************************************************


spu> get ncu_int_stat
ncu_int_stat
VALUE: 16#4001
| 15 | 14 | 13 | 12 # 11 | 10 |  9 |  8 #  7 |  6 |  5 |  4 #  3 |  2 |  1 |  0
|Hard| IA |Sclr| MB #ScnM|ScnM|ClkG|XCL #                  # SYSTEM INTERRUPT
|Err |Soft|Halt|Soft#Accs|Par |Par |Par #     Reserved      #  B |  A |  9 |  8


Bit 15 - Hard Error.                          Bits 3-7 (Reserved)
Bit 14 - NIA Soft Error.                      Bits 0-3 (System Interrupts)
Bit 13 - Scalar Halt.
Bit 12 - NMB Soft Error.
Bit 11 - Scan Memory Access Error.
Bit 10 - Scan Memory Parity Error.
Bit  9 - Clock Generator Parity Error.
Bit  8 - XCL Parity Error.


Note:  The memory board was installed into the cpu backplane but not powered
       up.  If this system had been booted to ConvexOS, excessive ecc errors
       would have been reported.
```

(This page intentionally blank)

Section Two

C3800 Series

NCU Troubleshooting

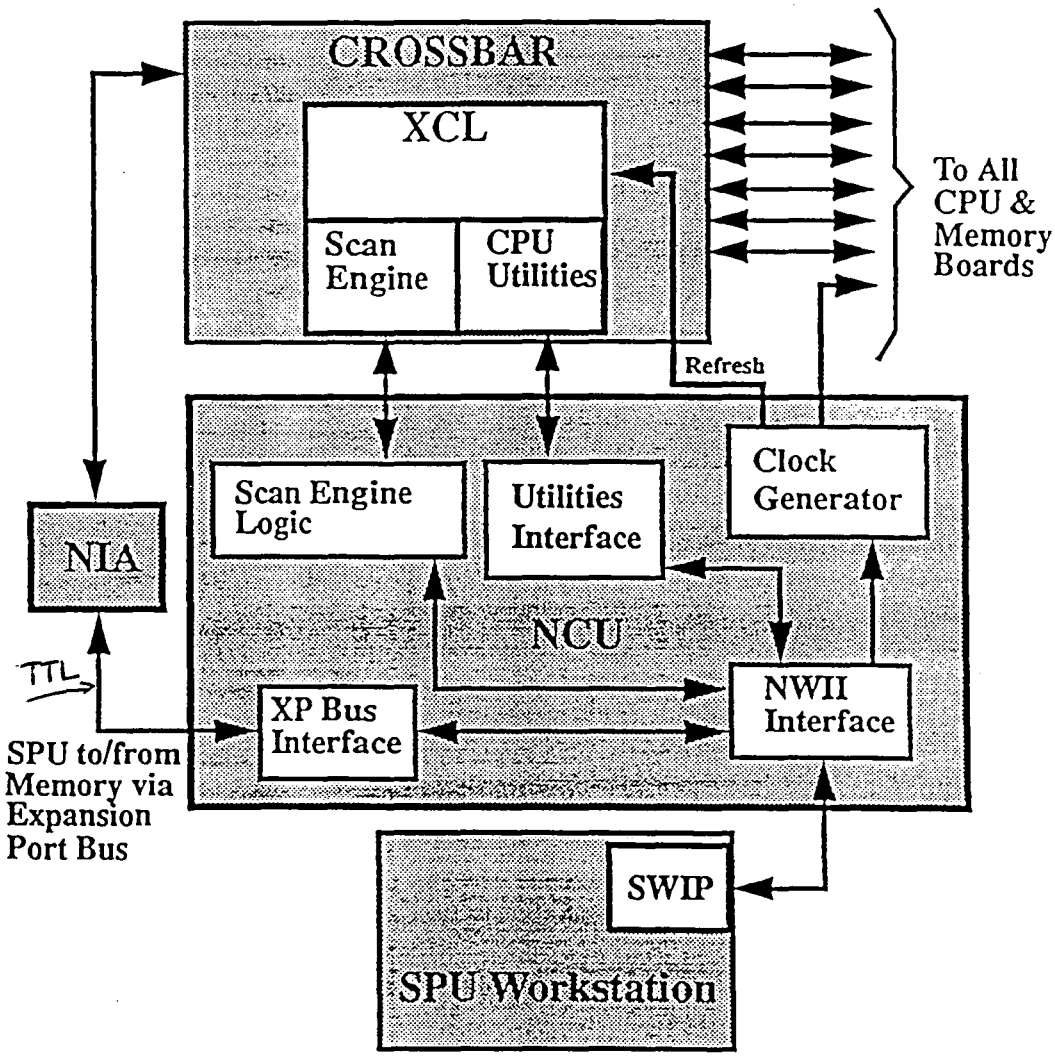
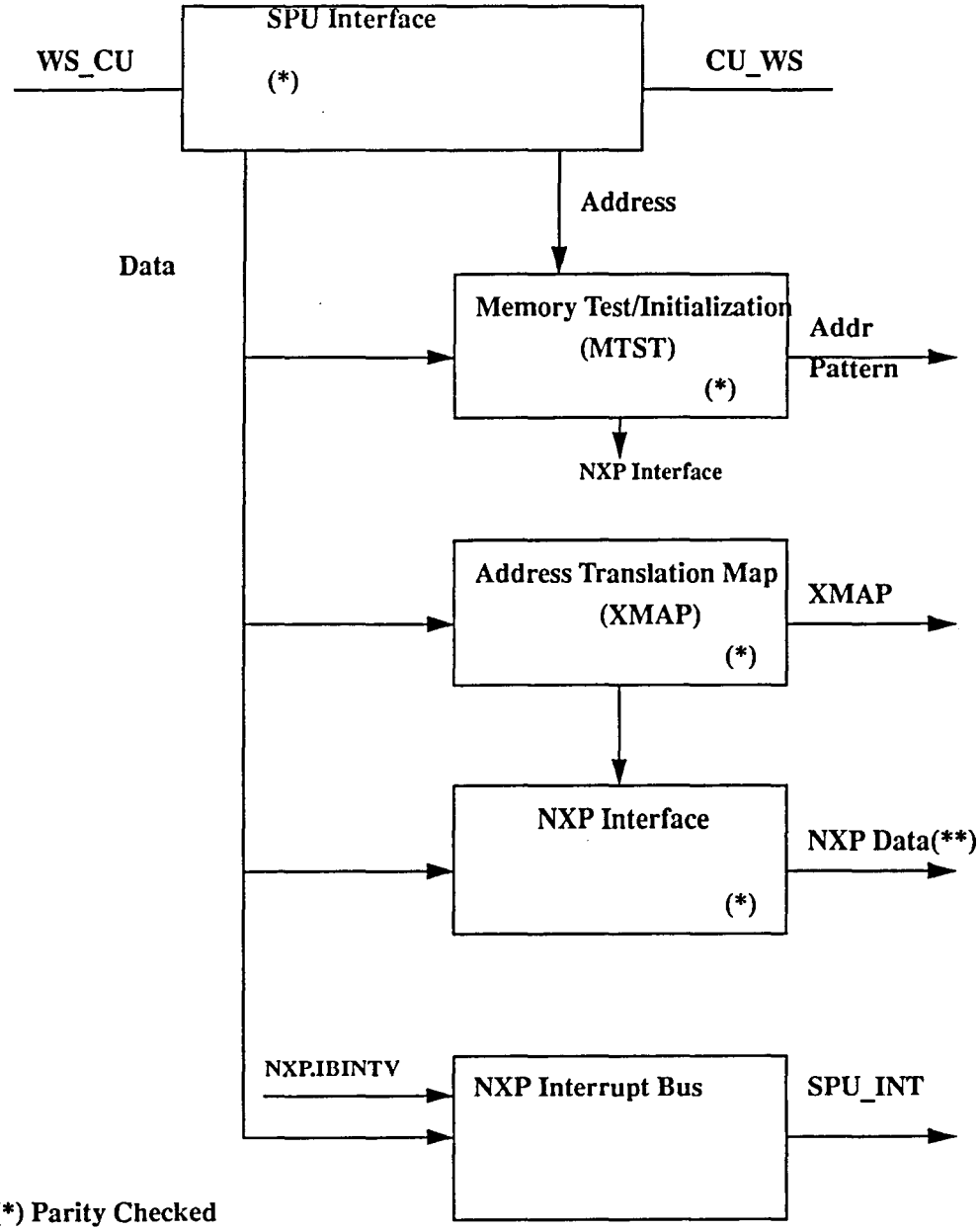


Figure 2-1

NCU Subsystems/Interfaces



(*) Parity Checked

(**) Bidirectional TTL to/from NIA

Figure 2-2
Simplified NWI Block Diagram

1.0 SPU Interface

The SWIP drives several control signals to the NCU. The address strobe (WS_CU.AS) clocks the ADBUS into the ADCTL register. The data strobe (WS_CU.DS) initiates a transaction. The read address strobe (WS_CU.RDADDR) puts the data path into a diagnostic mode. In addition to these, the SPU also provides a reset line (WS_CU.RESET), which resets the NWI portion of the NCU, putting it into a known state prior to the first SPU access.

In the other direction, the NWI (on NCU) drives three control signals on the cable. The first two, CU_WS.ACK and CU_WS.ERR, indicate successful and unsuccessful transactions to the SPU, respectively. These signals are handshake lines with the SPU. The other control signal is an interrupt request to the SWIP (CU_WS.INT). Assertion of this last signal indicates to the SPU that one or more of the NWI interrupts is currently enabled and asserted.

1.1 NXP Interface

The Neptune Expansion Port (NXP) is an ~~ECL~~ TTL version of the existing PBus architecture on the C3200's and C3400's. This port can run at up to 240 MB/sec, utilizing a 64-bit data path, and is **controlled by the NIA.** The NXP I/F portion of the NWI provides access to this port. In conjunction with the NXP Address Translation Map, the SPU can access any location in main memory or I/O space through this interface. In conjunction with the Memory Initialization Logic, the SPU can pattern test main memory at nearly the full bandwidth of the NXP.

1.1.1 NXP Error Conditions

Many different situations on the NXP Bus could result in one of three error bits getting set. Some of these conditions will actually result in multiple bits being set. To help further determine what the exact cause of the error is, it will often be necessary to examine the state of the NIA which is the master of the NXP Bus. Each of the three NXP errors are described below.

1.1.1.1 NXP Busy (NBSY)

When the Memory Test/Initialization (MTST) logic is active, it assumes ownership of the NXP I/F to access main memory. During this time, the NXP I/F is not available to the SPU for direct access to main memory or the I/O space. Should the SPU attempt to access the NXP while the MTST logic is active, the NCU will return an error. At the same time, the NBSY bit of the ncu_err_log register (bit 3), will be set, indicating the NXP was busy during the last SPU access.

1.1.1.2 NXP Bus (NBUS) Errors

There are three conditions which can produce bus errors on the NXP port; illegal header, parity errors, and PCM errors. For all types of accesses, the NIA checks the header information for both content and parity. An invalid header for either reason results in a bus error (NIA asserts IA_XIOP.BUS_ERR).

In the case of a **write operation,** data driven from the NCU is **parity checked on the NIA.** As in the case of an invalid header, a parity error in the write data results in a bus error.

In the case of a **read operation,** data driven from the NIA is **parity checked on the NCU.** A parity error results in a bus error, although the IA_XIOP.BUS_ERR signal is not asserted by the NIA.

When the NIA detects a PCM error, the IA_XIOP.BUS_ERR signal will be asserted by the NIA and the NBUS bit will be set. The NIA will have to be examined to determine if a PCM error has occurred.

All of these bus error conditions result in an error being reported to the SPU, and the NBUS bit of the ncu_err_log register (bit 4) being set. The parity error bits for the 8 byte data word in question (data or header) are saved in the ncu_err_log register (bits 31-24) as well (see Figure 2-3).

1.1.1.3 NXP Sequencing (NSEQ) Errors

The NXP I/F state machine expects a definite sequence of states during a valid NXP access. If the sequence is not correct, an error results. For instance, if the grant from the NIA is negated part way through a cycle, or if the SPU access in progress terminates as a bus time-out, the NXP I/F logic detects a sequencing error. This results in an error being reported to the SPU, if possible, and the NSEQ bit (bit 5) of the ncu_err_log register is set. Other conditions that can cause this bit to be set are by more than one of the following operations being specified; TAC, TAS, I/O and Scrub. These operations are mutually exclusive and are only used in the XMAP interface to main memory. The following figure and definitions are brief explanations of the ncu_err_log.

spu> get ncu_err_log

ncu_err_log

VALUE: 16 # ff00000000

31	30	29	28	# 27	26	25	24	# 23	22	21	20	# 19	18	17	16
I			NXPERR		<7..0>			#	Reserved			# XPERR <3..0>			

15	14	13	12	# 11	10	9	8	# 7	6	5	4	# 3	2	1	0
I APERR <3..0>				#	DPERR <3..0>			#res	libsy	lnseq	lnbus	#nbsyl	xmpl	wdpl	ap

1.2 NXP Address Translation Map (XMAP)

The XMAP is a block of RAM which implements a set of 1024 32-bit registers. These registers are accessed in two different modes, direct and indirect. Both modes require a collection of address bits from the address supplied by the SPU.

1.2.1 SPU XMAP Access

The SPU can read and write the RAM contents directly by addressing the XMAP registers. In this mode, the SPU can initialize the window mapping registers needed for accessing main memory. Parity is check on both reads and writes. If a parity error is detected, the XMP bit of the ncu_err_log (bit 2) is set, and an error is reported to the SPU. The parity error bits are also stored in the XPERR bits of the ncu_err_log (bits 19-16).

1.2.2 Using the XMAP with the NXP

The XMAP is also referenced by any SPU accesses which select the NXP. The XMAP is responsible for providing the window mapping information, used to translate SPU addresses to main memory addresses, as well as determining the type of operation to be performed.

1.3 Memory Test/Initialization (MTST) Logic

1.3.1 Purpose

The NCU has memory test and initialization logic designed in, to quickly initialize and pattern test main memory. The initialization and test functions are accessed through 4 registers. Two of the registers are for the Even and Odd pattern words. The third register contains the physical address in main memory, and the last register starts the desired operation and indicates the transfer count.

1.3.2 MTST Control Registers

The MTST logic is controlled by four 32-bit registers on the NCU. **The contents of these two registers are of little value to the System Engineer, but to see their contents, perform the following get commands.**

spu> get mem_test_odd

spu> get mem_test_even

The Main Memory Address register contains the current NXP transfer address. Since this interface only transfers longwords, the least significant three bits of this register are always forced to zero. **This register is of little value to the System Engineer, but to see the contents, perform the following get command.**

spu> get main_mem_addr

1.4 NWI Interrupts

This portion of the NWI maps NCU errors and System interrupts into an interrupt request line to the SPU. In addition to interrupt mapping, the NWI also provides an interface to the System interrupt Bus (NXP Interrupt Bus), allowing the SPU to send and receive system interrupts.

1.4.1 Interrupt Mapping

The NWI maps twelve interrupts into the SPU interrupt request line (CU_WS.INT). These interrupts have status and enable bits associated with them as follows:

spu> get ncu_int_stat

15	14	13	12	# 11	10	09	08	# 07	06	05	04	# 03	02	01	00
Hrd	IA	Sclr	MB	# MB	MB	CG	xcl	#	Reserved			# System Interrupt			
Err	Soft	Halt	Soft	#Accs	Par	Par	Par	#				# B	A	9	8

spu> get ncu_int_ena

15	14	13	12	# 11	10	09	08	# 07	06	05	04	# 03	02	01	00
		Reserved					ENA	#	Reserved			#	Int Enable		
							NCU	#				# B	A	9	8

The ncu_int_stat register provides information about the current state of interrupts mapped into the SPU interrupt request line. Once an interrupt has been asserted, the corresponding status bit remains set until cleared by the SPU. The SPU clears a bit position in the ncu_int_stat register by writing a one to that position. The individual bits of the ncu_int_stat register are as follows:

Bit 15 indicates a Hard Error has been pulled.

Bit 14 indicates the NIA has pulled a soft error (see dump_soft_log).

Bit 13 indicates a NSP has halted.

Bit 12 indicates a NMB has pulled a soft error (see dump_soft_log).

Bit 11 indicates a Scan Memory Access Error.

Bit 10 indicates a Scan Memory Parity Error.

Bit 9 indicates a Clock Generator Parity Error (Boards will not scan).

Bit 8 indicates an XCL (Xbar Control Logic board) Parity Error.

Bits 7-4 are reserved for future system interrupt use.

Bit 3 indicates an MBS error when ConvexOS is running.

Bit 2 - not reserved for any specific purpose.

Bit 1 - not reserved for any specific purpose.

Bit 0 - not reserved for any specific purpose.

The ncu_int_ena register is of no value to the System Engineer.

1.5 Diagnostic and Utility Registers

The remaining SPU-addressable registers in the NWI section of the NCU are of either a diagnostic or utility nature. The following are explanations of the purpose and use of some of these registers.

1.5.1 Loopback Data Register

The loopback data register provides the SPU with a means of testing the data path to the NCU. The register is physically implemented by the hardware in the data read mux register (RDMUX_REG). See the NCU Troubleshooting Manual for testing procedures.

1.5.2 Scalar Halt Register

The Scalar Halt register is a 32-bit read/write register that is used to read, mask and clear the state of the Scalar Processor Halt signals. This register is divided int three fields:

1) The Scalar Halt state which contains the state of the NSP Stop_cntr signals.

2) The Scalar Halt mask which is used to mask off individual Scalar Halt signals.

3) The SCALAR (NSP) HW HUNG field which reflects the state of the hw_hung signals received from each scalar processor in the system.

1.5.3 Hard Error Registers

There are two 32-bit read/write Hard Error Registers that contain registered versions of the hard error signals output from each of the boards in the system. When one of these bits become active, the clock generator will disable clocks to all of the boards in the system.

The following are examples of the hard_err1 and hard_err2 register contents.

```
spu> get hard_err1

hard_err1

VALUE: 16#00000000

|31 |30   |29|28 # 27|26|25|24      # 23|22|21|20 # 19|18    |17|16

|reserved  |       BAY 3                  #      BAY 2                |Base I/O

|15|14|13|12  # 11|10|9|8 # 7|6    |5|4 # 3|2|1|0

|Base I/O LSB   #          BAY1          |      BAY 0

spu> get hard_err2

hard_err2

VALUE: 16#00000000

|31|30|29|28 # 27|26|25  |24 # 23|22|21|20 # 19|18|17|16

|         Reserved              |          IA Soft Errors

|15|14|13|12 # 11|10|9|8     #  7   |6|5|4 # 3|2|1|0

|         MB Soft Errors           #Rsvd|      CROSSBAR
```

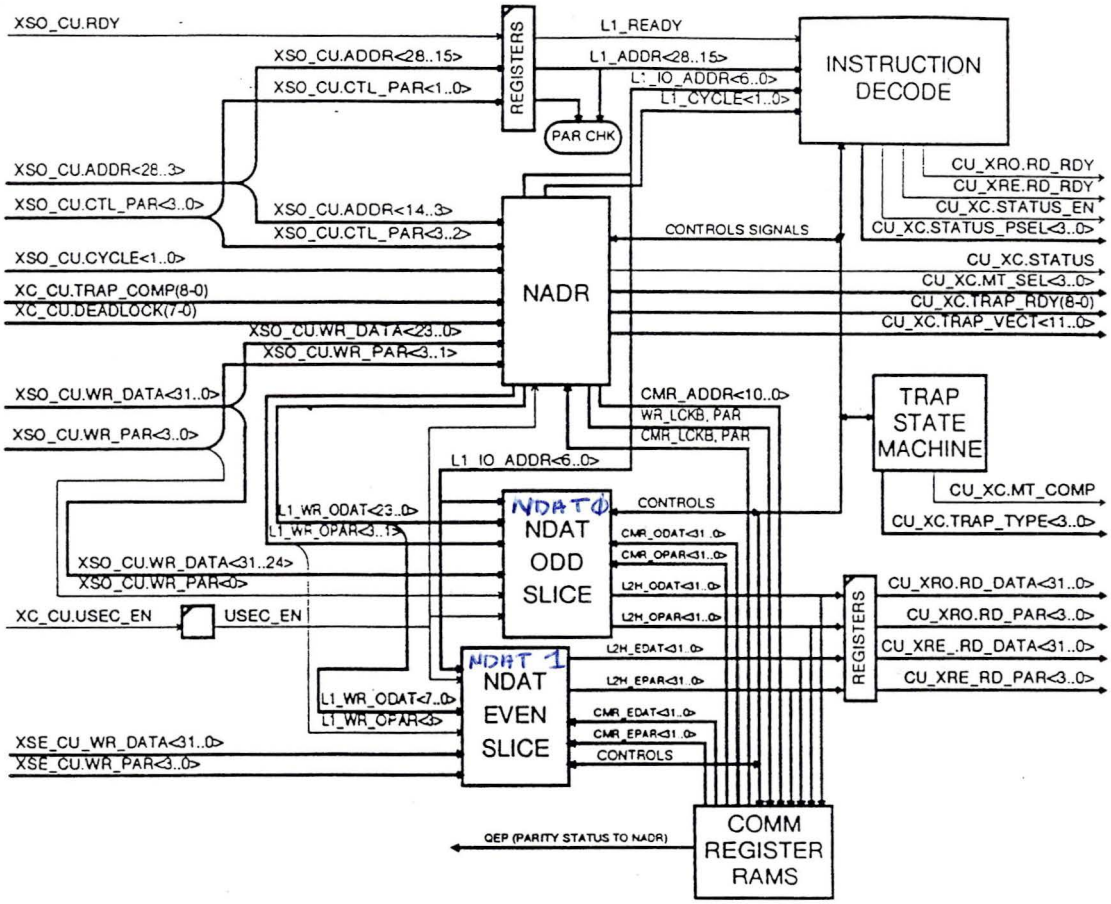


Figure 2-3. CU Block Diagram

2.0 NCU <-> SPU and Scan Troubleshooting

2.1 NCU <-> SPU and Scan Problems

2.1.1 spu4000 subtests

Subtest	Description
210-214	SWIS Tests (Will crash the system if run while the C3800 is powered up)
220	To verify the ability of the SPU to access the SWIP; force parity error register, miscellaneous register,interrupt enable, and the forced interrupt register.
221	To verify the data bus out of the NCU connector. This tests the SWIP data loopback register.
222	To verify the SWIP interrupt circuitry.
223	To verify the SWIP error detection circuitry.
310	SWIP-NCU Parallel Cable Test
311	NCU Memory-Test Registers Test
410	NCLK Register Test
411	NCU Scan Engine Register Test
413	SECG Register Uniqueness Test
414	Scan Memory Test
415	Scan Memory Uniqueness Test
416	Local Loopback Scan Test (scan engine only).
417	Local Loopback Scan With Verify Test
418	XCL Loopback Scan With Verify Test
420	XMAP Memory Test
421	XMAP Memory Uniqueness Test
510-614	Verify scan rings on logic boards.
615	Clock Cable Test

2.1.2 Bus Error/NCU Lost Communications during diaginit.

a. Tools

1) spu4000
2) NCU <-> SWIP cable test.
   put 0x214 (various patterns; i.e., 0x0 then 0xffffffff)
   get 0x214
3) Test the internal loopback address registers on the SWIP.
   put 0x210 (various patterns)
   get 0x210
4) Test the internal loop data registers on the NCU.
   put ncu_data_loop_back (various patterns)
   get ncu_data_loop_back

Note: A buserr during puts and gets to these registers may indicate a bad oscillator.

Note: For intermittent failures, it may be desired to loop on the above tests while you wiggle and check cable connections. The following is an example of how to loop on these tests.

spu> while true
More, please: do
More, please: put 0x214 0xffffffff
More, please: get 0x214
More, please: done

2.1.3 Buserr occurring during diaginit.

Powered VP 1
Powered SP 1
Initializing Scan Engine
Running spu_4000 st_310
Running spu_4000 st_410
bus_error_regs(): Bus error trapped. See spu console for info
Abort(coredump)

a. Possible Causes:
   Clock Cables (Xbar)
   Defective NCU (oscillator/transceivers)
   Defective SWIP (transceivers in thermal shutdown)
   NCU augat problem

b. Tools to Use
   spu4000
   0x214 (put/get)
   Reset NCU
      put swip_misc 0x20 (resets ncu)
      put swip_misc 0
      put swip_bsrc_lsb 0x0
      sysreset
      cleanup
      cleanup
      cleanup
      sysreset -cu
      sysreset -s
   diaginit
   sys_shutdown
   diaginit

2.1.4 SPU Hangs

There are two types of hangs on the SPU - Software and Hardware.

A software hang is when the SPU can still respond (i.e., the cursor can still be moved). This can be recovered from by going to another window, doing a "ps aux" to try and identify the hung process ID and killing this process. If this is not possible, then perform the following:

```
spu> ps aux | grep bpc
/diag/bin/bpccommd
/diag/bin/bpcwatchd
spu> ps aux | grep server
/diag/bin/cdbserver
/diag/bin/rbserver
```

If any of these are not running, perform a diaginit. If diaginit does not work, you will have to reboot the spu.

```
spu> su
spu> Password:
spu> /etc/reboot
```

If cdbserver will not start in the reboot process, try this:

```
spu> ps aux | grep server
```

Kill any server still running (rbserver).

```
spu> rbcdb_int -1
spu> su
spu> /etc/reboot
```

A hardware hang has occurred when the SPU will not respond to keyboard entries or cursor movement. At this point, a power cycle of the spu is required. Follow the above steps if there any problems during the reboot cycle. Expect repeated retrys due to errors in the fsck process.

2.1.5 NCU Errorlog.

The contents of the NCU errlog can be used as an aid to help isolate NCU related problems. The following is an example of an NCU errlog contents followed with a brief description of each register bit.

```
spu> get ncu_err_log
ncu_err_log
VALUE: 16#ff000000
|31 |30 |29 |28 | #27|26 |25 |24    # 23|22 |21 |20 | # 19 | 18 | 17 | 16
|      NXPERR <7..0>                  # Reserved          # XPERR <3..0>

|15 |14 |13 |12| # 11   |10 |9  |8  | #7 | 6| 5| 4# 3 | 2| 1 | 0
| APERR <3..0>  #    DPERR <3..0> # r|  I   N   N# N   X   W   A
                                     e   B   S   B   B   M   D   P
                                     s   S   E   U   S   P   P
                                         Y   Q   S   Y
```

NXPERR bits <31..24>, indicate which byte(s) had parity errors when a parity error is detected on reads from the NXP. If one or more of these bits is set, the NBUS bit <4> should also be set.

XPERR bits <19..16>, indicate which bank(s) on the XMAP RAM reported a parity error on the most recent access to the XMAP. APERR bits <15..12>, are the parity error bits associated with the contents of the ADCTL register, which is loaded by the SPU. One or more of these bits set implies that the AP bit should also be set.

DPERR bits <11..8>, are the parity error bits associated with data written from the SPU. One or more of these bits set implies that the WDP bit should also be set.

IBSY bit <6>, indicates that the NXP interrupt Bus had a pending interrupt when the SPU attempted to write a new vector to the SIB_XMT register.

NSEQ bit <5>, indicates that the NXP access resulted in a sequencing error.

NBUS bit <4>, indicates an NXP bus error, detected either by the NCU or the NIA.

NBSY bit <3>, indicates that the NXP was in use by the MTST logic when the SPU attempted to access the NXP directly.

XMP bit <2>, indicates an XMAP Parity error.

WDP bit <1>, indicates a Write Data Parity error in the data path from the SPU.

AP bit <0>, indicates an Address Parity error in the address/control information from the SPU.

2.2 SWIP Error Processing

There are two sources of errors from the SWIP: internal and NCU. Both of these sources perform error logging to the BSRL and BSRU registers.

```
spu> dump_swip
 swip_bsrc_msb: 16#00
 swip_bsrc_lsb: 16#10
 swip_force_par_err: 16#00
 swip_misc: 16#00
 swip_int_level_ctl: 16#90
 swip_int_ena: 16#80
 swip_int_stat: 16#00
 swip_force_int: 16#00

-----------------------------------------------------------------
swip_bsrc_msb
VALUE: 16#00
| 15  | 14  | 13  | 12  | # 11 | 10  | 9  | 8  | # 7  | 6    | 5   | 4   | # 3  | 2   | 1   | 0
|                                                 #Inv  | Int  |Re-  |NCU# PERR
|              unused                             #Add  |Lock  |run  |Err  #
```

+ INValid ADDRess (7) - An operation was attempted on or through the SWIP that violated the address map. Either address decoding failed due to hardware problems or software attempted to access an illegal address.

+ cable INTerLOCK (6) - The interlock signal detects breaks at several points. Back panel of the SPU, I/O bulkhead connection, I/O backplane connection, not installed or improperly installed NCU.

+ RERUN errors (5) - A rerun error is generated when the SWIP is unable to complete a requested transaction to the NCU. This can be caused by cable disconnects, powered down NCU, or clock incorrectly configured on the NCU (usually turned off). **Crossbar
and memory initialization problems can also interfere with transaction requests to main memory.**

+ NCU ERRor (4) - Error assertion is the default state of the NCU ERROR signal when
ever the NCU does not have power applied to the board. The NCU may issue an error to the SWIP for three different circumstances.

1) Parity or invalid address header that the SWIP has latched into the NCU - This problem is generally related to a hardware transceiver failure on either the SWIP or NCU, or a cable con nection problem. A write/read pattern test of the 0x214 register will usually detect the failed signal.

2) Data parity failure detected on data sent to the NCU by the SWIP during a write transaction. A read of the ncu_err_log register will indicate if the data parity bit is set, along with associated parity syndrome bits for the corrupted bytes of the transaction. This is normal a hardware failure.

3) Operations internal to the NCU and NXP interface on the NCU.

+ Parity ERRor (3..0) - Parity errors are generated on corrupted read transactions with the NCU. Parity errors are usually due to a failed transceiver on the SWIP or NCU, or a broken wire in the NCU cable.

---

swip_bsrc_lsb
VALUE: 16#10

15	14	13	12	# 11	10	9	8	# 7	6	5	4	# 3	2	1	0
								# Inv	Inv	Inv	Err				
			unused					#Add	Size	Req	Ena				

+ **INValid ADDRess (7)** - The SWIP was given an address during an SBus transfer that was not mapped to any device or function.

+ **INValid SIZE (6)** - The SBus protocol allows transactions that move 8, 16, or 32 bit data values. If an illegal width access transaction is attempted on a restricted register, the INVALID SIZE bit is set.

+ **INValid REQuest (5)** - The SBus protocol contains several type of transactions. The SWIP only supports simple byte, word, long word, and rerun transactions. If any other type of transaction is requested, such as burst or extended, the INVALID REQUEST bit will be set to 1

.

+ **ERRor ENAble (4)** - When the ERROR ENABLE bit is set, all error conditions will be reported. When the ERROR ENABLE is set to 0 only rerun errors will cause an error acknowledgment to occur.

3.0 NCU <-> System Troubleshooting

3.1.1 **CU Extractor Rules.** The following rules can be applied when troubleshooting NCU failures.

cu_lckb_par_err cu_ram_par_err	These error conditions are totally isolated to the CU and will not involve any other component of the system.
cu_trp_harderr	This error has been disabled by scan. Check to verify the scan ring of the CU, by executing spu400 subtest 611. If this passes then there is some software that is not initializing the board correctly.
cu_wr_dat_par_err	This error is on bad write data from the XS1O bits <23..0>. If this error is seen then there should also be a cu_ndat0_harderr. IF there is a cu_ndat0_harderr, check to see if there are any mb_iso_data_perr. IF there are, and there is bad data in the xbar and the memory board, the problem may involve the port identified by the MB. All receivers (in this case the CU is a receiver) could be detecting the error. This indicates a potential connectivity problem. Run cpu4332 from a proccessor that is not the one indicated by the mb_iso_data_perr. This will verify that there is basic connectivity at speed and spu4000 subtest 810 will problably not find anything. If this passes then run the test again from the processor that mb_iso_data_perr has identified. IF this passes, the problem could still be connectivity. It would be time for the meter. Check the value of the terminations at the CPU backplane for the three lower byte of the write data.
cu_ndat1_harderr	Check to see if there are any mb_ise_data_perr. IF there are, and there is bad data in the xbar and the memory board, the problem may be comming from the port which the MB has identified. All receivers (the CU is a receiver) could be detecting the error. Indicates a potential connectivity problem. Run cpu4332 from a proccessor that is not the one indicated by the mb_ise_data_perr. This will verify that there is basic connectivity at speed and spu4000 subtest 810 will problably not find anything. If this passes then run the test again from the processor that mb_ise_data_perr has identified. IF this passes, the problem could still be connectivity. It would be time for the meter. Check the value of the terminations at the CPU backplane for all the write data bits.

cu_ndat0_harderr

If this error is received and there is no cu_wr_dat_par_ err THEN the problem is likely to be on the most signif- icant byte of the data. (bits <31..24>). Use the same rules as above. Check to see if there are any mb_iso_data_perr. IF there are, and there is bad data in the xbar and the memory board, the problem may be comming from the port which the MB has identified.

1) Run cpu4332 from a proccessor that is not the one indicated by the mb_iso_data_perr. This will verify that there is basic connectivity to the CU at speed and spu4000 subtest 810 will problably not find anything.

2) IF this passes then run the test again from the processor that mb_iso_data_perr has identified.

3) IF this passes, the problem could still be connectivity. It would be time for the meter. Check the value of the terminations at the CPU backplane for the most significant byte of the write data.

4) IF cpu4332 fails at 1) then this failure would have to be diagnosed separately.

5) IF the failure was another cu_ndat0_harderr then the connectivity between the CU and the XBAR should be examined on bits <31..24> from the xbar at the I/O backplane.

6) IF the connectivity is found to be good in 5), then switch the XS1 even and odd boards.

7) IF the problem stays on the same side then pick the NCU or the Augat to replace.

NOTE The ndat gate arrays do not follow conventional naming conventions, as ndat1 is even and ndat0 is odd.

cu_addr_par_err_3
cu_addr_par_err_2
cu_addr_par_err_1
cu_addr_par_err_0

The Rules for these will be the same. The difference is in the actual bits involved with detecting the error. The same rules apply to the address that is applied to the data. That is the cpu4332 test. If the data in the xbar does not match the data in the board then the problem is the connectivity at the CU - XBAR interface. If there is no mb_iso_addr_perr then the processor is

probably the source of the problem. If the diagnostics do run without an error the next thing would be to meter the bits on the byte that was flagged as the error. (_0, _1, _2_3). If these were found to be normal impedance (52-54 ohms) then the next step would be to switch the XS0 even and odd boards and see if the problem moved sides.

3.1.2 cu4000 - Tests the CU portion of the NCU - NDAT, NADR, COMregisters, and 1 Mhz osc via scan.

1	100	Communication Register Parity Error Generator
2	105	Communication Register Lock Bit Test
2	110	Communication Register Pattern Test
2	115	CU Control Space Test
3	120	Communication Register Functionality Test
3	200	CU ASAP Logic Testing
3	210	CU RDCMR/WRCMR Testing
3	220	CU TOC and ITC Functionality Testing
3	230	CU Deadlock and Firmware Traps Testing
3	300	CU Interrupt Test

3.1.3 mem4000 (st350-370, 380-385) - Tests various fucntions between the NMB, NIA, and NCU.

4	350	Memory testing via NXP operations: read-modify-write
4	355	Memory testing via NXP operations: TAC instruction
4	360	Memory testing via NXP operations: TAS instruction
4	365	Memory testing via NXP operations: SCRUB operation
4	366	Memory testing via NXP operations: MB LOG Ring
4	370	Memory testing via NXP operations: page addressing
4	380	Memory testing via Memory Test Logic: address patterns
4	385	Memory testing via Memory Test Logic: various patterns

(This page intentionally blank)

Section Three

C3800 Series

XBAR Interface

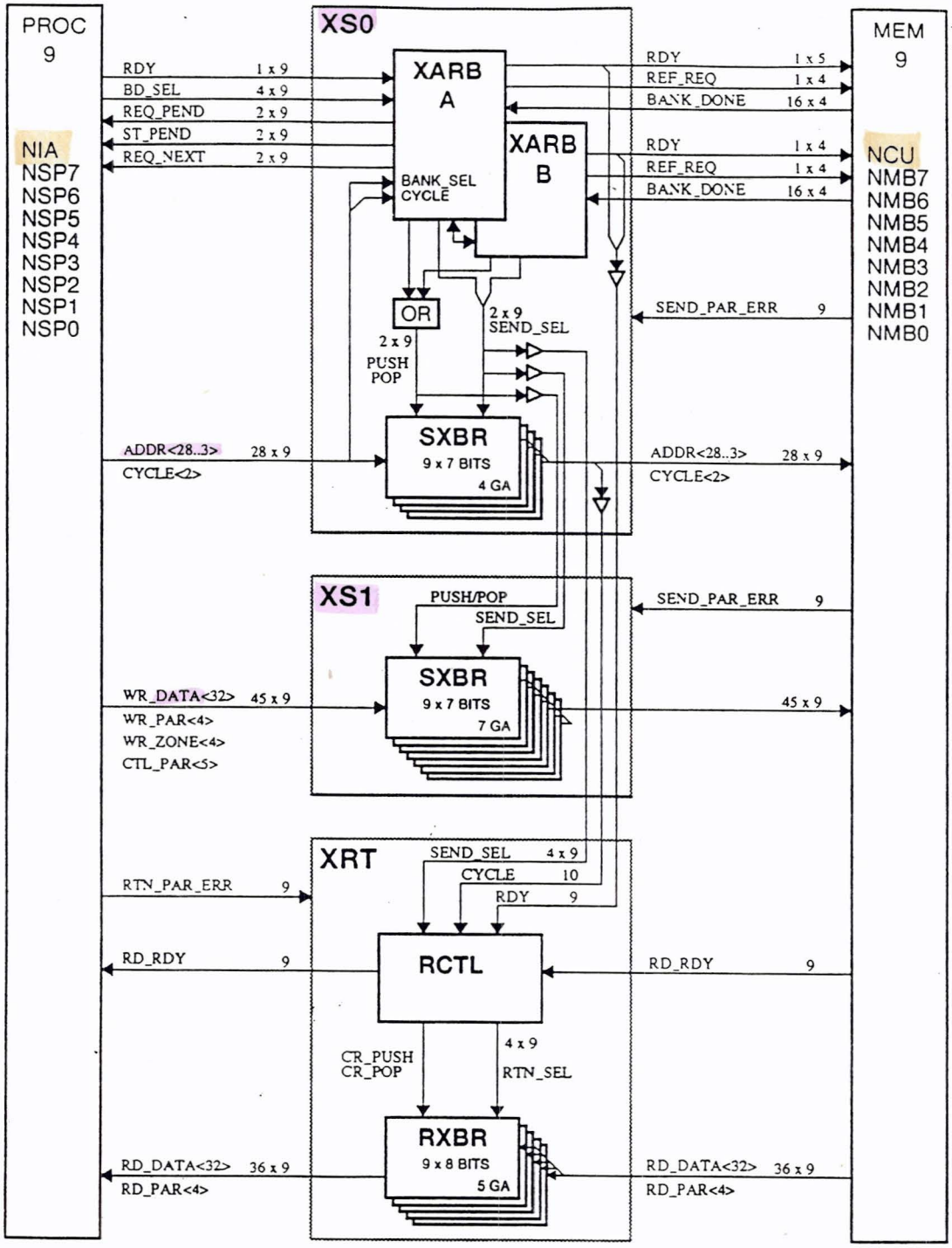


Figure 3-1

Crossbar Block Diagram

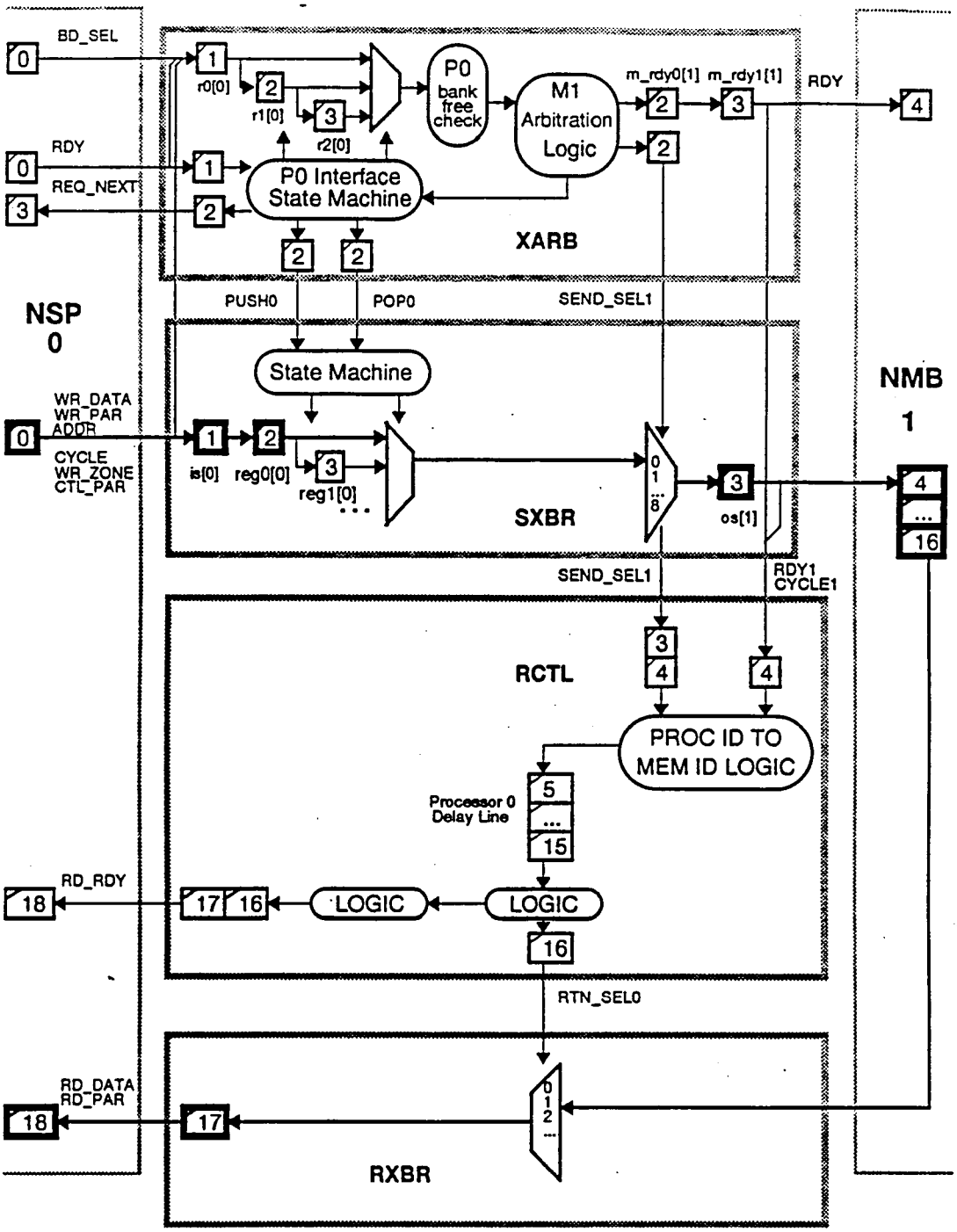


Figure 3-2
Crossbar Data Path to NMB

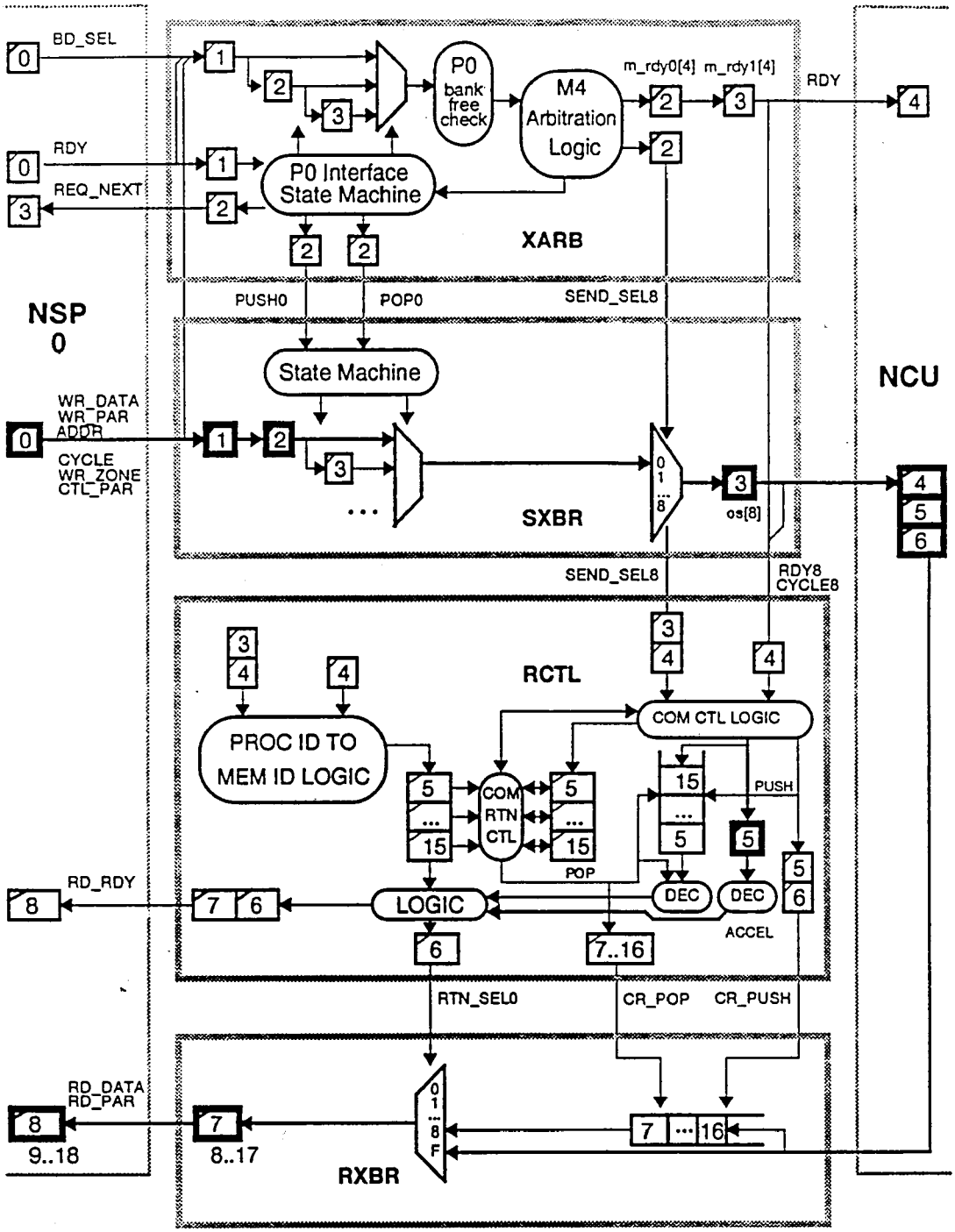
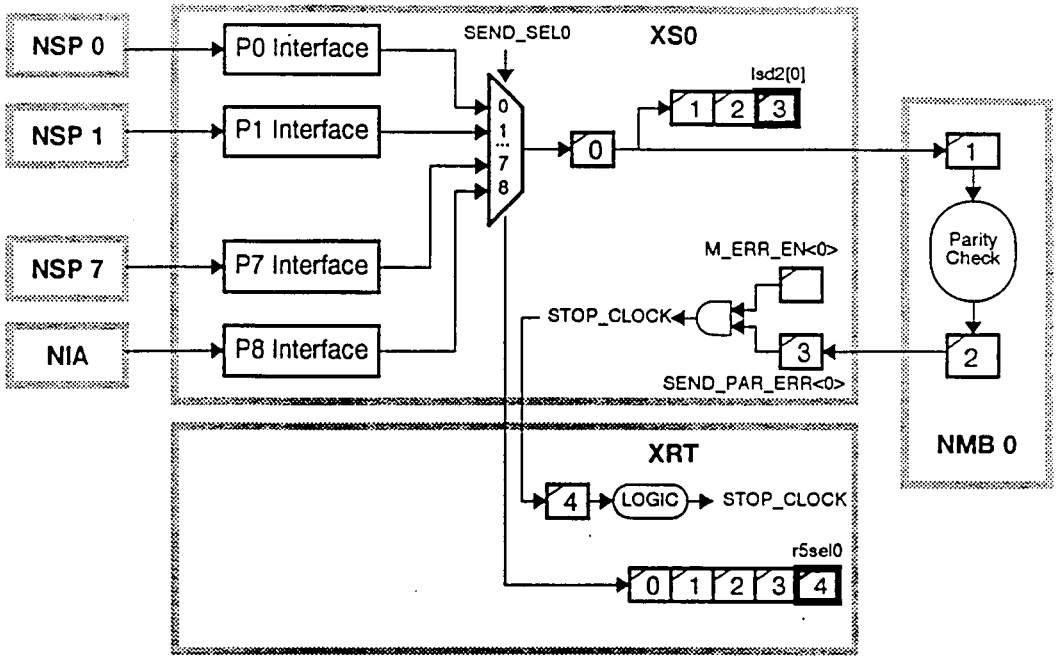


Figure 3-3
Crossbar Data Path to NCU



Crossbar Register	Corresponding NMB register	
xs1e:wr_data_lsd2[*m*]	mb*m*:ise_sys.rwre_data	
xs1e:wr_par_lsd2[*m*]	mb*m*:ise_sys.rwre_par	
xs0e:addr_lsd2[*m*]	mb*m*:bc*b*_sys_addr	*b* = 4 to 7 for even
	mb*m*:ise_sys.addr	(bits 28..22)
xs0e:cycle_lsd2[*m*]	mb*m*:bc[*b*].sys.is_cycle	*b* = 4 to 7 for even
xs1e:wr_zone_lsd2[*m*]	mb*m*:ise_sys.re_zone	
xs1e:ctl_par_lsd2[*m*]	mb*m*:bc[*b*].sys.ris_ctl_par	(bits 3..0)
	mb*m*:ise_sys.rctl_par	(bit 4)

Figure 3-4
Send Parity Error Capture

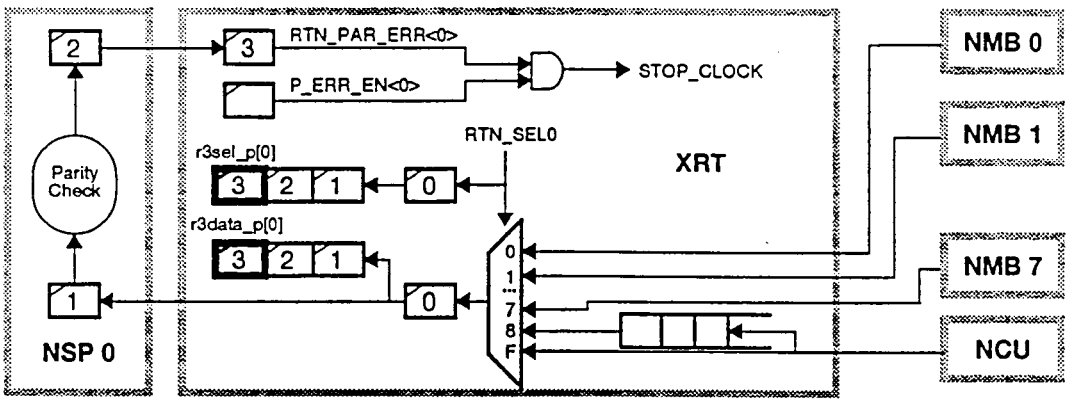


Figure 3-5
Return Parity Error Capture

spu4000 Scan based testing. Also checks connectivity of xbar boards.

mem4000 st100 - Scan-Based XS0 Standalone Testing.
st101 - Scan-Based XRT Standalone Testing.
st102 - Scan-Based XBAR Overrun Register Testing.
st103 - Scan-Based XBAR Processor Enable Testing.
st104 - Scan-Based XBAR PCM Check Testing.
st304 - Scan-Based XBAR Burst Mode Testing.

Board Scan Test - By using the clock and get commands, the scan ring of each board can be tested by getting the "signature". The following is an example of getting the signature information from the xs1e board.

```
spu> clock 1 xs1e
spu> get xs1e:*sig*
    xs1e:sxbr[0].signature = 16#0
    xs1e:sxbr[1].signature = 16#0
    xs1e:sxbr[2].signature = 16#0
    xs1e:sxbr[3].signature = 16#0
    xs1e:sxbr[4].signature = 16#1a9
    xs1e:sxbr[5].signature = 16#1a9
    xs1e:sxbr[6].signature = 16#1a9
spu>
```

All signature returns should equal 16#1a9. Returns other than this indicate a failure on the board. In the above example, the xs1e board is defective.

xbinteg Tests integrity of XBAR send_select lines. To initiate test, at the spu prompt, enter xbinteg e or o (e = even side; o = odd side). **Note: this script is no longer supported and should be used with the knowledge that false errors may be reported.** With the release of Systems Diagnostics 3.4, xbinteg will no longer be available. The following is an example of how this script is used.

```
xbinteg e

Starting send_sel and rdy/cycle path test for xs1e and xrte

TESTING - XBAR CONNECTIVITY TESTER
 TESTING - SENDSEL = 0

ERROR:****SXBR[3] PPUT[0] MEMPORT[0] failed test.
**********DATA: 11 EXPECTED: 10

ERROR:****SXBR[3] PPUT[0] MEMPORT[1] failed test.
**********DATA: 11 EXPECTED: 10
```

sys_con Tests connectivity from xs0, xs1 and xrt to ncu and nia; can also check to memory boards and processors. **This script is now a part of spu4000 and is no longer supported and should be used with the knowledge that false errors may be reported. With the release of Systems Diagnostics 3.4, sys_con will no longer be available.**

sst Can be used to verify a bad board.

2.0 Connectivity Problems

(1) All connectors to logic boards interact with the crossbar to some extent, except for the NVP augats.

(2) Connection problems usually show up as a bit stuck high or low. Whenever connection problems are suspected run spu4000.

(3) If the problem seems to be a send_select, xbinteg will help to isolate the it.

(4) The power to bay 4 should be physically turned off when checking the resistance on the xbar.

(5) For additional information concerning connectivity problems, see Section Six of this manual (Connectivity Troubleshooting.

(This page intentionally blank)

Section Four

C3800 Series

NIA Troubleshooting

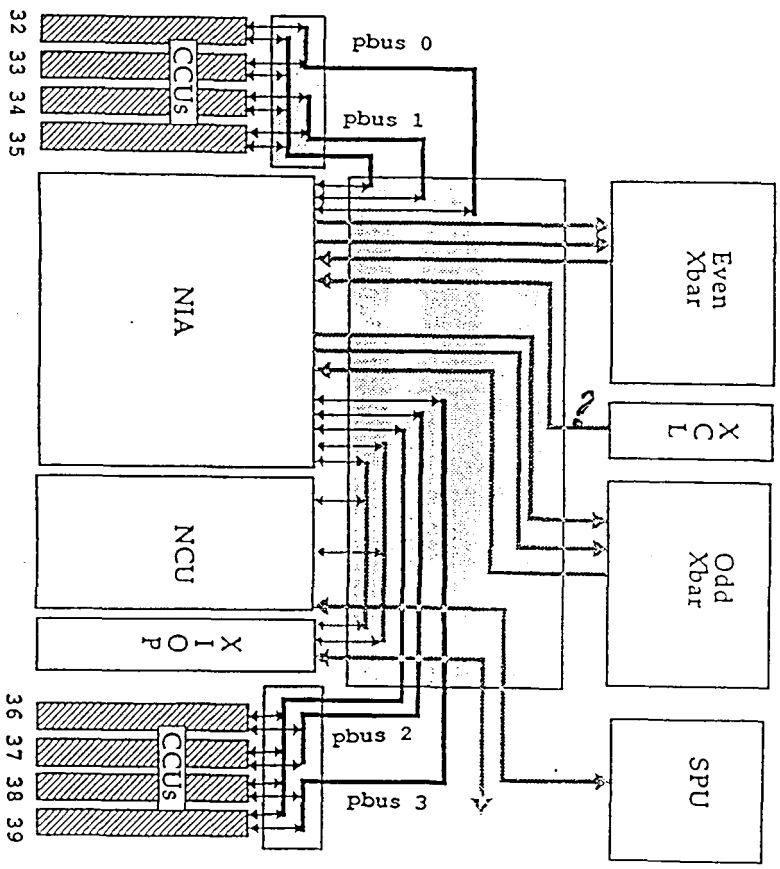


Figure 4-1

PBUS Interfaces

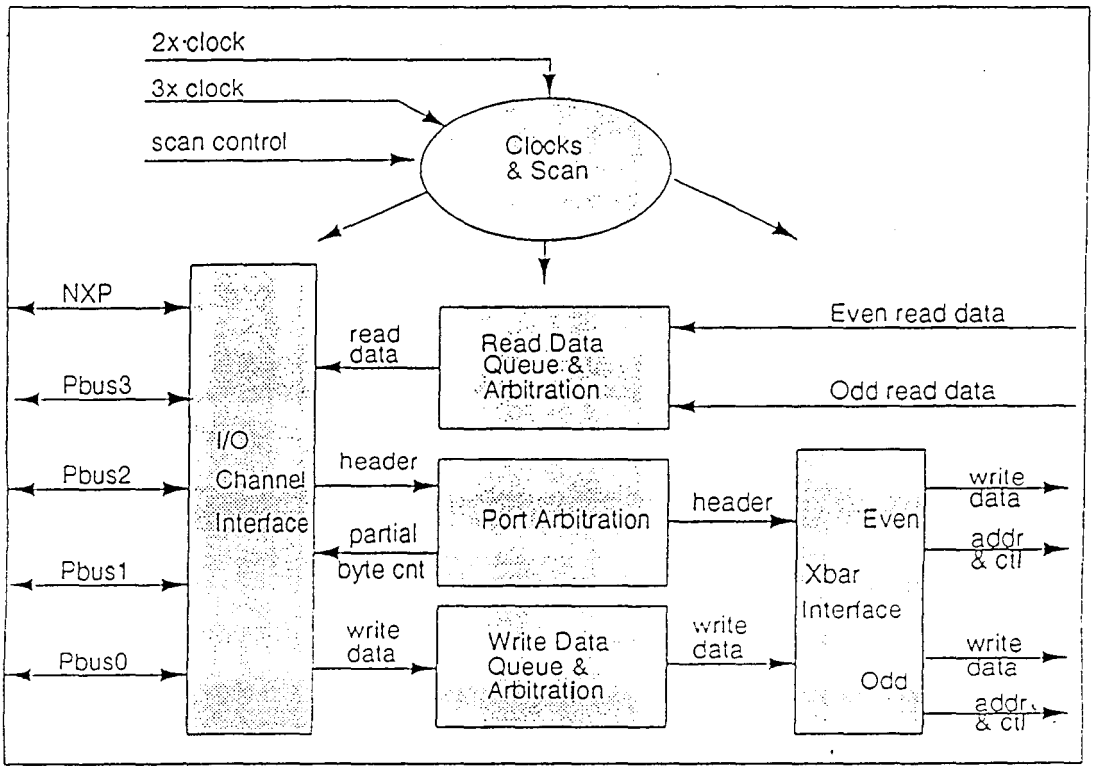


Figure 4-2

Major NIA Subsystems

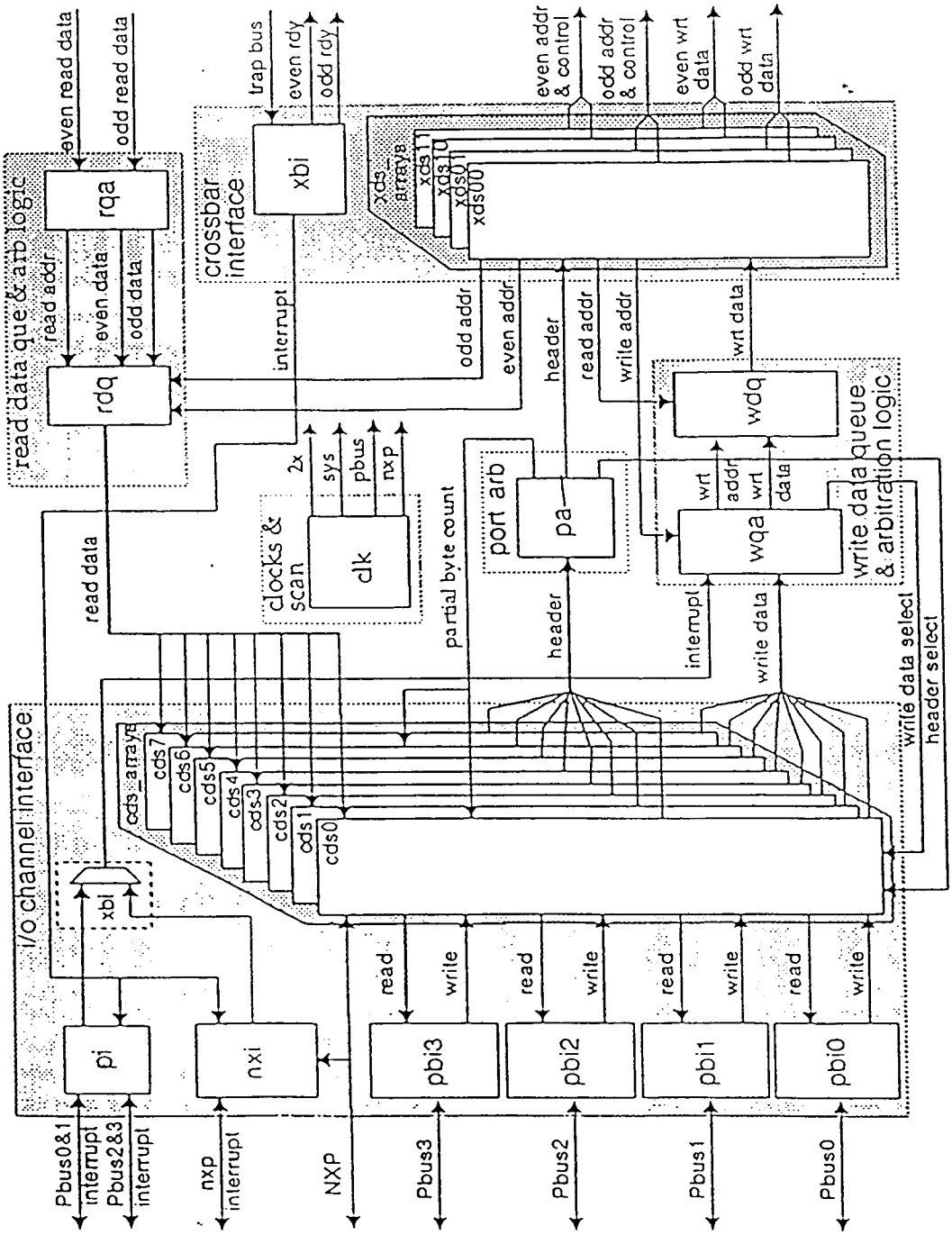


Figure 4-3

NIA Detailed Block Diagram

spu4000

Subtests 611-615, 710-713, 810
Does not test Pbus interfaces.

mem4000

Subtest 350. Read, modify, and write operations via NXP transfers - 4 secs/brd.
Subtest 355. TAC operations via NXP transfers. - 3 secs/per brd.
Subtest 365. SCRUB operations via NXP transfers. - 2 secs/per brd.
Subtest 370. Page addressing via NXP transfers. - 2 secs/per brd.
Subtest 375. Data retention via NXP transfers. - 24 secs/per brd.
Subtest 380. Addressing pattern via MTL transfers. - 7 secs/per brd.
Subtest 385. - Data retention via MTL transfers.

sst

All register components on the NIA are scannable.
- Little or no coverage on the self-timed RAMS (STRAM)

pb_walk

cable
Shards - yes
open - no

Basic NIA functional test for PBUS. Exercises the loopback path on the NIA from ECL read data to the TTL PBUS back to ECL write data. Also exercises the PBUS state machines and monitors reads of the memory base pointer on the NCU. Testing does not go off of the board on the PBUS end.

There are four subtests: loopback_test, pbi_test_NOP, pbi_test_WRITE, and mbp_test. Each of the subtests is run on all listed NIAs before the next subtest is run. Each of the first three subtests are run on all of the listed PBUS interfaces simultaneously. The mbp_test is run individually for each listed NIA and each listed PBUS. By default, all four interfaces of all installed NIAs are tested. Otherwise, list only the NIAs and/or interfaces to be tested. (See man page)

```
pb_walk -e
```

ccu_con

Connectivity checker for the PBUS and the system interrupt bus; between a CCU and an NIA. The test supports VIOP, IDC, ITC, TLI, and HPI. CCUs are selected by listing either the CCU numbers or NIA numbers. Listing an NIA essentially lists all of the 4 or 8 CCUs on that NIA. Then any boards installed in the listed CCU slots are tested. Testing of empty slots is not attempted.

There are three basic subtests: ccu_to_nia connectivity, ccu_to_memory functionality, and ccu_to_ncu_to_ccu interrupt connectivity and functionality. The first subtest is run on all of the selected CCUs, then the second is run on all, then the third. There is no way to run just one of the subtests, except that you can kill the program before it finishes.

For 88k-based CCUs, ccu_con uses the scan-based peeks and pokes of the 88K address space in order to initialize the CCU and download a rudimentary microcode for the PIGA. This microcode is then used to transfer data; first as a bit-by-bit connectivity test, then as a block transfer of data. For the VIOP, ccu_con also uses scan-based peeks and pokes, but the connectivity test is implemented by direct scan manipulation of the VIOP cache-to-PBUS interface. Also, the VIOP self-test is run in lieu of the PBUS block transfers. Tests Pbus data and parity signal connectivity between a CCU and the NIA.

```
ccu_con -e 32
```

nxp.newcon Scan script to test the connectivity of the NXP interface between the NIA and the NCU.

io5000.t Test viop.
Pbus Data Interface (-s 100-200).
Pbus Interrupt Interface (-s 400).

rslog This script reads and clears the soft errror log on the NIA.

Parity error bits have meaning only when "wrt", "pe", or "hdr pe" bits are set. Identifies ccu reporting error.

loadccu autoconf/loadccu failures

To help isolate loadccu failures, perform the following steps:

1) sysreset -l 2
   - viop: next to bottom LED should come on.
   - idc: bottom LED should come on.

2) `mminit -p cafebabecafebabe`

3) pqutil -I

4) sendint fe
   - viop: top two LEDs should come on.
   - idc: top LED flashing; next to top and bottom LEDs on.

5) loadccu -32 /mnt/os/idc (for idc boards)
loadccu -35 /mnt/os/viop (for viop boards)

- all ccu boards: top LED flashing; next-to-top LED on

autoconf (probe and attach) failures

1) Check ioconfig file, cabling, backplane pins, etc.

2) If you get past loadccu, things pretty much work between ccu and memory.

dis displays the stage of NIA data paths
(Not very helpful at this time)

rdq display the contents of the NIA's Read Data Queue

The rdq command performs a scan based read of the NIA's Read Data Queue (RDQ). One or more entries can be read with each invocation. The RDQ address, data, data parity, and error flags are displayed. The rdq command checks the data and error flag bits for odd parity and prints a " **parity error** " message at the end of each displayed entry that does not contain odd parity.

rdq -r 0-f0 Displays the RDQ contents starting at address 0 through f0.

rdq pbi2 40-ff Displays the RDQ contents for Pbus channel 2 starting at address 40 through ff.

rdq nxi toc Displays the two TOC entries for the NXP channel.

```
spu> rdq -r 0-f0

000 00000000 00000000 ff 01
001 00000000 00000000 ff 01
002 00000000 00000000 ff 01
.
.
0ee 00000000 00000000 ff 01
0ef 00000000 00000000 ff 01
0f0 00000000 00000000 ff 01
```

```
spu> rdq pbi0 40-ff

040 00000000 00000000 ff 01
041 00000000 00000000 ff 01
042 00000000 00000000 ff 01
.
.
.
0fd 00000000 00000000 ff 01
0fe 00000000 00000000 ff 01
0ff 00000000 00000000 ff 01


spu> rdq nxi toc
708 00000000 00000000 ff 01
709 00000000 00000000 ff 01
```

wdq display the contents of the NIA's WDQ Data Queue

The wdq command performs a scan based read of the NIA's Write Data Queue (WDQ). One or more entries can be read with each invocation. The WDQ address, data, data parity, and error flags are displayed. The wdq command checks the data and error flag bits for odd parity and prints a " **parity error** " message at the end of each displayed entry that does not contain odd parity.

wdq -r 0-f0 Displays the WDQ contents starting at address 0 through f0.

wdq pbi2 40-ff Displays the WDQ contents for Pbus channel 2 starting at address 40 through ff.

wdq int Displays the reserved interrupt entry (address 700). The reserved interrupt location contains the most recent interrupt vector processed by the NIA to the NCU.

nia_hrdlog basic nia hard error script

Very basic hard error script. It checks on the NIA to see if errors are present and doesn't really give info for the failures. All fields should have a 0, if any fields come up 1 there is a hard error present there.

Very basic hard error script. It checks on the NIA to see if errors are present and doesn't really give info for the failures. All fields should have a 0, if any fields come up 1 there is a hard error present there.

spu> nia_hrdlog (See page 4-3)

CDS ARRAYS	7	6	5	4	3	2	1	0
Par errors	0	0	0	0	0	0	0	0

	parity errors			
XDS ARRAYS	addr	cyc	dat0	dat1
XDS 00	0	0	0	0
XDS 01	0	0	0	0
XDS 10	0	0	0	0
XDS 11	0	0	0	0

RQA_RD_PAR_ERR EVEN 0 RQA_RDQ_WRT_PE 0

RQA_RD_PAR_ERR ODD 0 WQA_WDQ_WRT_PE 0

NXI_HARD_ERROR 0 PI_HARD_ERROR 0

IA_XC_HARD_ERROR 0 CK_HARD_ERROR 0

2.0 NIA Extractor Rules

```
# I - should be internal
# F - should be internal, but might fall through
#       from external
# X - could be external
# There are also some internals which might fall through to other internals,but those cases are handled by
#hard_logger extractor rules.The notation 'if ( extractor )' is true when the extractor reports an error.
# The notation 'extractor.word' refers to a value passed by the extractor in the event report.


ia_xds_hdr_pe I XDS arrays detected a header
                parity error.
        : reject NIA


ia_xds_wde_pe I XDS arrays detected even side
                write data parity error on data from WDQ.
        : reject NIA


ia_xds_wdo_pe I XDS arrays detected odd side
                write data parity error on data from WDQ.
        : reject NIA


ia_cds_rd_pe F CDS arrays detected read
                data parity error on data from RDQ.
        : if ( ia_rdq_wrt_pe )
        : { ignore this error }
        : else
        :               { reject NIA }


ia_rdq_wrt_pe F Parity error detected on
                write data during write access to RDQ.
        : if ( ia_rd_par_err_ *)
        : { ignore this error }
        :               { reject NIA }


ia_wdq_wrt_pe  I Parity error detected on write
                data during write access to WDQ.
        : reject NIA
```

ia_pi_hard_error

X Pbus interrupt state machine hard error. CCU unexpectedly dropped its interrupt request.

```
: if ( Ccu not installed ) { reject NIA }
: else
: {
:               /* start at 50% connection, 25%
                ccu, 25% nia */
: run sst on nia
: run ccu_con on Ccu
: }
```

ia_nxi_hard_error

X NXI interrupt state machine hard error. XIOP unexpectedly dropped its interrupt request.

```
: Xiop = ia_nxi_hard_error.xiop
: if ( Xiop > 1 ) { reject NIA } /
                * bad error report */
:               /* ia8 xiop0 is SPU */
: else
: {
:               /* start at 50% connection, 25%
                xiop, 25% nia */
: run sst on nia
: run nxp.newcon on Xiop
: }
```

ia_wdq_fflag_pe

I WDQ flush flag parity error.

```
: reject NIA
```

ia_rdq_flag_pe

I RDQ read error flag parity error.

```
: reject NIA
```

ia_rd_par_err_e

X Parity error on read data from even side crossbar.

```
: Port = ia_rd_par_err_e.port
: if ( any other errors reported by
                even side xbar )
: { ignore this error }
 : run xbar_err
: run spu4000 for the NIA
: }
```

ia_rd_par_err_o

X Parity error on read data from odd side crossbar.

```
: Just like even side.
```

(Intentionally left blank)

Section Five

C3800 Series

Hard_logger Interpretation

1.0 hard_logger initiated With no Hard Errors Detected

Example One

There have been cases where the hard_logger will be initiated, on aC3800 and no hard errors will be detected. The event will appear as below, the output of hard_err1 and hard_err2 will both be zero, indicating no harderr lines were active.

```
+++>
<Tue Jan 19 01:07:18 1993>   /diag/bin/hard_logger.exc:../hard.c:171
HardLog Start (DiagER368): Hard Error Register Contents

detected_harderrors[0] =  00000000     detected_harderrors[1] =  00000000
enabled_harderrors[0] =   00005fff     enabled_harderrors[1] =   0000007f
****

+++>
<Tue Jan 19 01:07:18 1993>   /diag/bin/hard_logger.exc:../hard.c:184
SW Error (DiagER338): No Valid Hard Errors Detected

detected_harderrors[0] =  00000000     detected_harderrors[1] =  00000000
enabled_harderrors[0] =   00005fff     enabled_harderrors[1] =   0000007f
****

+++>
<Tue Jan 19 01:07:18 1993>   /diag/bin/hard_logger.exc:../hard.c:187
HardLog End (DiagER483): Hard_logger completion
```

This can happen intermittently with all systems, so the first occurrence of this should be ignored. But, if the crashes persist, the following actions are recommended:

1) Execute spu4000

2) Check continuity, with meter, the XC.HARD ERROR nets on each board.

   These nets are located at segment 5 pin 401 on all of the "big" boards. The termination point for these nets are primarily in section 2 of the XCL. On the Xbar send (XS0 & XS1) the net is located in section 6 pin 401. For the Xreturn boards they are on section 5, pin 401. It is possible that a connectivity failure could initiate the process and then be missed. It is recommended that these nets be checked point-to-point.

3) Replace SWIP.

4) Replace SWIP/NCU cable

5) Replace NCU

6) Replace XCL

7) Disable hard_logger by editing the file /mnt/os/boot and remove -h option from /diag/bin/errintd -h -s &. This will likely result in a system hang on the next occurrence, allowing manual register dump and interrogation.

This problem should not be confused with the instance where anything is actually set in the hardlog registers and the hard_logger terminates with no hard errors detected. This event indicates a failure and unsuccessful interrogation of the registers.

Example Two

```
[SPU  @20:48:12] <Mon May 17 1993>

+++>
<Mon May 17 21:41:35 1993>   /diag/bin/hard_logger.exc(1356):../hard.c:279
HardLog Start (DiagER368): Hard Error Register Contents

detected_harderrors[0] =  00000020     detected_harderrors[1] =  00000000
enabled_harderrors[0] =   219c5fff     enabled_harderrors[1] =   0000007f
****
HARD LOGGER
System clocks halted

Machine  Device       Part      Serial   Ring Wire Assm   DC    AC
 Slot     Type       Number    Number   Rev  Rev  Rev    Rev   Rev
mb0       mb       416-001246  1061930   0    A    B     36     0
sp0       sp       416-006247  1047830   3    F    M     39     0
vp0       vp       417-002248  1035657   0    C    A      6     0

.
.

xs1o      xs1      426-001242  1004980   0    A    A     13     0
xrto      xrt      425-001239  1004895   0    A    A     15     0

System was running UPPER clock

System temperature dump:

.
.


Total extractors to process = 2
^MProcessing extractor 1
+++>
<Mon May 17 21:41:53 1993>   /diag/bin/hard_logger.exc(1356):../hard.c:1039
SW Error (DiagER340): Extractor Failed to Detect Harderror
```

```
Extractor Name:    mb_bcga_iso_low_addr
Port:  1
Board:    MB_TYPE
****
^MProcessing extractor 2


+++>
<Mon May 17 21:41:54 1993>    /diag/bin/hard_logger.exc(1356):../hard.c:1039
SW Error (DiagER340): Extractor Failed to Detect Harderror

Extractor Name:    mb_bcga_ise_low_addr
Port:  1
Board:    MB_TYPE
****

+++>
<Mon May 17 21:41:54 1993>    /diag/bin/hard_logger.exc(1356):../hard.c:468
SW Error (DiagER341): Hard Logger Failed to Detect Harderrors

****

+++>
<Mon May 17 21:41:54 1993>    /diag/bin/hard_logger.exc(1356):../hard.c:306
HardLog End (DiagER483): Hard_logger completion

****
```

Note(1): The hard error register did have a bit set for nmb 1.

Note(2): As noted, the system clock was set to UPPER. This indicates that the ncu powered down.

At this point, the event log (display_log), /usr/adm/messgaes file and pwr_util should be reviewed for possible entries related to this shutdown. Manual activation of hard_logger (prior to a reboot) may provide additional information. If the system was rebooted prior to your arrival, ask if diaginit was required before the system could be rebooted. This may indicate that a power down did occur.

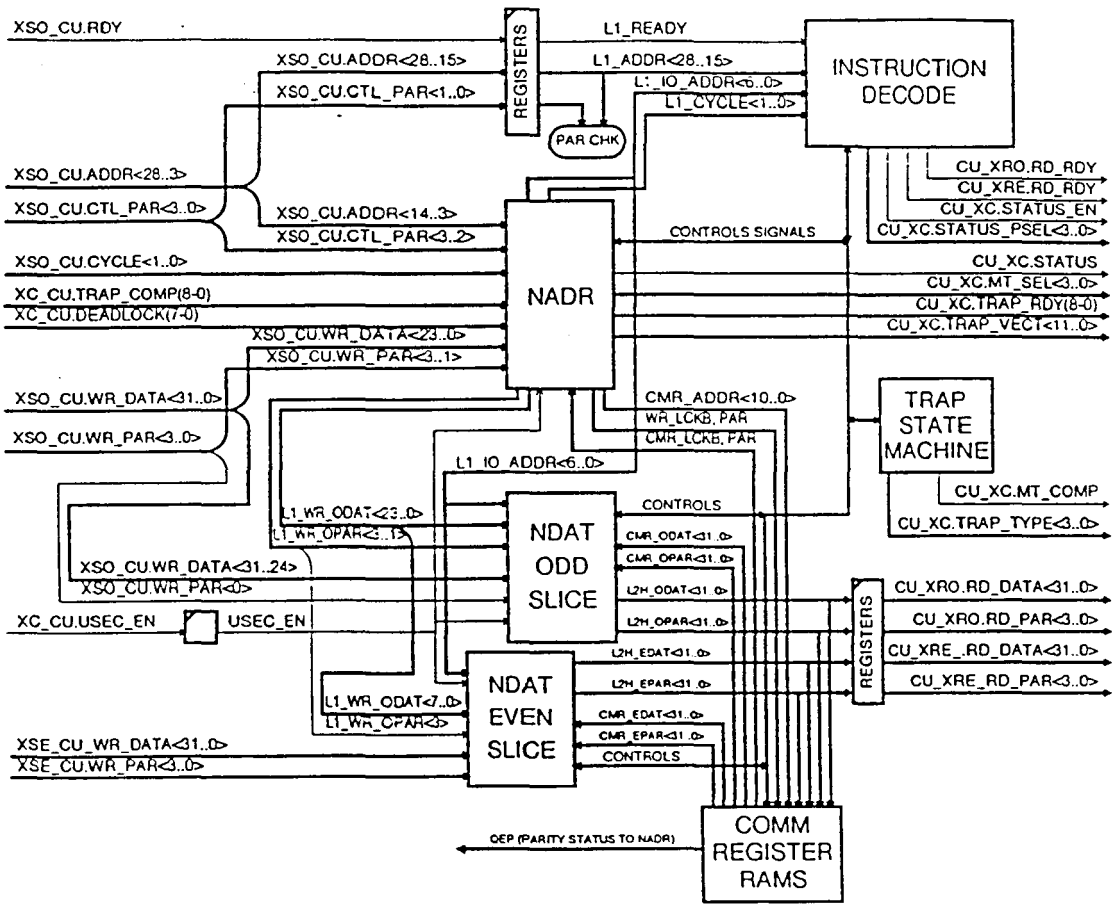


Figure 5-1
CPU Utilities Block Diagram

Example One

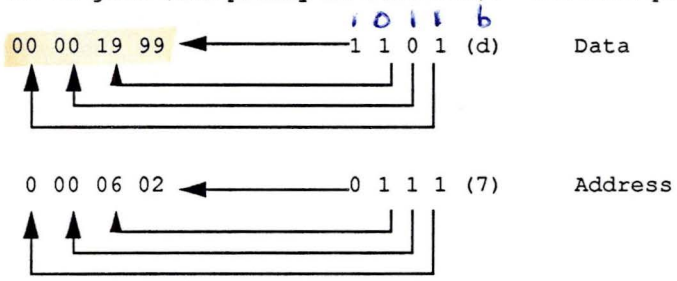
```
-------------------------------------------------------------------
HARDERROR #     BOARD TYPE        PORT/SIDE     EXTRACTOR
-----------     ----------        ---------     ---------
    0            CU_TYPE            N/A          cu_ndat1_harderr
===================================================================
        NCU harderror detected
        ndat[1].ndat_harderr = 1
        ---------------------------------------------------------
        Due to an NDAT design error, the NDAT does not hold it's error state
        correctly.  The hard error has been detected and can be one of
        the following conditions:
        1) Parity error from the XBAR on the signals
                XSE_CU.WR_DAT<31..0>/XSE_CU.WR_PAR<3..0>
        2) Parity error from the NADR on the signals
                L1_WR_ODAT<7..0>/L1_WR_OPAR<3>
        3) Parity error from even comm register rams on signals
                CMR_RD_EDAT<31..0>/CMR_RD_EPAR<3..0>
+++>
<Tue Feb 23 14:49:37 1993>   logmsg(904):cu_ndat1_harderr:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=cu_ndat1_harderr @board_type=cu @port=-1
****

-------------------------------------------------------------------
HARDERROR #     BOARD TYPE        PORT/SIDE     EXTRACTOR
-----------     ----------        ---------     ---------
    1           XS0_TYPE              0          xs0.send_par_err_ncu
===================================================================
 Control parity error detected by NCU.  EVEN xbar notified and
 halted.  Check to make sure that the ODD xbar has also pulled
 a control parity error.  The NCU only checks parity on the ODD
 side, but it notifies the EVEN xbar when it pulls an ODD error.
 If there is no error on the ODD side, then there is a flaw in the
 error circuits on either the XS0E, the XS0O, or the NCU.
-------------------------------------------------------------------
+++>
<Tue Feb 23 14:49:41 1993>   logmsg(930):xs0_send_par_err_ncu:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=xs0_send_par_err_ncu @board_type=xs0 @port=0
****

+++>
<Tue Feb 23 14:49:43 1993>   /diag/bin/hard_logger.exc(852):../hard.c:306
HardLog End (DiagER483): Hard_logger completion

****
```

```
spu> xbar_err

XBAR Error Logger/Identifier Version 0.7    (some testing done)

Halting system
   Checking xrte for errors
Error detected by xrte -- board has halted
   Checking xrto for errors
Error detected by xrto -- board has halted
   Checking xs0e and xs1e for errors
Error detected by xs0e and xs1e -- boards have halted
NCU detected parity error on data from NSP0
  Data  Par in XBAR:      0x00000000  0xf
  addr= 0000602  cycle= 0  wr_zone= 0  ctl_par= 17        -No Parity Check yet-
  Compare with NSP and NCU
   Checking xs0o and xs1o for errors
Error detected by xs0o and xs1o -- boards have halted
NCU detected parity error on data from NSP0
  Data  Par in XBAR:      0x00001999  0xb
  addr= 0000602  cycle= 0  wr_zone= 0  ctl_par= 17        -No Parity Check yet-
  Compare with NSP and NCU
```

Note(1): HARDERROR 0 tells us the data came from one of the xse boards.

Note(2): HARDERROR 1 states that the xbar was notified and halted.

Note(3): The xbar_err script shows the parity on both the data and address to be good (odd parity is utilized). The data parity is as follows:



In the above failure, a short was found on the ncu augat connector.

Example Two

```
-------------------------------------------------------------
HARDERROR #     BOARD TYPE         PORT/SIDE     EXTRACTOR
-----------     ----------         ---------     ---------
    0            CU_TYPE             N/A          cu_addr_par_err.1
=============================================================
        NCU harderror detected
        r_addr_par_err1 = 1
        Register                Value    Board signal name
        ------------------------------------------------------
        xso_l1_addr<18:12>      1c       XSO_CU.ADDR<21..15>
        ctl_par<1>              1*       XSO_CU.CTL_PAR<1>
          * indicates parity error
+++>
<Mon Sep  7 04:30:19 1992>   logmsg:../logmsg.c:56
Hard Error (DiagER349): logmsg: General purpose event
Source: cu_addr_par_err_1
@extractor=cu_addr_par_err_1 @board_type=cu @port=-1
****
-------------------------------------------------------------
HARDERROR #     BOARD TYPE         PORT/SIDE     EXTRACTOR
-----------     ----------         ---------     ---------
    1            CU_TYPE             N/A          cu_addr_par_err.3.2
=============================================================
        NCU harderror detected
        nadr.addr_par_err = 1
        Register                Value    Board signal name
        ------------------------------------------------------
        xso_l1_addr<11:5>       00       XSO_CU.ADDR<14..8>
        ctl_par<2>              1        XSO_CU.CTL_PAR<2>
        xso_l1_addr<4..0>/               XSO_CU.ADDR<7..3>/
          nadr.l1_cycle         0d         XSO_CU.CYCLE<1..0>  <-
        ctl_par<3>              1*       XSO_CU.CTL_PAR<3>
          * indicates parity error
+++>
<Mon Sep  7 04:30:20 1992>   logmsg:../logmsg.c:56
Hard Error (DiagER349): logmsg: General purpose event
Source: cu_addr_par_err_3_2
@extractor=cu_addr_par_err_3_2 @board_type=cu @port=-1
****
-------------------------------------------------------------
HARDERROR #     BOARD TYPE         PORT/SIDE     EXTRACTOR
-----------     ----------         ---------     ---------
    2           XS0_TYPE              1          xs0.send_par_err_ncu
=============================================================
 Control parity error detected by NCU from SP7 in XS0/1O
        NOTE - Address bit mapping is relative to the XBAR.
               addr<28> corresponds to addr<25> in the NCU.
-------------------------------------------------------------
 addr<28:22> <21:15> <14:8> {<7:3> cycle<1:0>}
    ----------------------------------------------
```

data was bad coming into XBar and bad into NCU. => MSP could be b

```
        ***Data was not preserved in SP7***
    ------------------------------------------------------
      17       1c       00       0d              xbar data
       1        1*       1        1*           xbar ctl_par<0:3>
    ------------------------------------------------------
      17       1c       00       0d               ncu data
       1        1*       1        1*           ncu ctl_par<0:3>
    ------------------------------------------------------
    ? indicates data was not staged
    * indicates parity error
-----------------------------------------------------------------
 The xbar data is saved in lookaside registers in the xs1 and xs0.
 This is the data that the xbar sent to the ncu for this request.
 Any difference between the xbar data and the ncu data may
 indicate a connectivity problem between the xbar and the ncu.
-----------------------------------------------------------------
 Scan fields:            ncu                          xbar
        --------------------------      -------------------------
       00b9c003 cu:xso_l1_addr         00b9c003 xs0o:addr_lsd2[8]
              1 cu:nadr.l1_cycle               1 xs0o:cycle_lsd2[8]
             0f cu:ctl_par                    1f xs1o:ctl_par_lsd2[8]
-----------------------------------------------------------------
 xs0o:m_err_en<8> = 1
 xs0o:send_par_err<8> = 1
+++>
<Mon Sep  7 04:30:24 1992>   logmsg:../logmsg.c:56
Hard Error (DiagER349): logmsg: General purpose event
Source: xs0_send_par_err_ncu
@extractor=xs0_send_par_err_ncu @board_type=xs0 @port=1
****
```

Note(1): HARDERROR 0 Extractor reported a "cu_addr_par.1" with address data from one of the xso boards.

Note(2): HARDERROR 1 Extractor reported a "cu_addr_par.3.2" with address data from one of the xso boards.

Note(3): HARDERROR 2 reports that the address data orginated on SP7 and the xs0o passed this on to the ncu. It also tells us that the problem is with address bits <21..15> and <6..3> or parity bits 1 and 3. The data received by the ncu is the same as the data in the xbar; which indicates the problem to be on sp7 or connectivity between the xbar and sp7.

The xbar_err script would have help isolate this problem. cpu4332 and spu4000 would have help isolate this problem. If cpu4332 or spu4000 can not detect the problem, sp7 should be swapped with another scalar board. If it fails again after the sp7 is swapped, then the xs0o board would be suspect. You would want to check the point-to-point resistance of the bits noted in Note(3) above.

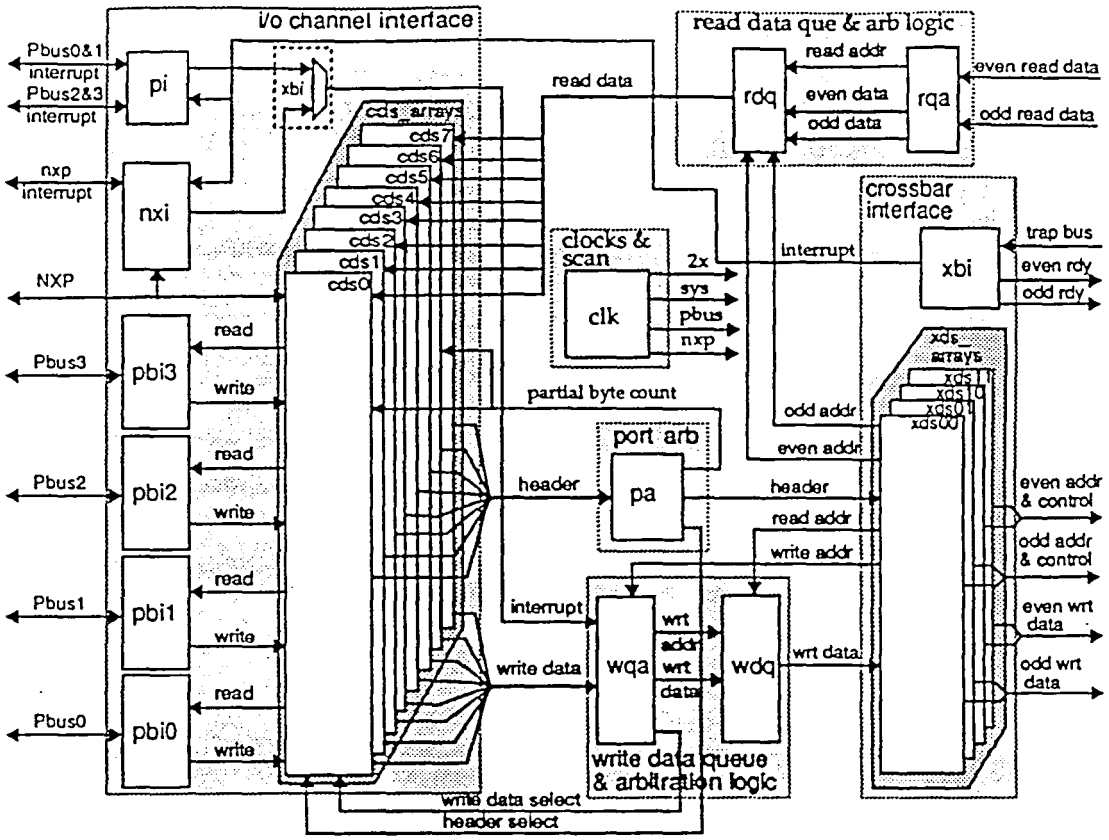


Figure 5-2

NIA Block Diagram

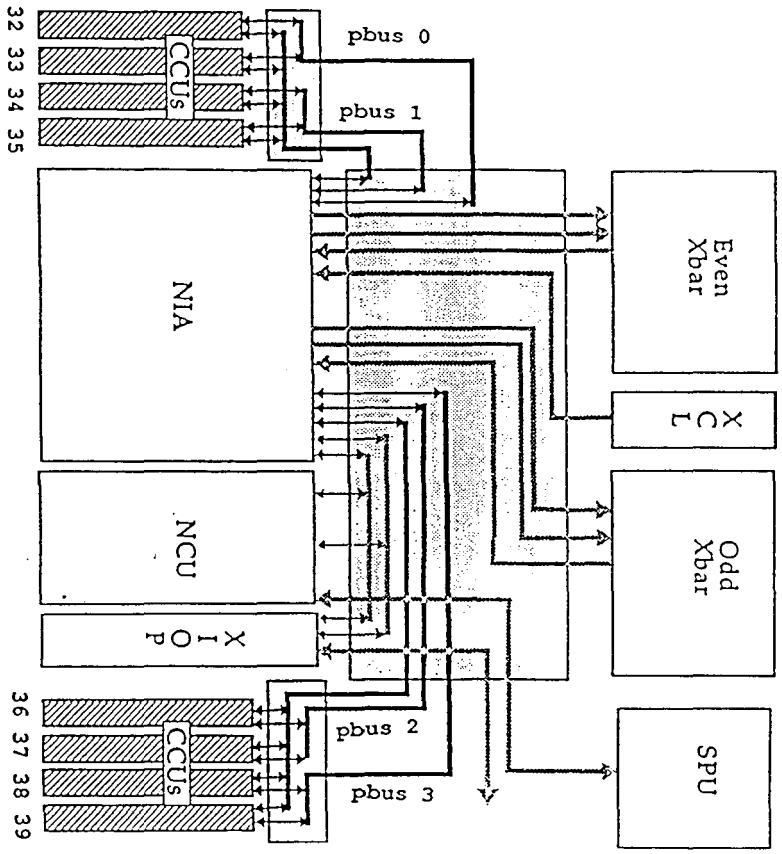


Figure 5-3
PBUS Interfaces

Example One

```
mm_sniff: aborting due to signal: SIGBUS          **SIGBUS = bus error**
mm_sniff: Fatal error. Sniffing terminated

   console: Aborting: window bus error
boot: 1526 Abort - core dumped

Cleaning up SPU processes...
errintd: Logging terminated.
* * * * * Last command returned status 1 * * * * *
```

SPU CONSOLE OUTPUT

```
swip_bus_error(): BUS ERROR trapped
        BSRU(0x10): NcuErr
        BSRL(0x10): No errors
        NCU: NcuErrLog(0xff000020)  ScnErrLog(0xf000c25) Laddr(0x1f442004)



spu>
spu> hard_logger

+++>
<Wed May 12 13:53:20 1993>   /diag/bin/hard_logger.exc(394):../hard.c:279
HardLog Start (DiagER368): Hard Error Register Contents

detected_harderrors[0] =  00000000     detected_harderrors[1] =  01000000
enabled_harderrors[0] =   0000503f     enabled_harderrors[1] =   0000007f
****

+++>
<Wed May 12 13:53:20 1993>   /diag/bin/hard_logger.exc(394):../hard.c:302
SW Error (DiagER338): No Valid Hard Errors Detected

detected_harderrors[0] =  00000000     detected_harderrors[1] =  01000000
enabled_harderrors[0] =   0000503f     enabled_harderrors[1] =   0000007f
****

+++>
<Wed May 12 13:53:20 1993>   /diag/bin/hard_logger.exc(394):../hard.c:306
HardLog End (DiagER483): Hard_logger completion

****
```

```
spu> rslog
                              IA8 soft log ring
    |     parity errors (bytes)       |ill|wrt|rd |wrt|hdr|ccu0|ccu1|ccu |
port| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |hdr|pcm|pcm|pe |pe |err |err |num |
----------------------------------------------------------------------------
nxi | 0   0   0   0   0   0   0   1   0   0   0   0   1   0    0    0
pbi0| 1   1   1   1   1   1   1   1   0   0   0   0   0   0    0    1
pbi1| 1   1   1   1   1   1   1   1   0   0   0   0   0   0    0    1
pbi2| 1   1   1   1   1   1   1   1   0   0   0   0   0   0    0    1
pbi3| 1   1   1   1   1   1   1   1   0   0   0   0   0   0    0    1


spu> spu4000
.
.
+++>
<Mon May 17 08:59:33 1993>   /diag/test/spu/spu4000(2634):../ia_cu_test.c:273
Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1
Source Signal: nxp.data(upper)                    Source Port: 8
Sink Signal:   nxp.data(upper)                    Sink Port:   8
Sink Field:    cu:sp_r_xbinreg_uh[8]
Source Signal Bit Position: 1          Sink Signal Bit Position: 30
Expected:                   00000002
Actual:                     40000002
****

+++>
```

Note(1): The ConvexOS crashed with a window bus error.

Note(2): The SPU CONSOLE window message "NcuErrLog 0xff000020" reported a NXP Sequencing Error.

Note(3): By running hard_logger manually, it was found that the nia did pull a soft error. The rslog script confirmed this as a nxi header parity error (nxp bus). The nia section in this document lists the mem4000 tests to run for nxp errors. In this failure, these tests passed.

Note(4): spu4000 detected a short on an nxp.data(upper), bit position 30. The nxp.data nets are bidirectional <63..0> nets between the nia and ncu. As this was the upper portion of the nxp.data <63..32>, you must add 32 to the bit position. In this case the problem was found on the nia augat (nxp.data<62> net).

Example Two

*** /mnt/errlog output ***

```
[CPU01@13:51:12] Accounting suspended

+++>
<Wed May 12 13:53:20 1993>   /diag/bin/hard_logger.exc(394):../hard.c:279
HardLog Start (DiagER368): Hard Error Register Contents

detected_harderrors[0] =  00000000     detected_harderrors[1] =  01000000
enabled_harderrors[0] =   0000503f     enabled_harderrors[1] =   0000007f
****

+++>
<Wed May 12 13:53:20 1993>   /diag/bin/hard_logger.exc(394):../hard.c:302
SW Error (DiagER338): No Valid Hard Errors Detected

detected_harderrors[0] =  00000000     detected_harderrors[1] =  01000000
enabled_harderrors[0] =   0000503f     enabled_harderrors[1] =   0000007f
****

+++>
<Wed May 12 13:53:20 1993>   /diag/bin/hard_logger.exc(394):../hard.c:306
HardLog End (DiagER483): Hard_logger completion

****
```

*** display_log output ***

```
+++>
<Wed May 12 13:53:20 1993>   /diag/bin/hard_logger.exc(394):../hard.c:279
HardLog Start (DiagER368): Hard Error Register Contents

detected_harderrors[0] =  00000000     detected_harderrors[1] =  01000000
enabled_harderrors[0] =   0000503f     enabled_harderrors[1] =   0000007f
****

+++>
<Wed May 12 13:53:20 1993>   /diag/bin/hard_logger.exc(394):../hard.c:302
SW Error (DiagER338): No Valid Hard Errors Detected

detected_harderrors[0] =  00000000     detected_harderrors[1] =  01000000
enabled_harderrors[0] =   0000503f     enabled_harderrors[1] =   0000007f
****

+++>
<Wed May 12 13:53:20 1993>   /diag/bin/hard_logger.exc(394):../hard.c:306
HardLog End (DiagER483): Hard_logger completion

****
```

```
+++>
                    ***    /usr/adm/messages file    ***

May 12 13:52:48 f2t_4 vmunix: swip_bus_error(): BUS ERROR trapped
May 12 13:52:48 f2t_4 vmunix:   BSRU(0x30): NcuErr Rerun
May 12 13:52:48 f2t_4 vmunix:   BSRL(0x0): No errors
May 12 13:52:48 f2t_4 vmunix:  NCU: NcuErrLog(0xff000010)  ScnErrLog(0xf000c25)
Laddr(0x1f442004)
May 12 13:52:58 f2t_4 vmunix: swipPoll(): disabling SWIP NCU interrupts(sys)-10:
May 12 13:52:58 f2t_4 vmunix: swip_bus_error(): BUS ERROR trapped
May 12 13:52:58 f2t_4 vmunix:   BSRU(0x10): NcuErr
May 12 13:52:58 f2t_4 vmunix:   BSRL(0x10): No errors
May 12 13:52:58 f2t_4 vmunix:  NCU: NcuErrLog(0xff000020)  ScnErrLog(0xf000c25)
Laddr(0x18400228)


                    ***     spu4000 results    ***

+++>
<Wed May 12 14:23:55 1993>   /diag/test/spu/spu4000(441):../ia_cu_test.c:273
Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1
Source Signal: nxp.data(upper)                    Source Port: 8
Sink Signal:   nxp.data(lower)                    Sink Port:   8
Sink Field:    cu:sp_r_xbinreg_lh[8]
Source Signal Bit Position: 0          Sink Signal Bit Position: 15
Expected:                   00000000
Actual:                     00008000

@EXPECTED=0x00000000
@ACTUAL=0x00008000
@SRC_PORT=8
@SOURCE="nxp.data(upper)"
@SNK_PORT=8
@SINK="nxp.data(lower)"
@SUBTEST=810


Note(1): The errlog and event_log pointed to the nia.

Note(2): The message file/SPU CONSOLE pointed to the nxp bus (nia-to-ncu).

Note(3): spu4000 detected a short on the nia augat (nxp.data<15>).
```

Example Three

```
mminit: using pattern 0xffffffff ffffffff
mminit: using Service Processor to initialize memory
mminit: using Service Processor to verify memory
mminit: using pattern 0x00000000 00000000
mminit: using Service Processor to initialize memory
mminit: using Service Processor to verify memory
mminit: memory initialization complete
loadccu: Timeout waiting for messages from CCU33
* * * * * Last command returned status 255 * * * * *
spu>


**AT THIS POINT THE FOLLOWING WAS PERFORMED**

spu> sysreset -l 2          **COMPLETED WITH NO PROBLEM**
                            **IDC BOTTOM LED CAME ON**

spu> mminit -p cafebabecafebabe      **COMPLETED WITH NO PROBLEM**
                                     **MEMORY NOT THE PROBLEM**

spu> pqutil -I                       **COMPLETED WITH NO PROBLEM**
                                     **MBS LOADS**

spu> sendint fe
spu> sending interrupt 0xfe done
spu> Last command returned **status 2      **IDC TOP LED NOT FLASHING **
                                           **NEXT TO TOP/BOTTOM NOT ON**
                                           **OPERATION FAILED         **



spu> rslog
                        IA8 soft log ring
    |    parity errors (bytes)      |ill|wrt|rd |wrt|hdr|ccu0|ccu1|ccu |

port| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |hdr|pcm|pcm|pe |pe |err |err |num |

-------------------------------------------------------------------------

nxi | 1   1   1   1   1   1   1   1   0   0   0   0   0   0    0    0
pbi0| 0   0   0   0   0   1   0   0   0   0   0   0   1   0    0    1
pbi1| 1   1   1   1   1   1   1   1   0   0   0   0   0   0    0    1
pbi2| 1   1   1   1   1   1   1   1   0   0   0   0   0   0    0    1
pbi3| 1   1   1   1   1   1   1   1   0   0   0   0   0   0    0    1

spu>


spu> ccu_con 33
```

```
ia8:  32       33 idc   34       35
ia8:  36       37       38       39
Initializing XBAR...
Initializing CU board...
Initializing IA board...
Initializing CCUs...
Initializing MB4,MB5...
Initializing SP4...
Initializing VP4...
Initializing SP5...
Initializing VP5...
checking PBI state
CCU to NIA PATTERN TEST
Testing  idc33 which is on nia8 pbus0
ERROR: pb0_dat_ms expected 0x00000100 actual 0x00000000 mask 0xffffffff
ERROR: PBI0 pattern not correct (ms)
Test FAILED.  End of step.  1 compare errors.
* * * * * Last command returned status 3 * * * * *
spu>
```

AS THESE SIGNALS (NIA-CCU) ARE TTL; 1 = OPEN, 0 = SHORT

```
spu> pb_walk
Testing NIAs:  NONE0  NONE1  NONE2  NONE3  NONE4  NONE5  NONE6  NONE7  nia8
Testing PBUS:  0  1  2  3
Initializing XBAR...
Initializing CU board...
Initializing IA board...
Initializing CCUs...
Initializing MB4,MB5...
Initializing SP4...
Initializing VP4...
Initializing SP5...
Initializing VP5...
Initializing IA board...
checking PBI state
LOOPBACK PATTERN TEST on nia8
ERROR: pb0_dat_ms expected 0x00000100 actual 0x00000000 mask 0xffffffff
while walking ones
Test FAILED.  End of step.  1 compare errors.
* * * * * Last command returned status 3 * * * * *
spu>
```

THIS PROBLEM WAS FOUND TO BE A SHORT ON THE WOVEN CABLE BETWEEN THE NIA (J13
**CABLE) AND CCU BACKPLANE. THE NET WAS BP0_BP0.P_DATA<40> (FOR PBX_DAT_MS   **
**NETS, ADD 32 TO THE BIT LOCATION TO DETERMINE ACTUAL BIT, E.G., IN THIS   **
**CASE, THE SHORT WAS ON MS DATA BIT 8 + 32 = 40.   **

Example Four

The following entry was observed on the CONVEXOS COSOLE window (Note: IA soft errors are not entered in the /mnt/errlog file on the SPU).

IA soft error detected in slot 8 (see event log)

The following is a extract of the dump_soft_log script.

```
**********************************************************************
*                          IA SOFT ERRORS                            *
**********************************************************************

C
C                                          #PBI0  #PBI1  #PBI2  #PBI3  #NXI
U      FIRST FAIL         LAST FAIL       FAILS  FAILS  FAILS  FAILS  FAILS
-  ----------------- ----------------- ------ ------ ------ ------ ------
33 May 18 14:33 1993 May 18 14:35 1993 000094 000000 000000 000000 000000
35 May 18 14:20 1993 May 18 14:32 1993 000000 000002 000000 000000 000000
```

The following was extracted from the event log (display_log)

```
+++>
<Tue May 18 14:33:32 1993>   /diag/bin/errintd(652):../errintd.c:411
Soft Error (DiagER495): errintd: IA soft error detected.

Source: errintd
IA soft error detected, slot=8, error data=0xff02, pbi0, ccu=33

|    parity errors (bytes)      |ill|wrt|rd |wrt|hdr|ccu0|ccu1|ccu |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |hdr|pcm|pcm|pe |pe |err |err |num |
-----------------------------------------------------------------------
  1   1   1   1   1   1   1   1   0   0   0   0   0   0    1    1


@Board=ia8
@Ccu=33
@Port=0
```

The following scripts were run.

```
spu> rslog
                          IA8 soft log ring
    |    parity errors (bytes)      |ill|wrt|rd |wrt|hdr|ccu0|ccu1|ccu |

port| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |hdr|pcm|pcm|pe |pe |err |err |num |

-------------------------------------------------------------------------

nxi | 0   0   0   0   0   0   0   0   0   0   0   0   0   0    0    0
pbi0| 0   0   0   0   0   0   0   0   0   0   0   0   0   0    1    1
pbi1| 0   0   0   0   0   0   0   0   0   0   0   0   0   0    0    1
pbi2| 0   0   0   0   0   0   0   0   0   0   0   0   0   0    0    1
pbi3| 0   0   0   0   0   0   0   0   0   0   0   0   0   0    0    1


spu> pb_walk    (This passed.  Remember, pb_walk will not find an open on
                 on the pbus cables between the nia and ccu backplane)

spu> ccu_con -e 33

ia8:  32       33 idc   34       35
ia8:  36       37       38       39
Initializing XBAR...
Initializing CU board...
Initializing IA board...
Initializing CCUs...
Initializing MB4,MB5...
Initializing SP4...
Initializing VP4...
Initializing SP5...
Initializing VP5...
checking PBI state
CCU to NIA PATTERN TEST
Testing  idc33 which is on nia8 pbus0
ERROR: pb0_dat_ls expected 0x00000001 actual 0x0000001f mask 0xffffffff
ERROR: PBI0 pattern not correct (ls)
ERROR: pb0_dat_ls expected 0x00000002 actual 0x0000001f mask 0xffffffff
ERROR: PBI0 pattern not correct (ls)
ERROR: pb0_dat_ls expected 0x00000004 actual 0x0000001f mask 0xffffffff
ERROR: PBI0 pattern not correct (ls)
ERROR: pb0_dat_ls expected 0x00000008 actual 0x0000001f mask 0xffffffff
ERROR: PBI0 pattern not correct (ls)
ERROR: pb0_dat_ls expected 0x00000010 actual 0x0000001f mask 0xffffffff
ERROR: PBI0 pattern not correct (ls)
ERROR: pb0_dat_ls expected 0x00000020 actual 0x0000003f mask 0xffffffff
.
.
.
```

PBUS CABLE J12 NOT PROPERLY SEATED

Example Five

```
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    0            MB_TYPE               2          mb_odd_bank_wredc_par_err
=================================================================
 ODD side 'bank_wredc_perr': PARITY error detected in MB2
-----------------------------------------------------------------
 WREDC PARITY error detected by BCGA 3 in bank 0 ( => BANK_co)
 during a WRITE.
        -----  --------------------------------
          1      mbs2:bc[3].sys.bctl0_merr_pe
          1      mbs2:nmcco_rerr
          1      mbs2:nmcco_rla_ecc_check1
       805d      mbs2:nmcco_data
          7      mbs2:nmcco_ecc (parity)
          2      mbs2:bc[3].bcga_log.bctl0_cycle
    1f8a8bc      mbs2:bc3_bank0_log_addr   (phys_addr is fc545e0)
-----------------------------------------------------------------
+++>
<Mon Apr 26 11:46:30 1993>   logmsg(7099):mb_odd_bank_wredc_par_err:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=mb_odd_bank_wredc_par_err @board_type=mb @port=2
****

-----------------------------------------------------------------
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    1            MB_TYPE               2          mb_iso_data_perr
=================================================================
 ODD side 'iso_data' ctl_par_err detected in discrete input staging
 registers by MB2.  Request came from NIA through the XS<odd>.
-----------------------------------------------------------------
  PARITY CHECK - The XBAR data is saved in lookaside registers in
                 the XS1 and XS0.  This is the data that the XBAR
                 sent to the NMB for this request.  Any difference
                 between the XBAR data and the NMB data may indicate
                 a connectivity problem between the XBAR and the NMB.
                 Otherwise the XBAR or the NIA-XBAR connection is bad.
-----------------------------------------------------------------
             Data                  Scan Field
        ----------------------------------------
          ***Data was not preserved in NIA***
        ----------------------------------------
        00   00   80   5d     xs1o:wr_data_lsd2[2]<31..0>
         1    1    1*   0      xs1o:wr_par_lsd2[2]<0..3>
        ----------------------------------------
        00   00   80   5d     mbs2:iso_sys.rwro_data<31..0>
         1    1    1*   0      mbs2:iso_sys.rwro_par<0..3>
        ----------------------------------------
        * indicates parity error.
 Note that quantities in "<>" delimiters are bit descriptors.
 They are not part of the scan field name.
```

```
-------------------------------------------------------------------
XBAR Error Logger/Identifier Version 0.7    (some testing done)
Halting system
   Checking xrte for errors
   Checking xrto for errors
Error detected by xrto -- board has halted
   Checking xs0e and xs1e for errors
   Checking xs0o and xs1o for errors
Error detected by xs0o and xs1o -- boards have halted
NMB2 detected parity error on data from NSP8   (NIA)
  Data  Par in XBAR:     0x0000805d  0x7          *** Parity Error ***  3
  addr= 1f8a8bc  cycle= 2  wr_zone= f  ctl_par= 1b       -No Parity Check yet-
  Compare with NSP and NMB                                  00|00|80|5d
+++>                                                  PAR= 1  1  0  0
<Mon Apr 26 11:46:32 1993>   logmsg(7106):mb_iso_data_perr:0   ↳ 0011 = 3
Hard Error (DiagER349): Non-diagnostic system event
@extractor=mb_iso_data_perr @board_type=mb @port=2     should be 0 1 ΦΦ = 7
****

-------------------------------------------------------------------
HARDERROR #     BOARD TYPE         PORT/SIDE     EXTRACTOR
-----------     ----------         ---------     ---------
    2           XS1_TYPE               1          xs1.send_par_err_mb2
===================================================================
 Send parity error detected by MB2 from NIA in XS0/1O
             Data                  Scan Field
        ----------------------------------------
          ***Data was not preserved in NIA***
        ----------------------------------------
        00  00  80  5d     xs1o:wr_data_lsd2[2]<31..0>
         1   1   1*  0      xs1o:wr_par_lsd2[2]<0..3>
        ----------------------------------------
        00  00  80  5d     mbs2:iso_sys.rwro_data<31..0>
         1   1   1*  0      mbs2:iso_sys.rwro_par<0..3>
        ----------------------------------------
        * indicates parity error


Note(1): HARDERROR 0 Extractor reports a "mb_odd_bank_wredc_par_err" on mb2,
         Bank co.

Note(2): HARDERROR 1 reports the request came from the NIA through the XSO.
         The data received by the memory board is identical to what was sent by
         the xs1o board (data bits <15..8>, parity bit <2>).  This would indicate
         that the problem is on the NIA side.

Note(3): The xbar_err script confirms Note(2).  Note that the NIA is referred
         to as NSP8.

Note(4)  Again confirms Note (2).

The problem in this failure was with the NIA board.  Two scripts that may have
helped isolate this problem are: (1) nia_hrdlog, and (2) wdq.
```

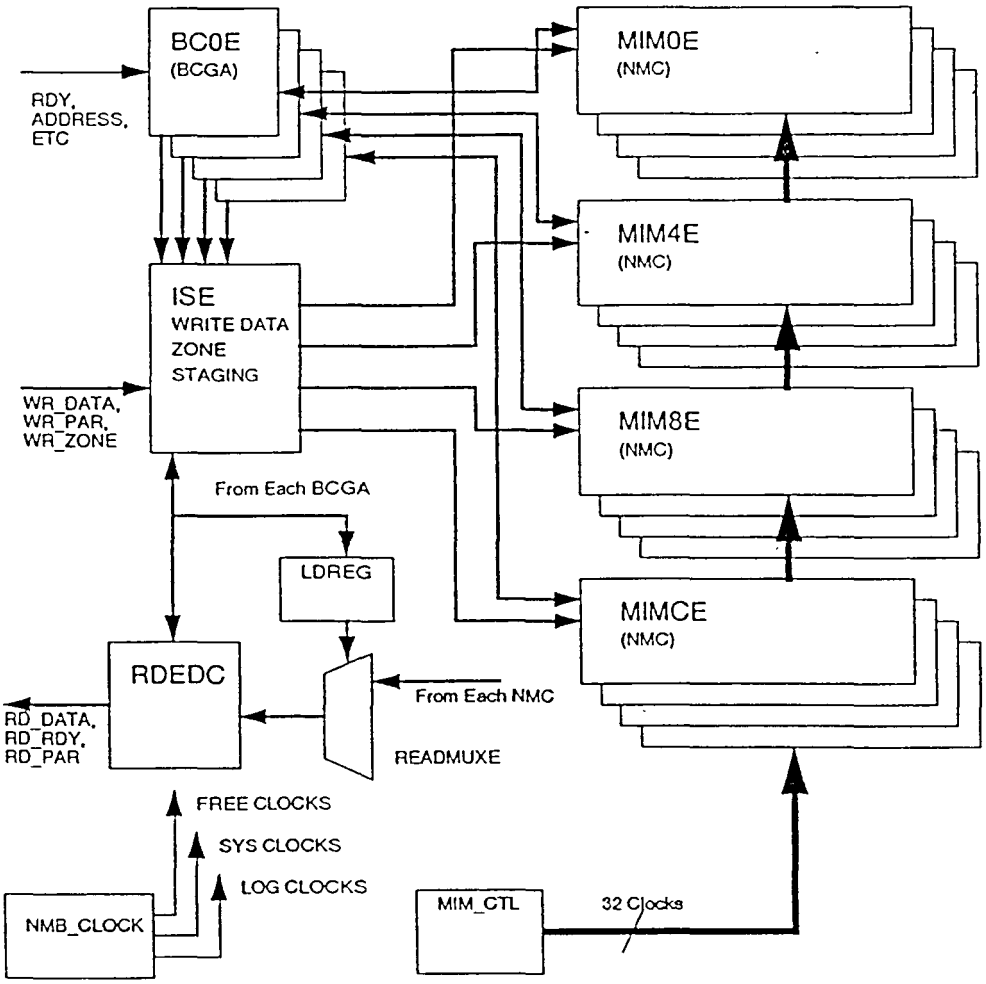


Figure 5-4

NMB Interfaces

Example One

```
Hard error Summary

-------------------------------------------------------------------
HARDERROR #     BOARD TYPE         PORT/SIDE     EXTRACTOR
-----------     ----------         ---------     ---------
    0            MB_TYPE              2          mb_even_bank_ctl_err
===================================================================
 EVEN side 'bank_ctl_err': Request to busy bank detected in MB2.
-------------------------------------------------------------------
  Request while bank busy error detected by BCGA 5 in bank 3
  ( => BANK_7e). No other error state saved for this bank.
       -----  --------------------------------
         1      mbs2:bc[5].sys.bctl3_ctl_he
-------------------------------------------------------------------
+++>
<Wed May  5 13:40:02 1993>   logmsg(459):mb_even_bank_ctl_err:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=mb_even_bank_ctl_err @board_type=mb @port=2
****
```

ADDRESS = XS0

```
spu> xbar_err

XBAR Error Logger/Identifier Version 0.7   (some testing done)

Halting system
System clocks halted
   Checking xrte for errors
   Checking xrto for errors
   Checking xs0e and xs1e for errors
   Checking xs0o and xs1o for errors


spu> nmb_errs 2

Halting System

System clocks halted

****************************

Checking for NMB2 Hard Errors

****************************
  mb_bank_ctl_err

  request while bank busy (no other error state saved for this bank)
  error occurred in BCGA5 BANK3( => BANK_7e)

  ERROR FIELD: mbs2:bc[5].sys.bctl3_ctl_he = 16#1
```

```
**************************************************************************
**Bank 7e (NMB) swapped and NMB2 swapped with NMB3 - Same Failure***

**swapped xs0 boards**
**************************************************************************


Hard error Summary

------------------------------------------------------------------------
HARDERROR #     BOARD TYPE         PORT/SIDE     EXTRACTOR
-----------     ----------         ---------     ---------
    0            MB_TYPE              2          mb_odd_bank_ctl_err
========================================================================
 ODD side 'bank_ctl_err': Request to busy bank detected in MB2.
------------------------------------------------------------------------
  Request while bank busy error detected by BCGA 1 in bank 3
  ( => BANK_7o). No other error state saved for this bank.
       -----  --------------------------------
         1      mbs2:bc[1].sys.bctl3_ctl_he
------------------------------------------------------------------------
+++>
<Wed May  5 15:01:08 1993>   logmsg(886):mb_odd_bank_ctl_err:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=mb_odd_bank_ctl_err @board_type=mb @port=2
****

+++>
<Wed May  5 15:01:08 1993>   /diag/bin/hard_logger.exc(860):../hard.c:306
HardLog End (DiagER483): Hard_logger completion

****


Note(1): HARDERROR 0 reported the problem to be on nmb2, even side, bank 7e.

Note(2): xbar_err found no problem.  This indicates that the problem should be
         on the memory board or between the xbar and memory.

Note(3): Swapping bank 7e did not fix the problem, so the xs0o and xs0e boards
         were rotated (Address data to the bcga's is from the xs0 boards).

Note(4): After rotating the xs0 boards, the failure moved to the odd bank.

Replacing the original xs0e board solved the problem.
```

Example Two

```
   HARDERROR #     BOARD TYPE         PORT/SIDE     EXTRACTOR
   -----------     ----------         ---------     ---------
    0              SP_TYPE             0           sp_nrfa3.cbus_par_err
=========================================================================
 sp0:nrfa3.cbus_par_err = 1
     Parity error script for cbus received by NRFA3 gate array.
        Register name        Ring value          Board signal name
        ------------------------------------------------------------
        cbus_ldata<7:0>         80               CBUS_DATA<31..24>
        cbus_lpar                1*              CBUS_PAR<4>
        cbus_udata<7:0>         00               CBUS_DATA<63..56>
        cbus_upar                1               CBUS_PAR<0>
        rd_mux_ctl3<1:0>         3 - NRC gate arrays
        * indicates parity error
+++>
<Tue Apr 27 16:03:34 1993>   logmsg(2458):sp_nrfa3_cbus_par_err:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=sp_nrfa3_cbus_par_err @board_type=sp @port=0
****



------------------------------------------------------------------------
HARDERROR #     BOARD TYPE         PORT/SIDE     EXTRACTOR
-----------     ----------         ---------     ---------
    1            SP_TYPE              0          sp_ndp.yuvc_upd_err
========================================================================
 sp0:ndp0.yuvc_par_err = 2
 sp0:ndp1.yuvc_par_err = 2  |  These should all have the same value
 sp0:ndp2.yuvc_par_err = 2  /
     Parity error script for Upd bus received by
        NDP0, NDP1, and NDP2 gate arrays.
   Register name      Ring value              Board signal name
   ------------------------------------------------------------------------
   upd<>    00  19  99  88  80  19  79  80    UPD_DATA<63..0>
   upd<>     1   0   1   1   1*  0   0   0    UPD_PAR<0..7>
   * indicates parity error
+++>
<Tue Apr 27 16:03:35 1993>   logmsg(2464):sp_ndp_yuvc_upd_err:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=sp_ndp_yuvc_upd_err @board_type=sp @port=0
****
```

```
-------------------------------------------------------------------
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    2            SP_TYPE               0          sp_nrc.xro_stop_in
===================================================================
 sp0:nrc0.par_err_stop_in = 8
 sp0:nrc1.par_err_stop_in = 8  |  These should all have the same value
 sp0:nrc2.par_err_stop_in = 8  /
     Parity error script for xro_data
        bus received by NRC0, NRC1, and NRC2 gate arrays.
   Register name      Ring value              Board signal name
   ------------------------------------------------------------------------
   xro_data<>       80  19  79  80           XRO_DATA<31..0>
   xro_data<>        1*  0   0   0           XRO_PAR<0..3>
   * indicates parity error
+++>
<Tue Apr 27 16:03:36 1993>   logmsg(2470):sp_nrc_xro_stop_in:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=sp_nrc_xro_stop_in @board_type=sp @port=0
****



-------------------------------------------------------------------
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    3           XRT_TYPE              1          xrt.rtn_par_err_sp0
===================================================================
 Return parity error detected by SP0 from NMB1 in XRTO
            Data                Scan Field
        ----------------------------------------
        80  19  79  80     sp0:xbo_rd_data<31..0>
         1*  0   0   0      sp0:xbo_rd_par<0..3>
        ----------------------------------------
        80  19  79  80     xrto:r3data_p[0]<31..0>
         1*  0   0   0      xrto:r3par_p[0]<0..3>
        ----------------------------------------
          ***Data was not preserved in NMB1***
        ----------------------------------------
        * indicates parity error
-------------------------------------------------------------------
 xrto:p_err_en<0> = 1
 xrto:rtn_par_err_p[0] = 1
+++>
<Tue Apr 27 16:03:38 1993>   logmsg(2480):xrt_rtn_par_err_sp0:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=xrt_rtn_par_err_sp0 @board_type=xrt @port=1 @orig=mb1 @side=mem
****

+++>
<Tue Apr 27 16:03:41 1993>   /diag/bin/hard_logger.exc(2435):../hard.c:306
HardLog End (DiagER483): Hard_logger completion
```

```
spu> xbar_err

XBAR Error Logger/Identifier Version 0.7    (some testing done)

Halting system
System clocks halted
   Checking xrte for errors
Error detected by xrte -- board has halted
NSP0 detected parity error on data from NMB1
  Data  Par in XBAR:      0x00199988  0xd
Compare with NSP and NMB
   Checking xrto for errors
Error detected by xrto -- board has halted
NSP0 detected parity error on data from NMB1
  Data  Par in XBAR:      0x80197980  0x1          *** Parity Error ***  0x0
  Compare with NSP and NMB
   Checking xs0e and xs1e for errors
   Checking xs0o and xs1o for errors
```

Note(1): HARDERROR 0 reported the problem was detected by sp0 (CBUS_PAR <4> and DATA<31..24>.

Note(2): HARDERROR 1 reported an error on UPD_DATA <32..24, PAR <4>

Note(3): HARDERROR 2 reported an error on XRO DATA <31..24>, PAR <4>

Note(4): HARDERROR 3 reported an error on the return read data from NMB1 on DATA<31..24> AND PAR <4>.

Note(5): The xbar_err script detected an error on the data coming from NMB1, PAR<4>.

Based on note(5), the problem should be with the xrto board, nmb1, or connectivity between the xrto and nmb1 boards.

spu4000 found a short on mb1_xro.rd_data<31>

Example Three

```
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    0            IA_TYPE               8          ia_rd_par_err_o
================================================================================
NIA #8 detected a parity error on the odd side read data bus.
        (xro_ia.rd_data<31..0>)
              Data                            Scan Fields
        ------------------------------------------------------------------
        80  21  a2  10                  ias8:r_rdata_o
         1*  1   0   0                  ias8:r_rpar_o_s1_3_0
        ------------------------------------------------------------------
         * indicates parity error
--------------------------------------------------------------------------------
+++>
<Tue May 11 16:32:49 1993>   logmsg(1303):ia_rd_par_err_o:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=ia_rd_par_err_o @board_type=ia @port=8
****

--------------------------------------------------------------------------------
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    1            IA_TYPE               8          ia_rdq_wrt_pe
================================================================================
NIA #8 detected a write data parity error during an RDQ write access.
              Data                            Scan Fields
        ------------------------------------------------------------------
                 0ce             ias8:xds10.s2,ias8:xds10.12 (RDQe write addr)
        00 00 00 00              rdq -r 0ce -----+++>
<Tue May 11 16:32:51 1993>   logmsg(1309):ia_rdq_wrt_pe:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=ia_rdq_wrt_pe @board_type=ia @port=8
****


--------------------------------------------------------------------------------
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    2            IA_TYPE               8          ia_cds_rd_pe
================================================================================
NIA #8 detected a read data parity error as reported by the CDS arrays.
              Data                            Scan Fields
        ------------------------------------------------------------------
        02 be 02 be 80 31 00 01    ias8:cds[7-0].rd_shadow
         0  1  0  1  1* 0  1  0    ias8:cds[7-0].rd_shadow_par
        ------------------------------------------------------------------
                              0ce  ias8:r_rdq_radr_s3<10..0> (RDQ address)
        00 00 00 00 80 21 a2 10    rdq -r 0ce  on ias8       (RDQ data)
         1  1  1  1  1* 1  0  0    rdq -r 0ce  on ias8       (RDQ parity)
        ------------------------------------------------------------------
```

```
spu> xbar_err

XBAR Error Logger/Identifier Version 0.7   (some testing done)

Halting system
System clocks halted
   Checking xrte for errors
   Checking xrto for errors
   Checking xs0e and xs1e for errors
   Checking xs0o and xs1o for errors
```

Note(1): HARDERROR 0 detected a ia_rd_par_0 (odd side read data bus) on data bits <31..24> and par bit <4>.

Note(2): HARDERROR 1 detected a write data parity error during an RDQ write access.

Note(3): HARDERROR 2 - The NIA CDS array reported a read data parity error on data bits <31..24> and PAR bit <4>.

Note(4): The NIA Block Diagram shows the ia_rd_data coming onto the NIA from the XRTO board (in this failure). The flow is then to the RDQ stage and then the CDS arrays. Based on this data flow, this failure could be caused by the NIA, XRTO, one of the Memory boards, or connectivity between the NIA-XBAR or XBAR-NMB?.

Note(5): The xbar_err script did not detect a problem so normally you would consider the problem to be isolated to the NIA side. Unfortunately, the ia_xro_rtn_par_err and ia_xre_rtn_par_err signals are very noisy an can cause system crashes. For this reason, these two signals are turned off and the xbar is not halted and does not retain data sent to the NIA during a rd_par error.

In this example, the failure was caused by a short on one of the nmb boards (rd_data<31>.

For debugging purposes, these signals can be turned on adding the following to /diag/bin/initall after the "mminit || exit" line.

```
halt
config=$(get -q xrte:p_config)
p_err_en=$(get -q xrte:p_err_en)
let "config|=16#100"
let "p_err_en|=16#100"
put xrt?:p_config $config
put xrt?:p_err_en $p_err_en
      scn_util -i
      print "System clocks started"
```

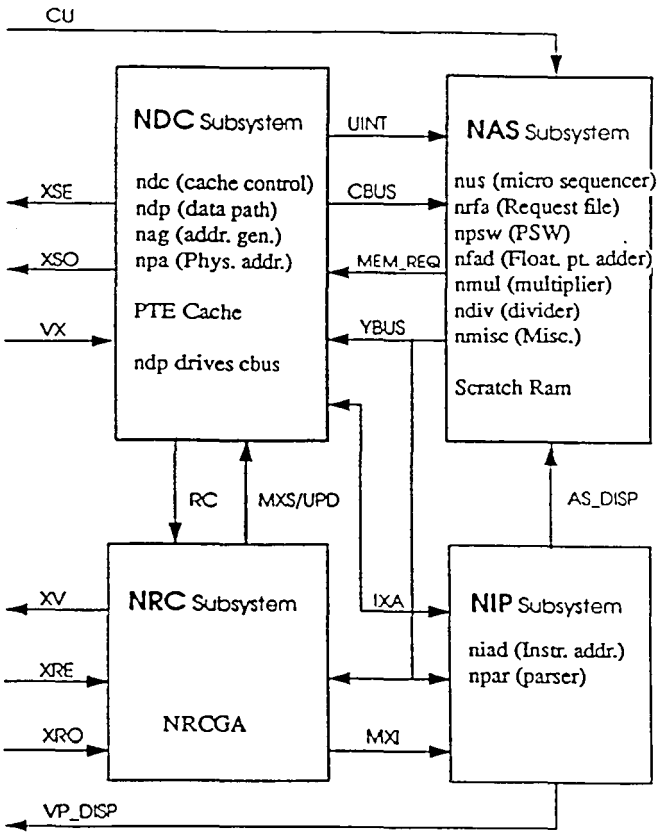


Figure 5-5
NSP Subsystem Interconnect

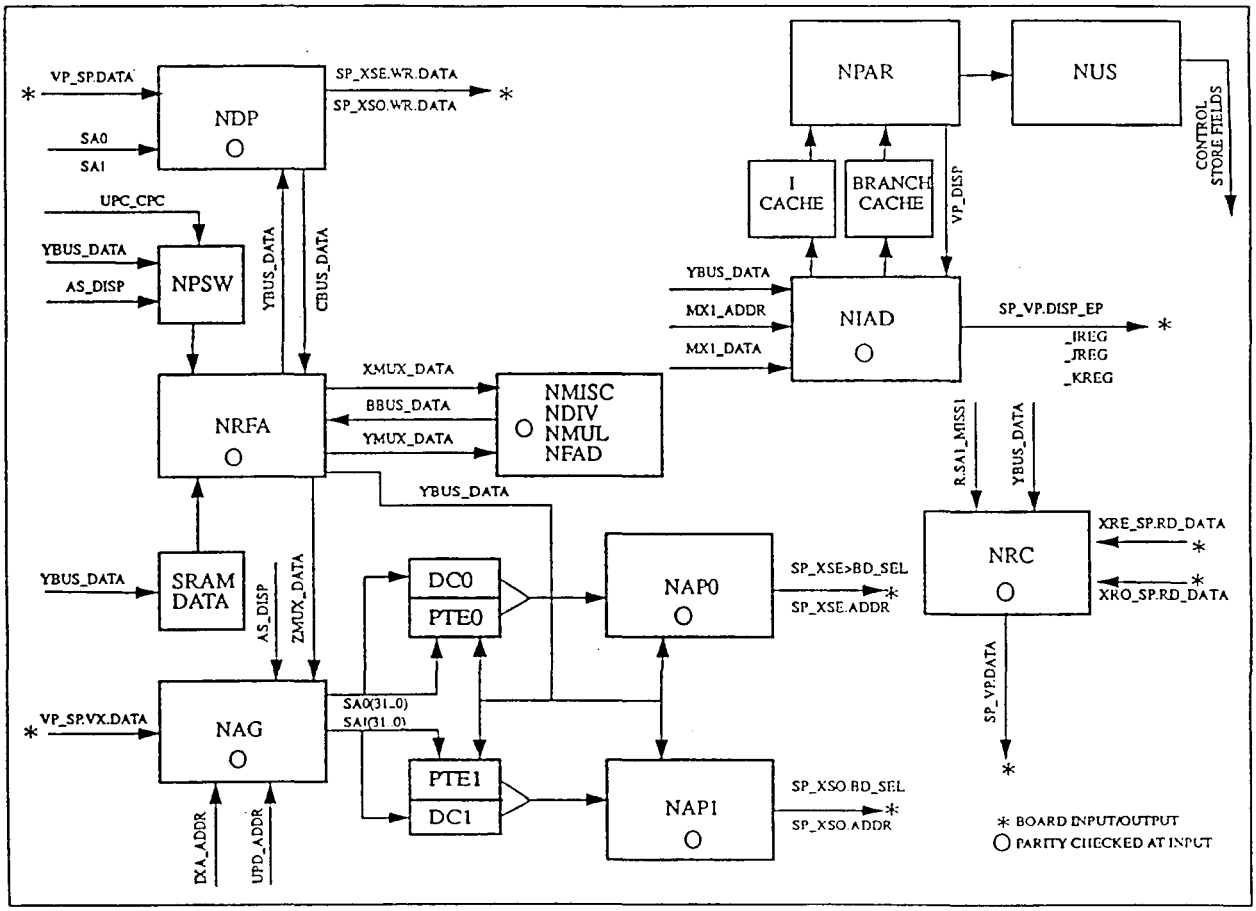


Figure 5-6
NSP External Interfaces

Example One

```
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    0            CU_TYPE              N/A          cu_addr_par_err.0
=============================================================================
        NCU harderror detected
        r_addr_par_err0 = 1
        Register                  Value    Board signal name
        ------------------------------------------------------
        xso_11_addr<25:19>         41      XSO_CU.ADDR<28..22>
        ctl_par<0>                 0*      XSO_CU.CTL_PAR<0>
          * indicates parity error
+++>
<Mon May  3 11:18:20 1993>   logmsg(2435):cu_addr_par_err_0:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=cu_addr_par_err_0 @board_type=cu @port=-1
****

-----------------------------------------------------------------------------
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    1           XS0_TYPE               1          xs0.send_par_err_ncu
=============================================================================
 Control parity error detected by NCU from SP1 in XS0/10
        NOTE - Address bit mapping is relative to the XBAR.
               addr<28> corresponds to addr<25> in the NCU.
-----------------------------------------------------------
 addr<28:22> <21:15> <14:8> {<7:3> cycle<1:0>}
    ------------------------------------------------
          ***Data was not preserved in SP1***
    ------------------------------------------------
      41       00       0c       00              xbar data
       0*       1        1        1            xbar ctl_par<0:3>
    ------------------------------------------------
      41       02       ??       ??              ncu data
       0*       0        ?        ?            ncu ctl_par<0:3>
    ------------------------------------------------
    ? indicates data was not staged
    * indicates parity error
-----------------------------------------------------------------------------
 The xbar data is saved in lookaside registers in the xs1 and xs0.
 This is the data that the xbar sent to the ncu for this request.
 Any difference between the xbar data and the ncu data may
 indicate a connectivity problem between the xbar and the ncu.
-----------------------------------------------------------------------------
 Scan fields:           ncu                          xbar
        ---------------------------      --------------------------
       0208200c cu:xso_11_addr          02080180 xs0o:addr_lsd2[8]
              1 cu:nadr.11_cycle                0 xs0o:cycle_lsd2[8]
             04 cu:ctl_par                     1e xs1o:ctl_par_lsd2[8]
-----------------------------------------------------------------------------
 xs0o:m_err_en<8> = 1
 xs0o:send_par_err<8> = 1
```

```
+++>
<Mon May  3 11:18:23 1993>   logmsg(2448):xs0_send_par_err_ncu:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=xs0_send_par_err_ncu @board_type=xs0 @port=1 @orig=sp1 @side=mem
****

-----------------------------------------------------------------
HARDERROR #     BOARD TYPE         PORT/SIDE     EXTRACTOR
-----------     ----------         ---------     ---------
    2           XS0_TYPE              0          xs0.send_par_err_ncu
=================================================================
 Control parity error detected by NCU.  EVEN xbar notified and
 halted.  Check to make sure that the ODD xbar has also pulled
 a control parity error.  The NCU only checks parity on the ODD
 side, but it notifies the EVEN xbar when it pulls an ODD error.
 If there is no error on the ODD side, then there is a flaw in the
 error circuits on either the XS0E, the XS0O, or the NCU.
-----------------------------------------------------------------
+++>
<Mon May  3 11:18:24 1993>   logmsg(2454):xs0_send_par_err_ncu:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=xs0_send_par_err_ncu @board_type=xs0 @port=0
****



spu> xbar_err

XBAR Error Logger/Identifier Version 0.7    (some testing done)

Halting system
System clocks halted
   Checking xrte for errors
Error detected by xrte -- board has halted
   Checking xrto for errors
Error detected by xrto -- board has halted
   Checking xs0e and xs1e for errors
Error detected by xs0e and xs1e -- boards have halted
NCU detected parity error on data from NSP1
  Data  Par in XBAR:      0x00000000  0xf
  addr= 0080180  cycle= 0  wr_zone= 1  ctl_par= 0e       -No Parity Check yet
  Compare with NSP and NCU
   Checking xs0o and xs1o for errors
Error detected by xs0o and xs1o -- boards have halted
NCU detected parity error on data from NSP1
  Data  Par in XBAR:      0x00199988  0xd
  addr= 2080180  cycle= 0  wr_zone= 0  ctl_par= 1e      -No Parity Check yet-  C
ompare with NSP and NCU
spu>


spu> spu4000
+++>
<Mon May  3 11:09:02 1993>   /diag/test/spu/spu4000(2049):../sp_xb_test.c:398
```

```
Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1
Source Signal: sp_xse.addr                          Source Port: 1
Sink Signal:   sp_xso.addr                          Sink Port:   1
Sink Field:    xs0o:addr_is[1]
Source Signal Bit Position: 5          Sink Signal Bit Position: 0
Expected:                   00000000
Actual:                     00000001
****

+++>
<Mon May  3 11:09:02 1993>   /diag/test/spu/spu4000(2049):../sp_xb_test.c:398
Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1
Source Signal: sp_xse.addr                          Source Port: 1
Sink Signal:   sp_xso.addr                          Sink Port:   1
Sink Field:    xs0o:addr_is[1]
Source Signal Bit Position: 6          Sink Signal Bit Position: 0
Expected:                   00000000
Actual:                     00000001
****
fail: "spu4000 subtest 810 ABORTED, Fail Threshold Exceeded"
```

Note(1): HARDERROR 0 - detected a cu_addr_par.0 on address bits <28..22>/parity bit <0> from one of the xs0 boards.

Note(2): HARDERROR 1 - reports that the address data came from SP1 and the data in the xs0o is the same as on the NCU. This would indicate the problem is located on the SP1 side of the xbar.

Note(3): The address parity, as shown in the xbar_err script, is all confused. This would also indicate a problem on the SP1 side of the xbar.

Note(4): spu4000 detected a short on the sp1. This was on the sp_xs0o.addr<3>. This was reported as the Sink Signal Bit Postion 0. For address data, you must add 3 to this bit position number.

With addres errors always add 3 to the Sink Signal Bit Positon !

Example Two

```
Mon Feb 22 15:58:26 1993>   /diag/test/spu/spu4000(361):../sp_xcl_test.c:363
Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1
Source Signal: SP_XC.DEADLOCK                    Source Port: 0
Sink Signal:   SP_XC.TRAP_COMP<8..0>             Sink Port:   0
Sink Field:    xcls:CU_TRAP_COMP_SPX
Source Signal Bit Position: 0          Sink Signal Bit Position: 0
Expected:                   00000000
Actual:                     00000001
****
Running Scalar Processor 1 to Crossbar Connectivity Test.
Running Scalar Processor 1 to XCL Test.

+++>
<Mon Feb 22 15:58:39 1993>   /diag/test/spu/spu4000(361):../sp_xcl_test.c:363
Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1
Source Signal: SP_XC.DEADLOCK                    Source Port: 1
Sink Signal:   SP_XC.TRAP_COMP<8..0>             Sink Port:   0
Sink Field:    xcls:CU_TRAP_COMP_SPX
Source Signal Bit Position: 0          Sink Signal Bit Position: 0
Expected:                   00000000
Actual:                     00000001
****

+++>
<Mon Feb 22 15:58:40 1993>   /diag/test/spu/spu4000(361):../sp_xcl_test.c:363
Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1
Source Signal: SP_XC.TRAP_COMP<8..0>             Source Port: 1
Sink Signal:   SP_XC.TRAP_COMP<8..0>             Sink Port:   0
Sink Field:    xcls:CU_TRAP_COMP_SPX
Source Signal Bit Position: 0          Sink Signal Bit Position: 0
Expected:                   00000000
Actual:                     00000001
****
```

Note(1): The above spu4000 failure was caused by a defective NSP in head 0.

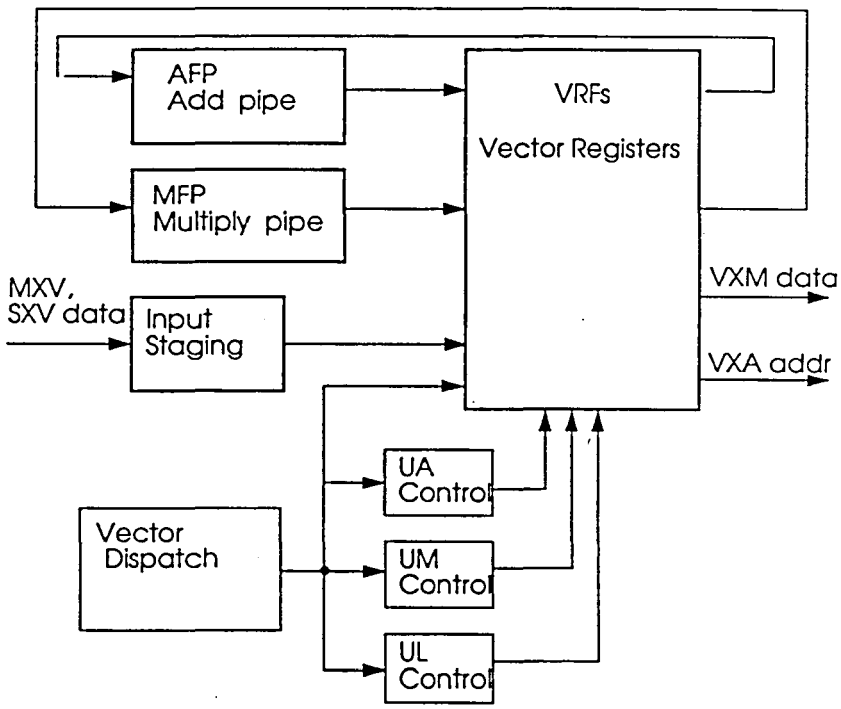


Figure 5-7

Basic NVP Organization

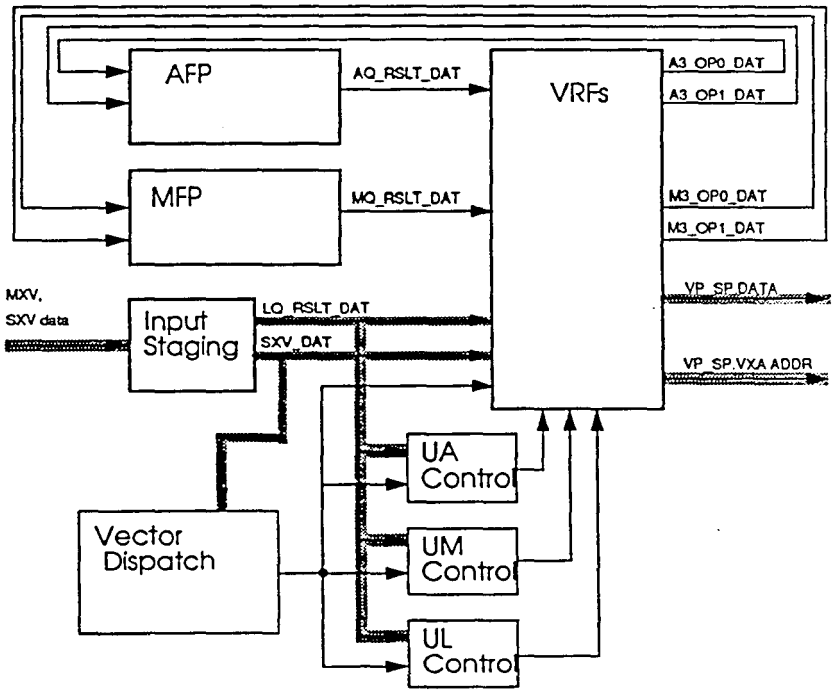


Figure 5-8
NVP Data Flow

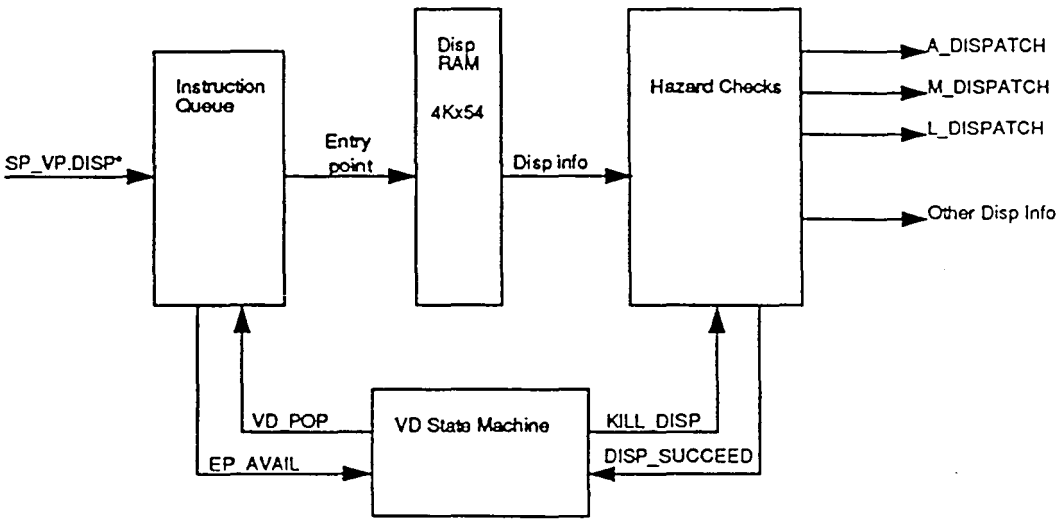


Figure 5-9
Vector Dispatch

Indicates parity is checked here

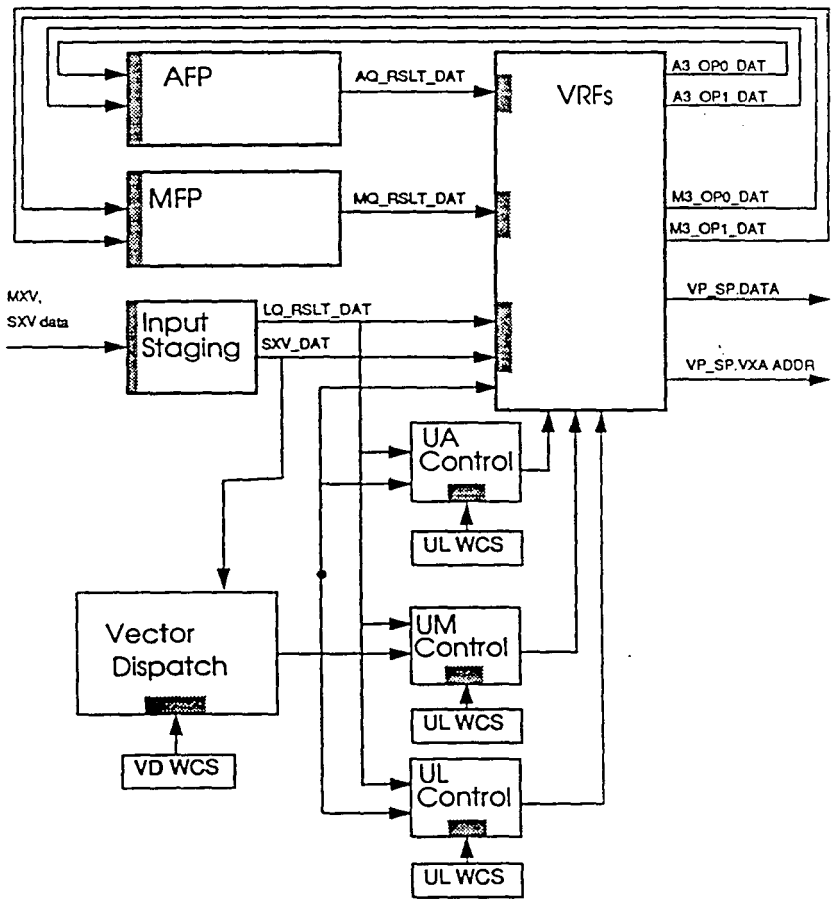


Figure 5-10
NVP Parity Checking

Section Six

C3800 Series

Connectivity Troubleshooting

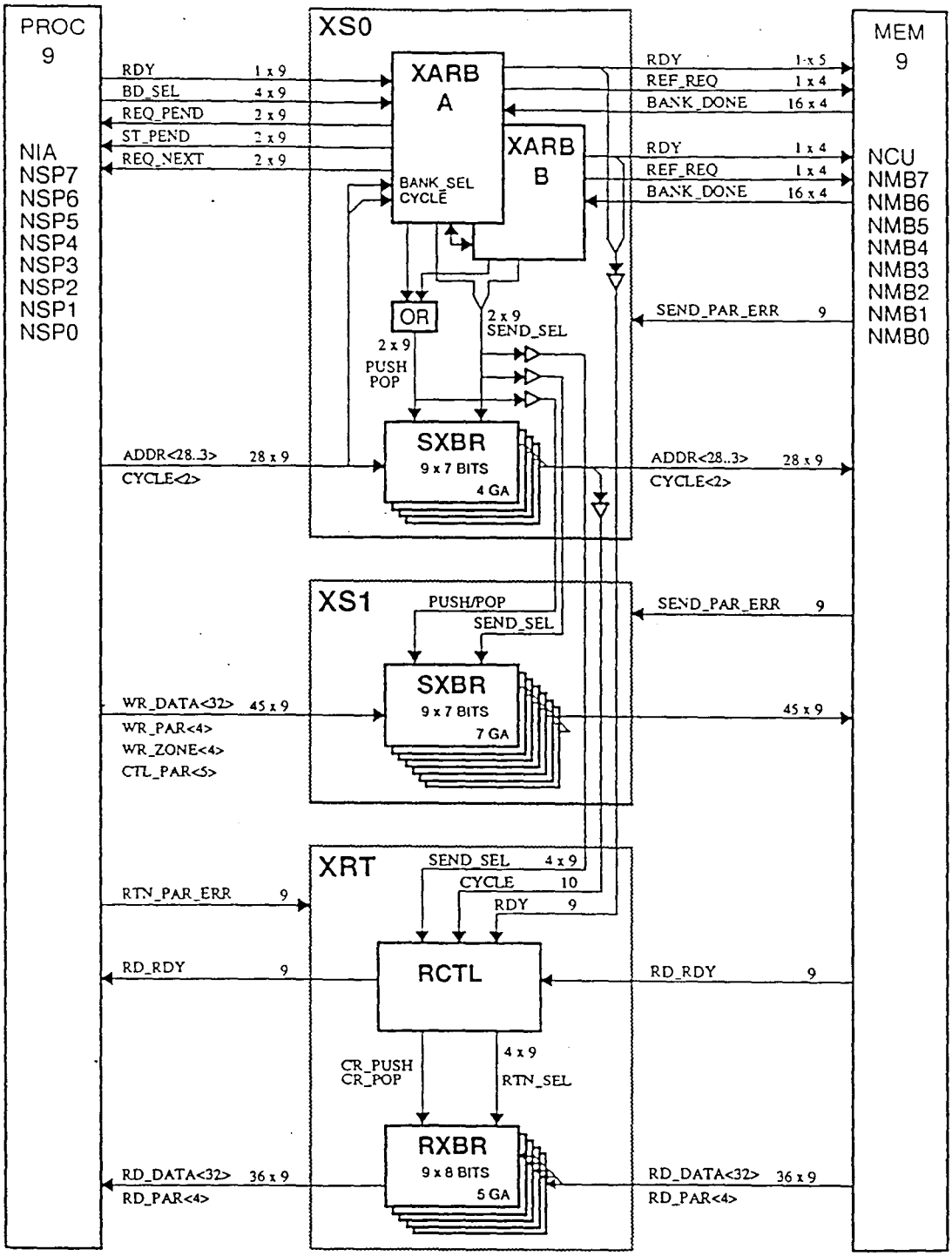


Figure 6-1
Signal Flow

Debugging Tips

1.0 Troubleshooting Backplanes

1.1 Net Resistance Rules.

The normal net resistance should read 52 ohms. If any net measures < 49 or > 56 ohms, it should be suspect.

Point-to-point measurements should be =< 2.5 ohms (i.e., xbar augat to nsp augat).

Measurements from one end of an epont to the other end should be < 1 ohm.

It takes a 21-25 ohm net to actually create a large enough voltage drop to be detected by spu4000.

A few ohm increase in point-to-point resistance will still not be sufficient to cause the termination resistance to vary outside of normal parameters, of 50-56 ohms. Therefore the only reliable measurement on intermittent nets is point-to-point.

The BPS power switch should be set to the off position when measuring nets on a backplane.

Termination is always located on the receiver end of a net name (i.e., the termination is located on the NSP for the XC_SP.CU_STATUS net).

1.2 spu4000 Failures

1.2.1 Intermittent Failures

It should be understood that spu4000 should not be relied on when trying to solve intermittent failures. This is because spu4000 is not an at-speed test and therefore will not detect subtle point-to-point net failures.

Example One

The following is an example of a spu4000 connectivity failure.

Source Signal:	sp_xse.wr_data	Source Port: 3
Sink Signal:	sp_xse.wr_data	Sink Port: 3
Sink Field:	xs1e:wr_data_is [3]	
Failing Bit Position:	10	
Expected:	00000400	
Actual:	00000000	

The source port indicates the point of origin for the failure. In the above example the net would become sp3_xse.wr_data. The port value will physically follow the configuration of the system. In the case of a value "9" in this location, the xbar will be the source.

The sink field is generally a register location on the board that detected the failure. In this case it is the input staging register on xs1e.

The failing bit position is self explanatory. The "Expected" value is the data pattern expected to be seen by the test and the "Actual" value indicates the results. Keep in mind that a 0 value for the failed bit will indicate a open and a 1 value indicates a short. Quite often multiple 1's will be displayed for shorted nets.

Example Two

The following is a second example of a spu4000 connectivity failure. Note that in this example the Source Signal and Sink Signal are different.

<Thu Dec 10 14:41:05 1992> /diag/test/spu/spu4000:../ia_xb_test.c:415

Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1

Source Signal:	cu_xro.rd_data<31..0>	Source Port: 8
Sink Signal:	xro_ia8.rd_data<31..0>	Sink Port: 8
Sink Field: ia8:r_rdata_o		
Failing Bit Position:	28	
Expected:	00000000	
Actual:	10000000	

****

In this example, the failing net was "xro_ia8.rd_data<28>.

1.3 ConvexOS Crash

As previously stated, spu4000 will not detect all connectivity problems. In some cases you may be required to analyze the hardlogger data to determine where the connectivity problem may be located. The following are two examples of this situation.

Example One

The following is an example of a nag failure caused by a faulty augat connector.

HARDERROR #	BOARD TYPE	PORT/SIDE	EXTRACTOR
0	SP_TYPE	6	sp_nag0.vxaq_par_err

sp6:nag0.vxaq_par_err = 1

Parity error script for vxa_addr received by NAG0 gate array.

vxaq_addr0<15:0>	ff	f8	VXA_ADDR<15..0>
vxaq_addr0_par<0:1>	1	1*	VXA_ADDR_PAR<2..3>
vxaq_addr1<15:0>	0a	cc	VXA_ADDR<15..0>
vxaq_addr1_par<0:1>	1	1	VXA_ADDR_PAR<2..3>
vxaq_addr2<15:0>	0a	d0	VXA_ADDR<15..0>
vxaq_addr2_par<0:1>	1	0	VXA_ADDR_PAR<2..3>
vxaq_rptr_las<1:0>	0		VXAQ_RPTR<1..0>

In this example, the error was detected by nsp6 NAG0 gate array. As the NAGs check parity at the board input from the Vector board, the suspects would be the Vector board or a connectivity problem between the Vector and Scalar boards. In this case, replacement of the Vector board did not fix the problem. As the hardlogger message indicates the problem area to be the vp6_sp6.vxa_addr<7-0> bits or vp6_sp6.par<3> bit, these points were checked on the backplane. The problem was found to be an intermittent short on the Vector board augat connector, pin 126, section 4 (vp6_sp6.vxa_addr<2>).

Example Two

Address bits on the Crossbar Interface are numbered from bit 28 through bit 3 to reflect their original position in the physical address word generated by the NSP or NIA. Bits 31 through 29 of the physical address are used in memory board select generation by the NSP and NIA and are therefore unnecessary at the memory board interface. Similarly, bit 2 of the physical address selects whether the address is to the even or odd port of memory. Bits 1 and 0 together with operand size are used by the processors to specify the zone bits for a write or TAM (Test and Modify) and are otherwise unnecessary at the NMB. This information must be considered when analyzing failures similar to this example.

HARDERROR #	BOARD TYPE	PORT/SIDE	EXTRACTOR
0	MB_TYPE	7	mb_bcga_ise_cycle

EVEN side 'cycle' ctl_par_err detected in BCGA input staging registers by MB7. Request came from SP0 through the XS<even>.

SANITY CHECK - Each BCGA should contain the same data as the other three BCGAs in the table below. Data mismatch indicates a failure on the NMB, or a slow signal coming from the XBAR that misses timing at the BCGA.

```
Scan fields:          NMB                              XBAR
                      -------------------------        -------------------------
                    0000601 mbs7:bc4_sys_addr        0000601 xs0e:addr_lsd2[7]
                          0 mbs7:bc[4].sys.is_cycle        0 xs0e:cycle_lsd2[7]
                        * 1 mbs7:bc[4].sys.rcycle_err      1 xs0e:send_par_err<7>
                          f mbs7:bc[4].sys.ris_ctl_par_3_0  ________________
                    0000601 mbs7:bc5_sys_addr        //////NOTE://////
                          0 mbs7:bc[5].sys.is_cycle  | The BCGAs do |
                        * 1 mbs7:bc[5].sys.rcycle_err | not check     |
                          f mbs7:bc[5].sys.ris_ctl_par_3_0 | parity on the |
                    0000601 mbs7:bc6_sys_addr        | zone bits.    |
                          0 mbs7:bc[6].sys.is_cycle  | There are 5   |
                        * 1 mbs7:bc[6].sys.rcycle_err | bits of ctl   |
                          f mbs7:bc[6].sys.ris_ctl_par_3_0 | parity, but |
                     0000601 mbs7:bc7_sys_addr       | only 4 bits   |
                            0 mbs7:bc[7].sys.is_cycle | are staged in |
                          * 1 mbs7:bc[7].sys.rcycle_err | the BCGAs.  |
                            f mbs7:bc[7].sys.ris_ctl_par_3_0 +---------------+
-----------------------------------------------------------------
```

PARITY CHECK - The XBAR data is saved in lookaside registers in the XS1 and XS0. This is the data that the XBAR sent to the NMB for this request. Any difference between the XBAR data and the NMB data may indicate a connectivity problem between the XBAR and the NMB. Otherwise the XBAR or the SP-XBAR connection is bad. Address bit mapping is relative to the schematic. Addr<28> corresponds to addr<25> in the scan fields.

```
-----------------------------------------------------------------
addr<28:22> <21:15> <14:8> {<7:3> cycle<1:0>}
-----------------------------------------------------
***Data was not preserved in SP0***
-----------------------------------------------------
   00    00    30    04        XBAR lookaside data
   1     1     1     1*        XBAR ctl_par<0:3>
-----------------------------------------------------
   00    00    30    04        MB7 data from BC4
   1     1     1     1*         MB7 ctl_par<0:3>
-----------------------------------------------------
* indicates parity error
-----------------------------------------------------------------
```

```
  ************************
**NOTE BIT ASSIGNMENTS**
  ************************

  ********************
**0000   0110 0    **
**8765   4321cycle **
**                  **
**Parity bit 3 for 04 **
  ********************
```

for address take 2 bits off and add 3

```
spu> xbar_err
+++>
XBAR Error Logger/Identifier Version 0.7 (some testing done)

Halting system
System clocks halted
 Checking xrte for errors
Error detected by xrte -- board has halted
 Checking xrto for errors
 Checking xs0e and xs1e for errors
```

```
Error detected by xs0e and xs1e -- boards have halted
NMB0 detected parity error on data from NSP0
 Data Par in XBAR: 0x00199988 0xd
 addr= 0000601 cycle= 0 wr_zone= 0 ctl_par= 1f          -No Parity Check yet-
 Compare with NSP and NMB
NMB1 detected parity error on data from NSP0
 Data Par in XBAR: 0x00199988 0xd
 addr= 0000601 cycle= 0 wr_zone= 0 ctl_par= 1f          -No Parity Check yet-
 Compare with NSP and NMB
NMB6 detected parity error on data from NSP0
 Data Par in XBAR: 0x00199988 0xd
 addr= 0000601 cycle= 0 wr_zone= 0 ctl_par= 1f          -No Parity Check yet-
 Compare with NSP and NMB
NMB7 detected parity error on data from NSP0
 Data Par in XBAR: 0x00199988 0xd
 addr= 0000601 cycle= 0 wr_zone= 0 ctl_par= 1f          -No Parity Check yet-
 Compare with NSP and NMB
 Checking xs0o and xs1o for errors

spu> spu4000
                    (The following is the results of spu4000 being run on this failure)
<Mon May 3 10:58:18 1993> /diag/test/spu/spu4000(2049):../mb_xb_test.c:480
Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1
Source Signal: mb_xre.rd_data      Source Port: 1
Sink Signal: xso_mb.addr           Sink Port: 1
Sink Field: mb1:bc0_sys_addr
Source Signal Bit Position: 8          Sink Signal Bit Position: 0
Expected:              00000000
Actual:                00000001
****

+++>
<Mon May 3 10:58:18 1993> /diag/test/spu/spu4000(2049):../mb_xb_test.c:480
Subtest Fail (DiagER466): Spu4000 Error: Connectivity Test Failure

Detected Short or Stuck-at-1
Source Signal: mb_xre.rd_data      Source Port: 1
Sink Signal: xso_mb.addr           Sink Port: 1
Sink Field: mb1:bc0_sys_addr
Source Signal Bit Position: 9          Sink Signal Bit Position: 0
Expected:              00000000
Actual:                00000001
****
fail: "spu4000 subtest 810 ABORTED, Fail Threshold Exceeded"
```

xs0o_mb1.addr <3>
mb1_xrte.rd.data <8>

Note (1): In the Sanity check there is an * in all 4 bcga fields. This would indicate a parity error but as all bcga's detected the error, the problem is probably prior to this stage.

Note (2): The Parity check shows a parity error was detected in the crossbar data and the data coming onto the memory board (addr bits 3-7 or parity bit 3). This would normally indicate a problem on the NSP, NSP-XBAR connectivity, or the XBAR board.

Note(3): The xbar_err script did not detect an error in the crossbar data but the parity for the address is in error. This would normally indicate that the address was bad when it arrived onto the crossbar board

```
data = 00 19 99 88      parity = 0xd
       1 0 1 1          = b  /inverted/ = d      (parity ok)
addr =  0 00 06 01      parity = 1f (zone; <7..3 and cycle>; <14..8>; <21..15>; <28..22>
       1 1 1 0          = e (reported as f)      (parity bad)
```

Note(4): The spu4000 failures indicate a short on the Source Signal (mb1_xre.rd_data) and Sink Signal (xso_mb1.addr). The short was reported in bit 0 position, but as this is address data, you must add 3 to this location which in this case equals bit 3.

The problem was a short on the xso_mb1.addr<3>. NOTE: This is an unusual example as the short on the mb backplane was feeding back up into the xbar and corrupting the data there.

Example Three

```
-------------------------------------------------------------------
HARDERROR #          BOARD TYPE      PORT/SIDE      EXTRACTOR
-----------          ----------      ---------      ---------
        0            MB_TYPE         0              mb_even_bank_ctl_err
===================================================================
 EVEN side 'bank_ctl_err': Request to busy bank detected in MB0.
-------------------------------------------------------------------
 Request while bank busy error detected by BCGA 4 in bank 0
 ( => BANK_0e). No other error state saved for this bank.
 ----- --------------------------------
 1 mbs0:bc[4].sys.bctl0_ctl_he
-------------------------------------------------------------------
+++>
<Tue May 11 16:56:21 1993> logmsg(1660):mb_even_bank_ctl_err:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=mb_even_bank_ctl_err @board_type=mb @port=0
****
```

mb ϕ - 0e BCGA (addr)

```
-------------------------------------------------------------------
HARDERROR #          BOARD TYPE        PORT/SIDE      EXTRACTOR
-----------          ----------        ---------      ---------
        1            XS0_TYPE          0              xs0.send_par_err_mb1
===================================================================
```

Control parity error detected by MB1 from SP0 in XS0/1E NOTE - Address bit mapping is relative to the XBAR. addr<28> corresponds to addr<25> in the NMB.

xs0e_mb0.addr <

```
-------------------------------------------------------------------
addr<28:22> <21:15> <14:8> {<7:3> cycle<1:0>} wr_zone<3:0>
-------------------------------------------------------------------
***Data was not preserved in SP0***
-------------------------------------------------
00    00    00    00    0        xbar data
1     1     1     1     1        xbar ctl_par<0:4>
-------------------------------------------------
00    00    04    00    0        mb1 data
1     1     1*    1     1        mb1 ctl_par<0:4>
----------------
```

The xbar data is saved in lookaside registers in the xs1 and xs0. This is the data that the xbar sent to the nmb for this request. Any difference between the xbar data and the nmb data may indicate a connectivity problem between the xbar and the nmb.

```
-------------------------------------------------------------------
Scan fields:          nmb                    xbar
              --------------------------     --------------------------
              0000080 mbs1:bc4_sys_addr      0000000 xs0e:addr_lsd2[1]
                    0 mbs1:bc[4].sys.is_cycle      0 xs0e:cycle_lsd2[1]
                    0 mbs1:ise_sys.re_zone         0 xs1e:wr_zone_lsd2[1]
                   1f mbs1:mbe_ctl_par            1f xs1e:ctl_par_lsd2[1]
-------------------------------------------------------------------
xs0e:m_err_en<1> = 1
xs0e:send_par_err<1> = 1
+++>
spu> spu4000
Spu4000:
Detected Short or Stuck-at-1
Source Signal: sp_xse.addr          Source Port: 0
Sink Signal: xse_mb.addr            Sink Port: 1
Sink Field: mb1:bc4_sys_addr
Source Signal Bit Position: 5       Sink Signal Bit Position: 7
Expected: 00000000
Actual: 00000080
****
**AS THIS IS AN ADDRESS NET, THE ACTUAL BIT POSITION IS 10**

spu> xbar_err
xbar_err:
XBAR Error Logger/Identifier Version 0.7 (some testing done)
```

Halting system
System clocks halted
Checking xrte for errors
Checking xrto for errors
Checking xs0e and xs1e for errors
Checking xs0o and xs1o for errors

Note (1); Harderror 0 indicates the problem is on the even side and that the 0e nmc could be at fault.

Note (2): Harderror 1 indicates that the data (address) is from the xs0 board. Note the address bit assignment in this report. The bits involved in this failure are address bits <14..8> and parity bit <2> The data in the crossbar and the memory board are NOT the same; therefore, the problem should be on the memory board or between the xbar and memory board.

Note (3): spu4000 shows the failure to be xse_mb.addr<10>.

Note (4): The xbar_err script did not detect any errors. This would also indicate the problem is on the memory side of the crossbar.

Example Four

```
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    0            CU_TYPE              N/A          cu_ndat1_harderr
================================================================
        NCU harderror detected
        ndat[1].ndat_harderr = 1
        ------------------------------------------------------
        Due to an NDAT design error, the NDAT does not hold it's error state
        correctly.  The hard error has been detected and can be one of
        the following conditions:
        1) Parity error from the XBAR on the signals
                XSE_CU.WR_DAT<31..0>/XSE_CU.WR_PAR<3..0>
        2) Parity error from the NADR on the signals
                L1_WR_ODAT<7..0>/L1_WR_OPAR<3>
        3) Parity error from even comm register rams on signals
                CMR_RD_EDAT<31..0>/CMR_RD_EPAR<3..0>
+++>

<Tue Feb 23 14:49:37 1993>   logmsg(904):cu_ndat1_harderr:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=cu_ndat1_harderr @board_type=cu @port=-1
****

----------------------------------------------------------------
HARDERROR #     BOARD TYPE          PORT/SIDE     EXTRACTOR
-----------     ----------          ---------     ---------
    1           XS0_TYPE               0          xs0.send_par_err_ncu
================================================================
 Control parity error detected by NCU.  EVEN xbar notified and
 halted.  Check to make sure that the ODD xbar has also pulled
 a control parity error.  The NCU only checks parity on the ODD
 side, but it notifies the EVEN xbar when it pulls an ODD error.
 If there is no error on the ODD side, then there is a flaw in the
 error circuits on either the XS0E, the XS0O, or the NCU.
----------------------------------------------------------------
+++>
<Tue Feb 23 14:49:41 1993>   logmsg(930):xs0_send_par_err_ncu:0
Hard Error (DiagER349): Non-diagnostic system event
@extractor=xs0_send_par_err_ncu @board_type=xs0 @port=0
****

+++>
<Tue Feb 23 14:49:43 1993>   /diag/bin/hard_logger.exc(852):../hard.c:306
HardLog End (DiagER483): Hard_logger completion

****
```

```
spu> xbar_err

XBAR Error Logger/Identifier Version 0.7    (some testing done)

Halting system
   Checking xrte for errors
Error detected by xrte -- board has halted
   Checking xrto for errors
Error detected by xrto -- board has halted
   Checking xs0e and xs1e for errors
Error detected by xs0e and xs1e -- boards have halted
NCU detected parity error on data from NSP0
  Data  Par in XBAR:      0x00000000  0xf
  addr= 0000602  cycle= 0  wr_zone= 0  ctl_par= 17        -No Parity Check yet-
  Compare with NSP and NCU
   Checking xs0o and xs1o for errors
Error detected by xs0o and xs1o -- boards have halted
NCU detected parity error on data from NSP0
  Data  Par in XBAR:      0x00001999  0xb
  addr= 0000602  cycle= 0  wr_zone= 0  ctl_par= 17        -No Parity Check yet-
  Compare with NSP and NCU
```

Note (1): Harderror 0 reported the data as coming from XS EVEN.

Note (2): Harderror 1 reported that the even xbar was notified and halted.

Note (3): The xbar_err script reported the parity for the data and address as being good. This would indicate the problem is either on the NCU or a connectivity problem between the NCU and XBAR.

Note (4): In this case, the failure was found to be a short on the NCU augat connector (xse_cu.wr_data<25>).

NCU ↔ XS1E

1.4 Shorted Net Problems

Assume the net is "XC_SP.CU_STATUS". Note: Unless otherwise stated, the meterprobe point is on the NSP.

Remove NSP - if short disappears, suspect the NSP or NSP augat. In this situation, install a different nsp to determine if the problem is with the board or augat. If short continues to exist after removal of the orginal nsp,

remove XCL - if short disappears, the suspects would be the XCL or XCL augat. In this situation it would be best to isolate this with a spare xcl or augat. If short continues to exist,

remove epont from xbar backplane - if short disappears from NSP side, ohm the net on the xbar side. If the short exists on the xbar side, the problem is in the xcl augat or the xbar backplane. In this case, remove the xcl augat to isolate the problem. If the short continues to exist on the NSP side,

remove epont from the cpu backplane - If the short disappears from the NSP backplane, the problem is in the epont. If short continues to exist on the NSP side,

remove the nsp augat - if short continues to exist, suspect the cpu backplane.

(Note: If it is known that two nets are shorted together, refer to the net lists to see if they are next to each other on an augat or epont. This will most likely be the area at fault).

1.5 Open Net Problems

Again, assume the net is "XC_SP.CU_STATUS".

Once the open net is found, ensure that the open exists on both the xcl and the nsp backplanes.

a. If the net is "open only on the xcl side", then suspect the epont, epont seating, xbar backplane or the cpu backplane (the termination is always contained on the board that is listed second in a net).

1) At this point you should check continuity between nsp and xcl. Once you confirm that continuity does not exist, move the meter probe from the xcl augat to the xbar epont test point for this net.

2) If you obtain continuity between this point and the nsp test point, the xbar backplane would be suspect.

3) If the open continues to exist, remove the epont connector and meter the net between the two connectors on the epont. CAUTION: The epont connector is easily damaged by meter probes.

b. If the net is open on both the xcl and nsp test points, suspects would be the nsp, nsp seating, or nsp augat.

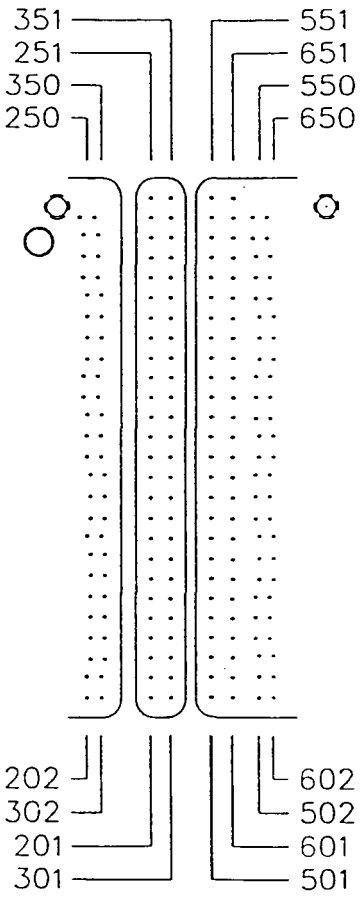
1) Reseat the nsp. If the problem disappears, this is either a seating problem or an intermittent augat problem.

2) If the problem continues to exist after reseating the board, replace the nsp with another nsp. If the open still exists, the main suspect would be the augat connector (though there is a possibility that the problem could be in the cpu backplane). If the open disappears, the problem is probably the nsp (there have been situations where an intermittent augat problem will disappear after the board has been reseated and come back at a later time).

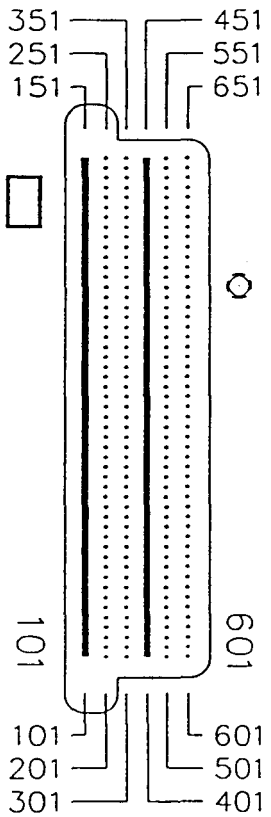
3) If it is believed to be an augat problem, remove, clean, and toruque to 14 inch pounds (torque sequence should be, starting from the center working out; 2, 4, 8, then 14 inch pounds). If problem continue to exist, replace the augat.

2.0 Backplane Connectors

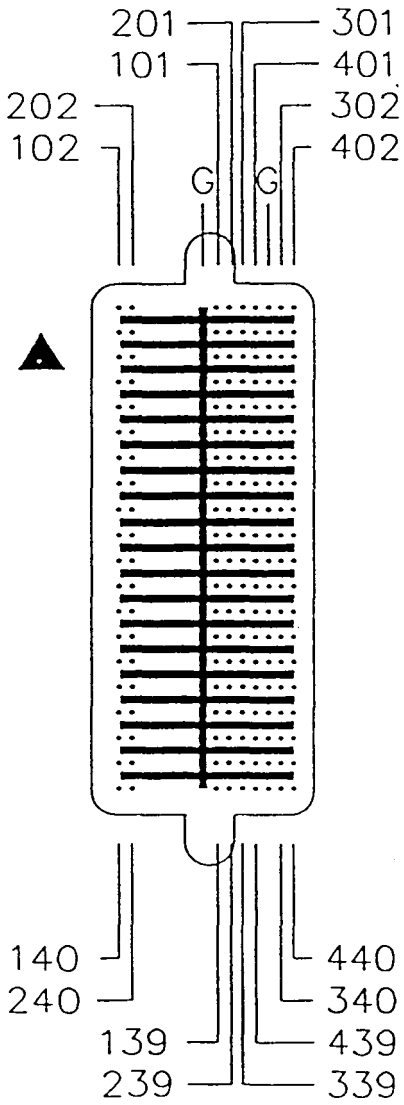
2.1 Xbar Epont Connector Pin out for Bays 1 and 2 (Bottom view).



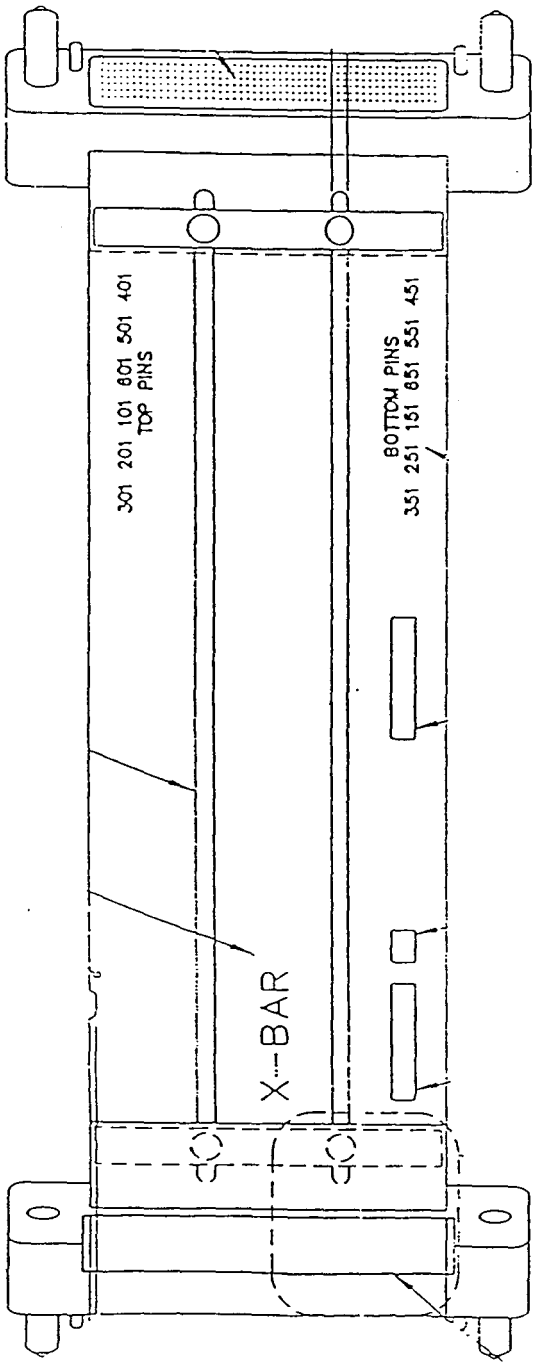
2.2 Xbar Epont Connector Pin out for Bays 0, 3 and 4 (Bottom view).



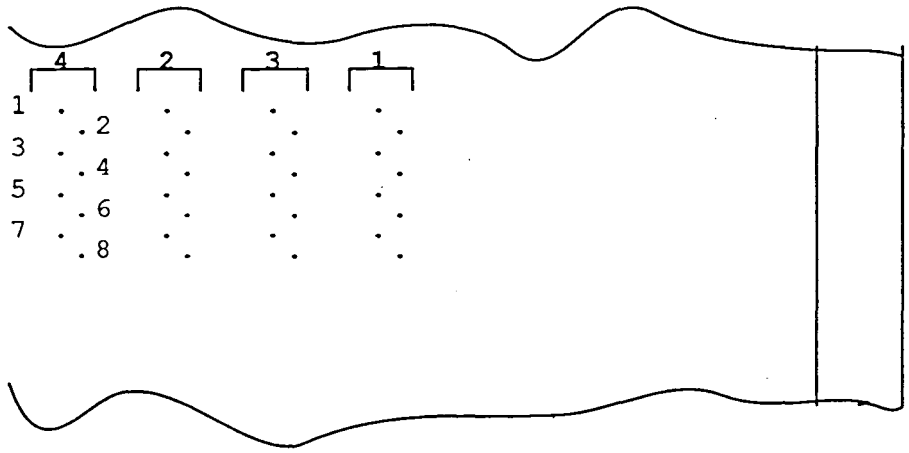
2.3 Xbar Augat Pin Outs (Bottom View).



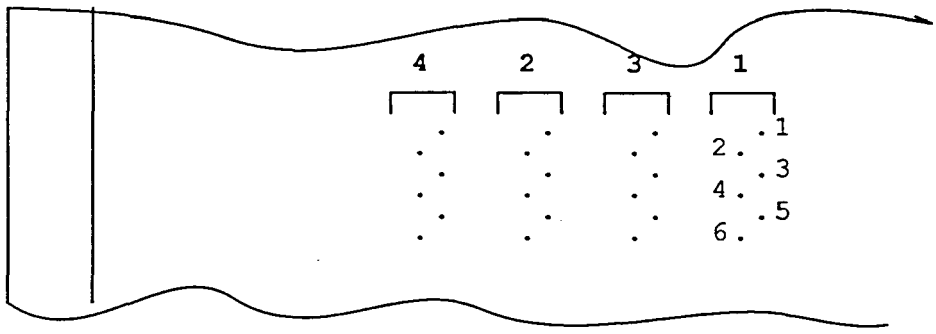
2.4 Epont Pin Out



2.6 Board via test points (board to backplane)

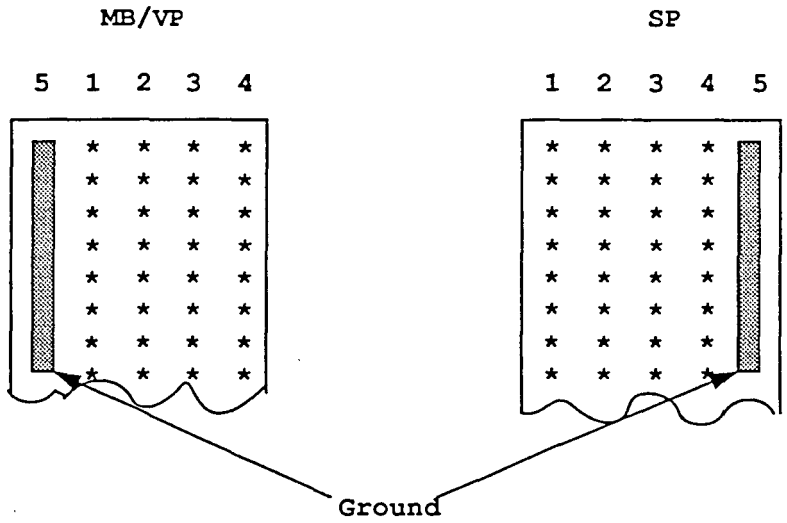


Left Facing Boards (NVP, NCU)
NON-GA Side View

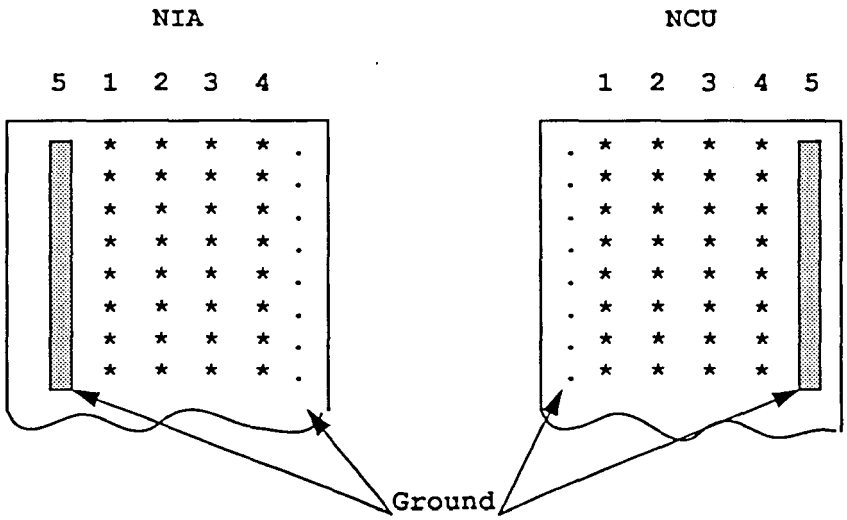


Right Facing Boards (NIA, NMB, NSP, XBAR Brds)
NON-GA Side View

2.7 CPU Backplane Augat Pin Outs



2.8 I/O Backplane Augat Pin Outs

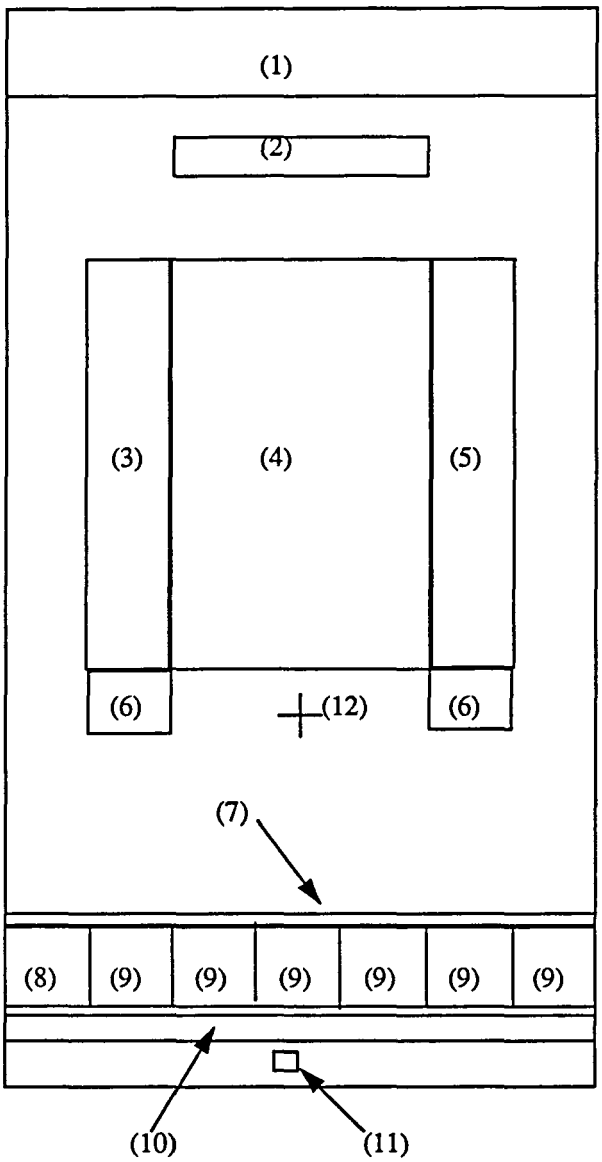


Appendix A

C3800 Series

I/O CABINET CABLING

I/O CABINET
FRONT VIEW



(1) FAN HOUSING

(2) DISTRIBUTION BRD.

(3) CCUBP, LEFT

(4) I/O BKPL

(5) CCUBP, RIGHT

(6) AIR FILTERS

(7) BAY POWER UNIT

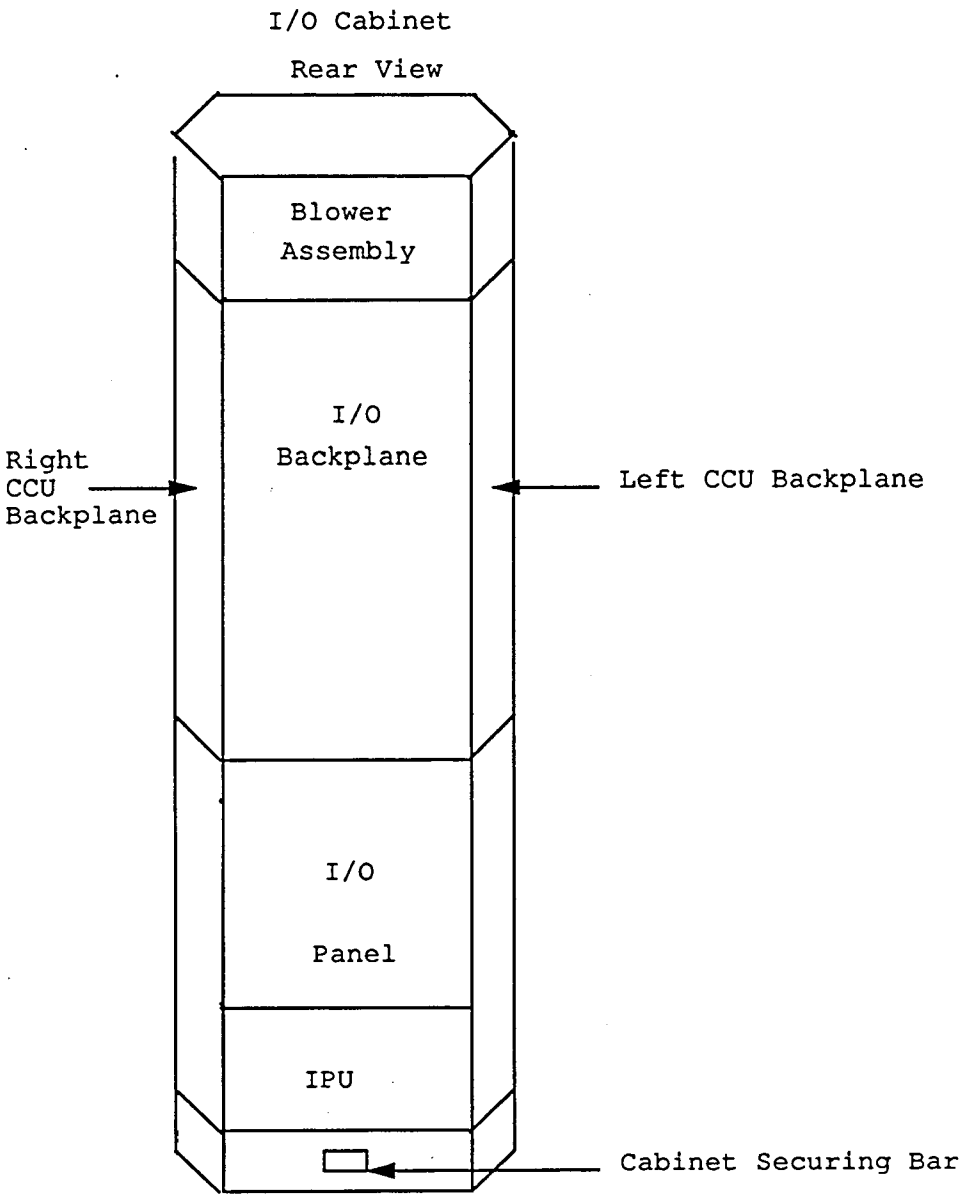
(8) BAY PWR CNTRL

(9) BAY PWR SUPPLIES

(10) AIR FILTER

(11) Lock Down Bolt

(12) Air Sensor


I/O Cabinet
Rear View
Blower
Assembly
I/O
Backplane
Right
CCU
Backplane
Left CCU Backplane
I/O
Panel
IPU
Cabinet Securing Bar

STANDARD IO PANEL CONFIGURATION
(Viewed from back of IO Bay)

DIFF. p2|p3|p4|p5 CCU39	VIOP p2y| p3y| p4y| p2x| p3x| p4x CCU35
DIFF. p2|p3|p4|p5 CCU38	DIFF. p2|p3|p4|p5 CCU34
DIFF. p2|p3|p4|p5 CCU37	DIFF. p2|p3|p4|p5 CCU33
VIOP p2y|p3y|p4y||p2x|p3x|p4x CCU36	DIFF. p2|p3|p4|p5 CCU32

I/O Cable Runs

BPU top left jacks (BPC):

J70	604-090003-200	I/O blowers and XBAR bulkhead J1
J71	604-600008-200	Front Distbrd J10
J72	**********	
J73	604-500020-200	Left Distbrd J3
J74	604-100008-200	XBAR CCABDIST BD J8

BPU back left jacks:

J10	603-010031-200	Jumper Interlock (1)
J11	604-210001-012	NCUPP J5 & J6 and Front Distbrd J4
J12	604-200001-200	Front Distbrd J7 and XBAR bulkhead J2 & J4
J20	604-210001-011	NIAPP J5 & J6 and Front Distbrd J3
J21	604-210001-011	CCULPP J5 & J6 and Front Distbrd J1
J30	(2)	
J31	(2)	

BPU back right jacks:

J40	(2)	
J41	(2)	
J50	(2)	
J51	(2)	
J60	604-210001-011	CCURPP J5 & J6 and Front Distbrd J8
J61	603-010031-200	Jumper Interlock (1)

(1) Each BPS has two output jacks (i.e., BPS 1 - J10 & J11; BPS 2 - J20 & J21; BPS 3 - J30 & J31, etc.). If one of these have a cable connected to it, the other one must have the Jumper Interlock installed. FOR THE I/O BAY, THE JUMPER MUST BE IN J10 & NUCPP TO J11.

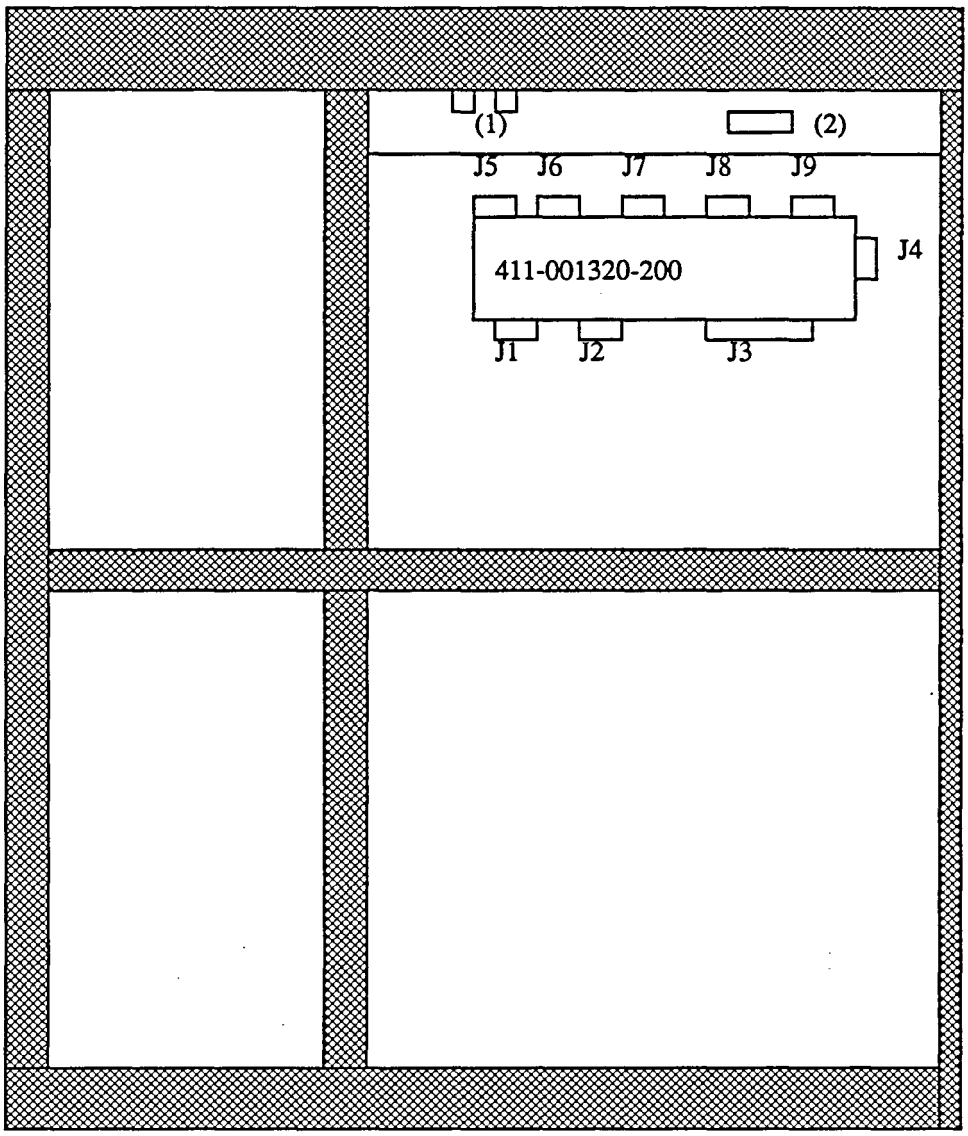
(2) Used as needed.

I/O Backplane

J84	Flex Cable	to XBAR BKPL J84
J83	Flex Cable	to XBAR BKPL J83
J82	Flex Cable	to XBAR BKPL J82
J81	Flex Cable	to XBAR BKPL J81
J25	601-100008-201	Left Distbrd J1
J24	604-100019-201	XBAR CCAB Shelf (See XBAR Cabling)
J23	Reserved for XPBUS	
J22	Reserved for XPBUS	
J21	Reserved for XPBUS	
J20	Reserved for XPBUS	
J16-19	604-640003-200	CCUBP Right J2-J5
J12-15	604-640003-200	CCUBP Left J2-J5
J11	604-080005-200	Ports 2 & 3 (Bay 1) J4 & J5
J10	604-080005-200	Ports 0 & 1 (Bay 0) J4 & J5
J9*	604-080005-200	I/O Backplane J4 & J5
J8	604-080005-200	XBAR Bkpl J4 & J5
J7	604-080005-200	Ports 6 & 7 (bay 3) J4 & J5)
J6	604-080005-200	Ports 4 & 5 (Bay 2) J4 & J5)
J5*	See J9 (A Connector)	
J4*	See J9 (B Connector)	

411-001320-200 (BD ASSY, OCABDIST_LEFT)

Located on the side panel of the left CCU cage.



(1) Blower jacks

(2) CCU air sensor (one on right ccu cage also)

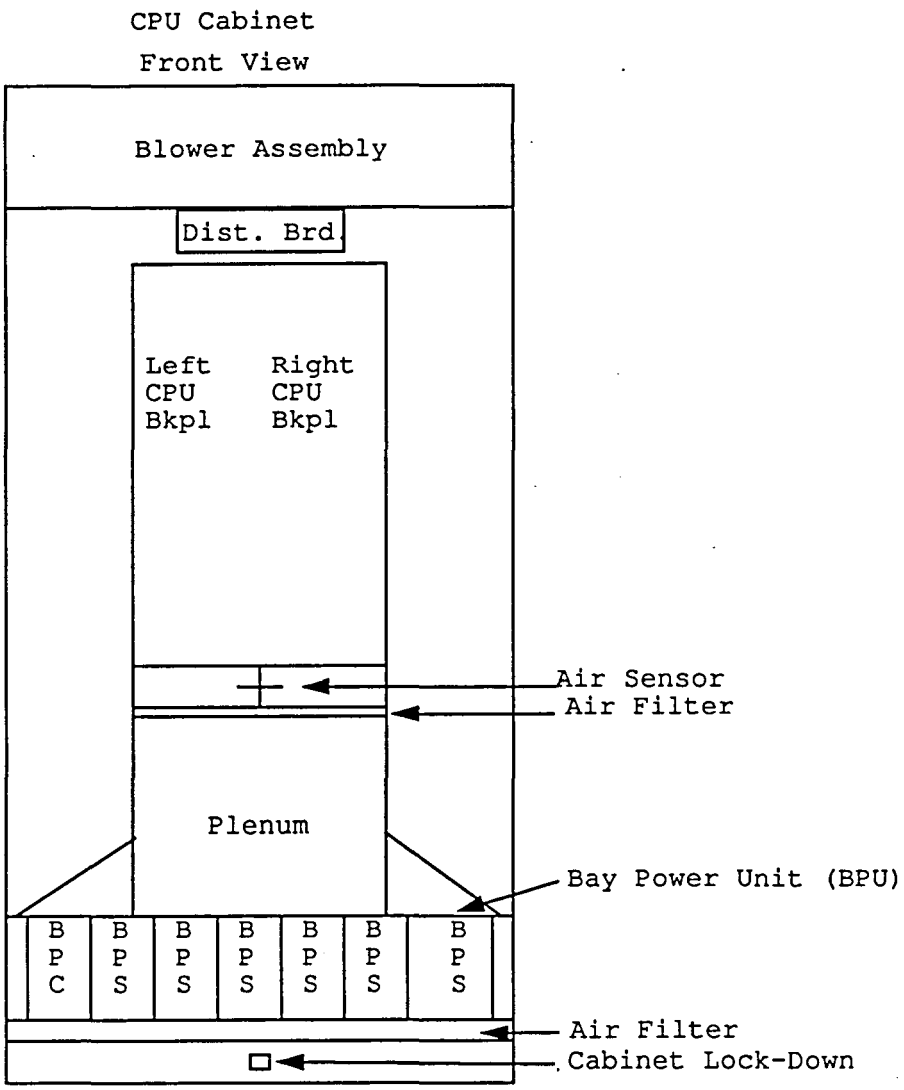
Cable runs from/to 411-001320-200 distribution board on I/O Cabinet.

Board Jack	Cable	Destination
J1	601-100008-201	To I/O backplane (J25)
J2	601-140002-201	To left CCU backplane (J1)
J3	604-500020-200	To BPC (J73)
J4	601-160003-200	To Front distbrd (J9)
J5	604-090002-200	To XBAR Cabinet Bulkhead J3
J6	603-040037-200	To left & right CCU air sensors
J7	603-100013-200	To I/O air sensor/two fan sensors
J8	601-140002-201	To CCU right backplane J1
J9	**Used only in CPU Cabinets**	

Appendix B

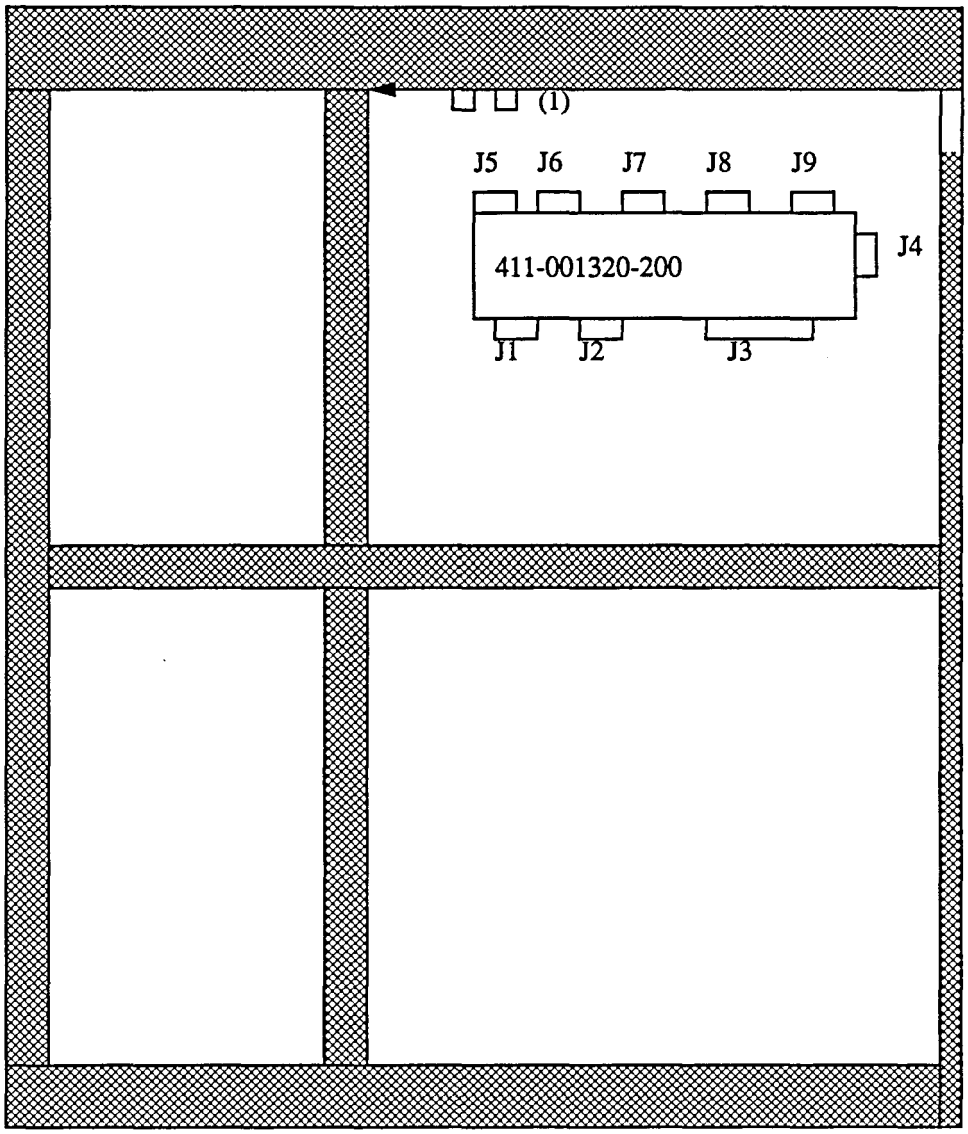
C3800 Series

CPU CABINET CABLING


CPU Cabinet
Front View
Blower Assembly
Dist. Brd.
Left CPU Bkpl
Right CPU Bkpl
Air Sensor
Air Filter
Plenum
Bay Power Unit (BPU)
BPC
BPS
BPS
BPS
BPS
BPS
BPS
Air Filter
Cabinet Lock-Down

411-001320-200 (BD ASSY, OCABDIST_LEFT)

Located (viewing cpu cabinet from front) on the left side of card cage.



(1) Blower Connectors

Cable runs from/to 411-001320-200 distribution board on CPU Cabinet.

Board Jack	Cable	Signal	Destination
J1	601-100008-201		CPU Bkpl_Left J6
J2	Not Used		
J3	604-500020-200		BPC (J73)
J4	601-160003-200		Front distbrd (J9)
J5	Not Used		
J6	Not Used		
J7	603-100013-200		Blower/Air Sensors (Top & Front)
J8	Not Used		
J9	601-100008-200		CPU Bkpl_Right J7

CPU BPC CABLES

Top Left Jacks

J70	604-060006-200	to	Cabinet Blowers
J71	604-600008-200	to	Front Distbrd J10
J72	Not Used		
J73	604-500020-200	to	Left Distbrd J3
J74	604-100008-200	to	Bay Assignment (See XBAR cables)

Back Left Jacks

J12	Jumper		
J10	604-210001-001	to	NMBPP
J11	No Connection		
J20	604-210001-001	to	NSPPP
J21	604-210001-002	to	NVPPP
J30	Spare		
J31	Spare		

Back Right Jacks

J40	Spare		
J41	Spare		
J50	604-210001-001	to	NVPPP
J51	604-210001-002	to	NSPPP
J60	604-210001-001	to	NMBPP
J61	Spare		

(1) Each BPS has two output jacks (i.e., BPS 1 - J10 & J11; BPS - J20 & J21; BPS 3 - J30 & J31, ETC.). If only one of these jack pairs have a cable connected to it, the other must have the Jumper Interlock (603-010031-200) installed.

(2) Used as needed.

CPU BACKPLANE CABLES

Note: The terms "right backplane" and "left backplane" refer to the cpu backplanes when viewed from the front of the bay.

Right Backplane

J5	604-080003-200	See I/O Backplane Cables.
J7	601-100008-200	from Left Distbrd J9.

Left Backplane

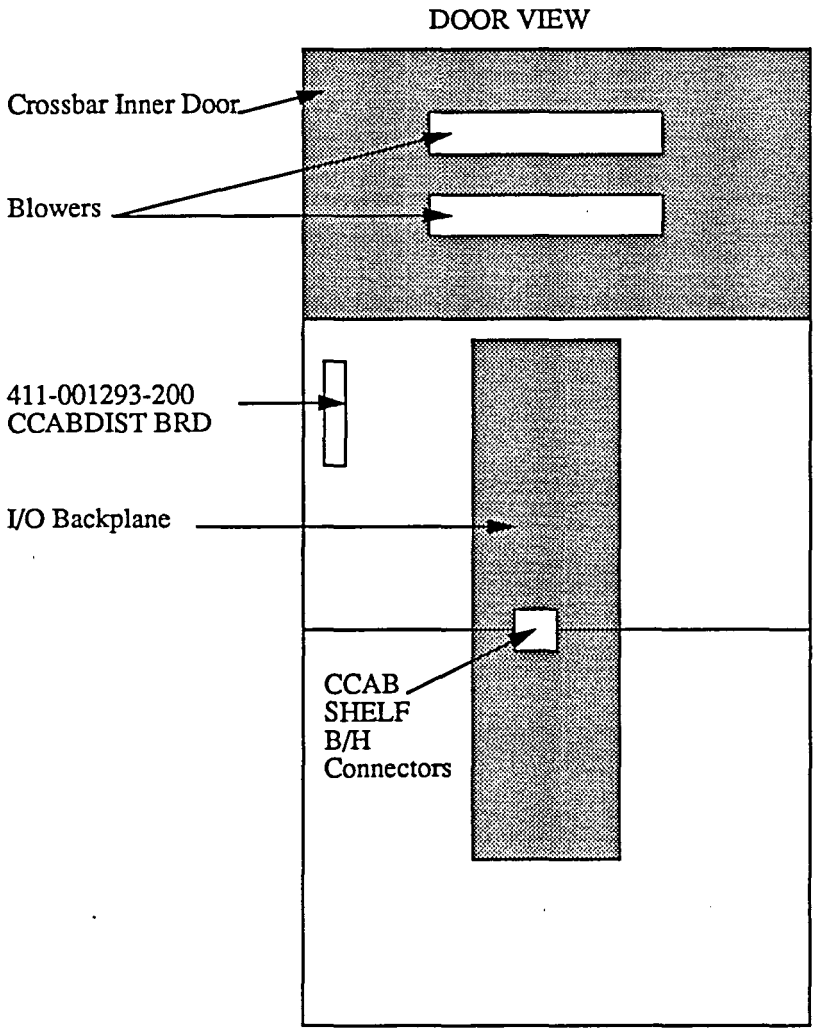
J4	604-080003-200	See I/O Backplane Cables.
J6	601-100008-201	from Left Distbrd J1.

Appendix C

C3800 Series

CENTRAL CABINET CABLING

XBAR CABINET

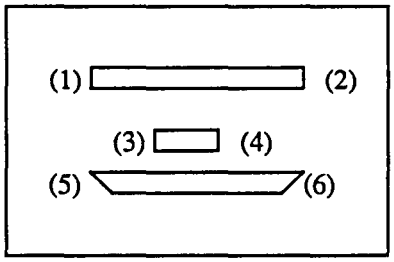


CABLE ROUTING

411-001293-200 CCABDIST Board

J1	602-500006-201	to CCAB SHELF B/H (5)
J2	603-020058-200	to CCAB SHELF B/H (3)
J3	500-000416-202	Keylock
J4	604-100008-200	to Bay 3 BPC J74
J5	604-100008-200	to Bay2 BPC J74
J6	604-100008-200	to Bay4 BPC J74
J7	604-100008-200	to Bay1 BPC J74
J8	604-100008-200	to Bay0 BPC J74
J9	500-000416-202	Keylock

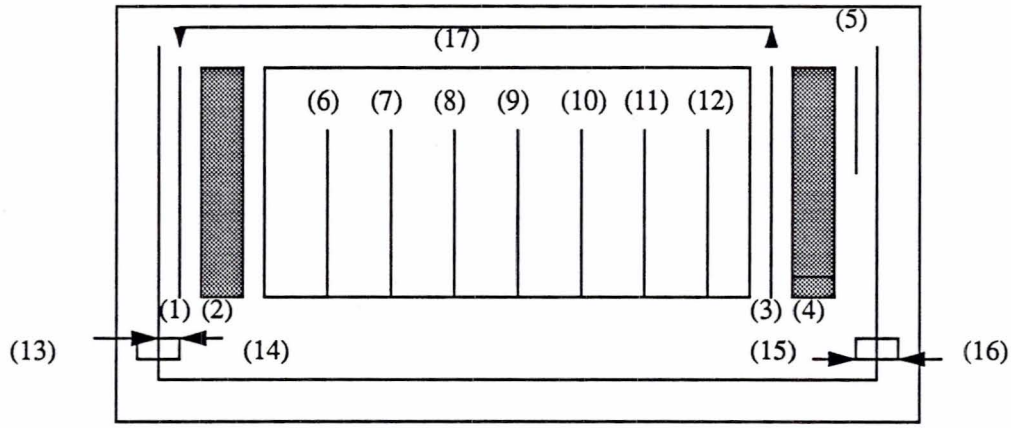
CCAB SHELF B/H



(1) 604-100012-200 to I/O Backplane J24
(2) 604-100019-201 to SPU SWIP Jack
(3) 603-020058-200 to CCABDIST Board J9
(4) 603-020004-200 to Peripheral Power Controller
(5) 604-500006-201 to CCABDIST Board J1
(6) 602-500006-201 to SPU BPC Jack

XBAR CARD CAGE

Door View

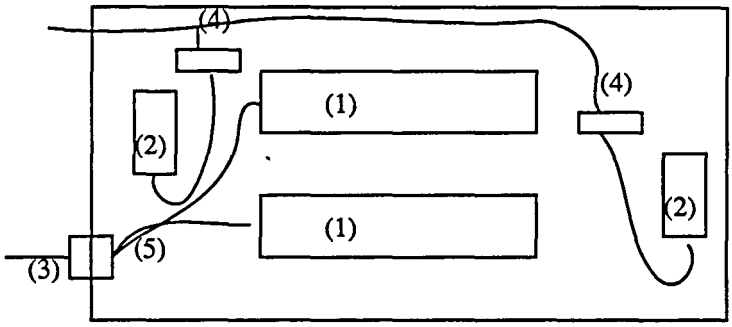


(1) 411-001308-200 (XPB)
(2) 500-000500-200 Power Supply (Includes 411-001308-200)
(3) 411-001308-200 (XPB)
(4) 500-000500-200 Power Supply
(5) 411-001285-200 (XPC)
(6) XRT Even
(7) XS1 Even
(8) XS0 Even
(9) XCL
(10) XS0 Odd
(11) XS1 Odd
(12) XRT Odd
(13) 604-060007-200 to XBAR B/H (See page 5)
(14) To Power Supply (2)
(15) To Power Supply (4)
(16) 604-060007-200 to XBAR B/H (See page 5)
(17) Buss Bars

Sensa ϕ = XPB
1 = XRTE
2 = XSϕE
3 = XS1E
4 = XCL
5 = XS1O
6 = XSϕO
7 = XRTO

XBAR BLOWER DOOR

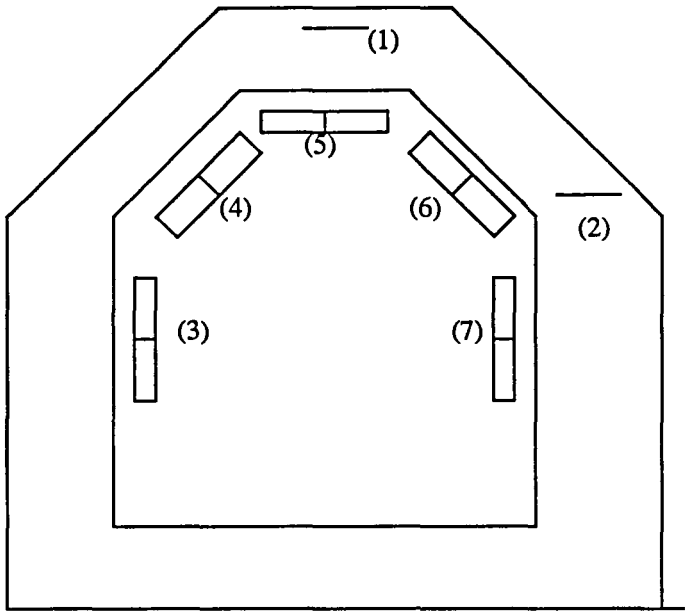
Inside View



(1) XBAR Cabinet Fans
(2) Capacitors
(3) 603-080025-200 to XBAR B/H (see page 5)
(4) 604-030003-200 to XBAR B/H (see page 5)
(5) 603-060044-200 to Fans

XBAR Backplane Connector Locations

Top View

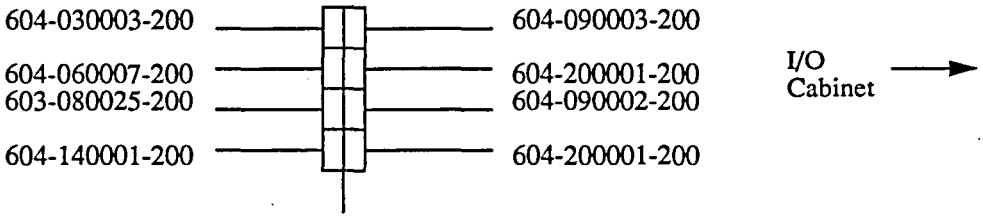


(1) Air Sensor (411-000119-201)

(2) XBAR B/H (see below)

(3) Flex connector from Bay 3 (J61, 62, 63, and J71, 72, 73)
(4) Flex connector from Bay 2 (J41, 42, 43, and J51, 52, 53)
(5) Flex connector from I/O Cabinet (J81, 82, 83, 84)
(6) Flex connector from Bay 1 (J21, 22, 23, and J31, 32, 33)
(7) Flex connector from Bay 0 (J01, 02, 03, and J11, 12, 13)

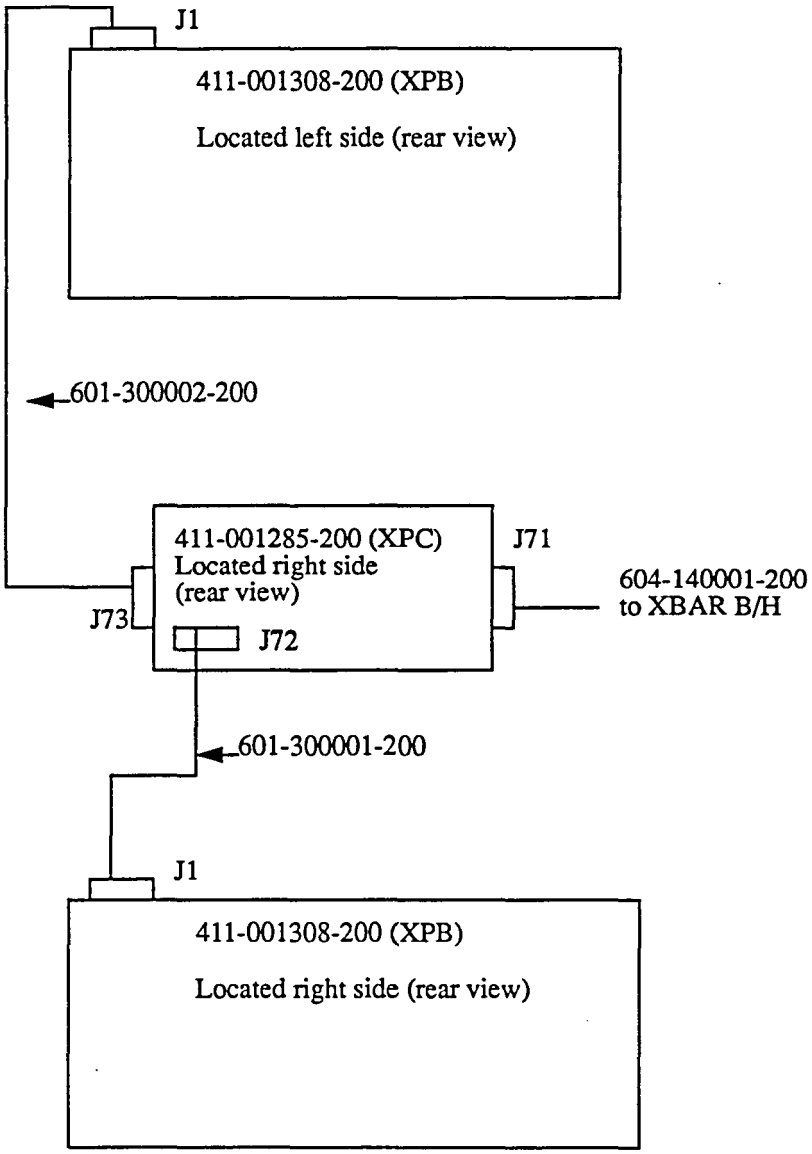
XBAR B/H

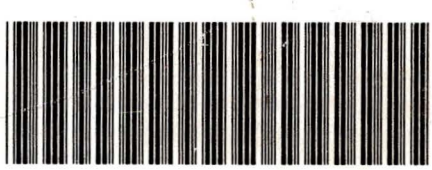


CABLE ROUTING FROM XBAR B/H

From XBAR B/H	To
604-030003-200	XBAR Blowers
604-060007-200	XBAR Power Supplies
603-080025-200	Back XBAR Air Sensor (411-000119-201) Blower Sensor Boards Front XBAR Air Sensor (411-000119-201)
604-140001-200	XPC Board (J71)

XBAR Backplane Power Boards





Document Number
710-027130-000