Britton Lee Technical Publications

BL700 MAINTENANCE MANUAL

(4.0)

October 1987
Part Number 201-1210-004

First Printing, February 1985
Second Printing, June 1986
Third Printing, February 1987
Fourth Printing, October 1987

This document supersedes all previous documents.

The information contained within this document is subject to change without notice. Britton Lee assumes no responsibility for any errors which may appear in this document.

The software and hardware described in this document are furnished under license and may only be used or copied in accordance with the terms of such license.

TABLE OF CONTENTS

GENERAL INTRODUCTION 1-1

INTRODUCTION 1-3
BL700 SYSTEM ARCHITECTURE 1-5

BL700 SYSTEM OVERVIEW 2-1

INTRODUCTION TO THE SYSTEM OVERVIEW 2-3
GENERAL 2-3
SPECIFICATIONS 2-3
LOGIC GROUND CONNECTIONS 2-3
MAIN ASSEMBLY 2-3
AC POWER DISTRIBUTION UNITS 2-11
INTERNAL AC POWER DISTRIBUTION 2-18
DC POWER SUPPLIES 2-19
MOTHERBOARD 2-20
FRONT PANEL 2-22
INTRODUCTION 2-22
MODES OF OPERATION:
REMOTE 2-22
FUNCTION KEY SWITCH 2-22
FRONT PANEL LED STATUS 2-22
CONTROL PANEL 2-25
PRINTED CIRCUIT BOARD (PCB) OVERVIEW 2-26
GENERAL 2-26
PCB LAYOUT 2-26
DATABASE PROCESSOR 6 MHz 2-27
TIMING AND CONTROL BOARD 2-28
MEMORY BOARD 2-29
SMD DISK CONTROLLER 2-30
TAPE CONTROLLER 2-30
DATA ACCELERATOR 2-31
CHANNEL I/O BOARDS 2-32
SERIAL ASYNCHRONOUS RS-232 I/O BOARD (SI/O) 2-32
PARALLEL IEEE-488 CHANNEL BOARD (PI/O) 2-33
ETHERNET CHANNEL BOARD 2-34
BLOCK MULTIPLEX 2-34

PREVENTIVE MAINTENANCE 3-1

INTRODUCTION 3-3
SCHEDULE 3-3

DIAGNOSTICS 4-1

INTRODUCTION 4-3
POWER ON SELF-DIAGNOSTICS AND START-UP SEQUENCE 4-5
INTRODUCTION 4-5
SYSTEM STARTUP (AC) POWER) 4-5
BL700 MAINTENANCE MODE 4-5
BL700 SAFE MODE 4-7
BL700 RUN MODE 4-9
RETURNING TO MAINTENANCE MODE 4-9
SCHEDULED SHUTDOWN 4-9
UNSCHEDULED SHUTDOWN 4-9

SERIAL AND PARALLEL SELF-DIAGNOSTICS 4-11
INTRODUCTION 4-11
SHARED MEMORY TEST 4-11
TIMING AND CONTROL SELF-DIAGNOSTICS 4-15
INTRODUCTION 4-15
BITS 4-15
DBP LEDS 4-17
INTRODUCTION 4-17
RUN 4-17
LMEN 4-17
BGRN 4-17
CGRN 4-17
NMI 4-17
FAIL 4-17
DISK CONTROLLERS AND TAPE CONTROLLERS POWER ON SELF-DIAGNOSTICS 4-19
INTRODUCTION 4-19
IDL MODE AND POWER-ON DIAGNOSTICS 4-19
MEMORY INTERFACE DIAGNOSTICS 4-19
TROUBLESHOOTING ETHERNET HARDWARE 4-21
INTRODUCTION 4-21
TEST PREREQUISITE 4-21
ETHERNET ADDRESS CONFIGURATION 4-22
IDM ETHERNET DIAGNOSTICS 4-22
EXPLANATION OF TEXT STEPS 4-24
LOOPBACK FOR SERIAL CHANNEL 4-29
TEST PREREQUISITES 4-29
TEST PROCEDURE 4-29
POWER SUPPLY TROUBLESHOOTING 4-31
INTRODUCTION 4-31
PROBABLE CAUSES 4-32
INITIAL POWER-ON SEQUENCE 4-33
WHEN YOU NEED ASSISTANCE 4-35

UTILITIES 5-1

UTILITIES AND DIAGNOSTIC PROGRAMS 5-3
GENERAL 5-3
ERROR MESSAGES 5-3
ILLUSTRATION CONVENTIONS 5-3
DEFINITION OF COMMON TERMS 5-3
DISK PAGE FORMAT 5-7
A REFERENCE MODEL IDM 5-9
GETTING READY TO USE ONE OF THE PROGRAMS 5-10
DISK FORMATTING UTILITY (DFU) 5-11
INTRODUCTION 5-11
IDM DISK ORGANIZATION 5-11
USAGE 5-12
STARTING DFU 5-12
DFU MODES 5-13
EXIT 5-13
SYSFORMAT 5-13
SYSCREATE 5-15
SYSDESTROY 5-16
MERGE 5-16
VALIDATE 5-17
LIST 5-18
FORMAT 5-18
UNFORMAT 5-18
CONFIGURE 5-19
SCAN 5-20
SOFT ERRORS 5-20
HARD ERRORS 5-20
MARKDB 5-22
RENUMBER 5-23
MIRROR OPERATION: 5-23
MIRRORFORMAT 5-23
MIRRORCOPY 5-24

COMPARE 5-24
PROMOTE DISK 5-25
DFU USAGE EXAMPLES 5-25
DISK DIAGNOSTICS (DD) 5-29
OVERVIEW 5-29
GETTING READY FOR DD 5-29
STARTING AND USING DD 5-30
DC DIRECT COMMAND DESCRIPTIONS 5-36
DC INDIRECT COMMAND DESCRIPTIONS 5-38
TAPE DIAGNOSTICS (TD) 5-47
OVERVIEW 5-47
LOOPBACK TEST 5-47
TD OPERATION 5-47
TPC COMMAND MENU 5-48
TPC DIRECT COMMANDS 5-54
TPC INDIRECT COMMANDS 5-53
TPC SPECIAL COMMANDS 5-58
CHECK DATABASE (CKDB) 5-61
INTRODUCTION 5-61
STARTING CKDB 5-61
CKDB PROMPTS 5-62
INITIAL OPERATION 5-62
CKDB CONVENTIONS 5-63
OPTIONS 5-64
OPTION LIST DISCUSSION 5-64
BOOLEAN VS NUMERIC OPTIONS 5-65
OPTION DESCRIPTIONS 5-65
CKDB COMMAND FUNCTIONS 5-69
ERRORS 5-83
EXAMINATION SEQUENCE 5-84

REMOVAL AND REPLACEMENT 6-1

INTRODUCTION 6-3
OPENING THE IDM (INSTALLED IN CABINET) 6-3
FRONT BEZEL ASSEMBLY 6-3
CONTROL PANEL 6-3
POWER SUPPLY (multi) 6-3
POWER SUPPLY (Bulk + 5V FCC) 6-4
TOP REAR PANEL 6-4
CARD CAGE AREA SIX INCH FANS 6-4
DEFECTIVE CABLES (internal) 6-4
T1 STEPDOWN TRANSFORMER 6-4
THERMAL SENSOR 6-5
LINE FILTER AC 6-5
(AC) POWER SWITCH / CIRCUIT BREAKER 6-5
(AC) POWERCORD AND PLUG 6-5
REPLACEMENT OF PRINTED CIRCUIT BOARDS 6-5
SERIAL DISTRIBUTION BOX 6-6
REPLACING SERIAL CABLE 6-6
REPLACING MOTHERBOARD 6-6
SERIAL DISTRIBUTION RACK 6-6
MEMORY CHIPS 6-7

APPENDIX A: CONSOLE COMMANDS A-1

INTRODUCTION A-3
CONSOLE COMMANDS A-3
conport <baud> or maintport <baud> A-3
list A-3
listtape A-4
load A-4
load <filename> A-4
loadtape A-4
loadtape <filename> A-4
loopback <slot number> A-4
kernal A-5

kernal ckdb or dfu A-5
kernal fileload A-5

kernal fileload -t A-5
kernal <filename> A-5
<return> A-5
slots A-5
td/dd A-6
version A-6

APPENDIX B: PERIPHERAL DEVICE STATUS & REPORTS B-1

INTRODUCTION B-3
LIST OF DEVICES B-3

APPENDIX C: SOFTWARE LOADING PROCEDURE C-1

INTRODUCTION C-3
LOADING FROM THE DSC-3 ..... C-3
SYSTEM SOFTWARE LOADING PROCEDURE (dsc-3) C-3
LOADING FROM THE IBM PC C-5
SYSTEM SOFTWARE LOADING PROCEDURE (IBM PC) C-5
LOADING FROM IDM TAPE C-8
SYSTEM SOFTWARE LOADING PROCEDURE (IDM Tape) C-8
LOADING FROM HOST TAPE C-10
SYSTEM SOFTWARE LOADING PROCEDURE (Host Tape) C-10

APPENDIX D: REPAIR PARTS LIST D-1

IDM BOARD PARTS ORDERING PROCEDURE D-3
REPAIR PARTS LIST D-5

APPENDIX E: WIRE LISTS E-1

IDM FCC WIRE CONNECTIONS (AC & DC) E-3
FCC MOTHERBOARD BUS BAR E-4
PS2 WIRING E-5
PS1 WIRING E-6
J-9 WIRING E-7
IDM FCC MOTHERBOARD RESISTOR PACK SCHEMATIC E-9
FCC MOTHERBOARD SLOT WIRING E-12
DATABASE PROCESSOR CONNECTOR PINOUTS E-19
INTRODUCTION E-19
CONNECTOR J-4 PINOUTS E-19
CONSOLE PORT PINOUTS J-5 ..... E-20
CONSOLE TERMINAL CABLE E-21
MAINTENANCE PORT J-6 ..... E-21
MAINTENANCE CABLE E-21
ADAPTER CABLE E-21
DISK CONTROLLER CONNECTOR AND CABLE DETAILS E-22
INTRODUCTION E-22
"A" CABLE PINOUT E-22
"B" CABLE PINOUTS J5, J6, J7, J8 ..... E-23
TAPE CONTROLLER READ WRITE CABLES E-24
WRITE CABLE PINOUTS J-1 ..... E-24
READ CABLE PINOUTS J-2 ..... E-25
SERIAL INTERFACE (RS-232) PINOUTS J1 ..... E-26
IEEE-488 HOST INTERFACE, GPIB STANDARD E-28
ETHERNET CONNECTOR PINOUTS E-29
ETHERNET CONSOLE CABLE TO J1 ..... E-29
ETHERNET INTERNAL TRANSCEIVER CABLE TO J2 ..... E-29
BLOCKMUX BOARD CONNECTOR PIN ASSIGNMENT E-30
BLOCKMUX INTERFACE IDM BUS AND TAG CABLES E-31

APPENDIX F: PRE-FCC DIFFERENCES F-1

INTRODUCTION F-3
BACK INTERFACE PANEL F-5

PRE-FCC AC/DC WIRE LIST F-9
POWER SUPPLIES F-11
FRONT PANEL F-17
MOTHERBOARD F-18
PRE-FCC MOTHERBOARD SLOT WIRING F-20

APPENDIX G: ERROR CODES G-1

TRAP MESSAGES G-3
DISK ERROR CODES G-7
SOFT (RECOVERABLE) AND HARD (NONRECOVERABLE) DISK READ ERRORS .. G-8
SOFT AND HARD DISK ERRORS G-9
TAPE ERROR CODES G-15
112 Tape Unavailable G-15
113 Tape Hard Errors G-17
114 Tape Caution Errors G-20
DATA ACCELERATOR (DAC) ERRORS G-21
CHANNEL SYSERR MESSAGES G-25
TAPE CONTROLLER LOOPBACK ERROR CODES G-29
CHANNEL SYSERRS FOR BLOCKMUX G-33
CHANNEL SYSERRS FOR ETHERNET G-35
DISK FORMATTING UTILITY (DFU) ERROR MESSAGES G-37
CKDB ERROR MESSAGES G-41
CKDB ERROR MESSAGE FORMAT G-41
CORRECTABLE ERRORS G-41
HARMLESS ERRORS G-44
USER CORRECTABLE ERRORS G-45
SEVERE ERRORS G-50
FATAL ERRORS G-54

LIST OF FIGURES:

Figure 2-1. IDM ACCESS PANELS 2-4
Figure 2-2. IDM REAR PANEL 2-6
Figure 2-3. IDM REAR VIEW 2-7
Figure 2-4. SERIAL INTERFACE 2-8
Figure 2-5. IDM CONFIGURATION 2-9
Figure 2-6. BLOCKMUX INTERFACE 2-10
Figure 2-7. MODEL 20801 FRONT VIEW 2-11
Figure 2-8. MODEL 20801 REAR VIEW 2-12
Figure 2-9. MODEL 20801 BLOCK DIAGRAM 2-13
Figure 2-10. MODEL 872C 001 FRONT VIEW 2-14
Figure 2-11. MODEL 872C 001 REAR VIEW 2-15
Figure 2-12. MODEL 872C 001 BLOCK DIAGRAM 2-15
Figure 2-13. MODEL 87201 FRONT VIEW 2-16
Figure 2-14. MODEL 87201 REAR VIEW 2-17
Figure 2-15. MODEL 87201 BLOCK DIAGRAM 2-17
Figure 2-16. AC INPUT CIRCUITRY 2-18
Figure 2-17. MOTHERBOARD 2-21
Figure 2-18. FRONT PANEL 2-24
Figure 2-19. CONTROL PANEL 2-25
Figure 2-20. DATABASE PROCESSOR 6 MHz 2-27
Figure 2-21. TIMING AND CONTROL BOARD 2-28
Figure 2-22. MEMORY BOARD 2-29
Figure 2-23. SMD DISK CONTROLLER 2-30
Figure 2-24. TAPE CONTROLLER 2-30
Figure 2-25. DATA ACCELERATOR 2-31
Figure 2-26. SERIAL ASYNCHRONOUS RS-232 I/O BOARD (SI/0) 2-32
Figure 2-27. EXPLODED VERSION OF LED 2-32
Figure 2-28. PARALLEL IEEE-488 CHANNEL BOARD (PI/0) 2-33
Figure 2-29. EXPLODED VERSION OF LED 2-33
Figure 2-30. ETHERNET CHANNEL BOARD 2-34
Figure 2-31. BLOCK MULTIPLEX 2-34

Figure 4-1. ETHERNET FEATURE 4-26
Figure 4-2. ETHERNET CONNECTION 4-27

Figure 5-1. DISK DIAGNOSTIC MEMORY BUFFERS 5-4
Figure 5-2. COMMAND BLOCK 5-5
Figure 5-3. COMMAND BLOCK STRING 5-6
Figure 5-4. MASTER BLOCK 5-7
Figure 5-5. BLOCKALLOC 5-8

Figure 6-1. IDM CHASSIS (FRONT) 6-8
Figure 6-2. IDM CHASSIS (REAR) 6-9

Figure C-1. IDM PORTS DSC-3 ..... C-12
Figure C-2. IDM PORTS PC C-13
Figure C-3. IDM PORTS HOST TAPE C-14

Figure D-1. BLI MANUFACTURING LABEL D-3

Figure E-1. MOTHERBOARD BUS E-4
Figure E-2. MOTHERBOARD E-8
Figure E-3. RESISTOR PACK SCHEMATIC E-9
Figure E-4. TTL LINE E-10
Figure E-5. ECL LINE E-11
Figure E-6. IDM 208V FCC CHASSIS E-32
Figure E-7. IDM CHASSIS WIRING OPTIONS E-33
Figure E-8. IDM CHASSIS WIRING OPTIONS E-34

Figure F-1. PRE-FCC REAR PANEL F-5
Figure F-2. FCC REAR PANEL F-6
Figure F-3. PRE-FCC POWER AREA F-7
Figure F-4. PRE-FCC ELECTRICAL CAUTION AREAS F-8
Figure F-5. PRE-FCC SYSTEM INTERCONNECT CHASSIS F-10
Figure F-6. PS2 ..... F-11
Figure F-7. PS1 ..... F-12
Figure F-8. PRE-FCC TM 11 WIRE POSITIONS F-13
Figure F-9. PRE-FCC TM 11 HOOK-UP F-14
Figure F-10. PRE-FCC TM 11 HOOK-UP (Power Supply and Circuit Breaker Connections) ... F-15
Figure F-11. PRE-FCC TM 11 HOOK-UP (Motherboard Connections) F-16
Figure F-12. PRE-FCC FRONT PANEL F-17
Figure F-13. MOTHERBOARD F-19

GENERAL INTRODUCTION

Introduction

BL700 System Architecture

INTRODUCTION

This third version of the BL700 Maintenance Manual, #201-1210-004, replaces previous editions and specifies the maintenance requirements for the BL700 database server. The scope of this manual includes general preventive maintenance procedures, self-diagnostic features, and the removal and replacement of components, making it relatively simple to efficiently maintain the database server.

The first section of the manual is the OVERVIEW of the database server's basic specifications and components. Power requirements, I/O channels, and printed circuit boards are described here.

Next, the PREVENTIVE MAINTENANCE section describes procedures for general physical maintenance of the database server including scheduled maintenance.

DIAGNOSTICS covers power on self-diagnostics and start-up sequence once the database server is in maintenance mode. Board self-diagnostics are provided for serial and parallel interfaces, timing and control, database processor LEDs, and disk and tape controllers. Also included are procedures for troubleshooting ethernet hardware, serial channel loopback, and troubleshooting the power supply.

The disk formatting utility (DFU), disk diagnostics (DD) and tape diagnostics (TD), and the check database utility (CKDB) are discussed in the UTILITIES section.

REMOVE AND REPLACE reviews the database server external and internal components. This section includes the removal and replacement of components beginning with the front bezel and the front panel. Once inside the IDM chassis, the power supply, fans, cables, transformer, thermal sensor, line filter (AC), power switch (AC), power-cord and plug (AC), printed circuit board (PCB), RS232 serial distribution box, motherboard, and serial distribution rack can be easily removed and replaced if each procedure is correctly followed.

The final section contains the following APPENDICIES: console commands (Appendix A), peripheral devices and status reports (Appendix B), software loading procedure (Appendix C), repair parts list (Appendix D), wire lists (Appendix E), pre-FCC differences (Appendix F), and lastly error messages and error codes (Appendix G).

NOTE

Additional Britton Lee publications that can be used with this manual are:

BL700 Installation Manual, #200-1077-xxx
BL700 Operation Manual, #201-1078-xxx

BL700 SYSTEM ARCHITECTURE

IDM Base Configuration	• 16 slot chassis, front panel and power supply • Database processor (1 slot) • Memory timing and control (1 slot) • 1M byte main memory (1 slot) • Disk controller (1 slot, supports up to 4 drives with SMD interface)
Options	• 1M byte main memory (1 slot, up to 6M bytes total) • Disk controller (1 slot, up to 4 controllers total, each controller supports up to 4 drives with SMD interface) • Tape controller (1 slot, supports up to 8 drives with Pertec formatter • IEEE-488 host interface (1 slot, supports up to 8 hosts, asynchronous, 19.2k baud maximum) • Ethernet (1 slot); 10M bit communication scheme used for local networking • Mirrored Disk Option supports up to 8 removable or fixed drives that can be mirrored or duplicated; resulting in a total of 16 separate drives.
BL700 Base Configuration	• 16 slot chassis, front panel and power supply • Database processor (1 slot) • Database accelerator (1 slot) • Memory timing and control (1 slot) • 2M byte main memory (1 slot) • Disk controller (1 slot, supports up to 4 drives with SMD interface) • RS232C or IEEE-488 host interface (1 slot, supports up to 8 hosts)
Options	• 1M byte main memory (1 slot, up to 6M bytes total) • Disk controller (1 slot, up to 4 controllers total, each controller supports up to 4 drives with SMD interface) • Tape controller (1 slot, supports up to 8 drives with Pertec formatter • IEEE-488 host interface (1 slot, supports up to 8 hosts, asynchronous, 19.2k baud maximum) • Ethernet (1 slot); 10M bit communication scheme used for local networking • Mirrored Disk Option supports up to 8 removable or fixed drives that can be mirrored or duplicated; resulting in a total of 16 separate drives.

BL700 SYSTEM OVERVIEW

Overview

AC Power Distribution Units

Internal AC Power Distribution

DC Power Supplies

Motherboard

Front and Control Panels

Printed Circuit Boards

OVERVIEW

INTRODUCTION TO THE SYSTEM OVERVIEW

This section presents an overview of the BL700 basic specifications and components. The System Overview covers a general description of power requirements, I/O channels, and printed circuit boards. Diagrams are included to assist the trained technician.

GENERAL

The BL700 has a common bus and integral switching power supply (bulk and multi). Several printed circuit boards with on board RAM (Random Access Memory) circuits are self-testing and microprocessor based. These particular boards process I/O, tape, disk, and host interface commands and data. Printed circuit board cables are internally connected and route to the back I/O panel. Cooling is provided by three to four fans in the back panel, depending on the model.

The BL700 has special automatic monitoring features that provide immediate status information. This information is displayed on the front panel indicator lights. It is then printed through the console port to a terminal.

SPECIFICATIONS

Input voltage:	200 - 240V ac
Input frequency:	47 - 63 Hz
Power consumption:	1600 Watts maximum
BTU:	6500 per hour
Weight:	170 pounds or 77.1 kilograms
Chassis:	16 slots
Memory:	6M bytes maximum
Dimensions:	172.5 x 107.2 x 113.03 inches or 67.9 x 42.2 x 44.5 centimeters

LOGIC GROUND CONNECTIONS

The BL700 comes from the factory with logic ground connected to internal chassis ground.

MAIN ASSEMBLY

Front Bezel

The BL700 assembly contains a die cast aluminum Front Bezel (see Figure 2-1). Horizontally across the bezel front, there are four ventilation slots.

Front Panel Assembly

The front panel is in the Front Bezel on the right side. The front panel has a Function Key Switch, a REMOTE detente switch, and indicator lights.

Access Panels

When the BL700 is pulled forward on its slide rails, five access panels are exposed (see Figure 2-1).

WARNING
Observe the **HIGH VOLTAGE** warning on the power supply access panel.

The panels are held in place with 6-32 1/4 inch screws. Vertical panels use positioning screws on their bottom edge. These positioning screws need to be loosened to remove a panel. The BL700 rear panel is not removable.

BL700 ACCESS PANELS

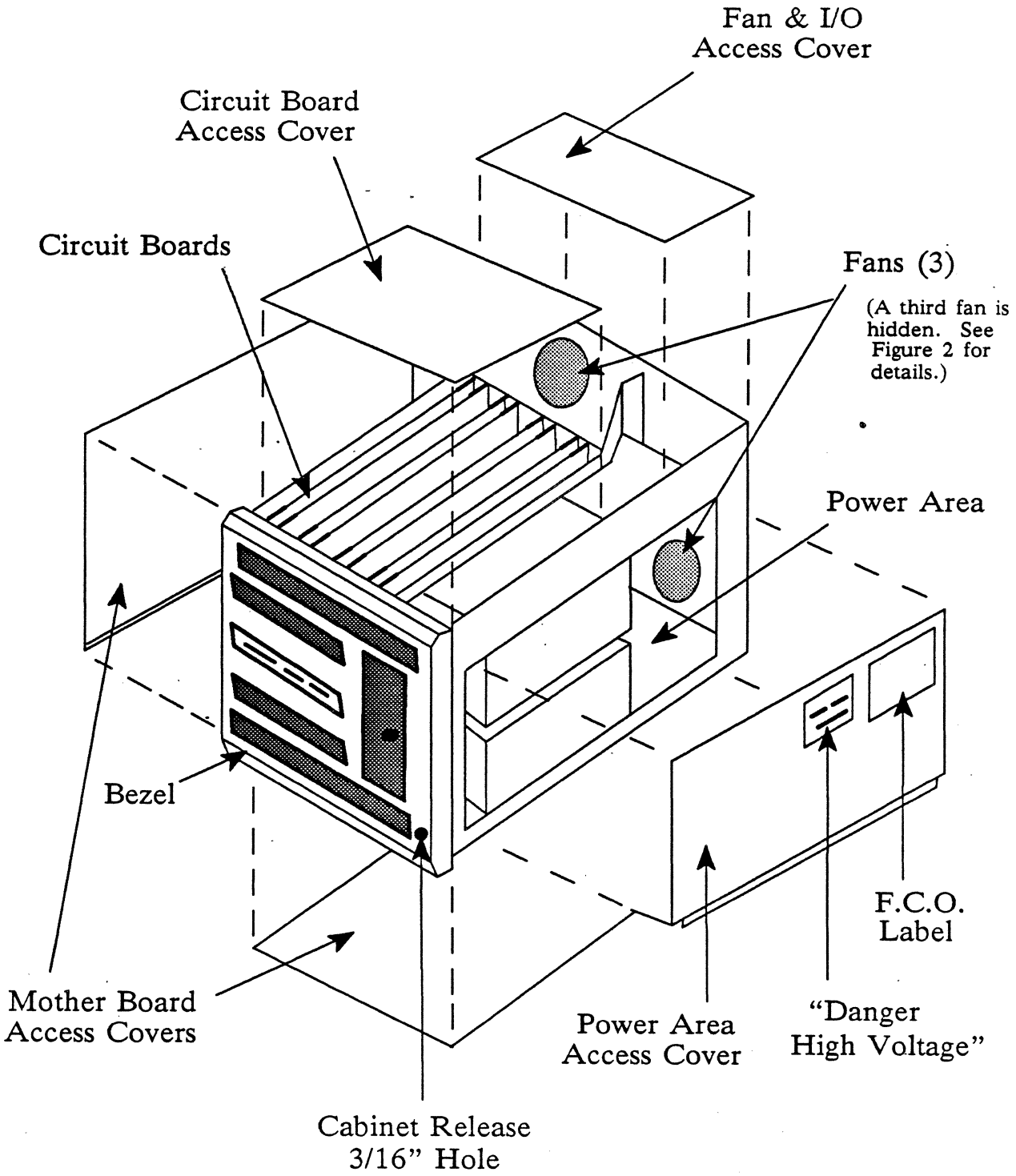


Figure 2-1: BL700 Access Panels

Rear Panel, Fans and I/O

Two vertical, 15.24cm (6 inch) diameter muffin fans are accessed through the fan and I/O access panel. These two fans cool the printed circuit boards (see Figure 2-2).

A smaller 10.16cm (4 inch) diameter muffin fan is on the lower left side of the rear panel. This smaller fan cools the BL700 internal power supply.

The fans are covered with EMI screens and finger guards for user safety.

The machine's serial number, and ac Main Power Circuit Breaker are on the Rear Panel.

Main Power Switch and Circuit Breaker

The Main Power Switch and Circuit Breaker (15A) is in the lower left hand corner of the rear panel. The switch glows when the BL700's power cord is plugged into a live ac outlet, and the circuit breaker is toggled to the ON position. The glowing light shows that ac power is supplied to the internal power supply.

WARNING

When working on the internal power supply, toggle the Main Power Switch OFF and disconnect the ac power cord from its outlet.

Maintenance and Console Ports

The maintenance and console ports are located above the smaller cooling fan. See the BL700 Installation Manual and the BL700 Operation Manual for more information on the ports.

Interface Panel

The interface panels are located on the rear panel above the maintenance and console ports (See Figure 2-3). The configuration is a maximum of eight 1-1/2" subpanels with the highest panel a strain relief for the heavy cables. The interface panels are configured by customer request and are designed to allow the downgrading or upgrading of the BL700. Each panel is held in place with screws (see Figure 2-5) BL700 Configuration.

Serial I/O Distribution

The serial I/O distribution box with eight ports is on the rear panel above the I/O panels (see Figure 2-4). This is the standard configuration. When more than eight ports are needed, the serial I/O distribution box is replaced with a serial I/O interface panel. The two ports on the serial I/O interface panel can be connected to a serial I/O distribution rack containing 16 ports using two 50 pin, ribbon cable connector.

Fan and I/O

The fans are under the rear top access panel (see Figure 2-1). A thermal switch to monitor the temperature of the PCBs is on the rear portion of the card cage. A stepdown transformer is mounted on the rear panel. The transformer provides low ac voltage to the front panel.

BL700 REAR PANEL

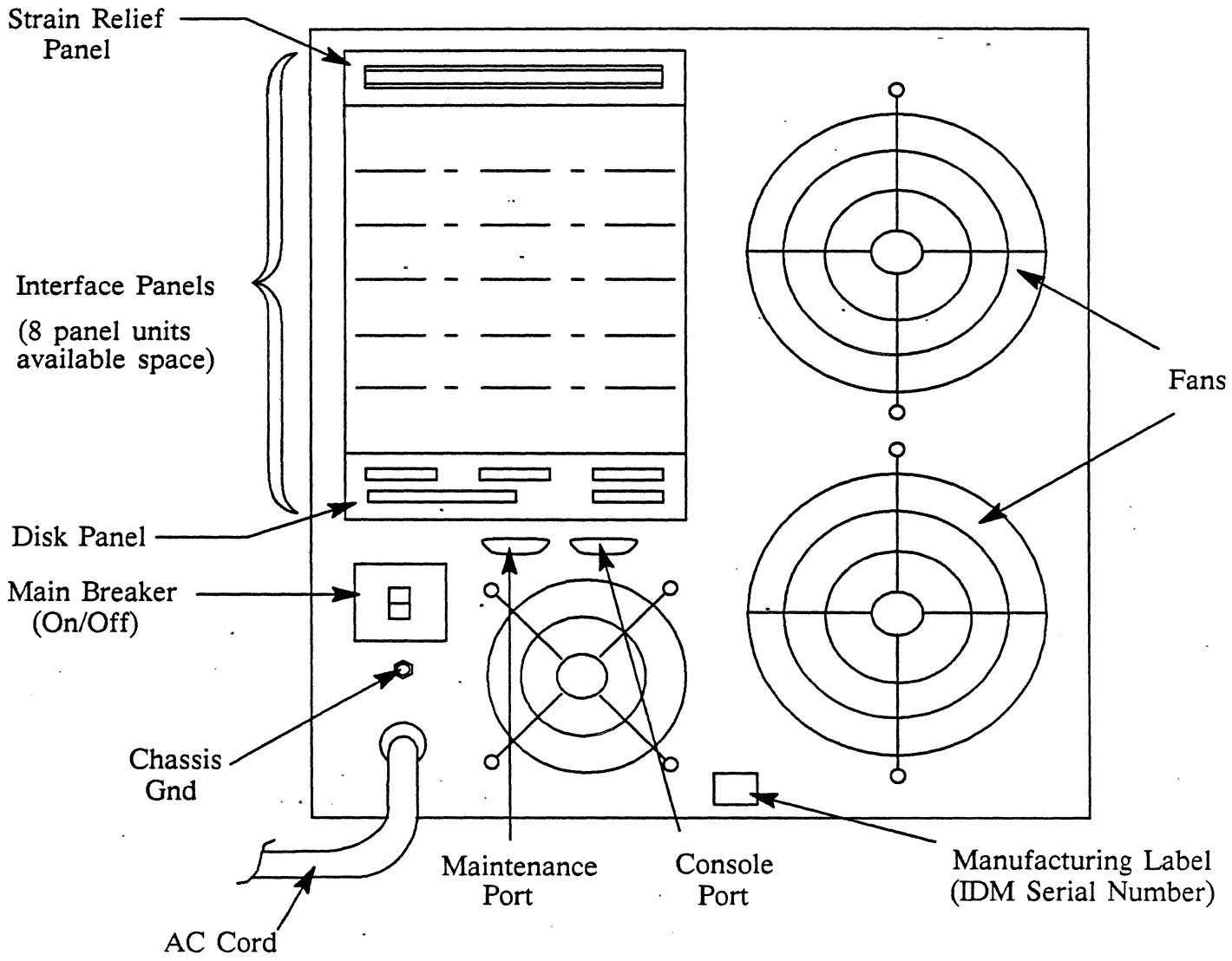


Figure 2-2: BL700 Rear Panel

BL700 REAR VIEW

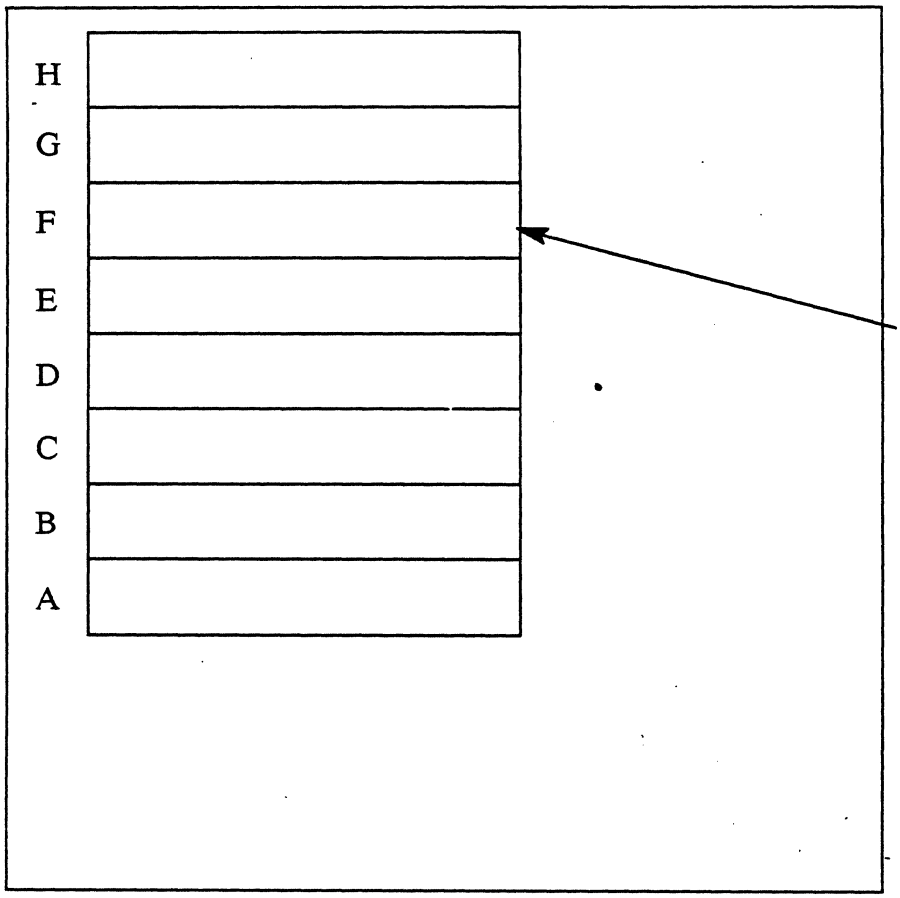


Figure 2–3: BL700 Rear View

SERIAL INTERFACE

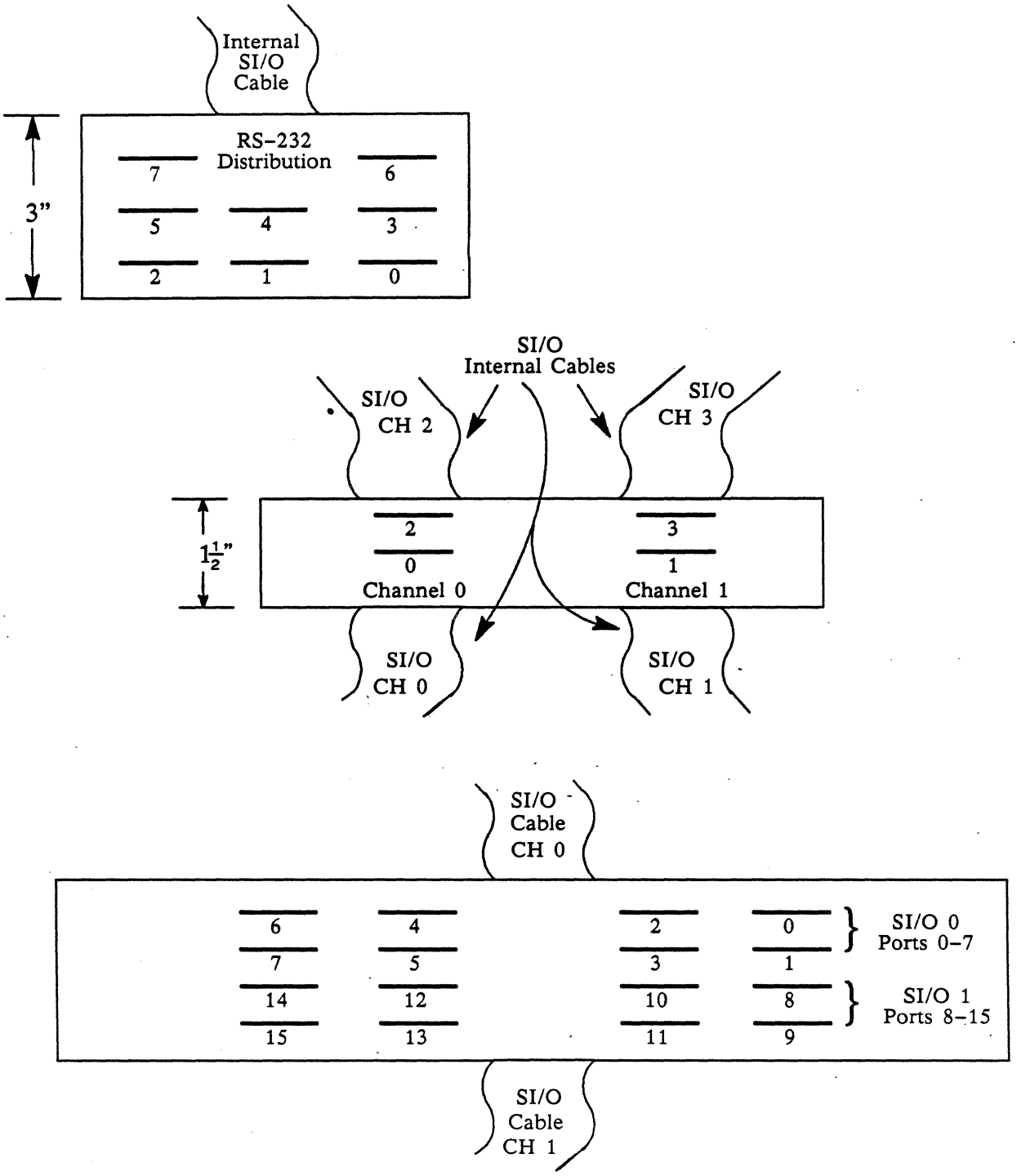
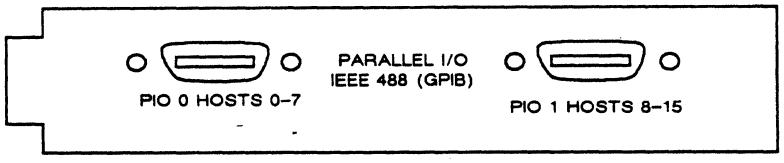


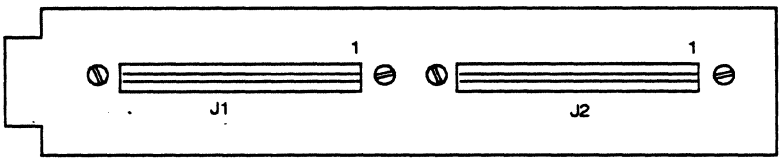
Figure 2–4: Serial Interface

IDM 500 CONFIGURATION

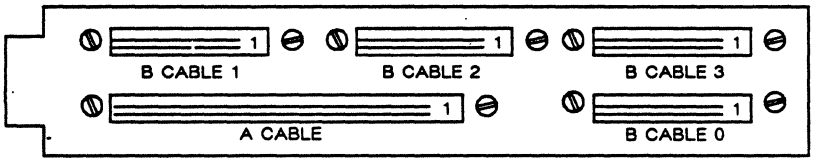
PARALLEL INTERFACE



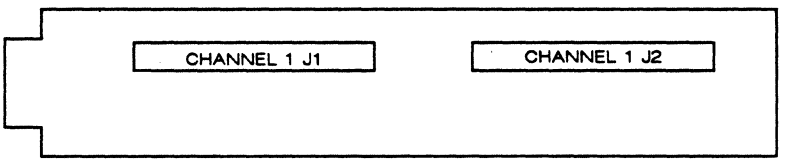
TAPE CONTROLLER



DISK CONTROLLER



BLOCKMUX INTERFACE



ETHERNET INTERFACE

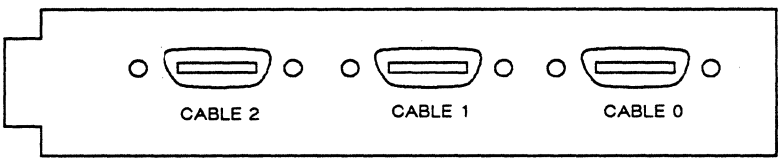


Figure 2–5: BL700 Configuration

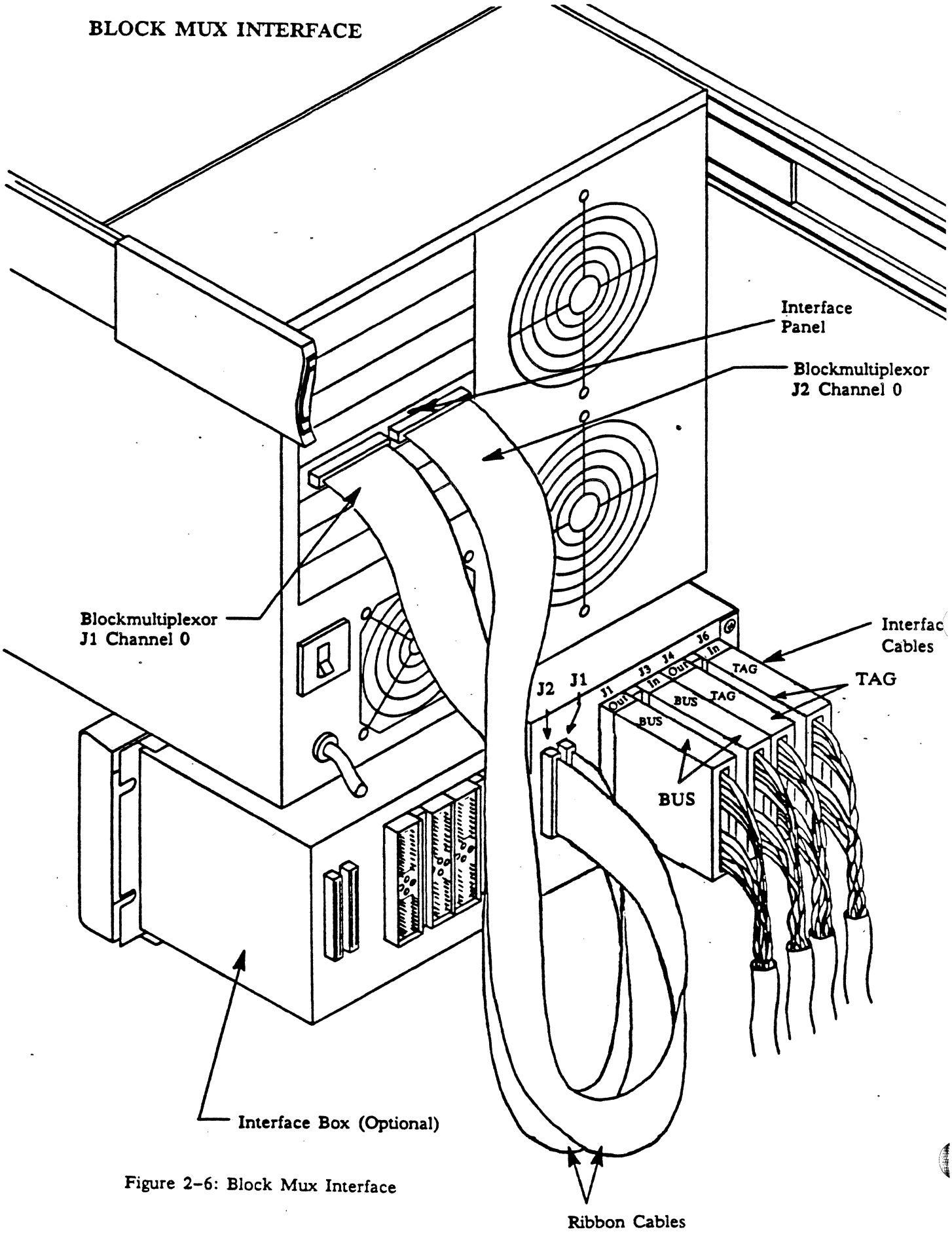


Figure 2-6: Block Mux Interface

AC POWER DISTRIBUTION UNITS

PDU, MODEL MPD 20801

Power Distribution Unit (PDU) model MPD 20801 is used in our 72" Retma cabinet. It delivers 14,000 watts of three phase, conditioned ac power on a 30A breaker. There are sixteen switched receptacles (six NEMA L6-15R, 250V ac,15A and ten NEMA 5-15R, 125V ac, 15A) and two unswitched receptacles (NEMA 5-15R 125V ac 15A) for peripheral ac power. The MPD 20801 is equipped with a EMI Filter to minimize low level high frequency interference. This unit has a heavy duty 12' power cord with a five conductor plug NEMA L21-30R. The remote feature is not used.

MODEL 20801
FRONT VIEW

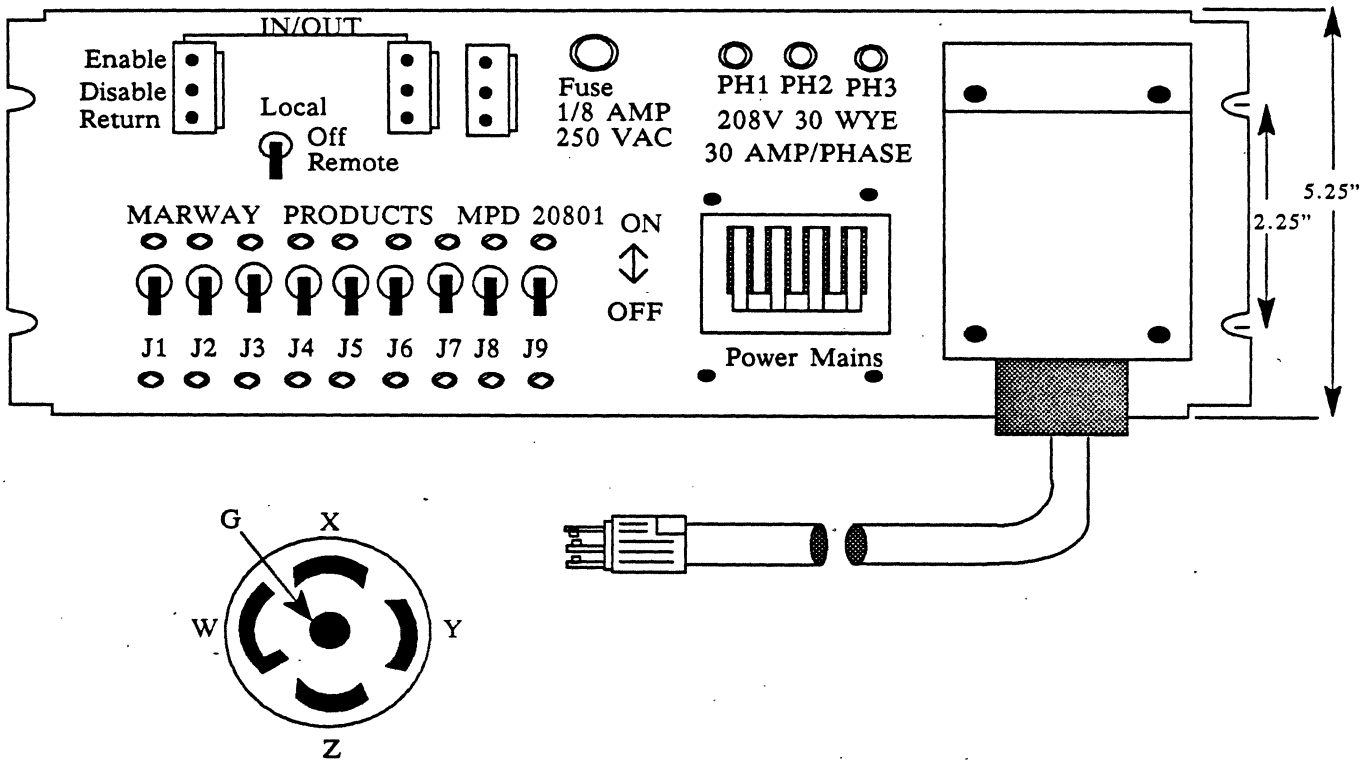


Figure 2-7: Model 20801 Front View

MODEL 20801
REAR VIEW

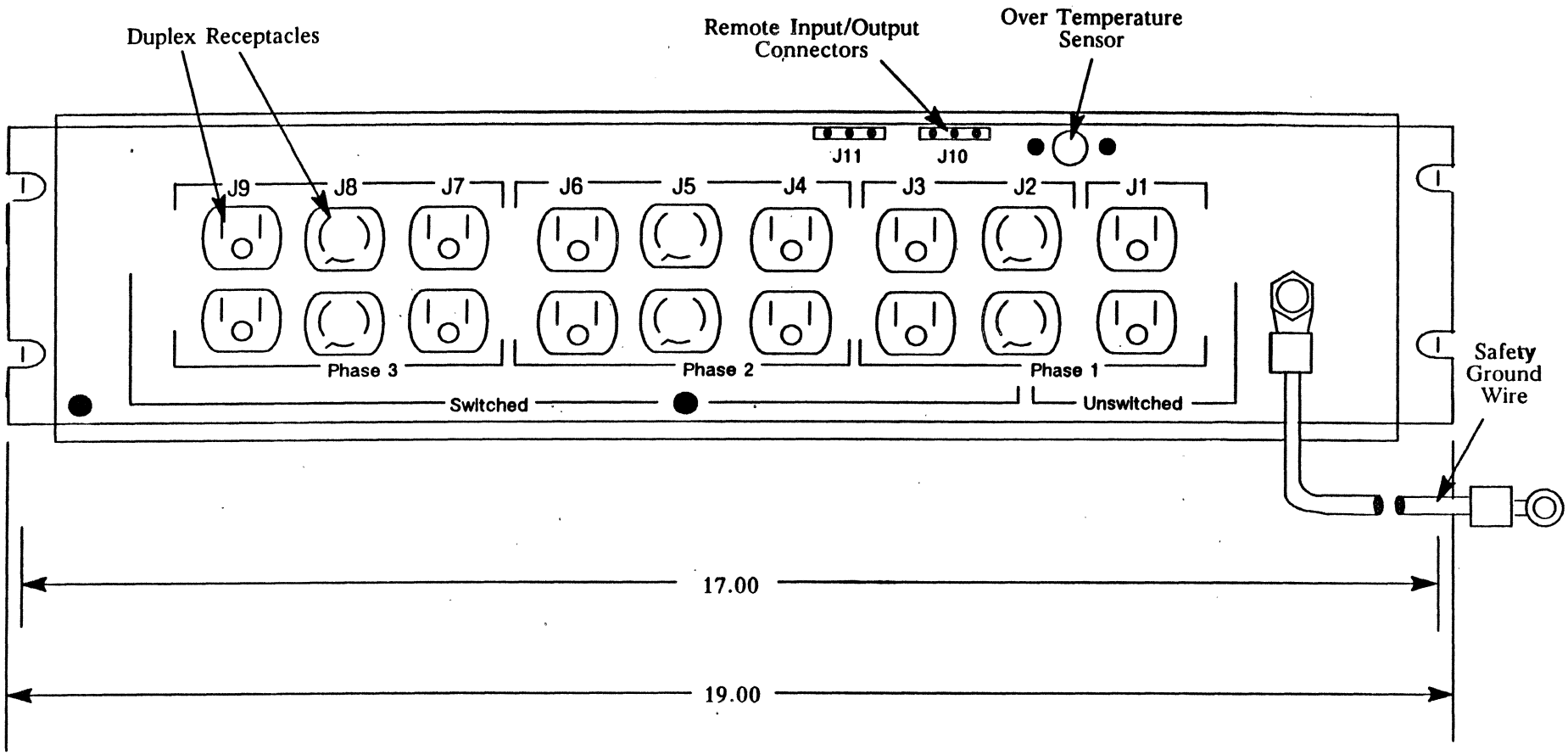


Figure 2-8: Model 20801 Rear View

MODEL 20801
BLOCK DIAGRAM

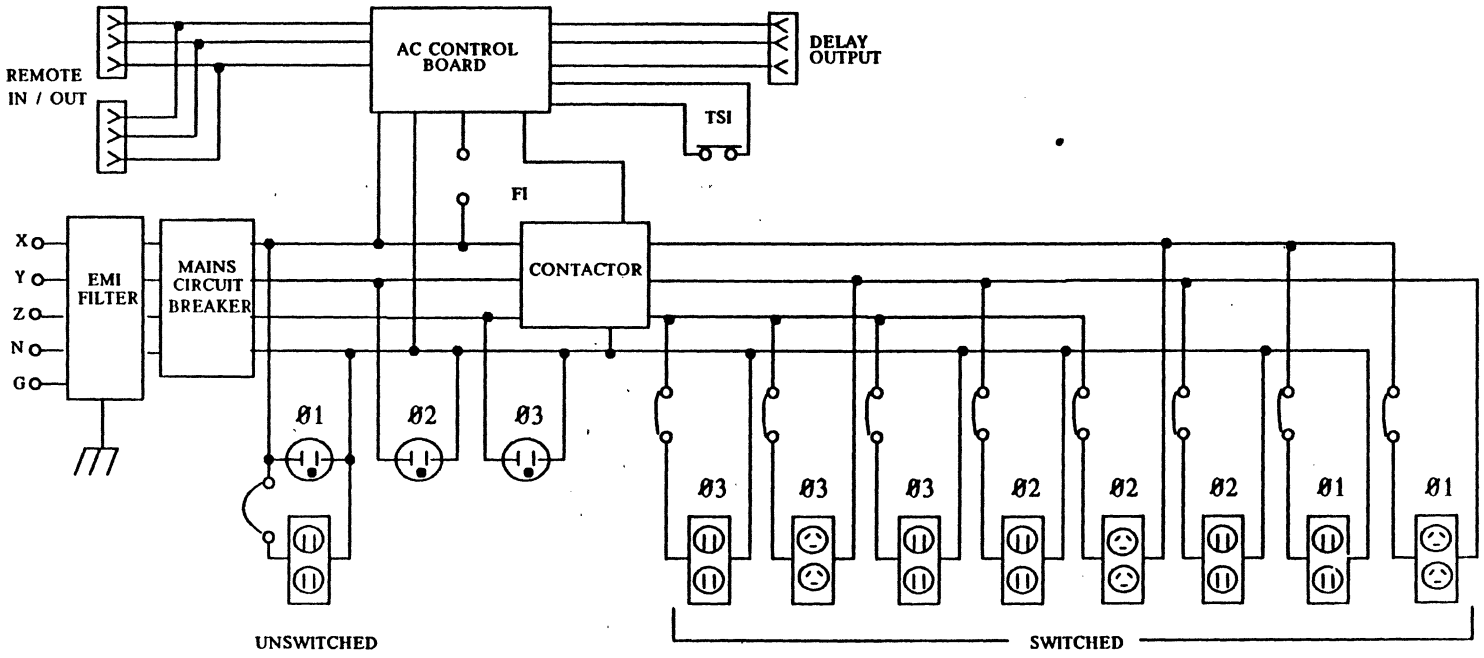


Figure 2-9: Model 20801 Block Diagram

PDU, MODEL 872C 001

Power Distribution Unit model MPD 872C-001 is used in our BL700 series 1,000 pound cabinet. There are four NEMA 6-15R, 250V ac, 15A and one NEMA L6-15R, 250V ac, 15A peripheral receptacles. The two-pole 20A circuit breaker is a motor starter high-inrush type, UL/CSA approved. It has a high performance EMI filter with high frequency ground noise isolator. The power cord has 3 conductors, is 15' long, and is equipped with a NEMA L6-20P plug. The remote feature is not used.

MODEL 872C 001
FRONT VIEW

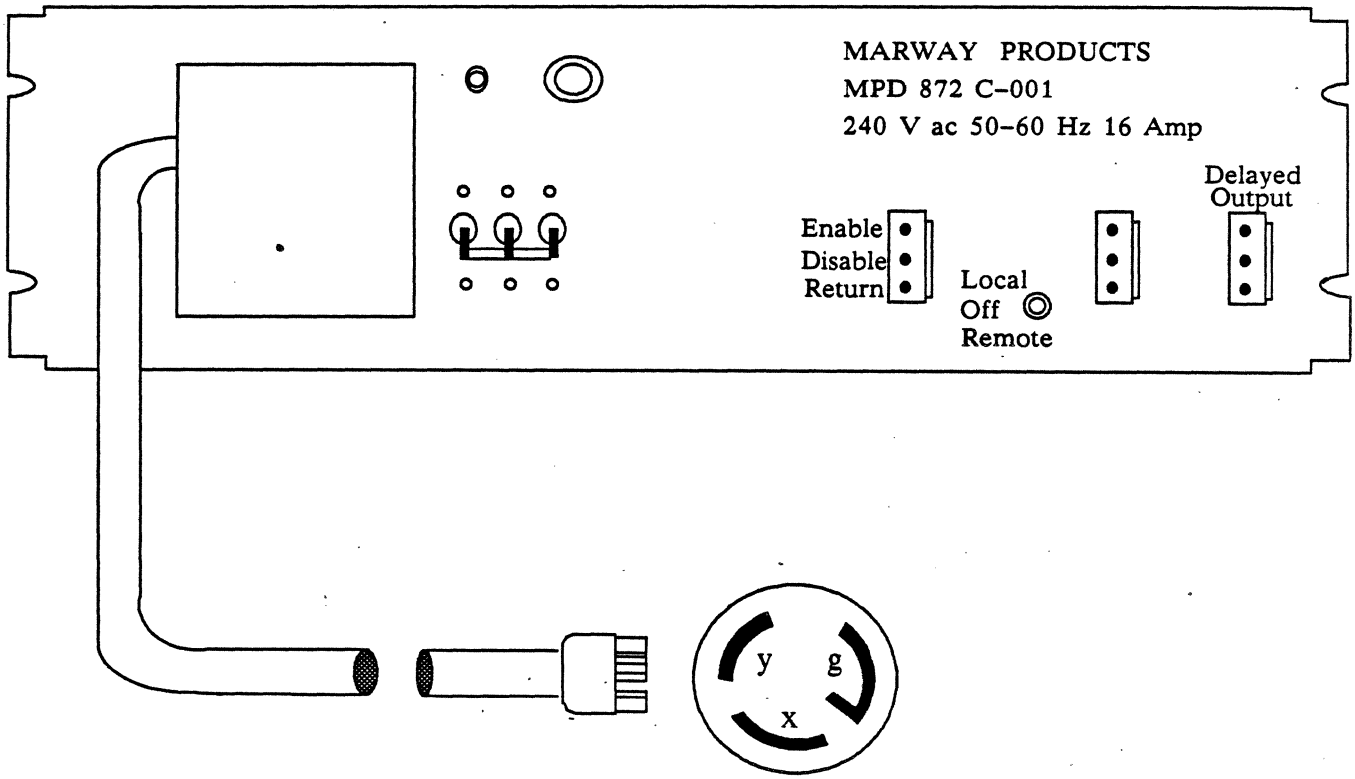


Figure 2-10: Model 872C 001 Front View

MODEL 872C 001
REAR VIEW

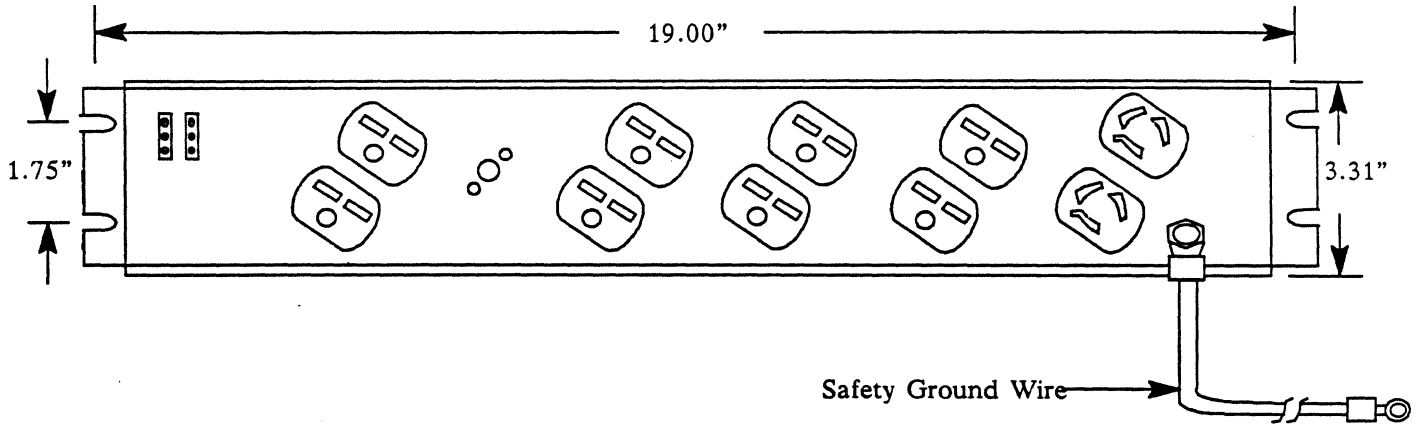


Figure 2-11: Model 872C 001 Rear View

MODEL 872C 001
BLOCK DIAGRAM

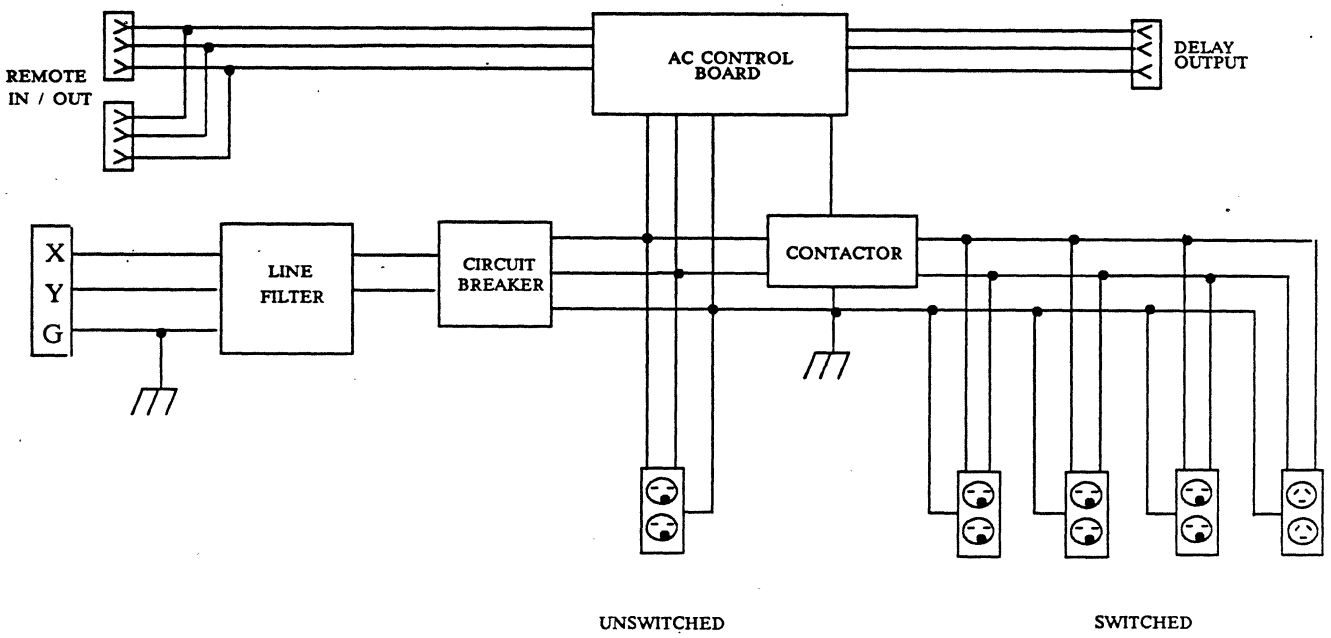


Figure 2-12: Model 872C 001 Block Diagram

PDU, MODEL MPD 87201

This Power Distribution Unit is used in DEC 40" and 60" equipment cabinets. There are eight switched receptacles (Four NEMA 5-15R, 125V dc 15A and four are L6-15R, 250V ac, 15A) and two unswitched receptacles (NEMA 5-15R, 125V ac, 15A) for peripheral ac power. This PDU is equipped with a high performance EMI filter with frequency ground loop isolation to minimize equipment interference. It provides up to 3,800 watts of single phase, conditioned ac power on a 20A breaker. The 15 foot four-wire, three-pole power cable is equipped with a NEMA L14-20R plug. The remote control feature is not used.

MODEL 87201
FRONT VIEW

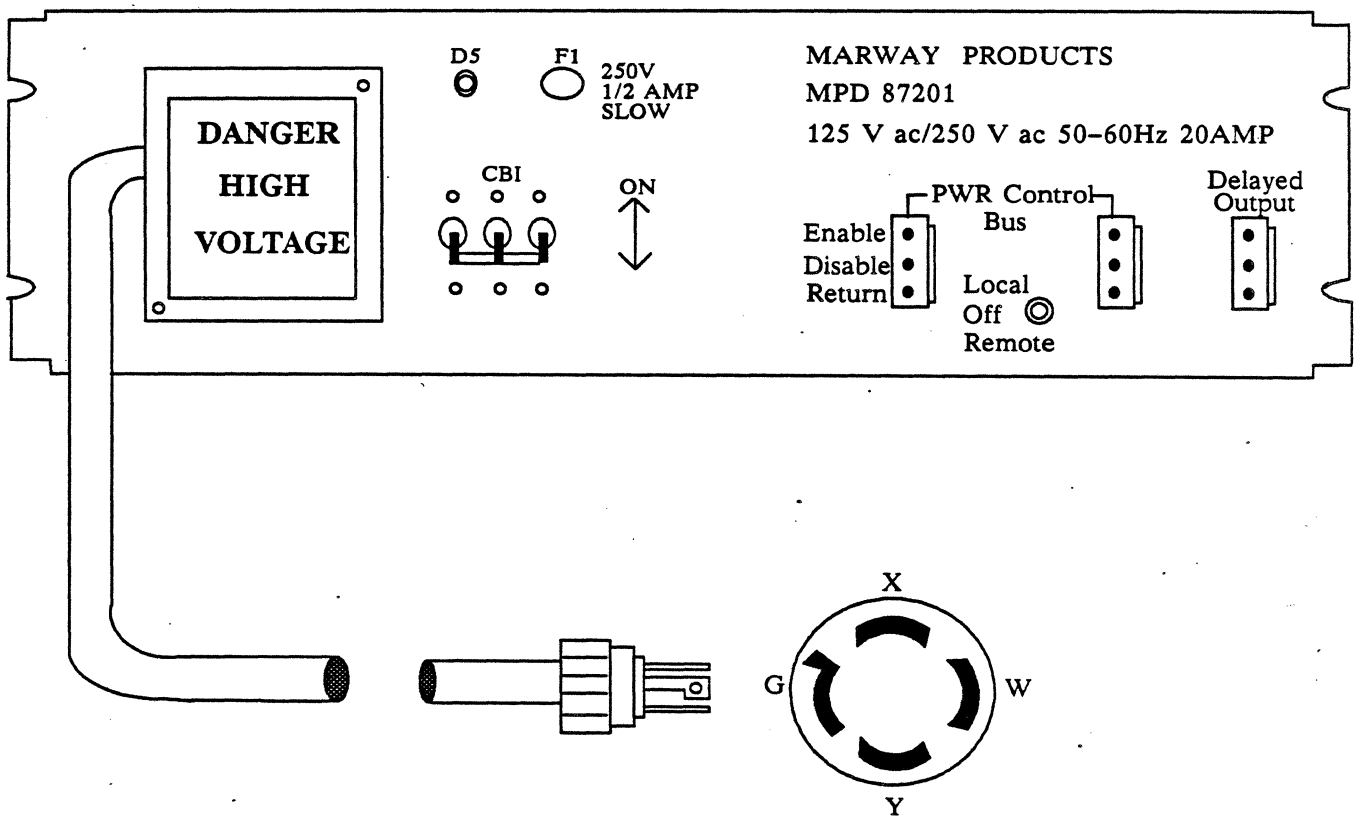


Figure 2-13: Model 87201 Front View

MODEL 87201
REAR VIEW

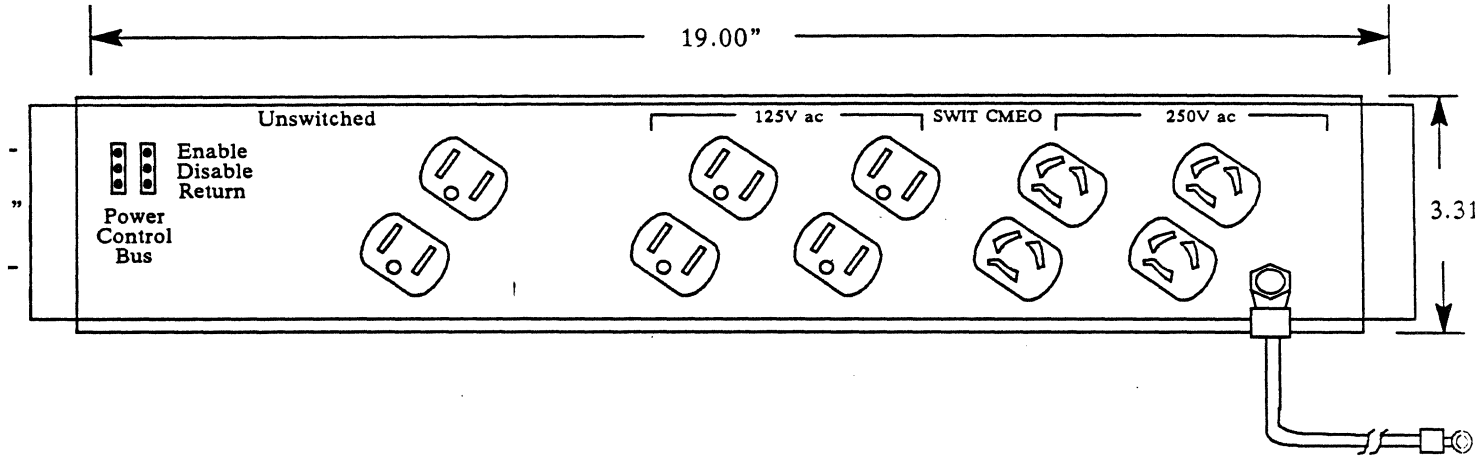


Figure 2-14: Model 87201 Rear View

MODEL 87201
BLOCK DIAGRAM

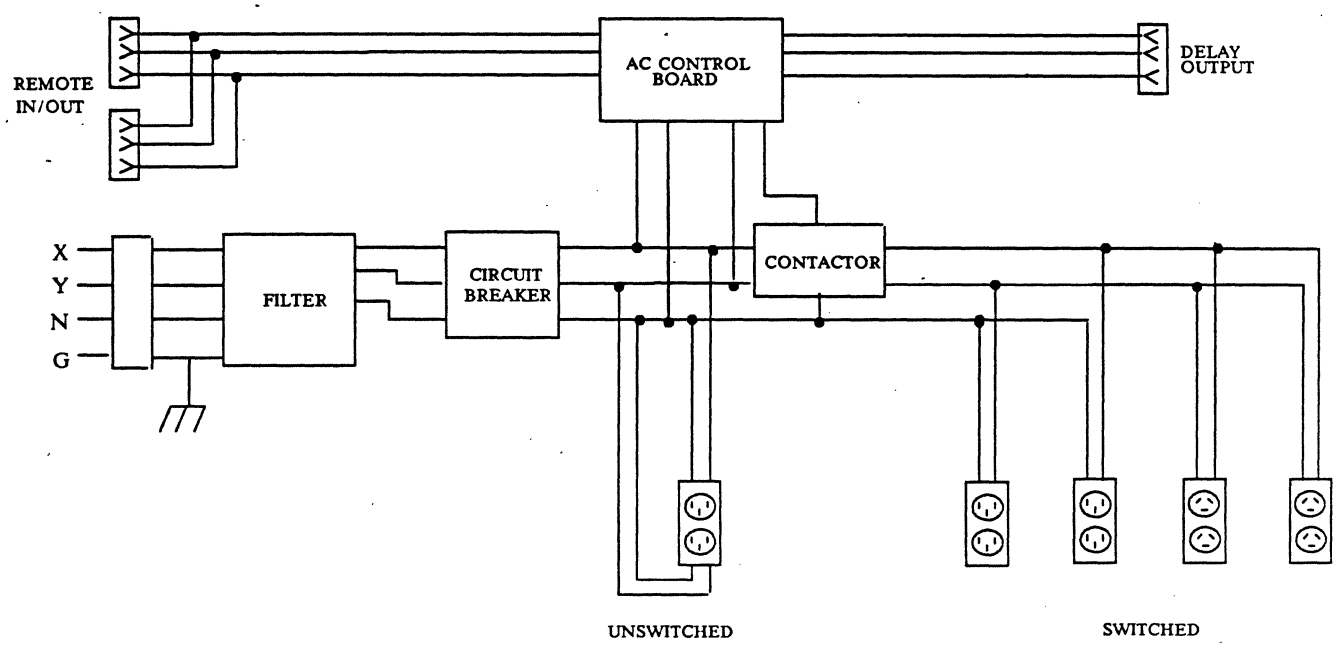


Figure 2-15: Model 87201 Block Diagram

INTERNAL AC POWER DISTRIBUTION

INPUT CIRCUITRY

Power Cable

The BL700 permanent ac power cable is 10 feet long, terminated with a NEMA L6-15P type plug.

Line Filter

The line filter cuts down low level, high frequency interference.

Breaker ON-OFF switch

The breaker is a 3-pole, 250V ac, 50/60Hz, 15A ganged actuator. It is trip free, opening on overload or when forcibly held closed.

AC Distribution Terminal Block

Input ac voltage between 200V and 240V, between 47 Hz and 63 Hz is brought into a terminal block, TB1. TB1 interconnects with PS1 and PS2, three cooling fans, and step down transformer.

Air Circulation Fans

The BL700 is equipped with three fans. These fans draw ambient temperature air in through slots in the Front Bezel and out the rear panel. Two six inch fans are located behind the card cage to keep the Printed Circuit Boards cool. A four inch fan located behind the power supplies assists the power supply fans.

AC INPUT CIRCUITRY

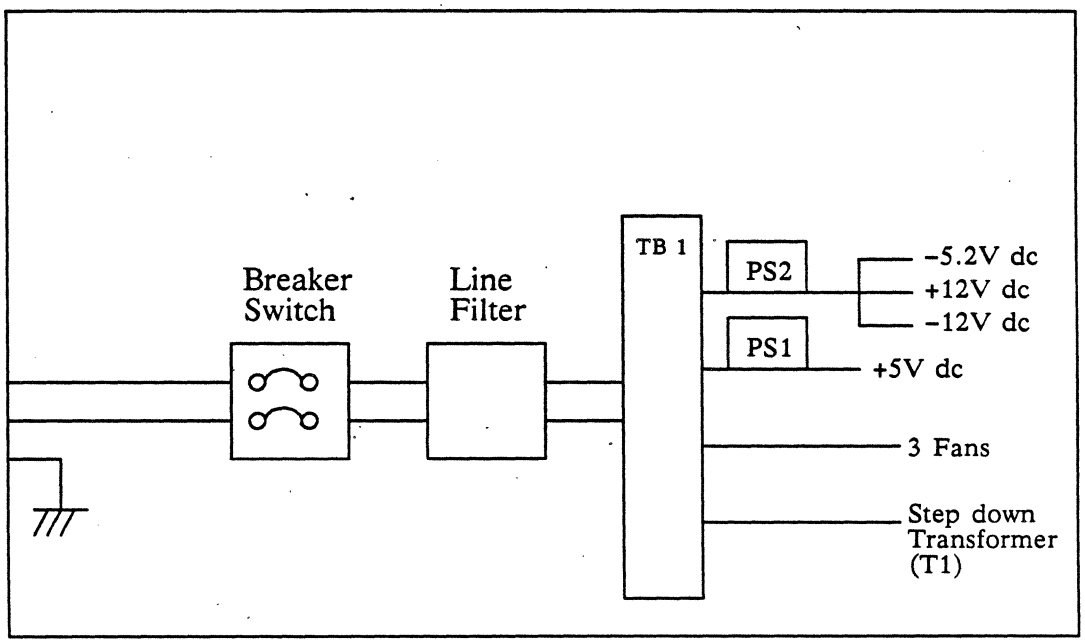


Figure 2-16: AC Input Circuitry

DC POWER SUPPLIES

BULK SUPPLY BL700

The bulk supply (PS1) is a single output power supply that drives all the TTL circuitry. Its input is 208V ac with a line frequency variation of 47 to 63 Hz. It delivers +5V dc at 150A. The power supply is self cooling, it also has an internal thermal sensor that will assert a POWER FAIL (PS1) if the fan stops working and the unit overheats.

MULTI SUPPLY BL700

The multi supply (PS2) is a multiple output power supply that drives ECL circuitry -5.2V dc and other devices requiring ± 12V dc. PS2 supplies the BL700 with -5.2V dc at 130A, 12V dc at 3A and -12V dc at 3A. Nominal ac line voltage is 208V ac and an allowable ac line frequency variation of 47 Hz to 63 Hz. It has internal thermal protection. The power supply will assert a POWER FAIL (PS2) when an excessive internal temperature is sensed.

POWER SUPPLY BL700

This power supply comes only in the BL700. It provides the BL700 with +5V dc at 60A, -5.2V dc at 33A, 12V dc at 1A and -12V dc at 1A. The input voltage is 208V ac and it has a frequency of 60 Hz. An internal thermal switch is provided with normally open contacts at external connector J1 that close when an excessive internal temperature is sensed. The power supply will remain operative when this occurs. Shutdown will be controlled by the user externally and will occur within 10 milliseconds. Additional internal thermal protection is provided in the event that shutdown by the user fails to occur under the above conditions.

Refer to the Power Supply Overview, Diagnostics Section.

MOTHERBOARD

The Motherboard is a passive platform electronically connecting five to sixteen printed circuit boards (see Figure 2–17). Each Printed Circuit Board (PCB) socket has three jacks containing 286 pins. The 286 pins support dc power (+5, –5.2, +12, & –12V dc), Address Bus, Data Bus, Control Signals, and Signal Return Grounds. The sixteen sockets are in parallel so any type of BL700 PCB may be plugged into any socket.

The Motherboard for an BL700 (part number 133–1089) only serves five to seven PCB's in any of the sixteen slots. The Motherboard for the BL700 (part number 133–1091) will serve five to sixteen PCB's.

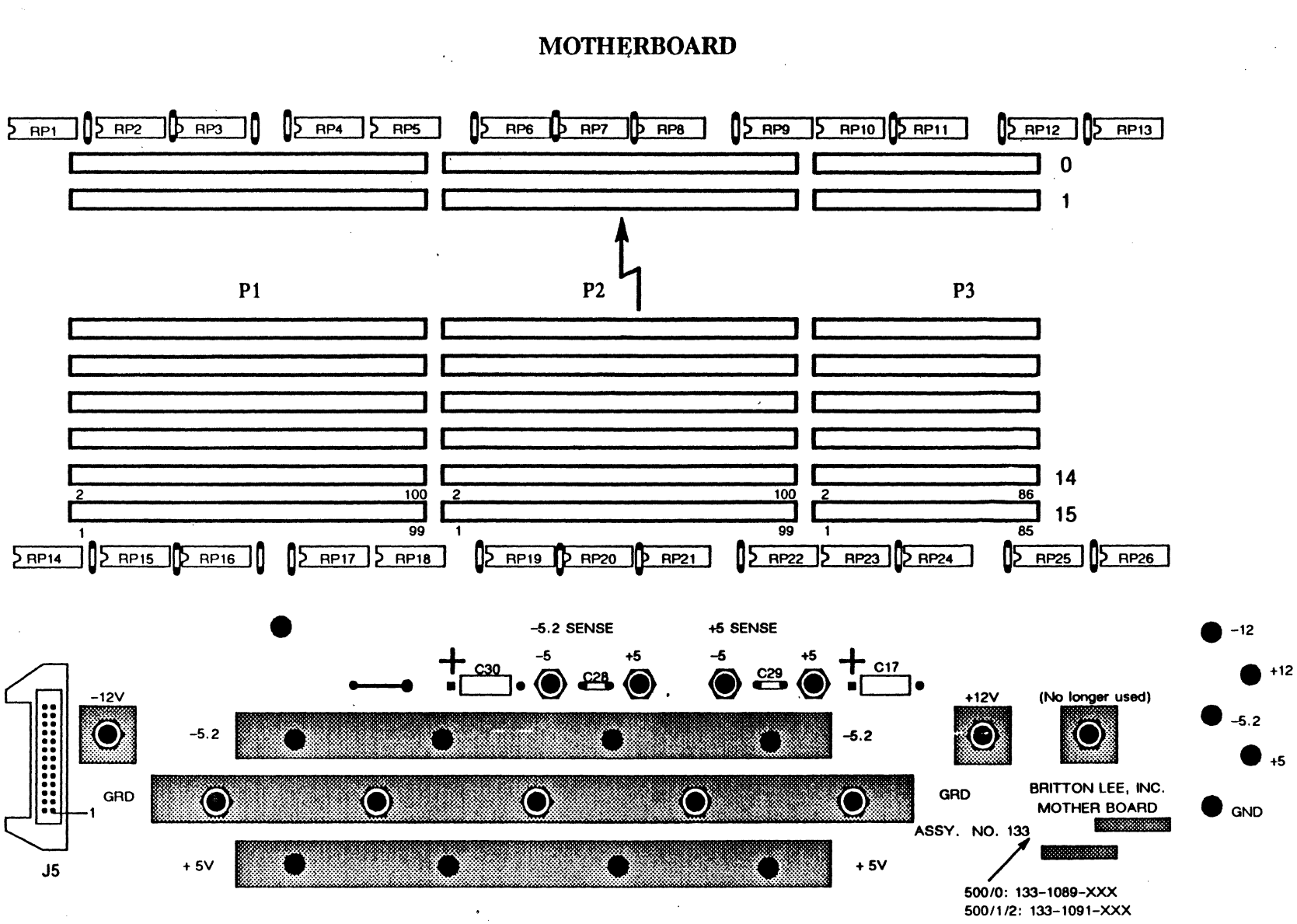


Figure 2-17: Motherboard

FRONT PANEL

INTRODUCTION

The front panel displays BL700 modes of operation and operating status (see Figure 2-18). The legends are lit by backlighted LEDS. They cannot be seen when the backlighted LEDs are OFF. The basic functions of each switch are explained in this section. The front panel is attached to the control panel, which is a printed circuit board. (See Figure 2-19.)

MODES OF OPERATION

REMOTE

Switch (press in, press out). When Remote is actuated, the ENABLED LED illuminates. The DBP will recognize a three second break key transmitted over a modem through the maintenance port as BL700 RESET. This initiates the self-diagnostics. When Remote is not actuated, a technician cannot initiate a RESET from a remote location nor can BLI attach for remote debugging.

FUNCTION KEY SWITCH

There are four positions on the function key switch; OFF, MAINT, SAFE, and RUN.

OFF The dc power supplies are extinguished. The BL700 cannot operate even when the ac power main switch is ON. All front panel LEDs are OFF. The function key may be inserted or removed. (Ac fans are still operating.)

MAINT Rotating the key from OFF to MAINT energizes the dc power supplies and initiates self-diagnostics. The POWER LED illuminates and stays on. READY, SAFE, and SERVICE LEDs flash momentarily and extinguish. The function key cannot be removed. You may now communicate with the BL700 over the DBP console port and load software from a diskette drive over the DBP maintenance port. Turning the function key from SAFE to MAINT also initiates the self-diagnostics, once the BL700 has backed out any current processes.

SAFE Rotating the key from MAINT or RUN to SAFE causes database housekeeping functions to close all relations (files) properly. When all databases are closed, the SAFE LED illuminates. The function key cannot be removed. When moving the switch to MAINT, always stop at SAFE until the SAFE LED turns on.

RUN Rotating the key from MAINT to RUN causes the System Software to be loaded into BL700 memory. As soon as the database is opened (refer to the Operations Manual for software commands.), the SAFE LED extinguishes. The function key may be removed or inserted.

FRONT PANEL LED STATUS

POWER dc power supplies are on and within specifications. This LED should be illuminated when the function key switch is in MAINT, SAFE, or RUN.

READY Function key switch is in RUN or SAFE and the System Software has been loaded. The BL700 is 'ready' to work.

SAFE This LED is ON when all databases are properly closed. The LED is OFF when any database is open, indicating it may be unsafe to turn the key switch directly to OFF.

SERVICE An unrecoverable error causes this LED to light. The console will report specific errors.

PWR FAIL Early Britton Lee database server's. The dc power supply is out of tolerance.

PS1 FAIL +5V dc output is out of tolerance. The BL700 has been shut down. Refer to the Power Supply Troubleshooting section.

PS2 FAIL −5.2, +12 or −12V dc output is out of tolerance. The BL700 has been shut down.

FAULT An unrecoverable hardware failure has occurred. See the console for specific information.

TFAULT BL700 chassis internal temperature exceeds 130 F. The BL700 has been shut down until internal temperature is below 130 F. Check chassis and power supply fans for proper air circulation.

FRONT PANEL

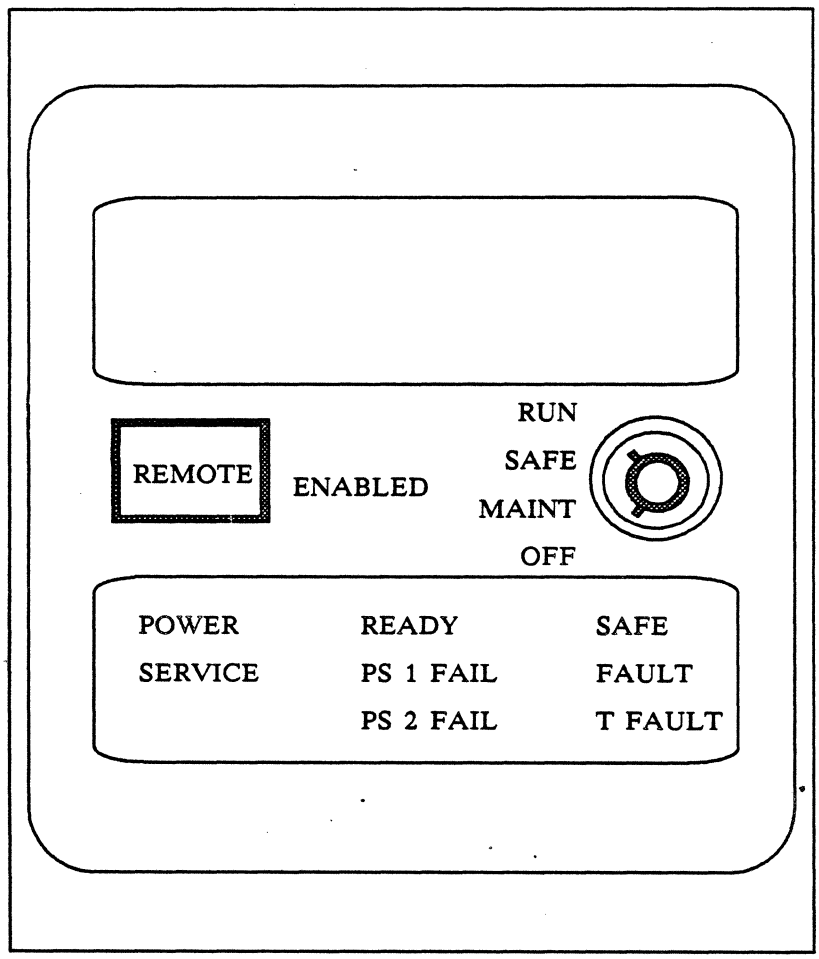


Figure 2–18: Front Panel

CONTROL PANEL

The control panel is attached to the main chassis and extends through a Front Bezel cutout. The control panel contains manual controls for basic system operating states, system status, and error conditions.

The control panel contains power monitoring and control circuitry needed for the BL700. It is powered by an auxiliary +5V dc source that is generated by an onboard regulator. The regulator is independent of the main dc power supply and dc controls and is fed by the auxiliary transformer (T1).

CONTROL PANEL

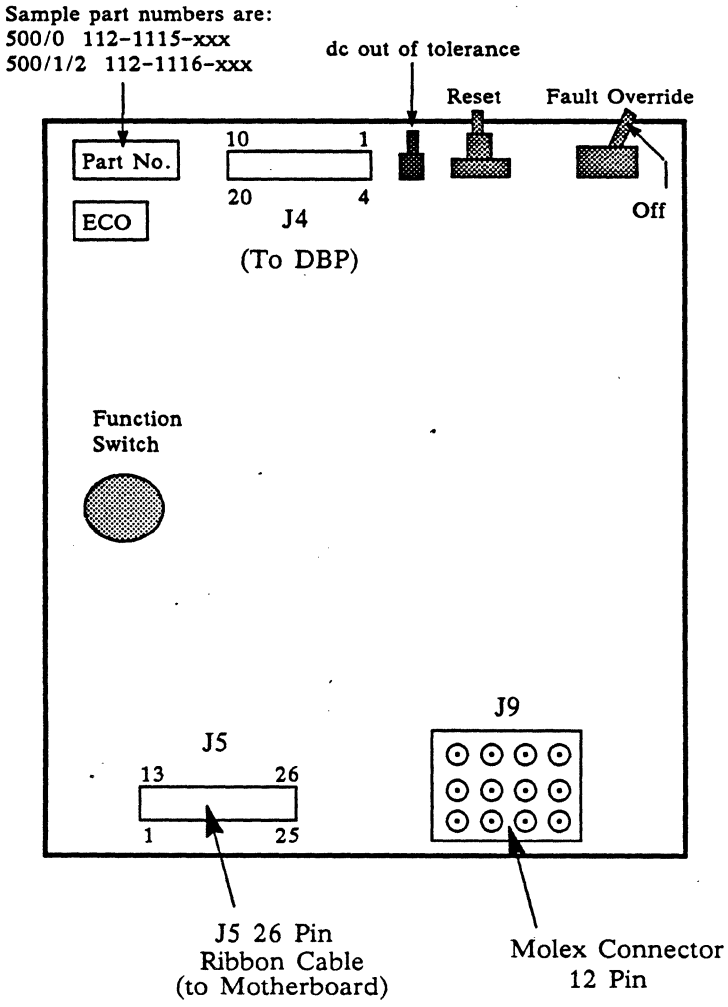


Figure 2-19: Control Panel

PRINTED CIRCUIT BOARD (PCB) OVERVIEW

GENERAL

The BL700 has 16 slots. The minimum configuration for a BL700 is:

1. Database Processor board
2. Timing and controller board
3. 1MB memory board
4. Disk controller board
5. Channel I/O board

Memory, I/O channel boards, disk or tape controllers, or a Database Accelerator can use the remaining eleven slots in the BL700. The maximum board amount of a type are:

Database processor	1
Timing and control	1
Memory boards	6 (1MB boards)
I/O channel boards	8
Disk controller boards	4*
Tape controller board	1

Two additional slots can be used beyond the minimum configuration for the BL700. These slots can have a tape controller or I/O channels.

The BL700 bus structure allows boards to be plugged into any slot. The general rule to follow is to install a Data Accelerator in slot 11, cabled boards toward the cable trough, and memory board to the left side.

Leave an open slot on both sides of the DAC board (slot 11). Balance the air flow through the system by evenly distributing the remaining open slots.

PCB LAYOUT

1. Printed circuit board (PCB) assemblies are 39.375 cm square. Each board slot in the Motherboard has three sockets, P1, P2, and P3 (see Figure 2-17). P1 and P2 have 100 pins and P3 has 86 pins. The boards are keyed to prevent incorrect board installation. Cam levers are provided for insertion and removal of PCBs.
2. Connectors are provided at the top of some PCBs extending the cables to the rear interface area.
3. Each PCB uses an alpha/numeric matrix system (see Figure 2-16) to name integrated circuit locations. Alpha characters start at the bottom with A and move up the rows. The letters G and I are not used. Numerics are used for the columns starting with 1 on the left side of the board at the component side. The alpha/numerics are marked on both sides of the board.
4. The PCB name, part number, and Engineering Change Order (ECO) are located on each board (see respective figures).
5. PCB serial numbers are represented in bar code and usually attached to the trace side of the board.

*If a tape controller is present and disk controllers are revision 27 or an earlier revision, three disk controllers can be used.

DATABASE PROCESSOR 6 MHz

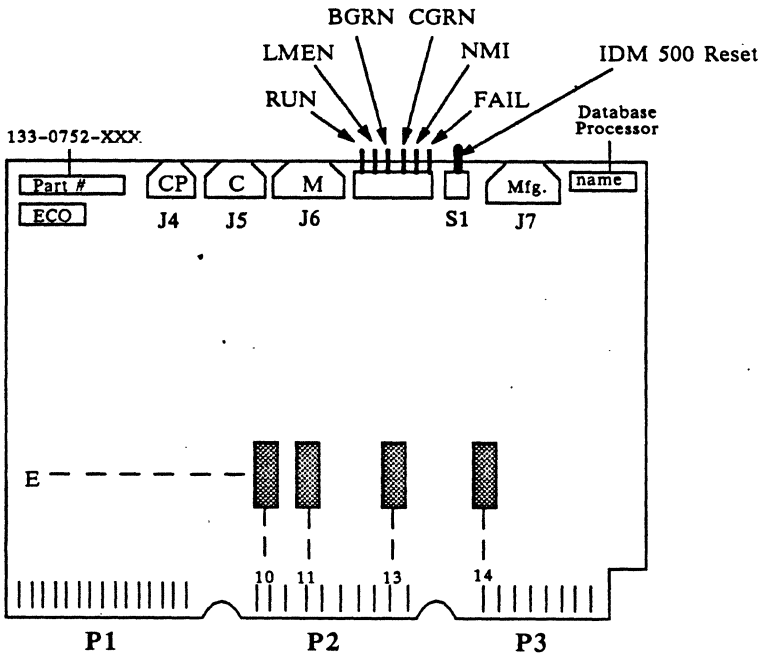
The Database Processor (DBP) is the BL700 Central Processing Unit. In maintenance mode, the DBP supports a special command language entered through the console and maintenance ports.

The Database Processor Board has two 26-pin connections for the console and maintenance ports located on the BL700 rear panel. J5 on the DBP is connected to the console port and J6 to the maintenance port. The 20-pin J4 is connected to the control panel. J7 (40-pin) is intended for factory diagnostics only.

Only BLI part numbers 133-0752 and 133-1180 are in production. The earlier modules are mature and continue to receive support.

133-0001-006 (4 MHz)
133-0719-004 (6 MHz)
133-0720-009 (6 MHz)
133-0752-xxx (6 MHz)
133-1180-xxx (6 MHz)

DATABASE PROCESSOR
6 MHz



*Firmware PROMs are dark rectangles located at coordinated positions.

Figure 2-20: Database Processor 6 MHz

TIMING AND CONTROL BOARD

A memory timing and control board controls Dynamic RAM (DRAM) Refresh Cycle. It buffers BYTE, WORD, and QUADWORD between the Data Bus TTL and memory ECL logic. A single bit error is corrected and reported to the console as a soft error. More than one bit errors are detected and reported to the console as uncorrectable memory errors. One timing and control board can control one to six memory boards in any combination of 256K bytes, 512K bytes, and 1MB boards.

TIMING AND CONTROL BOARD

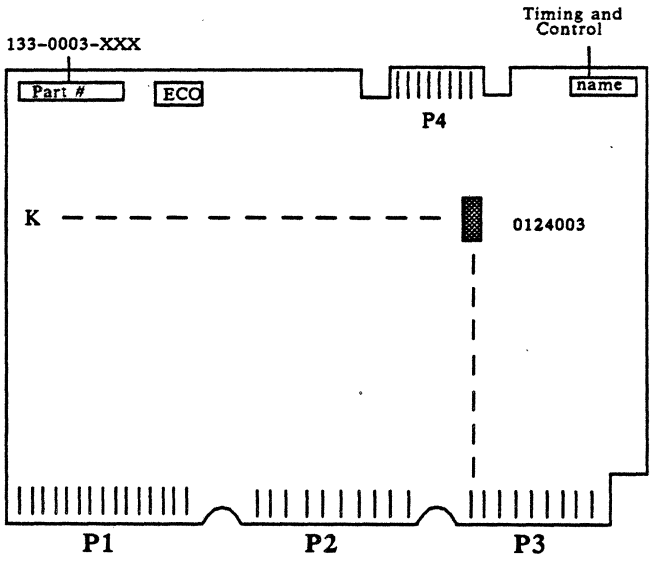


Figure 2-21: Timing and Control Board

MEMORY BOARD

The memory boards utilize 64K DRAM chips to provide a board capacity of 1MB. The BL700 can be expanded to a total of 6Mbytes for optimum throughput. Memory is used to hold system information, IDM/RDBMS software, disk caching, indices, and pre-processed stored commands.

MEMORY BOARD

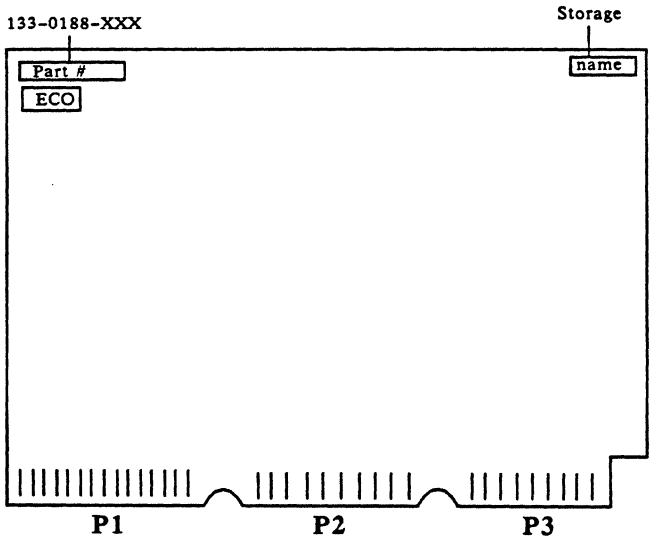


Figure 2-22: Memory Board

NOTE
Early version 256K* byte and 512K byte memory boards are interchangeable with 1MB memory boards, but only six boards of any combination are allowed.

* 256K use 16K DRAM.

SMD DISK CONTROLLER

A Disk Controller will support up to four industry standard SMD (storage module disk) compatible disk drives of fixed or removable media. The BL700 allows one to sixteen disk drives. The controllers perform burst error correction (5 bits) and retry without intervention from the Database Processor.

SMD DISK CONTROLLER

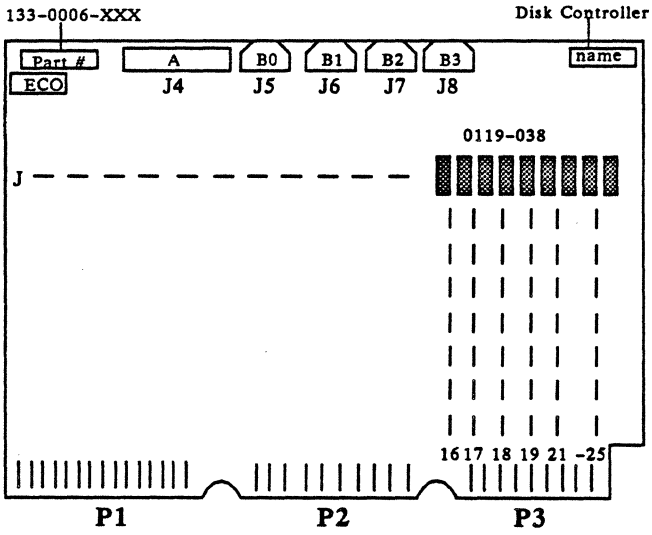


Figure 2-23: SMD Disk Controller

TAPE CONTROLLER

An optional tape controller module supports up to eight drives that can be used for direct disk to tape back-up, data loading and IDM/RDBMS software loading.

TAPE CONTROLLER

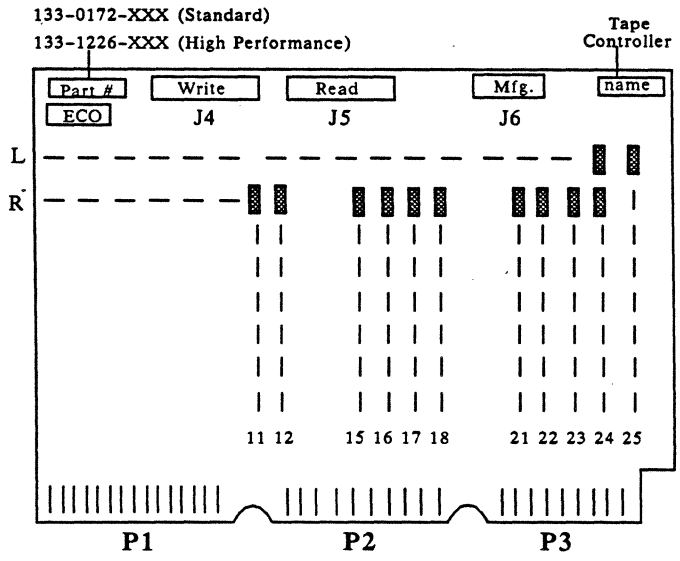


Figure 2-24: Tape Controller

DATA ACCELERATOR

By definition, a BL700 has a Data Accelerator board and two megabytes of memory. The DAC is a high-speed processor with an instruction set that optimizes relational database processing. This accelerator has a three stage pipeline that executes instructions up to 10 MIPS. It can initiate disk activity and search disk pages while pages are being transferred into memory. In most cases, the DAC can complete processing a page by the time the page has been completely read in. The Database Processor determines if the DBP or DAC will process instructions. The DAC is designed to perform the most time consuming work.

DATA ACCELERATOR

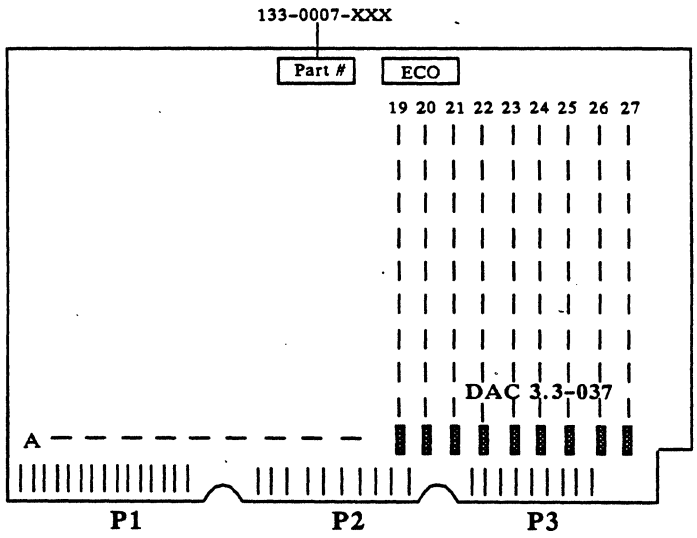


Figure 2-25: Data Accelerator

CHANNEL I/O BOARDS

The BL700 is a peripheral of host computers and must communicate with many hosts. Each kind of host has special communication requirements. A BL700 may be required to communicate with many different kinds of hosts simultaneously.

The communication interface is called a Channel Input/Output board. There are four kinds: Serial Asynchronous RS-232, Parallel IEEE-488, Ethernet with XNS protocol, and Block Multiplexor Channel for the IBM. A discussion of each Channel Input/Output board follows.

SERIAL ASYNCHRONOUS RS-232 I/O BOARD (SI/O)

A Serial Channel I/O board can support one to eight hosts on RS-232 ports. The baud rate (150 to 19,200 baud) and type of modem control are individually configured for each of the eight ports. Unused ports may be turned off to reduce interrupts caused by noise.

SERIAL ASYNCHRONOUS RS-232 I/O BOARD (SI/O)

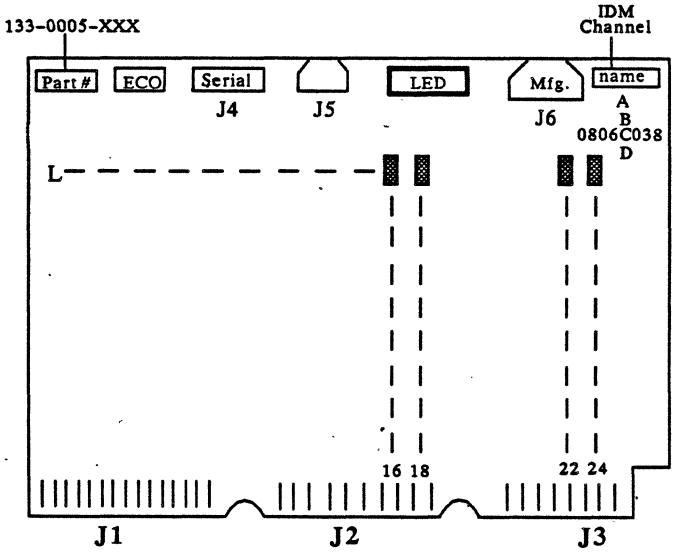


Figure 2-26: Serial Asynchronous RS-232 I/O Board (SI/O)

EXPLODED VERSION OF LED

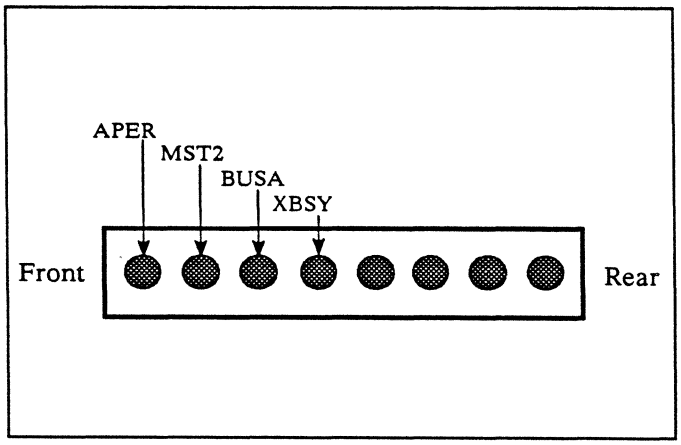


Figure 2-27: Exploded Version of LED

PARALLEL IEEE-488 CHANNEL BOARD (PI/O)

There are two parallel channel boards. PI/O-1 (133-0004) has the BL700 acting as communications controller. PI/O-2 (133-0807) has the host acting as communications controller. Eight hosts can be supported by one board. The external cable is designed to accept several cables in piggyback fashion. Each cable is connected to a different host.

A PI/O cable is connected to a host through a General Purpose Interface Board (GPIB) inserted in the host. The GPIB acts as an interface between a BL700 Parallel Channel board and the host.

PARALLEL IEEE-488 CHANNEL BOARD (PI/O)

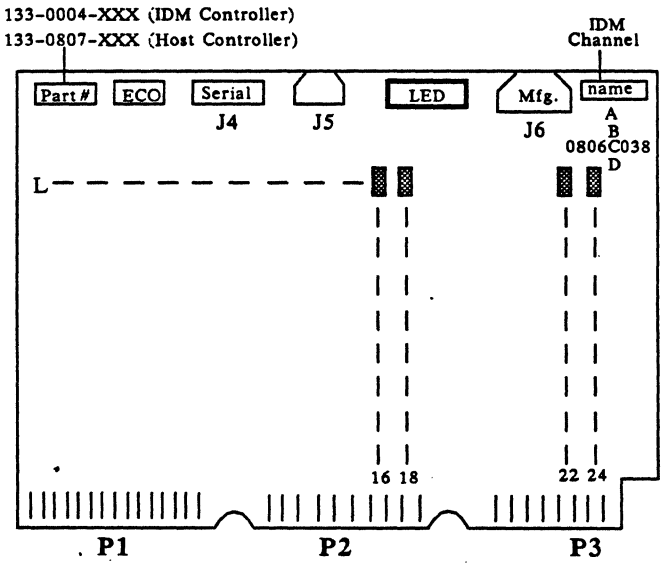


Figure 2-28: Parallel IEEE-488 Channel Board (PI/O)

EXPLODED VERSION OF LED

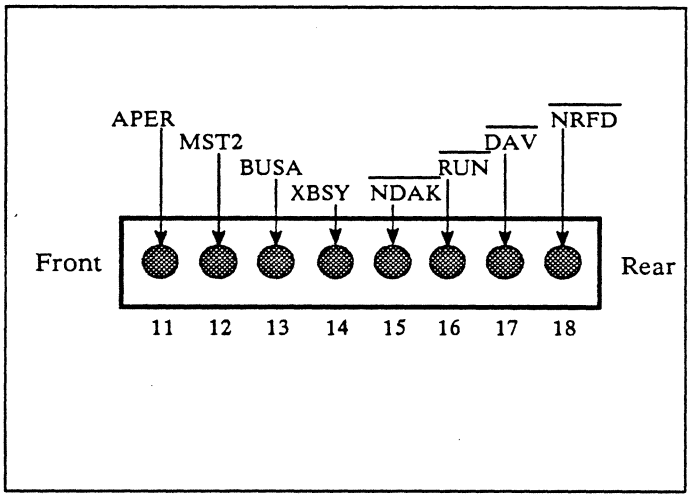


Figure 2-29: Exploded Version of LED

ETHERNET CHANNEL BOARD

The ethernet channel board includes a 50 foot, 20-lead cable and transceiver. The transceiver is connected by cutting and splicing, or tapping a terminated coax transmission line. All communicating devices are connected to the coax table. The ethernet card powers the ethernet transceiver with +12V dc.

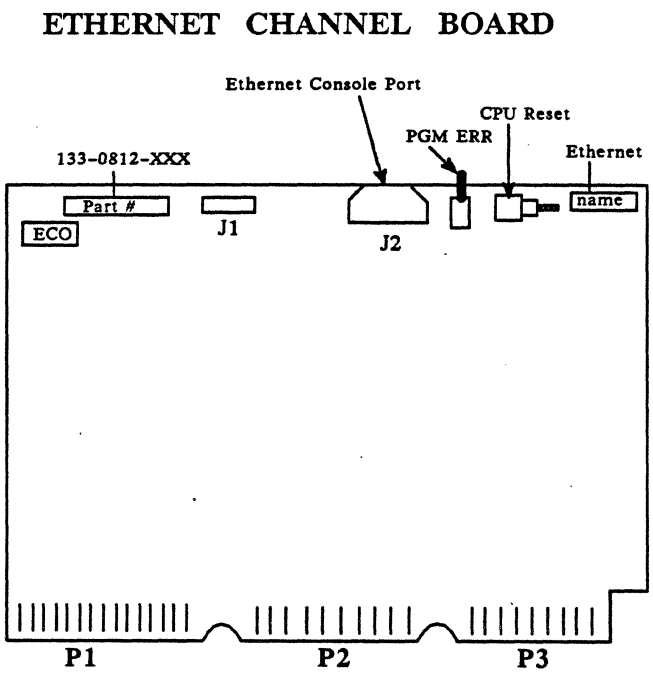


Figure 2-30: Ethernet Channel Board

BLOCK MULTIPLEX

The Block Multiplexor Channel board is connected to an IBM or IBM emulating computer on a Block Multiplex Channel.

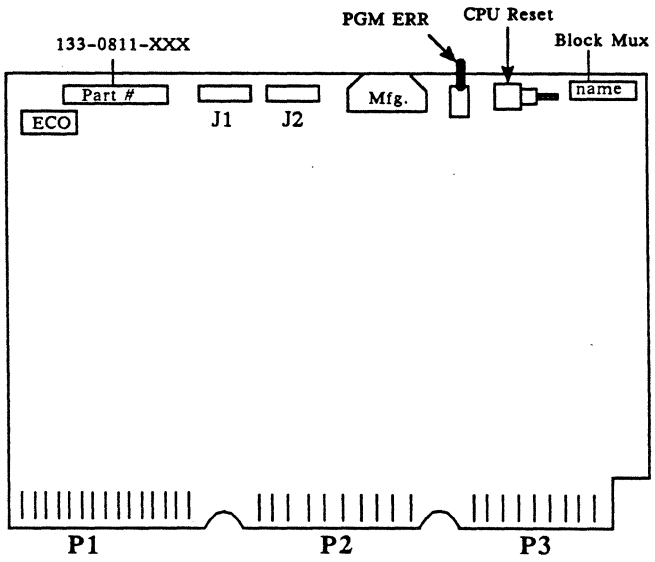


Figure 2-31: Block Multiplex

PREVENTIVE MAINTENANCE

Preventive Maintenance Procedures

PREVENTIVE MAINTENANCE

INTRODUCTION

The following is a suggested preventive maintenance procedure:

1. Check dc supply voltages using a Digital Volt Meter (DVM). If necessary adjust voltages. Refer to the Power Supply Troubleshooting Section for tolerances. Pre-FCC Britton Lee database servers with data accelerators should be readjusted from -5.2 to -5.35. Pre-FCC Britton Lee database servers can be identified by checking the back of the database server. The Pre-FCC model has four fans, three on the logic cage side and one by the power cord. The FCC model has two fans on the logic cage side and one by the power cord.
2. Check the power supply fans. Remove the side covers to gain access to the fans. The fans should be rotating while the BL700 ac switch is ON.
3. Vacuum the BL700 chassis, fan grills, and power supply fans.
4. Execute the "slots" command from the console port several times. The "slots" command executes the board's self-diagnostics. This tests the DBP, memory, channels, disk controller, tape controllers, and Data Accelerator (DAC). Replace bad memory chips indicated on the console.
5. Execute the "loopback command" to test the interface of serial channels and tape controller.
6. Clean the tape drive read/write head daily. It is not necessary for the BL700 to be down for tape drive cleaning. Refer to the manufacturer's suggested preventive maintenance items for additional information on disk and tape drives.
7. Execute the scan mode of the Disk Format Utility (DFU) to locate reoccurring soft disk read errors. Soft read errors have a return code of A008 or B008 with an error code of 0. This ensures that the sector can be read. Refer to the Utilities Section DFU.
8. Check cables. Cables should be securely fastened to the BL700 I/O panel. Cables should not bind or rub.

SCHEDULE

The Preventive Maintenance schedule should be followed on a three month basis. The frequency of unscheduled down time and the computer room environment may affect the schedule and make it necessary to shorten the interval between Preventive Maintenance checks.

DIAGNOSTICS

Introduction

Power on Self-Diagnostics and Start-up Sequence

Serial and Parallel Self-Diagnostics

Timing and Control Self-Diagnostics

DBP LEDs

Disk Controllers and Tape Controllers Power on Self-Diagnostics

Troubleshooting Ethernet Hardware

Loopback for Serial Channel

Power Supply Troubleshooting

When You Need Assistance

INTRODUCTION

Consider a troubleshooting problem this way:

1. Discovery: A problem exists!
2. Symptoms: There are System or IDM error messages.
3. Diagnosing: This section will help you to figure out how to eliminate the problem and how to use diagnostic tools.
4. Repairing:
5. Testing:
6. Restoring:

Diagnostic tools are discussed in the following order:

1. Power On Self-Diagnostics and Start-Up Sequence
2. Ethernet Self-Diagnostics
3. Timing & Control Self-Diagnostics
4. DBP LEDs
5. Disk Diagnostic
6. Tape Diagnostic
7. Disk Controller & Tape controller Power On Self-Diagnostics
8. Troubleshooting Ethernet Hardware
9. Loopback for Serial Channel
10. Power Supply Diagnostics
11. Disk Formatting Utilities

POWER ON SELF-DIAGNOSTICS AND START-UP SEQUENCE

INTRODUCTION

This section assumes that the following are true statements: The BL700 and all connected peripherals operate properly. A terminal is connected at 300 baud to the DBP console port. Correct ac power is available and connected to the BL700 through its ac power cord. The ac power switch is OFF. The front panel function key is OFF.

This section describes the events occurring to bring a BL700 from OFF to the fully operational mode, RUN. At system startup, the Database Processor (DBP) performs an internal self-check, tests memory, examines, and reports the state of each board in the BL700. All disks are tested for operating capability and the operating system is downloaded into memory. A recovery program brings all databases to a consistent (closed) state, and finally, each channel I/O board is programmed to its correct configuration.

SYSTEM STARTUP (AC POWER)

The ac power switch is located on the rear panel in the upper left corner. When the switch is pressed ON, these events occur:

- The ac power switch illuminates.
- The three cooling fans begin operating.
- Control panel "DC OUT" LED turns on.

The internal Control Panel's LED (light emitting diode) directly reports the dc power supplies are not delivering their specified output. The front panel function key is OFF. Indirectly, the LED shows the Control Panel is receiving correct ac power because its onboard dc power supply is operating.

BL700 MAINTENANCE MODE

A BL700 enters the maintenance mode when the front panel function key is turned to the MAINT position. Self-diagnostics operate immediately to determine if the BL700 is working. As long as the function switch is kept in the MAINT position, the terminal device (called console) is used for installation, maintenance, diagnostic, and troubleshooting purposes. Operation is NOT extended to users.

Observable milestones are seen as numbered statements. Detailed explanations follow each milestone. We begin with the function key switch in the OFF position.

1. Turn the function key switch to MAINT.

> **NOTE**
> If the function key switch is already in MAINT and the circuit board access cover has been removed (see Figure 2-1), either the DBP or Control Panel reset button may be pressed instead. Otherwise, turn the function key switch from MAINT to OFF and back to MAINT to start the self-diagnostics.

2. Front panel POWER LED turns ON.
   - The control panel performs the following functions.
     actuates the power supply remote control lines
     turns ON the front panel POWER LED
     issues a 1 second POWER ON RESET to the DBP
   - The dc power supplies stabilize in 800 milliseconds.
   - The control panel dc OUT LED extinguishes if the dc voltages are within specified limits.
   - The DBP issues Master Reset to each board.
   - The DBP begins self-test.
     Front panel LEDs READY, SAFE, and SERVICE flash ON and OFF.
   - When DBP self-test completes satisfactorily, send

```
"DBP Rev nn"
```

to the DBP console and maintenance ports.

3. DBP polls to identify memory and Channel I/O Boards.

   - Starting with Slot 0, DBP writes a 0 to the first RAM address. If DBP's command times out, continue at the next slot. If DBP's command does not time out, this slot contains a memory or channel board.
   - DBP writes a 0 to address 8K, of the memory or channel board. If DBP's command times out, it is a Serial or Parallel Channel board (Shared memory = 4K).
   - DBP writes a 0 to address 64K. If DBP's command times out, it is an ethernet or Block Multiplex Channel board. If the board responds, it is a memory board.
   - If this is the first memory board, DBP allocates the memory Mapping area.
   - If there is no memory board (or timing and control board), DBP issues "WHERE'S MEMORY" error message, turns ON the front panel service LED and halts.

4. On the second pass, the DBP issues an IDENTIFY command to each slot starting at Slot 0. If the slot times out, there is no I/O Register. If the slot responds, the DBP accepts the contents of this I/O Register as the board's ID.

   - For each disk controller (SMD or SMD-E), assign "CTLR" 0 through 3, in ascending order from slot 0.
   - For each type of channel I/O board, assign "CHNL" 0 through 7, in ascending order from slot 0.
   - When polled, conduct your own self-test before responding to the IDENTIFY command. If it is unable to respond, pass PCB type, firmware revision level, and fault code to the I/O Register.
   - If "Fault Code = 0", the board believes it is ready to work. DBP immediately prints one of these messages on the console:

```
Slot s: Serial chanl rev nn
Slot s: Parallel chan Controller rev nn
Slot s: Parallel Host Controller rev nn
Slot s: Ethernet chan rev nn
Slot s: BlkMux chan rev nn
Slot s: t&c rev nn
Slot s: dc rev nn
Slot s: tpc rev nn
Slot s: da rev nn
Slot s: dbp rev nn
Slot s: 10MHz dbp rev nn 256K RAM
Slot s: 256K Mem
Slot s: 512K Mem
Slot s: 1Meg Mem
```

   Slot "s" means the actual slot number. Revision "nn" refers to the revision level of the firmware contained in the board's PROMs.

   - When memory Boards are identified the DBP prints the name then tests the board. Each board's test runs for about five seconds.

```
READ 0's WRITE 1's (from low to high address)
READ 1's WRITE 0's (from low to high address)
READ 0's WRITE 1's (from high to low address)
READ 1's WRITE 0's (from high to low address)
```

Two different messages can appear during the test:

```
Correctable Memory Error: pg x - y: ic: z
Uncorrectable Memory Error: pg x - y
```

Memory is mapped out of the system in 2K byte sections. The total amount of memory that is mapped out is reported for each board. If any memory is mapped out, the BL700 may still be used BUT the defective memory board should be fixed at the earliest opportunity.

- If the I/O Register "Fault Code" (status) is 1, it means the board detected an error earlier but is now operational.

- If the I/O Register "Fault Code" (status) is 2-7, it means the board is unusable. The front panel service LEDs will turn ON, and a "defective board" message will appear on the slot line. Even so, the DBP continues the slots process, attempting to boot.

- With an unusable board, this error form appears with one of the following messages.

```
[Fault Code (2-7)] error message
Slot s: board rev nn    ***DEFECTIVE BOARD***
```

The error message is in English. The second line shows the faulting slot and board located in the slot. The list of error messages and their meanings are listed below.

ERROR MESSAGE	DESCRIPTION
`Unknown board ID`	No memory; ID not one of the above
`Channel without Memory`	No memory but I/O Register claims it is a Channel board
`Channel Mem with bad ID`	Channel memory but I/O Register does not have valid ID
`Channel Mem with no I/O`	Channel memory, but timeout on I/O write
`Unknown`	Board claims it is a DBP
`Board not responding`	Channel or Controller did not respond or did not respond in time
`Board selfcheck code`	1, 2, 3, 4 Bit oriented words describing problems These are defined later.
`Where's memory`	Either Timing & Control or memory or both are missing.
`Defective board, Code #`	Catchall for "none of the above"

For each message above, the front panel service LEDs turn on.

5. MAINT Self-diagnostics conclude with the maintenance mode prompt on the console and maintenance terminals. This prompt is

```
BL700 - Filename:
```

where # is 0, 1, or 2. The DBP waits for input from the console port, maintenance port, or the movement of the front panel function key to another position. The first character received from either port causes the DBP to read that port until RETURN or LINEFEED is received. The other port is ignored until then. The key has precedence over the ports. If the key is rotated, port input is ignored and dropped.

BL700 SAFE MODE

SAFE validates the state of each connected drive. The operating system is downloaded to main memory from the system disk, occupying about 500K bytes. The Recover program examines every database in all disks for consistency (closed). Each database will be reported as RECOVERED, ALREADY SAFE, or UNRECOVERABLE. Upon completion, the front panel READY and SAFE LEDs turn ON.

1. Rotate the function key from MAINT to SAFE.

- Download kernal into main memory.

```
BL700 - Filename:Kernal 39: 7/23/85 16:16:56
```

- For each disk controller (0-3), report each attached drive.

- For each attached drive, read logic drive number, 0 through F.

  If a drive is present:

  Read cylinder 0, head 0 sector 1 of the drive
  Can sector be read?
  Report as a foreign disk if it cannot be read or is not in a known format.
  Note if the drive is the system disk.

- Print out the drive condition, followed by the list of accessible disks.

NOTE
Comments or error messages concerning disks appear between the "kernal" line and "Accessible Disks" line.

Accessible Disks

NAME	Ctlr	Drive	Low	High	Status
'systemtest, '	0	0	1	73260	System
'user1 '	0	1	73261	146520	
'user2 '	0	2	146521	219780	
'user3 '	0	3	219781	293040	Mirrored User

This is an example of a two disk controller, five disk BL700 system. The low-high columns list "data blocks" allocated to each drive. Occasionally, BL700 systems are realigned for new tasks. That may result in overlapping data block assignments and several system disks assigned to one BL700. The kernal will complain. Invoke DFU to use RENUMBER and SYSDESTROY to clear those two situations.

Mirrored user or system disks are identical on different drives and controllers. Non-mirrored disks have no particular status.

The loading process follows the disk report.

```
loading syscalls
loading front
loading qryproc
loading support
loading sort
loading dumpload
loading compile
loading execute
loading recover
loading listen
number of available dbins: 94
Free process pages: 96
RECOVER Version 028 Wed Aug 14 17:03:41
Recovering system: recovered
Recovering memb: already safe

Recovering aloha: unrecoverable
recovery code deallocated
IDM Version 030 Wed Aug 14 16:16:30
SYSTEM Release 35
CONFIGURE to channel #
```

Only three databases are listed above. The term "recovered" is not stated until a database is actually recovered. "Already safe" means the database had already been closed before the last operating session. "unrecoverable" means that DFU mode MARKDB was invoked to mark this database unrecoverable. This is a tool to quickly restore a system that is down, (for example, a power outage). Very large databases take long a time to recover. By marking those databases unrecoverable, the BL700 ignores them until a more convenient time.

The CONFIGURE statement means all configure tuples concerned with this channel I/O board are downloaded to this channel.

At this point, the front panel READY and SAFE LEDs, which illuminate in the maintenance mode console terminal, is now a BL700 System Monitor. It cannot be used to enter commands. Part of the monitoring function permits sending two control characters, ^a and ^c, to peek at the Channel I/O board program counter (PC) and to see the last 128 bytes sent to the console, respectively.

BL700 RUN MODE

1. Rotate the front panel function switch from SAFE to RUN. The following message

   ```
   Switch in run
   ```

   is reported to the console as a system status report.

2. The System Database Administrator (DBA) will open the system database and permit users to go to work. As soon as the system database is opened, the SAFE LEDs turn OFF. These

LEDs stay OFF as long as any database is open. Typically, the DBA removes the function key for security.

RETURNING TO BL700 MAINTENANCE MODE

1. For a scheduled user shutdown, the DBA usually sends all users a broadcast message stating the system will be going down, and allows them a few minutes to find a stopping point.
2. The front panel function key is inserted and turned from RUN to SAFE. The following message

```
Switch in safe
```

is reported to the console as a system status report followed by

```
nn Processes, nn Active
```

where "nn" is a decimal number. Processes refers to the number of open databases. Active refers the number of active users.

Once the switch is turned to SAFE, no more users can access their databases. Current users are permitted to complete a query in process, others are locked out. Both **Processes** and **Active** decrease as databases are closed and users sign off. If no change in activity occurs in one minute, the last statement is repeated to show the system is still working.

The system database is still open and all users have signed off. The SAFE LEDs turn ON and the function switch may be rotated from SAFE to MAINT.

```
1 Processes, 0 Active

Switch in maintenance
BL700 DBP Dev 33
```

When the switch is turned to MAINT, the READY LEDs turn OFF. The DBP begins self-diagnostics.

```
BL700 - Filename:
```

Once again, you may use the console to conduct maintenance functions.

SCHEDULED SHUTDOWN

1. Rotate the front panel function key from MAINT to OFF. The POWER LED turns OFF as the power supplies are turned OFF. Ac power is still ON. The cooling fans are operating. Peripherals should still be on.
2. Press rear panel ac power switch OFF. The BL700 is in a full shut down state. The ac switch no longer glows; the cooling fans stop.

UNSCHEDULED SHUTDOWN

1. Ac power outage occurs in RUN mode with active users. The current query is "backed out" and ignored. Refer to the Operation Manual (part number 201-1078-xxx) for further details.
2. On power restoration, the BL700 will come up as though you had rotated the function switch from OFF to RUN. Recovery will take much longer than usual because of disruption.

SERIAL AND PARALLEL SELF-DIAGNOSTICS

INTRODUCTION

On startup, the channel identifies itself and then runs the following Channel test: (revision 37 – serial, revision 15 or above – parallel).

1. Test local memory with pattern: 00, F7, 7F, 00. If it fails, issue fault code 6. Then, Channel loops forever writing to location A5A5:
   - Test Number
   - Address
   - Expected Data
   - Real Data

2. Test Interrupt Controller chip, Timer chip, each SI/O, and each DMA chip. Fault code 4 occurs during any failure (see shared memory test).

3. Channel begins normal execution.

   The possible fault codes for the channels are:

   1. Channel previously reset itself due to internal parity error
   2. Channel not responding to DBP
   3. Combines 1 and 2 above
   4. Other channel problem (see Channel self-check codes)
   5. Combines 3 and 4 above
   6. local memory error

SHARED MEMORY TEST

The DBP issues a "test shared memory" command to the channel during startup or when "slots" is invoked. This is the sequence:

1. DBP zeros Shared Memory Address 0000.
2. DBP issues command to Channel's I/O port.
3. DBP polls Address 0000 waiting for it to become –1.
4. DBP timesout after 1/2 second.

If the channel responds, the DBP reports the following line:

```
Channel self-check codes 1, 2, 3, 4
```

(The numbers correspond to defined words below). The channel self-tests to determine if it can read and write shared memory. At the end of the test, the channel writes a –1 to location 0, 3 words of self-check to locations 2, 4, 6, and sets the rest of shared memory to hex AAAA.

First Word: From Shared Memory Test (Shared Memory Address 0002)

BIT	ERROR MESSAGE
0	DMA did not reach final count and it is not requesting service for DMA write.
1	DMA did not reach final count and it is requesting service DMA did not request for DMA write.
2	DMA write generated wrong interrupt.
3	DMA write had early EOP.
8	DMA did not reach final count and it is not requesting service for DMA read.
9	DMA did not reach final count and it is still requesting service for DMA read.
10	DMA read generated wrong interrupt.
11	DMA read had early EOP.
12	DMA read bad data.

Second Word: From Channel Self Check (Serial only) (Shared Memory Address 0004)

BIT	CHIP	ERROR MESSAGE
0	SIO 4	Did not write the same data as read for register 2.
1	SIO 5	Did not write the same data as read for register 2.
2	SIO 6	Did not write the same data as read for register 2.
3	SIO 7	Did not write the same data as read for register 2.

Third Word: From Channel Self Check (Shared Memory Address 0006)

BIT	CHIP	ERROR MESSAGE
0	9513	Did not write the same data as read for data register F0D1x.
1	9513	Did not write the same data as read for data register F0C1x. NOT TESTED ON PARALLEL BOARDS.
2	DMA 0	Did not write the same data as read for the first byte from a following address register: F000x, F002x, F004x, F006x.
3	DMA 0	Did not write the same data as read for the second byte from a following address register: F000x, F002x, F004x, F006x.
4	DMA 1	Did not write the same data as read for the first byte from a following address register: F010x, F012x, F014x, F016x. NOT TESTED ON PARALLEL BOARDS.

Third Word: From Channel Self Check (Shared Memory Address 0006)

BIT	CHIP	ERROR MESSAGE
5	DMA 1	Did not write the same data as read for the second byte from a following address register: F010x, F012x, F014x, F016x NOT TESTED ON PARALLEL BOARDS.
6	DMA 2	Did not write the same data as read for the first byte from a following address register: F020x, F022x, F024x, F026x. NOT TESTED ON PARALLEL BOARDS.
7	DMA 2	Did not write the same data as read for the second byte from a following address register: F020x, F022x, F024x, F026x. NOT TESTED ON PARALLEL BOARDS.
8	9519	Did not write the same data as read for data register.
9	9519	Generated at least one incorrect interrupt.
10	9513	Got no interrupt or incorrect interrupt for interval timer 0.
11	9513	Got no interrupt or incorrect interrupt for interval timer 1.
12	SIO 0	Did not write the same data as read for register 2 – OR – Did not write the same data as read for addr reg. of GPIB.
13	SIO 1	Did not write the same data as read for register 2. NOT TESTED ON PARALLEL BOARDS.
14	SIO 2	Did not write the same data as read for register 2. NOT TESTED ON PARALLEL BOARDS.
15	SIO 3	Did not write the same data as read for register 2. NOT TESTED ON PARALLEL BOARDS.

Fourth Word: Filled with AAAA, starting Shared Memory Address 0008

The DBP checks that remainder of shared memory is set to AAAA. If any location is not set, the DBP reports the value.

TIMING AND CONTROL SELF-DIAGNOSTICS

INTRODUCTION

The Timing & Control Self-Diagnostic is present in Revision 3 (or higher) memory timing and control boards. The Timing & Control board is tested to check its ability to properly handle single and double bit memory errors. A five number error code is printed if it fails some part of the test.

The first number (0 to 71) is the bit in which the error was detected. The second number is a code identifying the type of error. Problems are grouped into data Single Bit Errors (sbe) and Double Bit Errors (dbe), and check bit errors (cbit).

BITS

The data bit errors (bits 0 – 63) are:

0	–	sbe:	A Diagnostic Write changed the data causing a correctable memory, uncorrectable memory or read data parity trap.
1	–	sbe:	No-op Diagnostic Write caused word %d to change from %x to %x.
2	–	sbe:	Syndrome pattern is 0.
3	–	sbe:	Reported that checkbit %x was corrected.
4	–	sbe:	Syndrome %x indicates a multi-bit error.
5	–	sbe:	MST1 not seen.
6	–	sbe:	Expected syndrome %x, got %x.
7	–	sbe:	Caused read data parity trap.
8	–	sbe:	Correction failed: word %d expected %x, became %x.
9	–	dbe:	MST2 not seen.
10	–	dbe:	Write not cancelled; word %d expected %x, became %x.
11	–	dbe:	Caused read data parity trap.

The check bit errors (bits 64–71) are:

0	–	cbit:	MST1 not seen.
1	–	cbit:	Reported wrong syndrome type, bitsum %x ! = 1.
2	–	cbit:	Syndrome reported that checkbit %d failed.
3	–	cbit:	Caused uncorrectable memory error.
4	–	cbit:	Caused dead data parity trap.
5	–	cbit:	Error changed data; word %d was %x, became %x.

Some error messages have "%x"s (hex number) and "%d"s (decimal number). These give additional information such as an address in a quadword, or the value that should have been there. The first "%" is the third number of the error message. The second % is the fourth number. The third % is the fifth number.

Thus, "30, 8, 2, A0, B0" means that while testing bit 30, the correction logic failed and changed word 2 from "A0" to 'B0".

A defective timing and control board prevents normal execution of the system software, though programs may run if the front panel switch is in maintenance mode.

DBP LEDS

INTRODUCTION

This section describes the six LEDs located on the Database Processor (DBP). The LEDs indicate the status of various BL700 components. A detailed explanation of each LED is listed below.

RUN: Driven by the combination of:
1. BUSAK: Bus Acknowledge (Z8002 pin 24)
2. DS: Data Strobe (Z8002 pin 17)

Note: RUN LED is also on in MAINT Mode.

LMEN: DBP Z8002 "MICRO OUT", renamed "Local Memory Enable", is controlled by software. The logic state shows if the current memory address is on board the DBP or in System memory. DBP memory consists of 16K 16-bit words of firmware and 2K 8-bit bytes of RAM. LMEN is on when local memory is selected.

BGRN: "Bipolar Grant" is driven by DBP Bus Master arbitration logic to permit data accelerator access to BL700 main memory. BGRN and CGRN cannot be active at the same time.

USAGE:
1. Allows BL700 Data Bus (DB) data to enter the Address Bus (AD).
2. AD Bus data is passed to the Local Data Bus (LD) to be written into MMU window.
3. The data accelerator is the only board using this signal.

CGRN: "Controller Grant" is issued by the DBP in response to CREQ (Controller Request). This means a disk controller or tape controller is Bus Master and can initiate memory operations. CGRN and BGRN cannot be active at the same time.

NMI: "Non-Maskable Interrupt" sources:

PRIMARY ERRORS
1. Address parity
2. Read/write data
3. Uncorrected memory error

SECONDARY ERRORS
1. BACOK "buffered AC ok"
2. Correctable memory error
3. Memory time out
4. I/O time out
5. Abort access
6. Read only
7. Odd word
8. Stack limit

NOTE

It is permissible for NMI to flash several times during slots, however this will not occur once the BL700 is booted. The BL700 is in SAFE or RUN mode.

FAIL: TWO NMI's occurred without FCLR (Fail Clear) being issued by the DBP, or HFLT (Hardware Fault) has been set. The fail light may be reset by:

1. RESET on DBP
2. BREAK reset (REMOTE enabled diagnosing)
3. Power on RESET
4. Exchanging the board causing the "FAIL" indication

DISK CONTROLLERS AND TAPE CONTROLLERS POWER ON SELF-DIAGNOSTICS

INTRODUCTION

Any disk or tape controller revision 27 or a later revision, can be issued a controller "test memory path" command. If the board does not respond to the test or if the controller board detects a problem with its path to main memory, the following message appears:

```
Board not responding
```

The controllers fault code is printed.

The disk and tape controllers will perform a self check. All tape controller (TPC's) can loopback. The self check for both boards consists of three parts:

- Idle mode and power-on diagnostics
- Memory interface diagnostics
- Transport Interface diagnostics

USAGE:
1. Allows BL700 Data Bus (DB) data to enter the Address Bus (AD).
2. AD Bus data is passed to the Local Data Bus (LD) to be written into MU window.
3. The data accelerator is the only board using this signal.

IDLE MODE AND POWER-ON DIAGNOSTICS

2901 Check

1. Test
2. Internal and external register test

FIFO Check

The controller does checks while it is waiting for a command from the DBP or at power ON. If the tests pass, a self check code of '0' is issued with an IDENTIFY command. If any test fails and it is performing power on diagnostics, the status is updated with the correct self check code installed. Then the Controller executes a JUMP*. If a test fails and it is performing Idle Mode diagnostics, the controller executes a JUMP*.

The self check codes occurring during Idle Mode and power on Diagnostics are:

1. An error occurred before but the board is working now
2. A problem with the 2901 system
3. A problem with internal and external registers
4. A problem with FIFO

MEMORY INTERFACE DIAGNOSTICS

Virtual K-mode Writes
Virtual Q-mode Writes and Reads

This set of diagnostics is a Direct Command (11 dec) with the Most Significant Byte being the Virtual Address of a 64 word buffer in memory. If the test passes, the self check code is updated with '0' and the memory buffer is zeroed out, except for the first word which is a -1. If the test fails, the IDENTIFY response is updated with controller number, revision level, and self code. The busy bit is cleared and then the controller executes the idle loop.

The self check codes are:

5. Memory Data Error
6. Memory Timeout
7. Off board Error Interrupts

TROUBLESHOOTING ETHERNET HARDWARE

INTRODUCTION

This section will help you determine the cause of a problem and recommend some possible courses of action.

TEST PREREQUISITE

1. The Ethernet Interface (205-1325) consists of:
   - **A.** Ethernet Adapter PC Board (133-0812)
   - **B.** Ethernet Mechanical Interface Assy (112-1215)
   - **C.** Transceiver cable 15.25m (50') (428-0010-001)
   - **D.** Transceiver Unit (428-1246-xxx)
   - **E.** Ethernet XNS Software
     1. Magnetic Tape 1600 bpi (142-1230)
     2. 8" Diskette (CP/M 2.2) (142-xxxx)
     3. 5-1/4" Diskette (IBM PC-DOS) (142-xxxx)
2. The Ethernet Troubleshooting Kit consists of:
   - **A.** Loopback connector
   - **B.** Ethernet console cable (113-1139-000)
3. Two adm3a or equivalent terminals are required
   - **A.** Terminal attached to console port of DBP cable (113-1139-000)
   - **B.** Terminal attached to console port, ethernet cable

PROBLEM

Host will not communicate with the ethernet channel.

SOLUTION

- **A.** Verify correct address in Configure relation and host.
- **B.** Verify:
  1. Ether file is loaded into ethernet board during boot.
     a. Message on console.

     ```
     "Loading Ethernet code into board slot x"
     ```
  2. DBP revision 33 or higher.
  3. IDM/RDBMS software release 43 or higher.
  4. Host software release 3 or higher.
- **C.** Verify cables are connected between the following:
  1. Ethernet Board and BL700 Interface sub-panel.
  2. BL700 ethernet interface sub-panel and transceiver.
  3. Transceiver and Local Area Network Coax.
- **D.** Verify DC voltage is greater than 9.8 volts between pins 6 and 13 on the ethernet cable at the transceiver end.

E. Run ethertest diagnostics.

1. Part of ethernet board PROM set

F. If ethertest diagnostics fail, replace in order:

1. Ethernet board
2. Ethernet tranceiver
3. Ethernet external cable
4. Ethernet internal cable

ETHERNET ADDRESS CONFIGURATION

Xerox Corporation assigned 8002C000000 hex as Britton Lee's XNS base address. This address is now part of our ethernet software. A BL700 can support up to four ethernet channel boards at one time. A unique ethernet address is found by multiplying a BL700's serial number by four. Up to four addresses are available to one BL700.

A decimal number will be written into the BL700 "configure" table. A hexidecimal number will be written into the host as the equivalent host address.

Follow steps 1–5 to determine the unique ethernet address. A BL700 Serial number is included as an example.

1. Multiply the BL700 serial number by 4 and convert that number to hex.

   BL700 serial 05379411 X 4 = 21517644 dec = 148554C hex

2. Break the hex number into two parts. The LSB will have four characters. The MSB will have three characters.

   148554C hex : MSB = 148 LSB = 554C

3. Now convert the LSB to decimal. The four hex numbers will usually convert to five decimal numbers. You will enter this result in the Configure relation of ethernet for LSB in the "Value" field.

   LSB = 554C hex = 21836 dec

4. Convert the MSB to decimal. The three hex numbers will convert to three or four decimal numbers. You will enter this result in the Configure relation of ethernet for MSB in the "Value" field.

   MSB = 148 hex = 328 dec

5. Change the BL700 to maintenance mode, invoke Disk Formatting Utility (DFU), and use Configure mode to add two tuples illustrated next.

Type	Value	Number
(LSB) E	1	21836
(MSB) E	2	328

6. Host address [8002C00000hex + (BL700 serial # X 4 in hex)] is entered through the Host System console.

   8002C000000 + (05379411 X 4 hex) = 8002D48554C hex

BL700 ETHERNET DIAGNOSTICS

Prerequisites

A. BL700 in maintenance mode

B. Ether cable terminated at transceiver end

C. BL700 console connected to DBP console port

D. Ethernet console connected to ethernet board port 0

Procedure

Turn the BL700 front panel Function key to maintenance mode and observe both console and ethernet terminals. The following illustrates a typical sequence with relevant instructions. Descriptions of messages are listed below.

NOTE
Interrupt tests or loops in progress with <RETURN>.

1. Reset DBP and Ethernet boards.

BL700 CONSOLE	ETHERNET CONSOLE
BL700 DBP Rev 33 slot1:1Meg Mem slot2:T&C rev3 slot9:ethernet chan rev 1 slot13:dbp rev 30 slot15:dc rev 28	Etherboot Version 1.3 Tue OCT 16 Monitor Version 1.1

2. BL700 - Filename:loopback 9

If no error is found, BL700 prompt appears on the console. If an error code appears on the console, refer to Appendix G.

```
BL700 - Filename:
```

3. > teste<RETURN>

```
ETHERTEST Version 1.2
```

4. E> t 0<RETURN>

```
ttttttttttttttttttttttttttttttttt
ttttttttttttttttttttttttttttttttt
ttttttttttttttttttttttttt
```

(transmits to the transceiver chip until interrupted)

5. E> loop 0

```
.....
.....
.....
```

loops to the ethernet transceiver chip until interrupted)

#6. E> setchip<RETURN>

```
xmit stat: AOx:<RETURN>
xmit mask: 0x:<RETURN>
rec  stat: 0x :<RETURN>
rec mask: 80x :<RETURN>
tmode : 2x:<RETURN>
rmode : @9x:<RETURN>
node: 0: <RETURN>
node: 2: <RETURN>
node: 3: <RETURN>
node: 4: <RETURN>
node: 5: <RETURN>
```

BL700 CONSOLE	ETHERNET CONSOLE
	#7. E> t 0<RETURN>
	ttttttttttttttttttttttttttttttt ttttttttttttttttttttttttttttttt ttttttttttttttttttt (transmits to the terminator until interrupted)
	#8. E> loop 0<RETURN>
	.... (loops to the terminator until interrupted)

EXPLANATION OF TEXT STEPS

1. Resetting DBP (or control panel) and ethernet boards places everything in a known state before starting test.

2. (DBP console) BL700 - Filename: Loopback 9<RETURN>.
   The ethernet board in slot 9 is going to do a loopback test.

   NOTE

   DBP console and ether console do not appear on the screens. They are stated here for clarity.

3. (Ether console) >teste<RETURN>
   ETHERTEST Version 1.2
   E>
   Entering "test" through the ethernet console brings a new response.

4. E> t 0<RETURN>
   ttttttttttttttttttttttttttttttttt
   This command makes a packet consisting of STATE VARIABLES and transmits it "#" number of times. Number "0" means continue until interrupted. Each time a cycle completes, another "t" appears on the ether console. You may stop the test by pressing <RETURN>.

5. E> loop 0<RETURN>
   .....
   This command will cause the channel to transmit and receive packets and loop to itself. Each successful pass causes a dot to be printed on the ether console. You may stop the test by pressing <RETURN>.

6. E> setchip<RETURN>
   xmit stat: AOx:<RETURN>
   This command is used to set new values in the Fujitsu ethernet chip. "Setchip" returns a series of questions. Respond with the values given here. The chip starts with a default set. You are "programming" the chip to prepare for new testing.

7. E> t 0<RETURN>
   ttttttttttttttttttttttttttttttt
   This repeats the first ether test with another set of values.

8. E> loop 0<RETURN>

   This command will cause the channel to transmit and receive packets and loop to itself using the new values.

   When you have completed the ethernet testing, type "exit" to leave. Turn the front panel Function key to OFF. Turn the ether console power OFF. Remove the ether console signal cable.

   Turn the front panel Function key to MAINT and wait for the prompt. Turn the key to SAFE and wait for the SAFE LEDs to turn ON. Turn the key to RUN and wait for Configure to Channel.

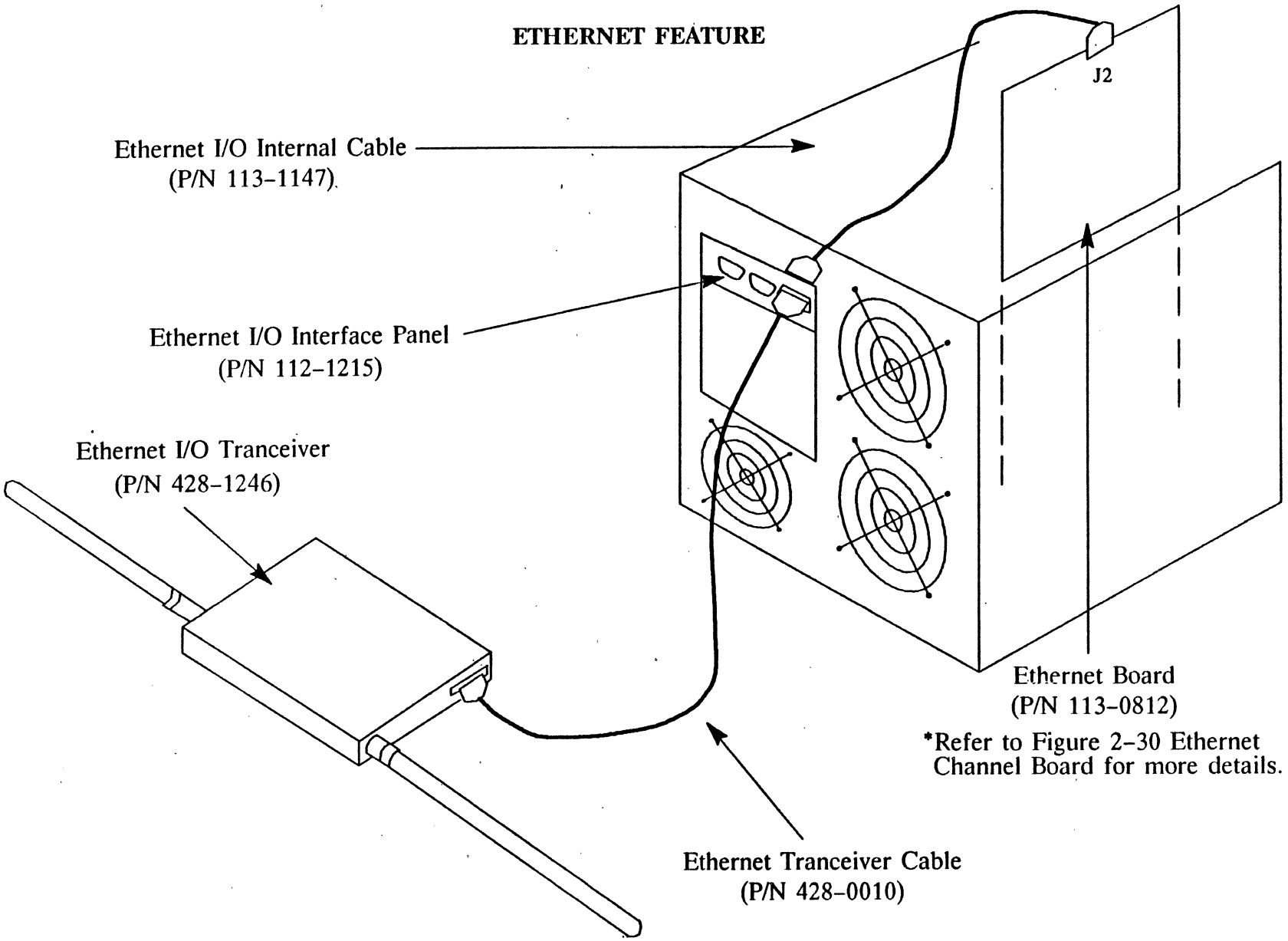


Figure 4-1: Ethernet Feature

ETHERNET CONNECTION

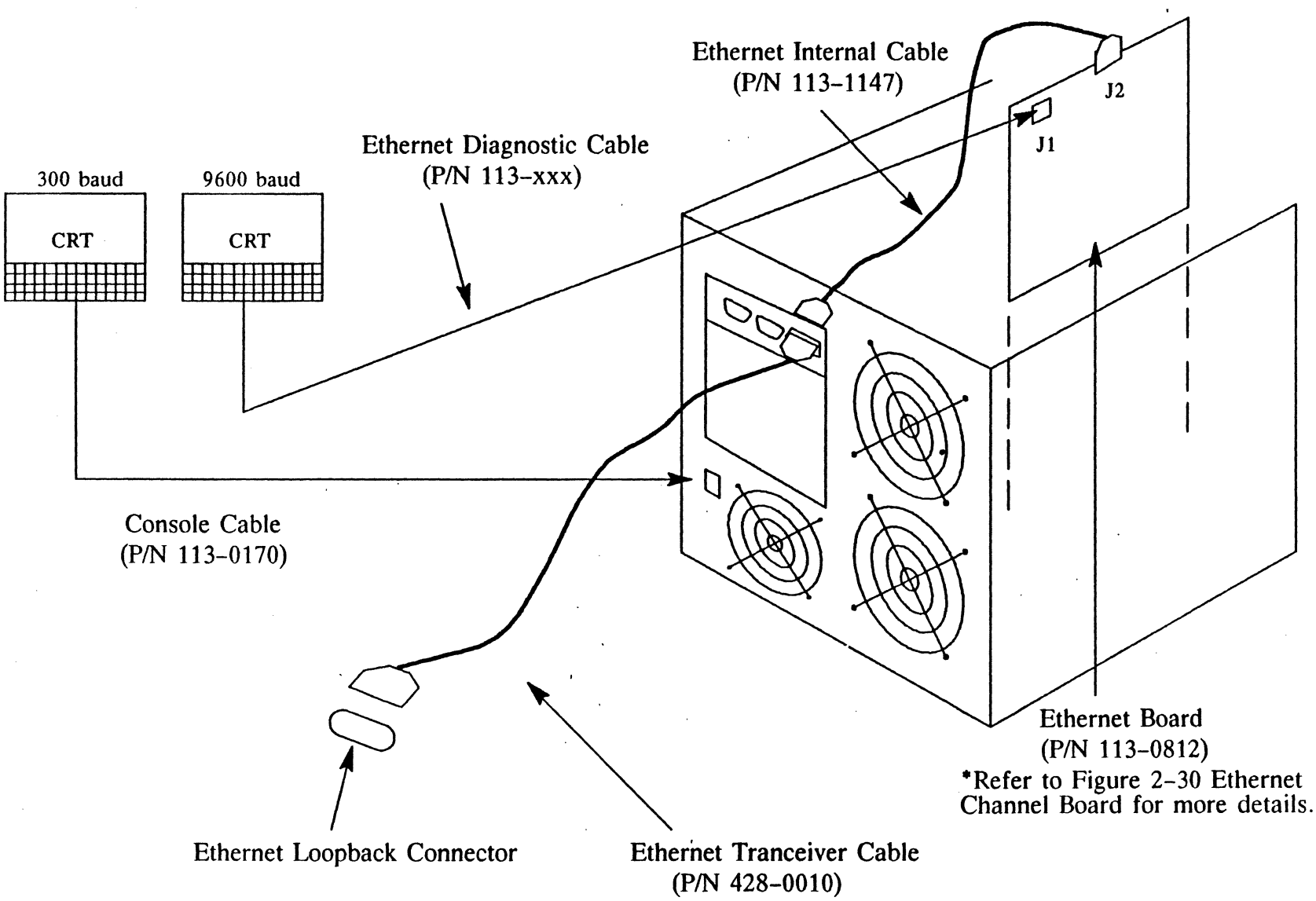


Figure 4-2: Ethernet Connection

LOOPBACK FOR SERIAL CHANNEL

The loopback command exists on revision 25 DBP or a later revision and serial channels revision 37 or a later revision.

TEST PREREQUISITES:

1. Disconnect all hosts from the channel to be tested.
2. The loopback test assumes that the technician has connected a loopback connector on each of the eight ports on the Serial board.

   NOTE

   Any ports not looped back will fail the test. This applies to any hosts connected to a port.

3. The loopback connector is an RS-232C male connector, DB25. Connect the following pins:

   pin 2 to pin 3
   pin 4 to pin 5
   pin 8 to pin 20

TEST PROCEDURE:

The loopback command is

```
BL700 - Filename: loopback nn <return>
```

where nn is the slot number indicating the location of the channel board.

The channel tests all eight serial ports and reports any errors on the console. A timeout will occur if the channel does not respond within 500 miliseconds. If the board discovers a defective port the message

```
port x failed - code yyyy
```

appears. Verify that the loopback connector is present and properly separated on the board and distribution panel. Any port that fails the loopback test is identified by the DBP with code yyyy (hex).

NOTE

Check voltages before replacing the board.

A board that is operating properly returns the message

```
BL700 - Filename
```

without additional comment.

POWER SUPPLY TROUBLESHOOTING

INTRODUCTION

This section will assist the technician in determining the probable cause of a power supply failure and possible course of action.

WARNING
High voltage is present. Use caution when servicing power supply area.

BL700

PROBLEM	DESCRIPTION	PROBABLE CAUSE
Fans not operating	No ac power	12, 13, 1
Console Error AC Low	ac power to the BL700 is below minimum level required for operation	1
(CONSOLE MESSAGE) Ac low system recovered	BL700 Control Panel sensed transient brown out condition but was able to recover.	1
(FRONT PANEL INDICATORS) PS1 FAIL	+5V out of tolerance	2, 5, 1, 10, 9, 6, 7, 3, 8, 16
PS2 FAIL	+12V, -12V, or -5.2V out of tolerance	1, 2, 10, 9, 6, 7, 11, 4, 8, 15
BL700 ONLY PS1 FAIL	+5V, +2V, -12V, or -5.2V out of tolerance	2, 5, 1, 10, 6, 7, 9, 11, 3, 8, 15

BL700

PS1	Low or out of tolerance +5V, +12V, -12V, -5.2V	2, 5, 1, 10, 6, 7, 9, 11, 3, 8, 15

PRE-FCC Britton Lee Database Servers

(Serial Number 05189311 and below)

PROBLEM	DESCRIPTION	PROBABLE CAUSE
Fans not operating	no ac power	12, 13, 1, 14
Console Error AC Low	ac power to the BL700 is below minimum level required for operation	1
(FRONT PANEL INDICATORS) PWR FAIL	+5V (PS1) out of tolerance +12V, -12V, -5.2V, or -2V (PS2) out of tolerance	2, 5, 1, 10, 6, 7, 3, 8, 9, 11, 4, 8, 15
PWR FAIL	Low or out of tolerance +12V, -12V, -5.2V	2, 1, 10, 6, 7, 9, 11, 4, 8, 15

PROBABLE CAUSES

1. Verify and correct line voltage source.
2. Verify or correct dc voltages within tolerance.

   DC voltage checks on the motherboard (below and behind power supplies).

Britton Lee database server - Pre Fcc

+5.00V (+/- .05V)	V1 on power supply 1 adjust to +5.00
-5.20V (+/- .05V)	V1 on power supply 2 adjust to -5.20
+12.00 (+/- .20V)	V2 on power supply 2 adjust to +12.00V
-12.00 (+/- .20V)	V3 on power supply 2 adjust to -12.00V

*With TM-11 current booster, -5.2V dc measures -5.35 (+/- 0.05V)

Early versions where PS2 has -2.00V the output should be the following:

-5.20V (+/- .05V)	V1 on power supply 2 adjust to -5.20V
+12.00V (+/- .20V)	V2 on power supply 2 adjust to +12.00V
-2.00V (+/- .20V)	V3 on power supply 2 adjust to -2.00V
-12.00V (+/- .20V)	V4 on power supply 2 adjust to -12.00V

BL700

+5.00V (+/- .05V)	V1 on power supply adjust to +5.00V
-5.20V (+/- .05V)	V2 on power adjust to -5.20V
+12.00V (+/- .20V)	V3 on power supply adjust to +12.00V
-12.00V (+/- .20V)	V4 on power supply adjust to -12.00V

NOTE

Do not attempt to adjust the current limit potentiometers; PRESET AT THE FACTORY.

3. Replace PS1.
4. Replace PS2.
5. Verify cable (J4) connection from the Control Panel to the DBP board. Replace it if it is necessary.
6. Both power supplies contain thermal sensors which activate power fail circuitry if the internal fans do not operate. Verify that the power supply fans are operating. SP -SEEMS CLUMSY, AWKWARD.
7. Verify that three fans are operating on the rear of the BL700.
8. Verify that the Transformer (T1) is operating. Confirm 30V from the outside tape to the outside tape on the secondary.
9. Turn on the fault override switch located on the Control panel (PCB). Check dc voltages.

CAUTION

Do not leave override switch in override position.

A. If dc voltages are correct, failure may be an intermittent or faulty Control Panel.

B. The following procedure isolates a power failure caused by a board drawing excessive power.

1. If dc voltages are not correct, turn off ac power.
2. Remove all boards, except the DBP.
3. Recheck voltages.
4. If voltages are correct, install one board at a time. Power must be off when a board is installed or removed.

10. Check for loose wiring and/or connectors in the power supply area.
11. Remove all of the RS-232 connections from the rear panel of the BL700.

A. If PS2 failure persists replace PS2.

B. If PS2 failure disappears, verify RS-232 cables meet Britton Lee pinout specifications.

12. Verify that the main breaker switch is toggled on. Refer to Figure 2-2, rear panel of the BL700.
13. Verify local remote switch on PDU is in the local position. Refer to Figures 2-7, 2-8, 2-9, 2-10, 2-11, 2-12, 2-13, 2-14, and 2-15 for diagrams of PDUs.
14. Check fuse.
15. Call your Maintenance Support Group.

INITIAL POWER-ON SEQUENCE

1. Confirm that the BL700's main circuit breaker (ac power rocker switch on the rear panel) is OFF.
2. Confirm that the BL700's front panel function switch is in the OFF position.
3. Plug the BL700 into the wall outlet.

WARNING

In early versions of the Britton Lee database server, the detachable power cord resembles that of many 115V devices. To avoid damage to equipment, do not connect the female end of this power cord to any 115V device. In addition, do not attach the female end of any other cord to the male receptacle in the BL700 rear panel.

4. Connect the console Terminal to the BL700's console port with the console cable.
5. With the key-operated function switch in the OFF position, toggle the circuit breaker rocker switch at the rear of the BL700 to the ON position. A red light on the circuit breaker switch appears, indicating that ac power is being applied to the BL700. The fans at the rear of the BL700 come on. Dc power is not yet applied to the PC boards. Verify that all rear panel fans are operating. The BL700 must not be used unless all fans are working.
6. While observing the indicator lights on the BL700's front panel, use the key to turn the function switch from OFF to MAINT. This applies dc power to the PC boards, causing the Database Processor (DBP) to perform a self-test. The POWER light on the front panel appears, indicating that dc power is on.

   If the DBP passes the self-test, the READY, SAFE, and SERVICE lights flash once briefly.

   If the DBP fails the self-test, these lights either flash on and off continuously or remain off. If they remain off, it could mean that the Database Processor is defective or missing, or that the ribbon cable from the DBP to the front panel is not connected securely.

   If the FAULT light appears, a message describing the fault appears on the console, and the SERVICE light also appears. Examine the console message to determine corrective action.

WHEN YOU NEED ASSISTANCE

If you have problems with the BL700, call your Hardware Maintenance Group. Please have the following information available when you call.

1. What was going on at the time of failure?
   Most often, a problem occurs when a change of any kind is made to equipment or software.
2. What actually happened?
   - Does the front panel status show an error condition?
     (FAULT, PS1 FAIL, PS2 FAIL, TFAULT)
   - Is the READY or POWER lamp turned off?
   - Is the SERVICE lamp turned on?
   - Does the System console report a problem?
   - What does the BL700 console report? Document all error messages/codes reported on the console.
3. What actions have you taken? Document the steps you followed to resolve the problem and the results.
4. Please have the following information when you call Hardware Maintenance:
   - Unit serial number
   - Model number
   - Number of disk drives
   - Number of tape transports
   - Software release number
   - Host software release number
   - Type of host
   - Name and location of each board (slot #)

If the problem is severe, we may need to dial-up your BL700 to solve the problem. In order to dial-up, the following conditions are necessary:

- An auto answer 300 baud modem (not a coupler) connected to the BL700 maintenance port
- BL700 front panel key switch in position at time of failure (MAINT or RUN)
- BL700 front panel REMOTE switch pressed in
- The modem telephone number
- The Disk and Tape Diagnostics on the system disk (will be loaded into memory)

When the above conditions have been fulfilled, we will be able to call the BL700. We will then be able to help you resolve the problem.

UTILITIES

Utility and Diagnostic Programs

Disk Formatting Utility

Disk Diagnostics

Tape Diagnostics

Check Database

CKDB Command Functions

Errors

UTILITY AND DIAGNOSTIC PROGRAMS

GENERAL

This section contains the BL700 utilities and diagnostics.

DFU	(Disk Formatting Utility)
DD	(Disk Diagnostic)
TD	(Tape Diagnostic)
CKDB	(Check Databases)

Their common thread is the BL700 hardware, in the Maintenance Mode of operation. An Administrator must leave the RUN Mode before any of these tools may be used.

The next subsections define common terms, set up a "model" BL700, and set the stage to start each program. Each program begins with the loading statement. Actual program loading is discussed in Appendix C, Software Loading Procedures.

ERROR MESSAGES

All error messages are located in Appendix G.

ILLUSTRATION CONVENTIONS

In this manual, all examples are indented and set in different type font than text for ease of identification.

All function keys are shown as bracketed <LABELS>, such as <RETURN>.

Each program has its own prompt, identified in the beginning. The DBP MAINT mode prompt is shown with a command to list the directory of the system database files.

```
BL700 - Filename: <RETURN>
```

DBP Maintenance Mode commands are listed and explained in Appendix A.

DEFINITION OF COMMON TERMS

Buffers
: This program has the use of four 2048 byte (1024 word) buffers, named a,b,c, and d. The buffers are used to transfer data to and from memory with read and write indirect commands. See Figure 5-1.

Command Block
: A Command Block (CB) consists of eight words (256 bits) linked together as a block of command information. Fixed bit positions include Return Code, Flags, Command, Drive number, Link to next block, Disk Address, Error Code, Retry Count, and Memory Address. Not all bits are used in each situation. See Figure 5-2.

Command Block String
: A CB String is a string of indirect commands that are linked to each other through memory addresses. The first block points to the second, the second points to the third, etc.; the last one points to the first one. The maximum number of command blocks in a string is 16. See Figure 5-3.

Command Register
: The Disk Controller (DC) Command Register is written into by the DBP and is read by the DC.

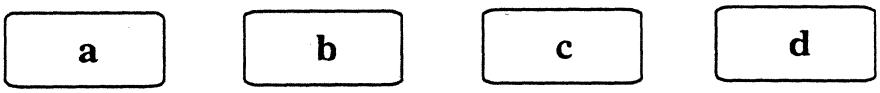
Direct Command
: Direct Commands are commands the DC can execute without getting further data from Main Memory (or does not need to put anything into Main Memory).

Indirect Command
: Indirect Commands are commands where the DC must go to a Main Memory location to obtain or place data for or from the command.

I/O Registers
: The 16-bit I/O Register allows the Disk Controller to communicate with the DBP. It is called the Command Register when DBP writes commands to the DC. The same register is called the Status Register when DC returns status to the DBP.

Status Register
: The Status Register is written by the Disk Controller with contents called "Return Code". The Return Code is updated as DC is executing a command. DBP reads the Status Register upon successful or unsuccessful completion of a command.

DISK DIAGNOSTIC MEMORY BUFFERS



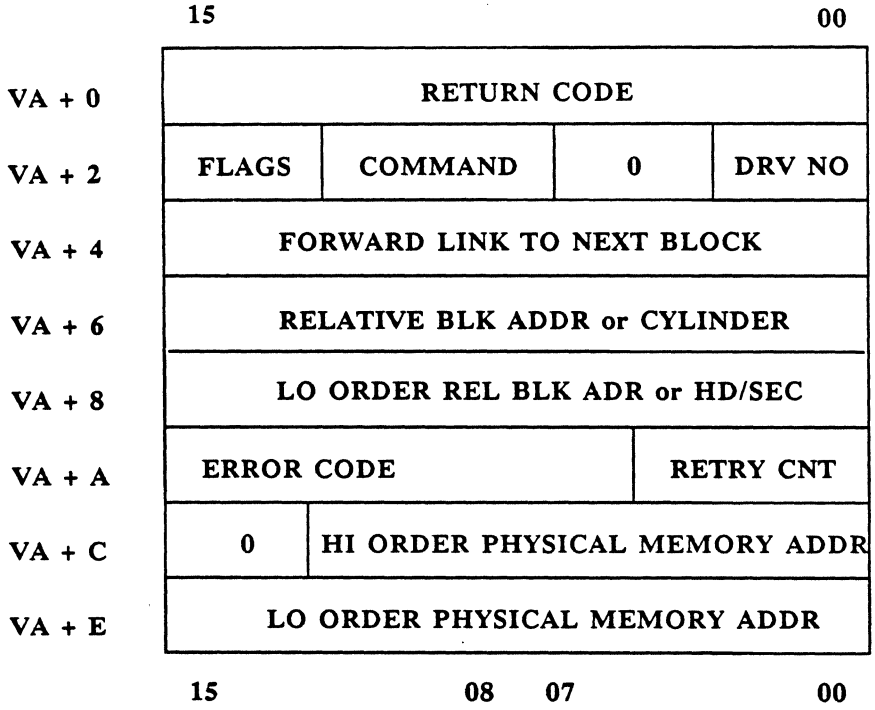
- 1024 words each buffer (2048 bytes)
- Word count in buffer
- Independently addressable
- Select first word (offset)
- Select number of words to display
- Displayed in hexadecimal
- Write to buffer
- Read from buffer
- Transfer buffer contents to sector
- Transfer sector contents to buffer
- Compare buffers
- Look at buffers

a + 0 FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE

a + 10 FEFE

Figure 5-1: Disk Diagnostic Memory Buffers

COMMAND BLOCK



VA = Virtual address

Figure 5-2: Command Block

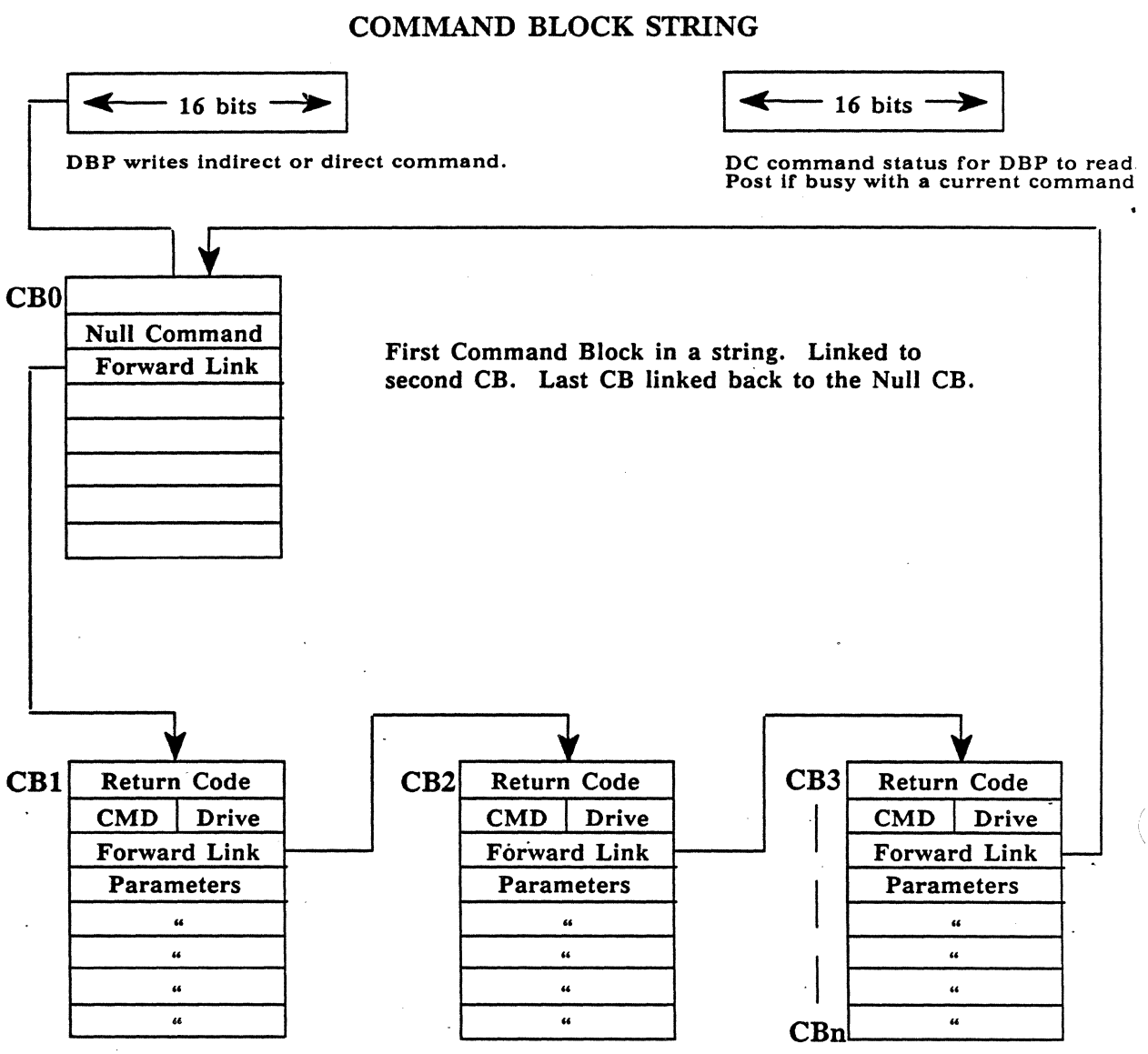


Figure 5-3: Command Block String

DISK PAGE FORMAT

MASTER BLOCK TUPLE (only one)

Bytes 14–46

Byte 14–15 = Disk Type (ASCII in hex)
- P: Promoted
- M: Mirror
- V: Virtual

Byte 16–27 = Disk Name (ASCII in hex)
(fill to end with 20 (space)

Byte 28–31 = DBTID for System Disk only (Rell2).
Non-system disks are zero-filled.

Byte 32–35 = High Block Number of disk in Longbyte hex using leading zeros if necessary.

Byte 40–41 = Blocks per Cylinder (Hds x Sectors/track)

Byte 42 = Sectors per Track

Byte 43 = Blocks per Zone

Byte 44–45 = Cylinders per disk

Byte 46 = Disk address logic plug number

SAMPLE MASTER BLOCK

Word										
							7	8	9	
As seen							50	7379	7374	
Byte							14 15	16 17	18 19	
Data							00 50	73 79	73 74	
ASCII or decimal equivalent							P	s y	s t	
Word	10	11	12	13	14	15	16	17	18	19
	656D	2020	2020	2020	0	D00	1	1E2C	0	1
Byte	20 21	22 23	24 25	26 27	28 29	30 31	32 33	34 35	36 37	38 39
Data	65 6D	20 20	20 20	20 20	00 00	0D 00	00 01	1E 2C	00 00	00 01
ASCII	e m					13	73260			1

Word	20	21	22	23
As seen	5A	9B4	337	0

Byte	40 41	42 43	44 45	46 47
Data	00 5A	09 B4	03 37	00 00
ASCII	90	9 180	823	0

SAMPLE TUPLE NUMBER TABLE

Starts at Word 1023, pointing to first byte of each tuple. Pointers work BACKWARDS. Second tuple pointer would be in Word 1022.
E is hex address for first tuple. Convert hex to decimal.

Word	1020	1021	1022	1023
Data	0	0	0	E

Figure 5–4: Master Block

BLOCKALLOC PAGE HEADER

Word	0	1	2	3	4	5	6
As seen	0	151	0	B5B4	B	446	1

Byte	01 01	02 03	04 05	06 07	08 09	10 11	12 13
Data	00 00	01 51	00 00	B5 B4	00 0B	04 46	00 01

BLOCKALLOC TUPLE (6 BYTES)

Bytes a–f (for all blockalloc tuples)

Byte a = Block number within this zone

Byte b = Status of Block (hex values)

- 02 Free Block
- 04 Used by relation "relid"
- 10 Defective, unused block
- 44 Used by file "relid"
- 84 Used, alternate, relation block
- 90 Defective, remapped file block
- C4 Used, alternate, fileblock
- D0 Defective, remapped, fileblock
- 08 State: can be set on any allocated block

Byte c = S1 (used by CKDB)

Byte d = Stat (used by CKDB)

Byte e–f = Relation ID of assigned relation

SAMPLE BLOCKALLOC TUPLES

Word	7	8	9	10	11	12	13	14	15
As seen	4	100	B	104	0	10	204	0	1

Byte	14 15	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31
Data	00 04	01 00	00 0B	01 04	00 00	01 00	02 04	00 00	00 01
Byte #	a b	c d	e f	a b	c d	e f	a b	c d	e f

SAMPLE TUPLE NUMBER TABLE

a + 840	0	0	0	0	440	43A	434	42E	428	422
.										
.										
.										
a + 1020	20	1A	14	E						

Figure 5–5: Blockalloc

A REFERENCE MODEL BL700

A single model is "constructed" for reference throughout these utilities and diagnostics. All examples are in reference to this model. BLI software release 35 is assumed. All programs and utilities run on this model unless specifically noted in the examples.

Two Disk Controllers each support 2 disk drives. The disk subsystem is defined as follows:

```
Controller 0 Drive 0 'systemtest ' (SysDB-mirror)
Drive 1 'user1 '
Controller 1 Drive 0 'systemtest ' (SysDB-mirror)
Drive 1 'user2 '
Each Drive
Cylinders              823
Heads                  10
Sectors                9
Total Zones            407
Total Blocks           73260
Total Blocks per Zone  180
```

The system database is considered to be residing on the system disk, except when using DFU to format or create a system disk.

One High Speed Tape Controller (TPC) is supporting a reel type tape transport and a cassette type tape transport. Each transport has its own micro-formatter.

At Power ON or RESET, this is the Slots message "appearing" on your Console Terminal. Both the standard DBP (6MHz) and optional DBP (10MHz) are illustrated in the slots message; only one would be present. (Slots 3, 4, 10, and 14 are not in use).

```
Mirrored DBP Rev 33          (6MHz DBP)
Mirrored DBP Rev 33          (10MHz DBP)
Slot 0:1Meg mem
Slot 1:1Meg mem
Slot 2:t&c rev 3
Slot 5:serial chan rev 38
Slot 6:parallel chan rev 28
Slot 7:block mux rev 0
Slot 8:ethernet rev 0
Slot 9:tpc rev 107
Slot 11:da rev 37
Slot 12:dc rev 28
Slot 13:dc rev 28
Slot 15:dbp rev 30M                    (6MHz DBP)
Slot 15:dbp rev 31M                    (10MHz DBP)

BL700 - Filename:
```

At this point, you may change the terminal's baudrate to 9600 baud. Type the following after the prompt:

```
BL700 - Filename: conport 9600<RETURN>
BL700 - Filename:
```

Set the terminal to 9600 baud. Then, press <RETURN> twice to see the system disk contents printed at 9600 baud. (The first time clears the registers of nonsense characters changing from 300 to 9600 baud).

CAUTION

If the DBP issues a comment while you're in the process of changing baudrates, you will miss the message. If nothing happens when you press <RETURN>, press <CTRL><c> together. This presents up to 128 bytes of the last DBP buffer to the Console Terminal. Front Panel status could also be useful.

GETTING READY TO USE ONE OF THE PROGRAMS

If resident, DFU, DD, TD, or CKDB may be loaded directly from the system disk drive into BL700 Main Memory. If not resident, they must be loaded from a source tape or diskette into BL700 Main Memory. Refer to System Loading Procedures in Appendix C.

We assume your version of our "model system" is inter-connected, AC power applied and turned ON, and all connected devices are READY.

Rotate the BL700 Front Panel function key from OFF to the MAINT position. When Self-Diagnostics are complete, the BL700 Maintenance Mode prompt is displayed on your Console Terminal.

```
BL700 - Filename:
```

Now refer to one of the four sections and begin.

DISK FORMATTING UTILITY (DFU)

INTRODUCTION

This chapter is written according to release 35 software. Other versions may not have the same enhancements discussed in this section.

The Disk Formatting Utility (DFU) lets a user format a system disk and up to 15 user disks. DFU is also used to create, recreate, or destroy a system database on any disk. Disks and their databases, newly attached to an existing BL700 system, may be merged into the system. DFU can validate the systemd database, display a list of all disks on line, unformat disks from the system, modify the configure relation, and mapout and remap blocks without reformatting.

When a mirror Database Processor (DBP) is in a BL700, disks may be paired as mirrors of each other. This applies equally to system and user disks.

BL700 DISK ORGANIZATION

A BL700 can be configured with 1 to 4 disk controllers (depending on the model). Each controller supports up to four disk drives. One disk, and only one disk per BL700, must be specified as the system disk, containing the system Database.

The system database contains relations with system level information and files that contain the IDM/RDBMS software. User databases may occupy the remaining space on the system disk. Because of the type of information stored on it, the system disk receives special attention during the formatting process.

Formatting a disk also tests its ability to record data properly and ensures that all operations on the disk will access usable blocks. Formatting each track consists of the following steps:

- Write a known test pattern onto each block of a given track of the disk.
- Read the data from each of the blocks on the track and compare it to the known pattern.
- If the data on the block does not match the known pattern, then the block is marked as bad.

An alternate block in a different area of the disk is designated to replace the bad block. This is known as remapping bad blocks, and is transparent to the user. From this point on, the disk controller will access the alternate block whenever an attempt is made to access the original bad block.

DFU divides each disk of cylinders, tracks, and sectors into zones containing up to 254 blocks of 2048 bytes each. The exact number of blocks per zone is optimized for the particular disk being formatted. The first block in each zone is called the blockalloc and is part of the blockalloc relation. The blockalloc relation contains information on the status of the other blocks in the zone, such as:

- What database owns this zone
- Blocks assigned to each relation
- Blocks that are allocated, free, or bad

After all blocks of a zone are tested, the program writes the blockalloc for the zone in the first good block of each zone. The above steps are repeated for each zone of the disk. The blocks of the disk are assigned numbers. These VIRTUAL block numbers must be unique throughout the system. When initially configuring a BL700, the system disk is assigned block numbers 1 through J, since it is always the first disk to be formatted. However, the system disk can be any range of numbers, not necessarily starting with 1. Additional disks for user databases are assigned block numbers J+1 through K, K+1 through M, and so on.

The lowest and highest block numbers assigned to a disk are recorded in its master block at formatting time. The master block is the second usable block on the disk, normally cylinder 0, track 0, sector 1. But if sector 0 is bad, then sector 1 is assigned to blockalloc and sector 2 is the master block.

A disk's block numbers must be unique, and the range of each disk's block numbers must not overlap the block number range of any other disk. While this is ensured during initial system configuration, it can become a problem if a user replaces a disk in the system with one of larger capacity. The block number gap left by the removed disk will not be large enough to accommodate the range of block numbers needed by the new, larger disk.

To prevent block number overlap between the new large disk and the other disks in the system, the new disk must be assigned block numbers greater than any previously assigned. If this causes the block numbers of the new disk to be greater than the maximum number of blocks allowable on the BL700 (2^{24} -1) the user must remove other disks with the Unformat Mode described below, to create enough space for the new disk.

Each disk must also be given a unique name of up to twelve characters. This allows the BL700 to keep a separate entry for each disk in the data dictionary relation: *Disks*.

CAUTION

Do not use two-word disk names. Delineating marks, such as the spacebar or a blank ( ), a period (.), or an underscore (_) may cause SYSERRs or loss of disk recognition.

DFU protects against the intentional or unintentional destruction of data on the system disk because of the importance of its contents. The user may not directly reformat or unformat a system disk. To reformat the system disk, you must first move the system to another disk using the SysDestroy and syscreate modes. Then Unformat the old system disk and Merge the system.

Much of DFU is intended to speed recovery from the loss of the system database. This can result from loss of the system disk or media failure on the system disk. All DFU modes are described in the DFU examples in this section.

USAGE

A BL700 must be in Maintenance Mode to run DFU. The BL700 reads input from and writes output to the console port in this mode. A console terminal is normally connected to the console port. Alternatively, a host can interface the BL700 console port to one of its terminals.

STARTING DFU

Refer to Appendix C for diskette and tape loading and starting processes.

```
BL700 - Filename: kernal dfu<RETURN>
BL700 - Filename: load kernal dfu<RETURN>
BL700 - Filename: loadtape kernal dfu<RETURN>
```

For all cases, the following response is a reasonable "facsimile" of what you will see on your console terminal appearing directly below your entry. You may see a hard disk error immediately below KERNAL if DFU cannot read the drive's master block on cylinder 0, head 0, sector 1. This means the disk is not formatted.

```
KERNAL 39 7/23/85 16:14:56

Foreign disk, ctlr0, drive0

loading syscalls
loading dfu
loading dfu2
number of available dbins: 94
free process pages: 50      DFU Version 9 Tue Jul 23 17:36:39
DFU Modes:
          (0) Exit
          (1) SysFormat......Format a system disk
          (2) Syscreate......Create system db on a formatted disk
          (3) SysDestroy.....Destroy the system database
          (4) Merge..........Merge user databases into system
          (5) Validate.......Check correctness of system database
          (6) List...........List all disks on line
          (7) Format.........Format a non-system disk
          (8) Unformat.......Remove a disk from the system
          (9) Configure......Add and delete  configure' tuples
          (10) Scan..........Remap and mapout sectors
          (11) MarkDb........Mark recovery status of databases
          (12) Renumber......Change block numbers of unused disk
          (13) MirrorFormat..Format a disk as a mirror of another
          (14) MirrorCopy....Copy data from one mirror to another
          (15) Compare.......Compare data on mirrored disks
          (16) PromoteDisk...Make a single mirror into a formatted disk

Your choice?
```

DFU MODES

Sysformat and syscreate require that the system database not exist. This prevents the creation of more than one system database. List and scan modes can be run with or without the system database. All other modes require the existence of the system database.

NOTE

1. The menu appears again only when you enter "?<RETURN>" in response to the prompt, "Your choice?".

2. Modes (13) through (16) appear only when a Mirror DBP board is installed in the BL700.

3. The formatting process is the same for sysformat, format, and mirrorformat. Mirrorformat has one difference which will be illustrated. Locate the disk manufacturer's flaw map. You need to use this when a disk is formatted.

4. Flaw maps come with Winchester disk drives and not Removable Packs.

5. You may return to DFU prompt (Your choice?) at any mode prompt by typing "quit<RETURN>".

Each mode is illustrated with the most common examples. Some error messages are also displayed. Select a mode by typing its number. DFU asks you to verify the selected mode. If you answer yes, DFU executes the mode and returns to the DFU prompt after completing the operation. DFU is then ready to execute another mode.

0. EXIT.

EXIT will return control to DBP with the BL700 prompt

```
BL700 - Filename:
```

When you invoke "0", the BL700 system is Reset.

```
Your choice?        0<RETURN>
You chose Exit.
Is this correct?    (y/n) y<RETURN>

DBP Rev 33
Slot 0:1meg mem.....starting self-diagnostics
```

1. SYSFORMAT.

Sysformat must be used first when configuring an BL700 system.

```
Your choice?        1<RETURN>
You chose SysFormat mode.
Is this correct?    (y/n) y<RETURN>
```

DFU prompts for controller number, drive number, disk name, sectors per track, tracks per cylinder, number of cylinders on the disk, and if the disk will be subdivided. (This example assumes the drive sector switches have been set for the correct number of sectors. The BL700 needs to see a minimum of 2240 bytes).

```
***SysFormat Mode***
Enter information on disk to be formatted:
Drive controller number?(0-3)   0<RETURN>
Disk drive number? (0-15)       0<RETURN>
Name of new disk?               systemtest<RETURN>
sectors per track?              9<RETURN>
tracks per cylinder?            10<RETURN>
cylinders per disk?             823<RETURN>
Subdivide disk? (y/n)           y<RETURN>
     Name of subdivision?    sys1<RETURN>
     Last cylinder?             108<RETURN>
     Name of subdivision?    badtracks<RETURN>
     Last cylinder?             114<RETURN>
     Name of subdivision?    sys2<RETURN>
     Last cylinder?             815<RETURN>
```

DFU asks if the disk should be subdivided into virtual disks. Subdividing the disk makes it look like more than one disk. If the user desires this, sysformat prompts for the name of each subdivision and the high block number of each subdivision. Subdivision cylinders must equal cylinders per disk minus the remap reserved area which is 1% of available disk space. DFU will prompt if you exceed the highest cylinder number. This example suggests a possible use for virtual disks by assigning subdivision #2 with 6 allegedly bad tracks.

```
Disk will be formatted as follows:
Name of disk..... 'systemtest  '
Disk controller number.. 0
Disk drive number..... 0
sectors per track..... 9
tracks per cylinder..... 10
cylinders per disk..... 823
Disk subdivisions:
          Subdivision #1: name = 'sys1          ' highest cylinder = 108
          Subdivision #2: name = 'badtracks     ' highest cylinder = 114
          Subdivision #3: name = 'sys2          ' highest cylinder = 815
Is this correct? y/n y<RETURN>
```

If you enter n', you will reenter data starting with the drive controller. Or, type "quit<RETURN>" to return to "Your Choice?". If the specified disk is already formatted, sysformat displays a warning message and asks if the format procedure should continue.

NOTE

If the disk controller is at least Rev. 27, DFU asks whether special formatting patterns are required. This allows more thorough testing of the disk's magnetic properties and helps detect defective sectors. The user may enter up to four hexadecimal patterns of four digits each.

```
Do you wish special formatting? y/n y<RETURN>
Enter up to 4 patterns consisting of four hex digits
Pattern: d9b3<RETURN>
Another pattern? y<RETURN>
Pattern: 366c<RETURN>
Another pattern? n<RETURN>
2 patterns entered
Pattern 1 = D9B3
Pattern 2 = 366C
Is this correct? (y/n): y<RETURN>
Disk divided into 407 zones.
```

The formatting process divides each disk into zones and blocks. In this example, 407 zones will be reported in the master block. A disk's sector contains a block of 2048 data and 192 overhead bytes. The exact number of blocks per zone is determined by the format program, optimized for the kind of disk being formatted. Note that DFU's own formatting pattern (DB6C) follows any requested special patterns.

DFU displays the approximate time to complete formatting. The time is based on 3600rpm, and is incorrect for other speeds. About every fifteen seconds, DFU displays the number of cylinders left to be formatted.

> Approximate format time (at 3600 rpm): 17 minutes, 45 seconds with 2 special patterns. 814 cylinders left to be formatted.

Any corrupted block (sector) found during format is locked out. RESERVED area corrupted blocks are mapped out. WORKING area corrupted blocks are remapped to the reserved area. (Two examples are given for clarity). Mark all errors on the disk manufacturer's flaw map.

```
Disk hard err code xC000
cmd 14, ctlr 0, drive 0, cyl 818, head 5, sec 4
          errcode 101, errstat 0, phyaddr 0
cyl 818, head 5, sec 4 mapped out.

22 cylinders left to be formatted.
Do you wish to mark any sectors as defective? y/n y<RETURN>
```

When DFU has finished formatting the surfaces, it politely reminds you now is the time to use the disk's flaw map. A flaw map has columns to identify cylinders, heads (or tracks), and byte position from index. The map may also show the number of bit errors at that byte position. If the number of bits is reported, then only remap flaws with 3 or more bits not already remapped by DFU. Calculate the flawed sector with this formula:

sector # = byte position/2240

(The constant is BL700 2K sector with 192 byte overhead). Discard any fraction. Numbering starts with sector 0. If this disk's flaw map lists cylinder 27, head 4, and 12512 bytes, then 12512/2240 = 5.587. Discard 0.587 and use sector 5.

NOTE

Century Data systems ams-513 drives use 2270 bytes per sector.

```
Enter cylinder?    27<RETURN>
Enter head?        4<RETURN>
Enter sector?      5<RETURN>

Cyl 27, head 4, sector 5, remapped to cyl 814,
head 0, sector 1, on Ctlr 0, Drive 0.
Another? y/n n<RETURN>

2 blocks were remapped
1 block was mapped out
```

The last statement includes the two previous mapping examples. Finally, DFU asks whether databases will be created in ASCII or EBCDIC.

```
Databases can be created in ASCII or EBCDIC representation
(Enter  a' for ASCII or  e' for EBCDIC)
Character type for database? (a/e) a<RETURN>
You chose ASCII.
Is this correct? y/n y<RETURN>
```

DFU checks the cylinder and head selection on the drive. If any of the cylinder or head selection lines are shorted or open, sysformat displays a warning message. If all is still well, DFU builds the master block of the disk and writes it on the second good block of the disk. Then DFU creates an empty system Database on the disk.

```
***SysFormat complete***
(Type '?' for menu.)
Your choice?
```

To load the system database, leave DFU with EXIT and load the files from diskette or tape. See Appendix C for procedures.

2. SYSCREATE.

Build a new empty system database on a formatted disk. (Syscreate is similar to the last step of sysformat.) This mode is disallowed if the system database already exists.

This mode can be used to recover from the loss of the system database. After syscreate, the user must run the Merge mode to merge user databases into the system database from the other disks on line.

After running syscreate, the only tuple in the database relation **databases**, is that of the system database, and the only tuple in the relation **disk**, is that of the system disk. Merge mode of DFU must be run to insert tuples for other databases and disks on line, and the system software must be reloaded before the BL700 can become fully operational again.

```
Your choice? 2<RETURN>
You chose Syscreate mode.
Is this correct? (y/n) y<RETURN>
*** Syscreate Mode ***
Future system disk name? systemtest<RETURN>
This disk has zones numbered 0 through 406.
Zone size is 180 blocks.
Lowest zone number of system database? 0<RETURN>
Allocating system database zones: 0 1 2.
Databases can be created in ASCII or EBCDIC representation.
(Enter  'a' for ASCII or  'e' for EBCDIC.)
Character type for database? (a/e) a<RETURN>
Default character type is ASCII.
Is this correct? (y/n) y<RETURN>
*** Syscreate complete ***
(Type  '?' for menu.)
Your choice?
```

Sysformat creates the system database on the first few zones on the disk. However, syscreate will ask where you want to create the system database. Syscreate displays an error message if there is not enough space on the disk for the system database or if there is no disk with the given name.

CAUTION

All system and user relations and files in the system database are destroyed when SysDestroy is run. Syscreate does not regenerate user relations or files.

3. SYSDESTROY.

This mode finds and destroys the system database on a given disk. The disk will still be formatted and will still exist in the Disks Relation with corresponding block numbers. SysDestroy is used to move the system Database to another disk. For example, suppose the disk storage capacity on a BL700 system is being expanded at the expense of other BL700 systems. Several system disks may end up attached to the expanded system. The extra system disks' system Databases may be removed one at a time with SysDestroy.

```
Your choice? 3<RETURN>
You chose SysDestroy mode.
Is this correct? (y/n) y<RETURN>
*** SysDestroy Mode ***
system disk name? systemtest<RETURN>
system disk  'systemtest  ' found.
freeing system database zones: 0 1 2.
*** SysDestroy complete ***
(Type  '?' for menu.)
Your choice?
```

SysDestroy finds, deallocates, and displays a list of all system Database zones deallocated. The disk is marked as a nonsystem (User) disk.

4. MERGE.

Merge is run following syscreate or sysformat when there are other disks on line with user databases. Merge is also necessary when a system has more disk packs (removable media) with databases than disk drives. Each pack must be separately identified.

Merge searches for all online disks, including foreign disks. (A disk is online when it is ON and READY at the last BL700 Reset). Each identified disk is compared with a list in the disk relation. Those not already in the disk relation are merged into the list with their own tuple.

```
Your choice? 4<RETURN>
You chose Merge mode.
Is this correct? (y/n) y<RETURN>
*** Merge Mode ***
Merging disks...
Disk 'systemtest ' found in 'disks'
Disk 'user1      ' found in 'disks'
Disk 'user2      ' merged in 'disks'
1 disks were merged.
```

In this example, 'systemtest' and 'user1' have tuples in the disk relation. Disk user2' is new. A tuple is constructed, added to disk relation, and disk relation status is given.

Each disk in the disk relation is searched for existing databases and compared with the tuples in the database relation. Those not already in the database relation are merged as a new tuple.

```
Merging databases...
Database 'system      ' dbid 1 is in 'databases' relation.
Database 'prospecting ' dbid 2 is in 'databases' relation.
Database 'westernregn ' dbid 3 is in 'databases' relation.
Database 'easternregn ' dbid 4 merged in 'databases' relation.
1 database were merged.
*** Merge complete ***
```

Three databases pre-exist. (As an example, we will call them: "system", "prospecting", and "westernregn".) Easternregn is discovered, a tuple constructed, and added to the database relation. Merge also warns of any incomplete databases for which a tuple could not be formed and of any other problems or inconsistencies.

NOTE

Disks from one BL700 system can be merged into another system. However, this can result in overlapping block numbers, duplicate disk and database names, and multiple system disks. See other DFU modes for correction.

5. VALIDATE.

Validate confirms all online user disks and databases have tuples in their respective system database disks and Databases Relations. Validate locates existing databases, attempts to open them, and checks for any databases or disks recorded in the system database that are not on line.

```
Your choice? 5<RETURN>
You chose Validate mode.
Is this correct? (y/n) y<RETURN>
*** Validate Mode ***
Validating disks on line...
Disk 'systemtest  ' found in 'disks' relation.
Disk 'user1       ' found in 'disks' relation.
Disk 'user2       ' found in 'disks' relation.
Virtual disk 'sys1          ' found in 'disks' relation.
Virtual disk 'badtracks     ' found in 'disks' relation.
Virtual disk 'sys2          ' found in 'disks' relation.
All disks on line are in the 'disks' relation.
Validating 'disks' relation.
Disk 'systemtest  ' is on line.
Disk 'user1       ' is on line.
Disk 'user2       ' is on line.
All disks in 'disks' relation are on line.
Validating databases on line...
Database 'system          ' dbid 1 is in 'databases' relation.
Database 'easternregn     ' dbid 2 is in 'databases' relation.
Database 'prospecting     ' dbid 3 is in 'databases' relation.
Database 'westernregn     ' dbid 4 is in 'databases' relation.
4 databases are in 'databases' relation.
All databases in 'database' relation are on line.
*** Validate complete ***
(Type '?' for menu.)
Your Choice?
```

Validate warns of inconsistencies by displaying messages about missing disks, databases, or respective tuples. However, it does not update the data dictionary relations, disks or databases. Run Merge to insert any disks or databases that are online but not recorded in the system database.

NOTE

Validate should not be confused with CKDB which tests user as well as system databases. CKDB performs a much more thorough verification of databases than DFU Validate.

6. LIST.

List displays disk name including foreign disks, controller number, drive number, block numbers, and status of each disk on line. It also displays the name, block numbers, and type of all disks in the disks relation, including virtual disks.

```
Your choice? 6<RETURN>
You chose List.
Is this correct? (y/n) y<RETURN>
*** List mode ***
Disks on line:
       Name          Ctlr    Drive      Low      High      Status
    'systemtest  '    0        0          1     73260    Mirrored system
    'user1       '    0        1      73261    146520
    'systemtest  '    1        0          1     73260    Mirrored system
    'user2       '    1        1     146520    219780
Disks in 'disks' relation:
       Name             Low     High  Type
    'badtracks   '     82981    83520   V
    'sys1        '     73261    82980   V
    'sys2        '     83521   146520   V
    'systemtest  '         1    73260   P
    'user1       '     73261   146520   P
    'user2       '    146521   219780   P
*** List complete ***
(Type '?' for menu.)
Your choice?
```

7. FORMAT.

This mode is nearly identical to sysformat. An unformatted, or foreign disk is used to create a User disk for User databases instead of a system disk for the system database. Format mode does not ask for ASCII or EBCDIC representation because they are system entities. All user disks follow their system disk. disks formatted with this mode are inserted into the disks relation.

```
Your choice? 7<RETURN>
You chose Format mode.
Is this correct? (y/n) y<RETURN>
*** Format Mode ***
.
.
.
*** Format complete ***
(Type '?' for menu.)
Your choice?
```

8. UNFORMAT.

This mode deallocates blocks belonging to a disk and logically removes the disk from the system by marking it as a foreign disk. The disk's tuple is removed from the Disks Relation. Unformat scans the disk to be unformatted, looking for allocated block numbers which are then freed.

If the specified disk is a system disk, run SysDestroy followed by Unformat.

```
Your choice? 8<RETURN>
You chose Unformat mode.
```

```
Is this correct? (y/n) y<RETURN>
*** Unformat Mode ***
Name of disk to be unformatted? systemtest<RETURN>
WARNING: system database does not exist.
         Unformatting will only mark disk foreign.
Continue unformatting? (y/n) y<RETURN>
WARNING: zone 4 belongs to dbid 2.
Unformatting will destroy all data on this disk.
Continue unformatting? (y/n) y<RETURN>
Marking this disk 'foreign'.
Deleting 'disks' tuple(s).  *
** Unformat complete ***
(Type '?' for menu.)
Your choice?
```

If the disk is not online but has an entry in the Disks Relation, Unformat prompts the operator, asking if the process should continue. If you respond yes, Unformat deletes the disk's tuple and returns to the main menu.

If the disk does not have an entry in the Disks Relation but is online, Unformat prompts the operator, asking if the process should continue. If you respond yes, Unformat checks all zones of the disk for user databases. If any are found, it warns you by asking whether the process should continue. If you answer yes, Unformat deletes the disk's tuple for the disk and marks the disk as foreign to the system.

9. CONFIGURE.

Configure is used to insert, change, and delete Configure tuples. Any value, even a nonsense value, may be inserted in a Configure relation tuple field. A BL700 doesn't recognize a nonsense value until the tuple is loaded. False information could render a BL700 completely unusable.

Configure mode lets you add new tuples, change or delete existing tuples. Configure operational detail is found in the BL700 Operation Manual and the System Administrator's Manual. In the following example, Serial port #6 is turned off.

```
Your choice? 9<RETURN>
You chose Configure Mode.
Is this correct? (y/n) y<RETURN>
*** Configure Mode ***
Configure Relation.
type ... D    number ... 0    value ... 0
type ... M    number ... 0    value ... 1
type ... R    number ... 0    value ... 35
Configure Mode commands
          (0) Return.....  Return to Main Program
          (1) Add........  Add a tuple
          (2) Delete.....  Delete a tuple
Your choice? 1<RETURN>
You chose Add.
Is this correct? (y/n) y<RETURN>
Enter 'configure' tuple values:
          Type?      S<RETURN>
          Number?    6<RETURN>
          Value?     11<RETURN>
You entered this 'configure' tuple
type ... S    number ... 6    value ... 11
Is this correct? (y/n) y<RETURN>
Tuple inserted.
Configure Relation.
type ... D    number ... 0    value ... 0
type ... M    number ... 0    value ... 1
type ... R    number ... 0    value ... 35
type ... S    number ... 6    value ... 11
Configure Mode commands
          (0) Return.....  Return to Main Program
          (1) Add........  Add a tuple
          (2) Delete.....  Delete a tuple
Your choice? 0<RETURN>
You chose return.
Is this correct? (y/n) y<RETURN>
*** Configure complete ***
(Type '?' for menu.)
Your choice?
```

10. SCAN.

CAUTION

All databases must be SAFE prior to running scan mode. If scan is used on a BL700 system with unsafe databases (no recovery since the last time the system was brought down), data may be lost if blocks are remapped or mapped out by scan. The system database is never "already safe", but it must have had recovery run on it anyway.

NOTE

If the system Database is being recreated you should run MERGE and bring the system software up to SAFE prior to running scan to ensure that all databases are safe. This means you must EXIT from DFU.

Scan's job is to scan (read) a disk looking for bad sectors. The disk is divided into zones when first formatted. The first good sector of each zone is the blockalloc page. Blockalloc's function is to keep the track of free, allocated, and defective sectors within the zone. Three types of disk read problems on the BL700 are identified.

disk soft error A008, error code 0
disk soft error B008, error code 0
disk hard error E00X, error code 100

When a soft error occurs consistently on a particular block of a disk, you should remap the bad block with scan as soon as the BL700 system can be dropped to MAINT Mode. This avoids reformatting the disk and reloading the databases that were on the disk. Disk hard errors cannot be remapped if the errored sector is allocated to a database.

SOFT ERRORS. Recovery from a failed sector, called a soft error, is shown with an "error code 0". The "B" category soft error may have characters other than "B00". However, the trailing "8" is always true.

A008 is a return code from the Disk Controller I/O register. The code means 1 of 8 data subsectors of the stated sector had up to 5 sequential bit errors when read. The command was retried and the sector was read correctly on the second attempt. The digit "8" means that all 8, 256-byte subsectors were processed correctly.

B008 return code is similar to the A008 return code . A soft read error occurred. Read was re-executed and was still soft on the second read. Disk Controller ECC (error correction code) circuitry was enabled, allowing up to 60 additional read retries before the subsector was fixed. The digit "8" means that all 8, 256-byte subsectors were eventually processed correctly.

HARD ERRORS. More than 5 bits in error constitutes a hard read error: unrecoverable.

E00X return code is a sample only. The first three characters are bit sensitive, refer to the error code section (see Appendix G). "X" represents a number between 0 and 8 to show how many subsectors were processed before the bad subsector was encountered. The message means that a subsector within the sector read had more than 5 bits in error. The Disk Controller's ECC circuitry could not correct the data with any certainty. The sector may not be recovered without **GREAT** difficulty, usually it is unrecoverable.

Scan was devised to run in three modes: manual mapping, automatic scanning without mapping, and automatic scanning with mapping. Both automatic and manual scan use the same remapping routines that sysformat, format, and mirrorformat modes use.

Manual Mapping mode. The Manual mode of scan is used to map out a soft allocated sector without scanning a complete disk. Typically, this mode is used to quickly map out a recurring soft read error. Unlike the other scan modes, manual allows an operator to remap hard or soft failed sectors IF the sector IS MARKED free IN blockalloc. An allocated hard error sector cannot be remapped.

```
Your choice? 10<RETURN>
You chose Scan mode.
Is this correct? (y/n) y<RETURN>
*** Scan Mode ***
Name of disk to be scanned?          systemtest<RETURN>
1676 free out of 1728 in remap zone.
Manually map out sectors? (y/n) y<RETURN>
         cylinder? 5<RETURN>
         head?    16<RETURN>
         sector?   7<RETURN>
Cyl 5, head 16, sector 7, remapped to cyl 814,
head 0, sector 1, Ctlr 0, drive 0.
Another? (y/n) y<RETURN>
         cylinder? 5<RETURN>
         head?    16<RETURN>
         sector?   7<RETURN>
Remapping a remapped block: 1231.
cyl 5, head 16, sec 7, remapped to
cyl 814, head 0, sec 2, on ctlr 0, drive 1
Another? (y/n) n<RETURN>
*** Scan complete ***
```

In this example, the Manual Mode is used to remap a defective sector. In a mistake, the same sector is remapped again. Scan complains but remaps to another sector. Remap sector 814,0,1 will not be available because of the double remap.

Automatic mode with no remap capability This mode also scans a specified disk. However, it does not remap sectors. It just makes reports to the console port and continues. This mode was designed in case the drive being scanned had broken hardware. In this mode, scan read without remapping and noted two soft errors. This mode is particularly useful if there is reason to believe the disk drive is defective. It is conceivable that DFU might attempt to remap or map out every allocated sector that it reads, unable to discern that something other than bad media is the cause.

```
*** Scan Mode ***
Name of disk to be scanned?          systemtest<RETURN>
1676 free out of 1728 in remap zone.
Manually map out sectors? (y/n) n<RETURN>
Report errors without remapping? (y/n) y<RETURN>
Number of scans to be performed? 1<RETURN>
Scanning zone 0.
.
.
Disk soft err code xA008
cmd 3, ctlr 0, drive 1, cyl 518, head 4, sec 7
         errcode 0, errstat 1, phyaddr 14C800

WARNING: soft read error on block 85796.
         Not remapping the block.

Scanning zone 400.

Disk soft err code xB008
cmd 3, ctlr 0, drive 1, cyl 397, head 1, sec 1
         errcode 0, errstat 1, phyaddr 18A800

WARNING: soft read error on block 111899.
         Not remapping the block.
Scanning zone 520.
0 blocks were reported bad.
2 soft read errors.
*** Scan complete ***
```

In this example, scan read without remapping and noted two soft errors.

Automatic mode with remap capability This mode scans (reads) each sector on a specified disk. When it encounters a soft read error, it inspects that zone's blockalloc block to see if the sector is allocated, free or bad. If it is allocated, scan will retry the read. If it is still soft (B008 category above), then scan will

move the data to the first free sector in the reserved area. The original sector is marked defective in its zone's blockalloc Block. If the retry read was A008 category (above) or blockalloc block says that the sector is free, then scan continues without making changes. If a permanent read error is encountered on an allocated sector (E00X category above), scan can do nothing with it since the contents of the sector are unknown.

```
*** Scan Mode ***
Name of disk to be scanned?          systemtest<RETURN>
1676 free out of 1728 in remap zone.
Manually map out sectors? (y/n) n<RETURN>
Report errors without remapping? (y/n) n<RETURN>
Number of scans to be performed? 1<RETURN>
Scanning zone 0.
.
.
Disk soft err code xB008
cmd 3, ctlr 0, drive 1, cyl 397, head 1, sec 1
        errcode 0, errstat 1, phyaddr 18A800
cyl 518, head 1, sec 1 remapped to
cyl 814, head 0, sec 2.
1 block was remapped
0 blocks were mapped out.
*** Scan complete ***
(Type '?' for menu.)
Your choice?
```

11. MARKDB.

This mode lets the user mark user databases as recoverable or unrecoverable. It is useful if a user database has been damaged in a way that causes the recover program to fail. The system database cannot be marked unrecoverable. A database marked unrecoverable cannot be opened but can be "destroyed". Use CKDB to analyze database damage.

Results of this mode are seen when recovery operates after the system files are loaded. The specified database is listed as "unrecoverable". All other databases are listed as "recovered" or "already safe".

```
Your choice? (y/n) 11<RETURN>
You chose MarkDb mode.
Is this correct? (y/n) y<RETURN>
*** MarkDb Mode ***
Databases:
        'system      ' is marked recoverable.
        'easternregn ' is marked recoverable.
        'prospecting ' is marked recoverable.
        'westernregn ' is marked recoverable.
MarkDb mode commands:
        (0) Return to Main Program
        (1) Mark a database unrecoverable
        (2) Mark a database recoverable
Your choice? (0-2): 1<RETURN>
You chose the Mark Unrecoverable command.
Is this correct? (y/n): y<RETURN>
Name of database to be marked? easternregn<RETURN>
easternregn marked unrecoverable.
MarkDb mode commands:
        (0) RETURN to Main Program
        (1) Mark a database unrecoverable
        (2) Mark a database recoverable
Your choice? (0-2): 0<RETURN>
You chose the Return to Main Program command.
Is this correct? (y/n): y<RETURN>
*** MarkDb complete ***
(Type '?' for menu.)
Your choice?
```

12. RENUMBER.

This mode allows changing the block numbers of a disk, provided there is no information on the disk. This is useful if a disk is removed and the block numbers used by it must be reused. A formatted disk of the same size with no information on it can be renumbered to use the same block numbers as the old one. This changes the block numbers on the master block of the disk, the 'disks' tuple(s) and the page numbers of the blockallocs on each zone. In most cases, Renumber saves reformatting because of overlapping blocks.

```
Your choice? 12<RETURN>
You chose Renumber mode.
Is this correct? (y/n) y<RETURN>
*** Renumber Mode ***
Name of disk to be renumbered? local<RETURN>
New low block number?          219781<RETURN>
*** Renumber complete ***
(Type '?' for menu.)
Your choice?
```

MIRROR OPERATION.

Mirror operation requires a Mirror DBP and disk controllers with Rev 26 or higher firmware. Modes 13 through 16 format, Copy, Compare, and Promote (make single) disks. The mirror function is to use disk pairs that are exact copies. During operation, DBP and the DC's assure correct WRITE and READ operations. Upon failure of one disk, DBP informs the console port. If a Mirrored disk is corrupted, it is marked foreign and operation automatically continues with a "promoted" single disk.

13. MIRRORFORMAT.

A BL700 can mirror if this is reported on the Console Terminal at system Reset time:

```
          Mirrored DBP Rev 33 (6MHz)
or        Mirrored DBP Rev 33 (10MHz)
          Slot 0:1Meg mem
          Slot 1:1Meg mem
          Slot 2:t&c rev 3
          Slot 5:serial chan rev 38
          Slot 6:parallel chan rev 26
          Slot 7:block mux rev 0
          Slot 8:ethernet rev 0
          Slot 9:tpc rev 107
          Slot 11:da rev 37
          Slot 12:dc rev 28
          Slot 13:dc rev 28
          Slot 14:dbp rev 30M (6MHz)
or        Slot 14:dbp rev 31M (10MHz)

BL700 - Filename:
```

Sysformat and format are similar. In mirrorformat, DFU does not ask for cylinder, head, and sector because DFU expects the same type disk to be used. Different sized disks may be formatted so long as the SMALLEST disk is sysformatted or formatted and the LARGER is mirrorformatted. The larger disk must have an equal or larger number of tracks per cylinder, cylinders per disk, and sectors per track. The larger disk will have wasted space as the total number of blocks are based on the smaller disk.

However the initial disk is formatted, the mirrorformatting must be the same. (Disk subdivision is omitted below). Recommended: separate disk controllers.

```
Your choice? 13<RETURN>
You chose MirrorFormat mode.
Is this correct? (y/n) y<RETURN>
*** MirrorFormat Mode ***
Enter information on disk to be mirror formatted:
Disk controller number? (0-3) 1<RETURN>
Disk drive number? (0-15) 0<RETURN>
Enter information on existing disk:
Name of existing disk? systemtest<RETURN>
Disk will be formatted as follows:
Name of disk.............  'systemtest  '
Disk controller number..  1
Disk drive number.......  0
sectors per track.......  9
tracks per cylinder.....  10
cylinders per disk......  823

Is this correct? y/n y<RETURN>
```

The remainder of mirrorformat is the same as sysformat and format.

NOTE

MirrorCopy always follows mirrorformat.

14. MIRRORCOPY.

This mode identifies both disks and copies files from the first disk onto the second or "mirrored" disk. This is file copying, NOT sector by sector copying. Each disk has its own physical characteristics for remapping and is transparent to the user. Only allocated sectors are copied.

This mode assumes the original disk has been loaded with its system or user files before attempting to MirrorCopy.

```
Your choice? 14<RETURN>
You chose MirrorCopy mode.
Is this correct? (y/n) y<RETURN>
*** MirrorCopy Mode ***
Name of disk to be copied? systemtest<RETURN>
Copying 'systemtest  ' from ctlr 0 drive 0 to ctlr 1, drive 0.
Copying zone 0.
.
.
.
Copying zone 400.
*** MirrorCopy complete ***
(Type '?' for menu.)
Your choice?
```

15. COMPARE.

This mode compares the mirrored disks, checking for inconsistencies. Compare should be selected following MirrorCopy when disks are originally mirrored and any time a disk is recovered from corruption with MirrorCopy. It is noteworthy: this mode may also be used to compare other duplicate disks, possibly with separate names.

```
Your choice? 15<RETURN>
You chose Compare mode.
Is this correct? (y/n) y<RETURN>
*** Compare Mode ***
Name of first disk? systemtest<RETURN>
Controller number? (0-3) 0<RETURN>
Drive number? (0-15) 0<RETURN>
Name of second disk? systemtest<RETURN>
Controller number? (0-3) 1<RETURN>
Drive number? (0-15) 0<RETURN>
Comparing zone 0.
.
.
.
Comparing zone 400.
Disks are identical.
*** Compare complete ***
```

16. PROMOTE DISK.

Promote disk lets you keep a single user or system disk operating after one mirrored disk has failed. When a mirrored disk fails, DBP reports the problem to the console port.

```
Your choice? 16<RETURN>
You chose Promote mode.
Is this correct? (y/n) y<RETURN>
*** Promote Mode ***
Name of disk to be promoted? user1<RETURN>
*** Promote complete ***
(Type '?' for menu.)
Your choice?
```

DFU USAGE EXAMPLES

Suppose a user requires five databases, V, W, X, Y, and Z, on three disks, D1, D2, and D3. To create this system, the user must perform the following steps:

- In MAINT Mode, format the system disk (D1) using DFU sysformat.
- Leave DFU and MAINT Mode for RUN Mode to create user databases V, W, X, Y, and Z on the disks using the IDL parser through an I/O Channel's Host Computer. There is usually enough space on the system disk for user databases in addition to the system database.
- Re-enter MAINT Mode to use DFU Validate to ensure the system Database is consistent with the user databases and disks online. This is optional, but should be done when the system is first created or whenever it is rebuilt.

Suppose the user has the following:

system database on disk D1
user database V on disk D1
user database W on disks D1, D2, and D3
user database X on disk D2
user database Y on disks D2 and D3
user database Z on disk D3

You can use several modes of DFU in combination to recover from almost any disk defect or crash. DFU has other useful features for maintenance of the disks on line and for adding new disks to the system. Given the above configuration, some possible scenarios requiring DFU are explained below.

EXAMPLE 1

Disk D2 is lost or damaged beyond use, but the system disk D1 and system database are still intact. To recover, the user must:

- In RUN Mode via a host, destroy any databases that were partially or completely on D2 (W, X, and Y). This removes any traces of the missing databases from the system Database.
- In MAINT Mode with DFU, Unformat D2 to free the block numbers that were formerly allocated to D2. This is required only if too few unallocated block numbers remain for the new disk.
- In MAINT Mode with DFU, format a new disk, D2" with format.

- In RUN Mode via a host, create and load databases W, X, and Y, and roll forward as necessary. User databases need not be placed on D2" if there is enough space elsewhere.

EXAMPLE 2

The system disk D1 is lost or damaged beyond use. The system database and user databases V and W have been lost along with the disk. To recover:

- In MAINT Mode with DFU, use sysformat to create a new system disk.
- In MAINT Mode with DFU, Merge D2 and D3 into the new system Database. This adds information about the User Databases on D2 and D3 to the system Database.
- In MAINT Mode, leave DFU and load the system Database from the source media on a new disk, D1".
- In RUN Mode through a Host, load databases V and W and roll forward as necessary. It may be necessary to destroy V and W before loading if any zones of D2 and D3 are allocated to V or W.

Alternatively:

- Run syscreate on one of the remaining disks on line. This may be done if there is enough space on that disk for databases V and W as well as the system Database.
- Merge D2 and D3 into the system.
- Load databases V and W and rollforward as required.

EXAMPLE 3

The user wants to replace the system disk D1 with a new disk, D4. A new disk may be required if D1 is of insufficient size, has too many defective blocks, or has been damaged. The user must:

- In RUN Mode through a Host, dump databases V and W. Destroy databases V and W.
- In MAINT Mode with DFU, run SysDestroy on D1. If necessary, Unformat D1 and remove it.
- Install D4 and in MAINT Mode, run DFU sysformat on it.
- In MAINT Mode, leave DFU and load the system Database from the source media.
- In RUN Mode through a Host, create and load V and W.
- In MAINT Mode with DFU, Merge D2 and D3 into the system Database.

EXAMPLE 4

Suppose you have created a new system database and there are more disk packs containing user databases than there are drives connected to the BL700. You must Merge all the databases into the system database. While not all user databases can be on line at once, the system database contains information on all databases. Should you attempt to access a database that is not currently on line, the BL700 returns an error message to the host program.

Given two drives, D0 and D1, and three disk packs P1, P2, and P3, complete the following:

- Format the system disk on drive D0.
- Insert pack P1 in drive D1 and run Merge on that drive.
- Remove P1 and repeat with packs P2 and P3, leaving P3 in drive D1.
- Run Validate to determine which databases are offline along with packs P1 and P2.

EXAMPLE 5

Disks from BL700 A must be moved to BL700 B to increase the storage available to BL700 B. After connecting them to a disk controller, the disks are immediately added to the list kept by DFU of on line disks.

NOTE

The correct method to transfer databases between BL700s is to:
1. dump the databases from BL700 A to tape, and
2. load the databases from tape to BL700 B.

The proper method of transferring disks between BL700s is:

- Destroy databases on the disks being moved with **destroy database** command (Refer to the *IDL* or *SQL Reference Manual* for details). If a system disk is being moved, DFU SysDestroy must be used before attempting the move.
- Move the disks to BL700 B.
- Connect the drives to BL700 B and use DFU List mode to ensure the drives are connected properly.
- Use DFU Renumber to make the new blocks contiguous.
- Use DFU to Merge the newly added disks names and block numbers to the system Database of BL700 B.
- After a user disk is moved, run Unformat on BL700 A to remove the old disk blocks and name from the system database.

Several problems may result from transferring disks if the above procedure is not followed: overlapping block numbers, multiple system disks, and incomplete databases remaining on the transferred disks. The Command List displays the status of each disk, showing any problems.

Solve the overlapping block number problem with Renumber on a disk whose block numbers overlap. Duplicate disk name and incomplete database problems are solved with Unformat. Follow with format. Merge is useless if the block numbers overlap.

Format is useless in the multiple system disk problem. It disallows formatting a system disk. SysDestroy must be used to "unsystem" one or more of the system disks. The remaining system disk will become the new system disk for BL700 B.

If there are multiple system disks with overlapping block numbers, SysDestroy must be used to unsystem all but one system disk. Renumber should then be used to assign distinct block numbers to the overlapping disks.

DISK DIAGNOSTICS (DD)

OVERVIEW

Purpose

Disk Diagnostic (DD) exercises a disk controller and connected disk drive(s). The program has utilities that allow common disk functions to be performed easily.

Program Description

DD is written for Britton Lee disk controllers. DD is a menu driven program – designed so the user is prompted for all necessary information. The program reports the result of the operation and displays any error messages in standard English.

Definition of Terms

Buffers
: This program has the use of four 2048 byte (1024 word) buffers, named a,b,c, and d. The buffers are used to transfer data to and from memory with read and write indirect commands.

Command Block
: A Command Block (CB) consists of eight words (256 bits) linked together as a block of command information. Fixed bit positions include return code, flags, command, drive number, link to next block, disk address, error code, retry count, and memory address. Not all bits are used in each situation.

Command Block String
: A CB String is a string of indirect commands that are linked to each other through memory addresses. The first block points to the second, the second points to the third. This process continues on through to the last block which then points back to the first block. A CB string can have a maximum of 16 command blocks.

Command Register
: The disk controller (DC) Command Register is written into by the DBP and is read by the DC.

Direct Command
: Direct Commands are commands the DC can execute without retrieving or adding data to Main Memory.

Indirect Command
: Indirect Commands are commands where the DC must go to a Main Memory location to obtain or place data for or from the command.

I/O Registers
: The 16-bit I/O Register allows the disk controller to communicate with the DBP. It is called the Command Register when DBP writes commands to the DC. The same register is called the status register when DC returns status to the DBP.

Status Register
: The status register is written by the disk controller with contents called "return code". The return code is updated as the DC is executing a command. DBP reads the status register upon successful or unsuccessful completion of a command.

GETTING READY FOR DD.

If resident, DD may be loaded directly from the system disk drive into BL700 Main Memory. If not resident, DD must be loaded from a source tape or diskette into BL700 Main Memory. Refer to System Loading Procedures in the *BL700 Installation Manual.*

The BL700 Front Panel function key is in the MAINT position. When power ON RESET and Self-Diagnostics are complete, the BL700 Maintenance Mode prompt is displayed on your console terminal,

```
BL700 - Filename:
```

where /x represents model 0, 1, or 2; "/2" is used in all examples. All keyboard operations are given with appropriate responses. Note that <RETURN> means the single key whose cap is named Carriage Return, RTN, RETURN, New Line, and sometimes Enter.

Examples are given for every DD command. These commands can be used to ensure that BL700 disk controllers, drives, and associated connections are functioning properly. The drives must be properly addressed before communication across the A and B cable can occur. A disk drive can only SEEK, READ, and WRITE.

NOTE

This is not an automatic diagnostic. A trained service technician can use DD to discern failure modes of a disk versus the disk controller.

An operator can create small, diagnostic programs by building command blocks in memory. Command blocks are executed and decoded by the disk controller. The disk controller then toggles the appropriate signals on the SMD interface (A and B cable) for a selected disk drive. It completes the requested operation and reports the status to the Controller. If the program works correctly, the specified result is displayed on the console. Errors are decoded and displayed on the console.

STARTING AND USING DD.

Refer to Appendix C for software loading procedures. Use one of these three entry lines to start DD.

```
BL700 - Filename: load dd<RETURN>
BL700 - Filename: loadtape dd<RETURN>
BL700 - Filename: dd<RETURN>

-->disk controller Diagnostics 1.2<--

ENTER ALL NUMBERS IN DECIMEL EXCEPT FOR BUFFER DATA

disk controller found in slot 13.
disk controller found in slot 14.

What slot is the disk controller located in? (0 - 15): 14<RETURN>

disk controller located in slot 14(dec)
```

In this exchange, DD has identified two disk controllers in slots 13 and 14, and needs to know which controller will be tested: 14, in this case. DD confirms there is a DC in slot 14. Then, DD asks for drive information. Table 1 contains a list of drives used with BL700's.

Table 1

Manufacturer	Model	Copy	Cyl.	Heads	Sectors
CDC (Pack)	9710-80	80	823	5	9
	9762	80	823	5	9
	9766	300	823	19	9
CDC (HDA)	9715-160	160	823	10	9
	9715-340	340	711	24	9
	9730-160	160	823	10	9
	9775	675	842	40	9
Fujitsu (HDA)	m2284	160	823	10	9
	m2283	130	823	8	9
	m2282	65	823	4	9
CDS (Pack)	t-306	300	823	10	9
CDS (HDA)	ams 315	315	845	19	9
	ams 380	380	845	14	14
	ams 513	513	845	19	14

```
Enter Cylinders per Pack: 823<RETURN>
Enter Sectors per Track:  9<RETURN>
Enter Heads per Cylinder: 10<RETURN>
Do you want to have the menus printed? (y or n):y<RETURN>
```

There are three menus: Exctlr Command Menu, DC Direct Commands, and DC Indirect Commands. The menus are printed when accessed, upon completion of any menu command, and when '?' is invoked, if you wish them. Answer yes if you wish the menus. If you respond no, the menus are never presented.

Upon entering a command, you will be prompted (questioned) for responses. The question "Continue as specified? (y or n)" is an opportunity to continue (complete) or cancel this command, respectively.

The EXTERNAL CONTROLLER COMMAND MENU contains the commands that:

- Allows execution of a Command String.
- Provides a path to both Direct and Indirect Command menus.
- Allows the operator to set, use and examine memory buffers through indirect commands. Each command is defined and illustrated in the following subsections.

```
 EXCTLR COMMAND MENU
**************************
COMMAND             COMMENT
**************************
a                   direct
b                   indirect
c                   read status port
d                   CB string init
e                   execute CB string
f                   loop on string
g                   display CB's
h                   compare buffers
i                   preset buffers
j                   init buffers
k                   look at buffers
l                   format drive
m                   exit to DBP
o                   change slot
>
```

Exctlr Command Menu

a: DIRECT.

This command accesses the DC Direct Command menu. Direct Commands follow EXCTLR Command menu descriptions.

```
>a<RETURN>

 DC DIRECT COMMANDS
**************************
COMMAND             COMMENT
**************************
a                   ident
b                   set cont.pri.
c                   poll drive
d                   please wait
e                   abort
f                   memory diag.
g                   return
>
```

b: INDIRECT.

This command accesses the DC Indirect Command menu. Command descriptions follow the DC Direct Command Menu descriptions.

```
>b<RETURN>

 DC INDIRECT COMMANDS
*************************
COMMAND            COMMENT
*************************
a                  seek
b                  write
c                  read
d                  noop
e                  copyp
f                  copyv
g                  fmttrk
h                  fmtsec
i                  wrhead
j                  rdhead
k                  wrtbad
m                  verify
n                  vfyfmt
o                  return
>
```

c: READ status port.

Displays the return code of the disk controller selected when DD was invoked. The result displayed is bit oriented.

```
>c<RETURN>
Status is 1C4
>

0000 0001 1100 0100

BSY Flag 15:
 0 = status register contents
 1 = Command Register contents
Self Check Code    14-12:
000 = no error
001 - 011 = soft error code (operational)
100 - 111 = hard error code (not operational)
Microcode Revision Level (in decimal) 11-04:
Board ID Number = Hex 4 (0100) 03-00:
```

d: CB STRING INIT.

Initializes the Command Block string by erasing current Indirect Commands. Always start a Command Block String with this command XQ.O, otherwise, your string will be added to the end of an existing CB String.

```
>d<RETURN>
Continue as Specified: (y or n): y<RETURN>
***DONE***
>
```

e: EXECUTE CB STRING.

This command starts the execution of the Indirect CB String. When the command has been executed, a result is displayed on the console terminal. If the command string completed error free, the following is printed. See figure 5-3.

```
e<RETURN>
Continue as Specified? (y or n): y<RETURN>
***DONE***
>
```

If retries were necessary, the following is printed:

```
e<RETURN>
***DONE WITH RETRIES***
>
```

If the disk controller was unable to complete the command even after retries, written error messages stating the reason for the failure are printed on the console terminal.

f: LOOP ON STRING.

This command executes and repeats (loops on) the Indirect CB String. Loop on string offers either stop on error (which aborts on first error) or no stop on error. In this case, the command string continues until stopped with a <RETURN>. Number of passes and errors are printed on the console terminal. DC Direct Commands do not use this command. See DC Direct Commands.

```
f<RETURN>
Exit on error? (y or n): n<RETURN>
><RETURN>
Pass count = 40
Error count = 0
>
```

g: DISPLAY CB's.

The function of this command is to examine an Indirect CB string just written to read the command and other parameters. After execution, this command is used to read the return code status and any error codes present. The return code is the first word of the Command Block. It is preset to zeros by the DBP unless the DBP wants to direct retry operations by relying on the internal retry codes of the DC. The error code is the fifth word of the CB. If an error does occur, it will be presented to the console terminal in standard English. The following example is of a freshly written but not executed READ Indirect Command. A properly executed Indirect Command should have "8008" in the return code word and nothing in the error code word. See figure 5-3.

```
>g<RETURN>
COMMAND BLOCK 0(hex) at paddr A000(hex):

Command = C (hex)
Forward link = 4000 (hex)

Do you want to continue on to the next CB? (y or n): n<RETURN>
>
```

In the above example, the CB string has just been initialized. There is only one CB, 0. An interesting way to study the CB structure is to build a WRITE, READ, COMPARE diagnostic program. Look at the CB's after construction, and then after execution. Be sure to "continue on the the next CB". The menu appears after the last CB.

h: COMPARE BUFFERS.

This command lets you compare two buffers. (Two buffers that have been previously written with some kind of data.) Four buffers, a through d, are available for this purpose. For example, something in buffer a is to be compared with something in buffer b. You may also select the number of words for comparison, 0 to 1024. If you select 0 words, you are returned to the menu. A miscompare example follows the menu prompt. See figure 5-1.

```
>h<RETURN>
First Buffer Name (a, b, c, or d): a<RETURN>
Second Buffer Name (a, b, c, or d): b<RETURN>
Number of Words? (0 - 1024): 1024<RETURN>
Absolutely, no miscompares
>
```

```
>h<RETURN>
First Buffer Name (a, b, c, or d): a<RETURN>
Second Buffer Name (a, b, c, or d): b<RETURN>
Number of Words? (0 - 1024): 1024<RETURN>
$a + 0(dec): 0(hex)
$b + 0(dec): 400(hex)
>
```

The error shows that buffer a word 0 is 0 but buffer b word 0 is 400.(0 does not equal 400).

i: PRESET BUFFERS.

This command presets all four buffers (a,b,c,d) to an incrementing hex value, starting at zero in buffer a.

Buffer a = 0-3FF
buffer b = 400-7FF
buffer c = 800-BFF
buffer d = C00-FFF

```
>i<RETURN>
***DONE***
>
```

j: INITIALIZE BUFFERS.

This command sets the buffers (a,b,c,d) to selected hex data. In this example, buffer a is completely filled with FEFE hex. You can also fill other buffers if necessary by answering yes to "entering more data". The command repeats from the first line. The number of words available is reduced by the starting word offset.

```
>j<RETURN>
Buffer (a, b, c, or d): a<RETURN>
Starting word offset? (0 - 1023): 0<RETURN>
Number of words? (1 - 1024): 1024<RETURN>
Word value in hex: FEFE<RETURN>
Continue as Specified? (y or n): y<RETURN>
Do you want to enter more data? (y or n): n<RETURN>
***DONE***
>
```

k: LOOK AT BUFFERS.

This command lets you look at any buffer (a,b,c,d), selecting beginning and number of words. A good time to look at a buffer is just after you have set some value into it (as in the last example) or just after some value is read into it from a sector.

```
>k<RETURN>
Buffer (a, b, c, or d): a<RETURN>
Starting word offset? (0 - 1023): 0<RETURN>
Number of words? (0 - 1024): 24<RETURN>

a + 0 FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE
a + 10 FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE
a + 20 FEFE FEFE FEFE FEFE FEFE
```

l: FORMAT DRIVE.

This lets you format a selected portion of the disk with any desired hex word. You are prompted for beginning and ending cylinders along with the drive to be formatted. Each cylinder number is reported after it is formatted for this example, only a single cylinder, 822.

CAUTION

Do not confuse this command with the DFU format modes. This command destroys the BL700 format set up by DFU, necessary for each disk.

```
>l<RETURN>
Enter Low Cylinder? (0-822): 820<RETURN>
Enter High Cylinder? (0-822): 822<RETURN>
Enter drive # (0-15): : 1<RETURN>
What data do you want in the data fields? (hex word): ABCD<RETURN>

820 821 822
>
```

m: EXIT TO DBP.

This command exits DD and returns to the DBP to execute the Reset Self-Diagnostics. Unless your console terminal is already set to 300 baud, you will be unable to read the console report.

```
> m<RETURN>

Mirrored DBP Rev 33
Slot 0:1Meg mem
Slot 1:1Meg mem
Slot 2:t&c rev 3
Slot 5:serial chan rev 38
Slot 6:parallel chan rev 26
Slot 7:block mux rev 0
Slot 8:ethernet rev 0
Slot 9:tpc rev 107
Slot 11:da rev 37
Slot 13:dc rev 28
Slot 14:dc rev 28
Slot 15:dbp rev 30M

BL700 - Filename:
```

n: CHANGE SLOT.

This command asks the operator again for the disk controller slot and drive parameters. Use this to change disk controllers, drive parameters, or to change menu availability.

```
>n<RETURN>

ENTER ALL NUMBERS IN DECIMAL EXCEPT FOR BUFFER DATA

disk controller found in slot 13.
disk controller found in slot 14.

The disk controller is in what slot? (0 - 15): 13<RETURN>

disk controller is located in slot 13(dec).
Enter Cylinders per pack: 823<RETURN>
Enter Sectors per track: 9<RETURN>
Enter Heads per cylinder: 10<RETURN>
Do you want to have the menus printed? (y or n):y<RETURN>
```

DC DIRECT COMMAND DESCRIPTIONS

The DC Direct Commands are sent from the DBP to the disk controller Command Register. The DC operates directly upon these commands without further reference. DD does not permit Direct Commands to loop from the main menu. Each of these commands may be looped for 'scoping by responding yes to the question:

```
Loop on direct command? (y or n): _<RETURN>

Stop looping by pressing <RETURN>.
```

a: IDENT.

Identify yourself. This command causes the disk controller to post the status port with the disk controller board ID code (Hex 4), the microcode revision level, and its self-check diagnostics, as it does during DBP Slots. The disk controller self-check codes are:

0 = good board
1 = an error occurred, but the board is now working
2 = problem with the 2901 system
3 = problem with internal-external registers
4 = problem with FIFO
5 = memory data error
6 = memory timeout
7 = off-board error interrupts

```
>a<RETURN>
Loop on direct command? (y or n): n<RETURN>
dc IDENTIFY RESPONSE
ID = dc
REV = 28
Self Check = 0
>
```

b: SET CONTROLLER PRIORITY.

This command assigns the controller arbitration priority number to the controller. This number is used in the arbitration between multiple contending disk or tape controllers and must be issued prior to any command that results in bus arbitration (any indirect command). You must set the controller priority to one of four settings (0,1,2, or 3).

```
b<RETURN>
Set controller priority? (0-4) 2<RETURN>
Loop on direct command? (y or n): n<RETURN>

***DONE***
>
```

CAUTION

If other controllers are plugged in and the controller priority is changed to a value of an existing controller, a trap message will occur.

c: POLL DRIVE.

The poll command retrieves information about the drive regarding its readiness to receive a command. The states are:

On cylinder or not
Drive Ready or not
Drive Write Protected or not
Cable 0 - 3 or no cable
Hardware error
Seek error
Re-zero issued due to SMD fault

```
>c<RETURN>
Enter Drive # (0 - 15): 1<RETURN>
Loop on direct command? (y or n): n<RETURN>

Status of Selected Drive

Drive is positioned on cylinder
Drive is ready
Drive is not write-protected
Selected drive is on cable 0
>
```

d: PLEASE WAIT.

When the most recently issued command is Please Wait, the DC does not fetch a new CB, or write the "Complete" bit into the return code of any CB. However, the DC can continue to modify or inspect the contents (other than link) of a previously read CB so long as bit 15 of the first word is zero (incomplete). This way, the DBP can modify the CB string without fear of DC-DBP interference. The response to Please Wait in the status port is the DC status word.

```
>d<RETURN>
Loop on direct command? (y or n): n<RETURN>
***DONE***
>
```

NOTE

If the CB reference bit in the status port is active, then Please Wait has no effect until CB reference is reset. See figure 5-2.

e: ABORT.

This command sends abort to the disk controller, stopping its current command execution.

```
>e<RETURN>
Loop on direct command? (y or n): n<RETURN>
 ***NONMASKABLE INTERRUPT***

HFAULT

disk controller found in slot 13

>
```

f: MEMORY DIAG.

This command sends a memory transfer test command to the disk controller, testing memory transfer capabilities of the disk controller. The response to this command is one of two results.

PASS: The test was executed without error.
FAIL: The disk controller was unable to complete the test without error.

```
f<RETURN>
Loop on direct command? (y or n): n<RETURN>
PASS
>
```

g: RETURN.

This command returns the operator to the EXCTLR Command Menu.

```
g<RETURN>

 EXCTLR COMMAND MENU
**********************
COMMAND            COMMENT
**********************
a                  direct
b                  indirect

>
```

DC INDIRECT COMMAND DESCRIPTIONS

An Indirect Command prompts for information needed to prepare its own Command Block parameters. The disk controller must make a memory reference using CB's while processing these commands. One to fifteen CB's may be set up as a chain of events to diagnosticaly exercise a disk drive. In addition to individual questions, two questions are asked for every Indirect Command. One question is:

```
Do you want to add this CB to the string? (y or n):
```

Are you sure you didn't make a mistake? If you answer yes, this Indirect Command becomes the next CB in the string. If you answer no, this Indirect Command is ignored.

```
Do you want the diagnostic bits? (y or n):
```

Answering yes to this question raises five questions having to do with sector READ error recovery attempts. If you respond no, the additional questions are not asked. Each question is independent of the others. A NO response keeps to normal operation, bypassing that subject. A YES response modifies the normal operation. Each item is explained as presented.

```
Do you want use of the diagnostic bits (y or n): y<RETURN>

Indexing? (y or n): _<RETURN>
```

NO. Normally, a disk controller recognizes the Index pulse and reads each sector header until it finds the desired sector header to read.

YES. Alternatively, a disk controller recognizes the Index pulse and counts sector pulses before reading the desire sector header. This becomes very useful when a preceding sector is corrupted and no further reads on that track are permitted. This could permit recovering 8 of 9 sectors on a track.

```
Subsectors Bypassed? (y or n): _<RETURN>
```

NO. The disk controller starts reading at data subsector 0.

YES. The disk controller counts data subsector gaps "keying" from the sector pulse until it reaches the data subsector to read. This is useful when a data subsector is corrupted and no further reads are permitted. This could permit recovering 7 of 8 data subsectors.

```
Inhibit retries? (y or n): _<RETURN>
```

NO. The disk controller uses up to 60 retries.

YES. The disk controller does not use retries. You already know this sector has failed. Retries are usually not desirable during recovery.

```
Inhibit ECC recovery? (y or n): _<RETURN>
```

NO. The disk controller uses ECC correction.

YES. The disk controller does not use ECC correction. This is generally not desirable during recovery.

```
Enable Skip defect? (y or n): _<RETURN>
```

NO. The disk controller writes header information at the sector pulse.

YES. The disk controller delays 50μ after the sector pulse to skip a possible defect in the media.

a: SEEK.

This test checks that the disk drive can seek to a specific cylinder and head. Note this Indirect Command is automatically included in all Indirect read and write Commands.

EXAMPLE: Seek between cyl 822, head 2, and cyl 15, head 1. Then loop. Stop looping by pressing <RETURN>. The example starts from the EXCTLR Command menu and eventually returns there.

```
b<RETURN>
a<RETURN>
Do you want use of the diagnostic bits? (y or n): n
Drive # (0 - 15): 1<RETURN>
Cylinder (0 - 822): 822<RETURN>
Head (0 - 9): 2<RETURN>
Do you want to add this CB to the string? (y or n):y<RETURN>
>a<RETURN>
Do you want the diagnostic bits? (y or n): n<RETURN>
Drive # (0 - 15): 1<RETURN>
Cylinder (0 - 822): 15<RETURN>
Head (0 - 9): 1<RETURN>
Do you want to add this CB to the string? (y or no): y<RETURN>
>o<RETURN>
>f<RETURN>
Exit on error? (y or n): n<RETURN>
><RETURN>
Pass count = 40
Error count = 0
>
```

b: WRITE.

A specified memory page (2048 bytes) is written into a specified sector. The write command performs a seek to the specified track (if not already on cylinder), prior to initiating a disk write.

CAUTION

Write destroys any data on the selected sector.

EXAMPLE. Start the CB string. Write a predetermined value into buffer a for cylinder 822, head 1, sector 2. When done, read the data into buffer d and compare buffers. This ensures that what is written is the same as what is read on the sector. If this matches, the WRITE and READ to the disk has been accomplished properly. Display the buffers and look at the command blocks. Starting at the EXCTLR COMMAND MENU:

```
>d<RETURN>
Continue as specified: (y or n): y<RETURN>
***DONE***
>j<RETURN>
Buffer (a, b, c, or d): a<RETURN>
Starting word offset? (0 - 1023): 0<RETURN>
Number of words? (1 - 1024): 1024<RETURN>
Word value in hex: FEFE<RETURN>
Continue as specified? (y or n): y<RETURN>
Do you want to enter more data? (y or n): n<RETURN>
***DONE***
>j<RETURN>
Buffer (a, b, c, or d): d<RETURN>
Starting word offset? (0 - 1023): 0<RETURN>
Number of words? (1 - 1024): 1024<RETURN>
Word value in hex: 0000<RETURN>
Continue as specified? (y or n): y<RETURN>
Do you want to enter more data? (y or n): n<RETURN>
****DONE***
>b<RETURN>
Do you want the diagnostic bits? (y or n): n<RETURN>
Drive # (0 - 15): 1<RETURN>
Cylinder (0 - 822): 822<RETURN>
Head (0 - 9): 1<RETURN>
Sector (0 - 8): 2<RETURN>
Buffer (a, b, c, or d): a<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>
>0<RETURN>
>c<RETURN>
Do you want the diagnostic bits? (y or n): n<RETURN>
Drive # (0 - 15): 1<RETURN>
Cylinder (0 - 822): 822<RETURN>
Head (0 - 9): 1<RETURN>
Sector (0 - 8): 2<RETURN>
Buffer (a, b, c, or d): d<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>
>0<RETURN>
>e<RETURN>
***DONE***
>h<RETURN>
First buffer name (a, b, c, or d): a<RETURN>
Second buffer name (a, b, c, or d): d<RETURN>
Number of words? (0 - 1024): 1024<RETURN>
Absolutely, no miscompares
>k<RETURN>
Buffer (a, b, c, or d): a<RETURN>
Starting word offset? (0 - 1023): 0<RETURN>
Number of words? (0 - 1024): 24<RETURN>

a + 0 FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE
a + 10 FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE
a + 20 FEFE FEFE FEFE FEFE FEFE

>k<RETURN>
Buffer (a, b, c, or d): d<RETURN>
Starting word offset? (0 - 1023): 0<RETURN>
Number of words? (0 - 1024): 24<RETURN>

d + 0 FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE
d + 10 FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE FEFE
d + 20 FEFE FEFE FEFE FEFE

>
```

c: READ.

Subsectors n through 7 of an 8-subsector disk page are transferred to memory buffer (a - d).

EXAMPLE: Start the Command Block. Create a program that reads a sector at cylinder 15, head 0, sector 1. Transfer the data from the sector into buffer b. This reads the complete 2k block. Starting at the EXCTLR COMMAND MENU:

```
>d<RETURN>
Continue as specified? (y or n): y<RETURN>
***DONE***
>b<RETURN>
>c<RETURN>
Do you want the diagnostic bits? (y or n): n<RETURN>
Drive # (0 - 15): 1<RETURN>
Cylinder (0 - 822): 15<RETURN>
Head (0 - 9): 0<RETURN>
Sector (0 - 8): 1<RETURN>
Buffer (a, b, c, or d): b<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>
>o<RETURN>
>e<RETURN>
Continue as specified? (y or n): y<RETURN>
***DONE***
>k<RETURN>
Buffer (a, b, c, or d): <RETURN>
Starting word offset? (0 - 1023): 0<RETURN>
Number of words? (0 - 1024): 40<RETURN>

b + 0 DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C
b + 10 DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C
b + 20 DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C
b + 30 DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C DB6C
```

This is the data read from cylinder 15, head 0, sector 1.

d: NOOP.

No operation is performed on the specified drive. This command is only used to end a string of consecutive sector command blocks.

```
d<RETURN>
Do you want use of the diagnostic bits? (y or n): n<RETURN>
Drive # (0-15): 1<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>

>
```

e: COPYP.

This command is no longer valid. If called, it will return an error message. Do not use.

f: COPYV.

Copy Virtual Address tests the MMU function in the DBP.

EXAMPLE: Initialize the CB's, initialize buffer a with some value, shift to the Indirect Command for Copyv using buffer a's value for source and buffer b for destination. Return to the main menu, execute the CB string and compare values.

NOTE

Only the "f" command is given in full detail. The remainder of the commands have already been discussed.

```
d<RETURN>
j<RETURN>
b<RETURN>
f<RETURN>
Do you want use of the diagnostic bits? (y or n): n<RETURN>
Number of quadwords? (0 - 256): 256<RETURN>
Source buffer (a,b,c,d): a<RETURN>
Destination buffer (a,b,c,d): b<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>
o<RETURN>
e<RETURN>
h<RETURN>
```

g: FORMAT TRACK.

This command formats a disk track by writing a sector header field and data field on each sector of the specified track. The sector at the track index point is written as sector 0. Incremental sector ID's are written for subsequent sectors. The data field is written with the repeating data pattern of Hex DB6C. The command performs a seek to the specified track prior to initiating the disk write if not already on the correct cylinder. CYL/HD/SEC format for the disk address must be used. The sector field specifies the number of sectors per track and must be non-zero.

```
g<RETURN>
Do you want use of the diagnostic bits? (y or n): n<RETURN>
Drive # (0-15): 1<RETURN>
Cylinder (0-822): 822<RETURN>
Head (0-9): 9<RETURN>
Sector (0-8): 8<RETURN>
Buffer (a,b,c,d): a<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>

>
```

h: FORMAT SECTOR.

The FMTSEC command formats a single disk sector by writing a sector header field and data field on the specified sector. CYL/HD/SEC format of the disk address must be used. The physical sector formatted corresponds to the value in the SECTOR field of the CB, and the header byte "SECTOR ID" is given the same value. The data field is written with repeating data pattern, DB6C hex. This command performs a seek to the specified track prior to initiating the disk write if not already on the correct cylinder.

```
h<RETURN>
Do you want use of the diagnostic bits? (y or n): n<RETURN>
Drive # (0-15): 1<RETURN>
Cylinder (0-822): 822<RETURN>
Head (0-9): 9<RETURN>
Sector (0-8): 8<RETURN>
Buffer (a,b,c,d): a<RETURN>
What data do you want in the data fields (hex word): ABCD<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>

>
```

i: WRITE HEADER.

This command should only be used to mark a sector as defective. FMTTRK or FMTSEC commands should be used for initializing good sectors. This command writes the specified sector with a header which optionally contains a pointer to the alternate sector, or is used to flag a deleted defective sector. The data field is not written. If the skip-defect flag is set, the header is located 50µ beyond its normal position and preceding portions of the sector are erased. After executing this command, old data in the data area may be illegible.

```
i<RETURN>
Do you want use of the diagnostic bits? (y or n): n<RETURN>
Drive # (0-15): 1<RETURN>
Cylinder (0-822): 822<RETURN>
Head (0-9): 9<RETURN>
Sector (0-8): 8<RETURN>
Buffer (a,b,c,d): a<RETURN>
Enter 0 for Alt.sector or 1 for Delete sector or 2 for good: 0<RETURN>
Enter alt cylinder: 821<RETURN>
Enter alt.head:8<RETURN>
Enter alt.sector:7<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>
>
```

NOTE

Entering 0 or 1 brings same response pattern. Alternate data is not requested for a good sector.

j: READ HEADER.

The entire 10 byte header is read into a memory buffer (a - d). The command performs a seek to the specified track prior to initiating the disk read if not already on the correct cylinder.

EXAMPLE: Begin a new program by building a Command Block. Select option j from the INDIRECT COMMAND MENU. It reads the header area of a selected cylinder, head, and sector, and places that data in a specified DD buffer. This verifies the seek to a selected cylinder. Start the test from the EXCTLR COMMAND MENU as follows:

```
>d<RETURN>
Continue as specified? (y or n): y<RETURN>
***DONE***
>b<RETURN>
>j<RETURN>
Do you want the diagnostic bits? (y or n): n<RETURN>
Drive # (0 - 15): 1<RETURN>
Cylinder (0 - 822): 15<RETURN>
Head (0 - 9): 1<RETURN>
Sector (0 - 8): 2<RETURN>
Buffer (a, b, c, or d): a<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>
>o<RETURN>
>e<RETURN>
Continue as specified? (y or n): y<RETURN>
***DONE***
>k<RETURN>
Buffer (a, b, c, or d): a<RETURN>
Starting word offset? (0 - 1023): 0<RETURN>
Number of words? (0 - 1024): 5<RETURN>
```

(Words	-1-	-2-	-3-	-4-	-5-	-6-)
	a + 0	F	102	0	F	102

Words in -1- means buffer "a" at word offset 0. Words -2- through -6- are the 5 requested words. Word -2- is the primary cylinder number in hex. Word -3- is the primary head and sector numbers in hex. Word -4- is the alternate flag. The alternate is 0 when the sector is good; and set to 1 when the sector is defective. Words -5- and -6- equal words -2- and -3- respectively when the alternate flag is zero. When the flag is 1, then words -5- and -6- equal the alternate or replacement cylinder, head, and sector.

k: WRITE BAD.

This command forces every ECC field to correspond to one generated for a data field of all zeros. Any 1 bits in the data field represent the equivalent of a disk media error. The DC will perform automatic correction of a single error burst with span of 1-5 bits on a read per subsector, and this command allows the simulation of error burst of 0-2047 bits span in each of the 8 subsectors of a disk sector for use by diagnostic programs.

```
k<RETURN>
Do you want use of the diagnostic bits? (y or n): n<RETURN>
Drive # (0-15): 1<RETURN>
Cylinder (0-822): 822<RETURN>
Head (0-9): 9<RETURN>
Sector (0-8): 8<RETURN>
Don't forget to initialize the buffer to all zeros and the last byte the
error.

Buffer (a,b,c,d): a<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>

>
```

l: VERIFY.

Reads back disk data to detect read errors. Verifies a previously written memory buffer (a - d).

EXAMPLE.
Clear Command Blocks.
Initialize Buffer A with FEFE.
Write Buffer A to Cylinder 822, Head 1, Sector 2.
Verify Sector 2 with Buffer A.

```
>d<RETURN>
Continue as specified: (y or n): y<RETURN>
***DONE***
>j<RETURN>
Buffer (a, b, c, or d): a<RETURN>
Starting word offset? (0 - 1023): 0<RETURN>
Number of words? (1 - 1024): 1024<RETURN>
Word value in hex: FEFE<RETURN>
Continue as specified? (y or n): y<RETURN>
Do you want to enter more data? (y or n): n<RETURN>
***DONE***
>b<RETURN>
>b<RETURN>
Do you want the diagnostic bits? (y or n): n<RETURN>
Drive # (0 - 15): 1<RETURN>
Cylinder (0 - 822): 822<RETURN>
Head (0 - 9): 1<RETURN>
Sector (0 - 8): 2<RETURN>
Buffer (a, b, c, or d): a<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>
m<RETURN>
Do you want use of the diagnostic bits? (y or n): n<RETURN>
Drive # (0-15): 1<RETURN>
Cylinder (0-822): 822<RETURN>
Head (0-9): 1<RETURN>
Sector (0-8): 2<RETURN>
Buffer (a,b,c,d): a<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>
o<RETURN>
e<RETURN>
***DONE***
>
```

m: VERIFY FORMAT.

Reads back disk data to detect ECC errors, or compare errors. No buffer is specified; the format pattern is used for compare.

```
>n<RETURN>
Do you want use of the diagnostic bits? (y or n): n<RETURN>
Drive # (0-15): 1<RETURN>
Cylinder (0-822): 822<RETURN>
Head (0-9): 9<RETURN>
Sector (0-8): 8<RETURN>
Buffer (a,b,c,d): a<RETURN>
Do you want to add this CB to the string? (y or n): y<RETURN>

>
```

n: RETURN.

This command returns control to the EXCTLR Command menu.

```
> o<RETURN>

 EXCTLR COMMAND MENU
**************************
COMMAND            COMMENT
**************************
```

TAPE DIAGNOSTICS (TD)

OVERVIEW

PURPOSE

This program, Tape Diagnostics (TD), deals with tape controller use in providing utilities that allow common functions to be performed easily. In addition, both controller and drive testing abilities are provided to verify their correct operation.

LOOPBACK TEST

You have the option of running a Tape Controller loopback test in TD or as a DBP command (Rev 27 or above). The test verifies that the TPC is sending, and correctly receiving, Tape Interface signals. The loopback cables may be connected to the BL700 Rear Panel TPC Interface or directly to the top of the TPC. See instructions for either command.

CAUTION

Unwind the loopback tool ribbon cables before fastening to the Interface or PCB connector. Wrapped, inductive reactance is strong enough to cause interface error messages.

TD OPERATION.

Refer to Appendix C for Software Loading Procedures. Use one of these three entry lines to start TD.

```
BL700 - Filename: load td<RETURN>
BL700 - Filename: loadtape td<RETURN>
BL700 - Filename: td<RETURN>
```

ENTER ALL NUMBERS IN DECIMAL EXCEPT BUFFER DATA

```
What slot is the tape controller in? (0 - 15): 9<RETURN>
Do you want to run automatic tape system test? (y or n): y<RETURN>
```

Enter the TPC's slot number shown when the system self diagnostics were run. Respond yes for automatic tape test and turn to "k" in the TPC COMMAND MENU. Otherwise, answer no.

```
TPC COMMAND MENU
************************
COMMAND      COMMENT
************************
a            direct
b            indirect
c            init command string
d            execute string
e            loop on string
f            look at command block
g            preset buffer value
h            init buffer value
i            look at buffers
j            compare buffers
k            automatic tape test
l            summary of errors
m            exit to dbp boot

>
```

TPC COMMAND MENU

The TPC COMMAND MENU contains the commands to start examination of memory buffers used in the direct and indirect commands. This menu also lets you move to the Direct and Indirect Command menus. They contain the commands to start examination of the memory buffers used in the direct and indirect commands. It also contains a command to execute a long term test to assist in verifying the correct operation of the tape controller.

The "Continue as specified" question permits continuing or escaping the selected command in the TPC Command and Direct Command menus. A similar form, "Continue to next CB" is present in the Indirect Command menu.

a: DIRECT.
This command accesses the TPC Direct Command menu. The TPC DIRECT COMMANDS menu contains commands the tape controller can execute without further information. Individual Direct Commands are described later.

```
>a<RETURN>

TPC DIRECT COMMANDS
************************
COMMAND COMMENTS
************************
a  identify
b  set cont.  priority
c  abort
d  memory inter
e  loopback
f  return
>
```

b: INDIRECT.
This command accesses the TPC Indirect Command menu. One or more Command Blocks (CB's) are set up in memory to diagnosticaly exercise a drive. The menu and descriptions follow the TPC Direct Command Menu descriptions.

```
>b<RETURN>

TPC INDIRECT COMMANDS
*********************
COMMAND COMMENT
*********************
a  read
b  write
c  poll
d  special
e  return
>
```

c: INIT COMMAND STRING.
Always start a Diagnostic Command Block String with this command. It erases existing Diagnostic CB's. Otherwise, your string will be added to the end of an existing Diagnostic CB String.

```
>c<RETURN>
***DONE***
>
```

d: EXECUTE STRING.

This command starts the execution of the Diagnostic CB String. When the command has been executed, the result is displayed on the Console Terminal. If the command string completed error free, the following is printed.

```
d<RETURN>
Continue as Specified? (y or n): y<RETURN>
***DONE***
>
```

If retries had to be used, the following is printed.

```
d<RETURN>
***DONE WITH RETRIES***
>
```

If the tape controller was unable to complete the command, even after retries, written error messages stating the reason for the failure are printed.

e: LOOP ON STRING.

This command executes and repeats (loops on) the Diagnostic CB String. The command string continues until the first error or until stopped with a <RETURN>. Number of passes and errors are printed on the Console Terminal. DC Direct Commands do not use this command. See DC Direct Commands.

```
e<RETURN>
Stop on error? (y or n): n<RETURN>
><RETURN>
Pass count = 40
Error count = 0
>
```

f: LOOK AT COMMAND BLOCK.

This command allows you to examine the Diagnostic CB just written. The CB contains a return code as first word of the Command Block.

```
f<RETURN>
COMMAND BLOCK 0(hex) at virtaddr 5800(hex):
7(hex)
5800(hex)
5(hex)
0(hex)
Continue to next CB? (y or n): n<RETURN>
```

g: PRESET BUFFER VALUE.

This command presets all four buffers (a,b,c,d) to an incrementing hex value, starting at zero.

```
Buffer  a  = 0-3FF
buffer  b  = 400-7FF
buffer  c  = 800-BFF
buffer  d  = C00-FFF


>g<RETURN>
***DONE***
>
```

h: INIT BUFFER VALUE.

This command sets the buffers (a,b,c,d) to selected hex data. In this example, buffer a is completely filled with FEFE hex. You can also fill the other buffers if you need by answering yes to "entering more data". The command repeats from the first line. The number of words available is reduced by the starting word offset.

```
>h<RETURN>
What buffer do you want to use? (a,b,c,d): a<RETURN>
Starting word offset?: 0<RETURN>
Number of words?: 1024<RETURN>
Word value: FEFE<RETURN>
***DONE***
>
```

i: LOOK AT BUFFERS.

This command lets you look at any buffer (a,b,c,d) selectable portion and selectable beginning. A good time to look at a buffer is just after you have set some value into it, as in the last example.

```
>i<RETURN>
What buffer do you want to use? (a,b,c,d): a<RETURN>
Starting word offset?: 0<RETURN>
Number of words?: 24<RETURN>

a + 0 FEFEFEFEFEFEFEFEFEFEFEFEFEFEFEFE
a + 16 FEFEFEFEFEFEFEFEFEFEFEFEFEFEFEFE
a + 32 FEFEFEFEFEFEFEFEFEFEFEFEFEFEFEFE
```

j: COMPARE BUFFERS.

This command lets you compare two buffers. (Two buffers that have been previously written with some kind of data). Four buffers, a through d, are available for this purpose. Something in buffer a is to be compared with something in buffer d. You may also select any number of words for comparison, 1 to 1024. If you select 0 words, you are returned to the menu. A "no miscompare" (note the = sign) example is followed by a "miscompare" example where Word 100 in buffers "a" and "b" differ.

```
>j<RETURN>
First Buffer Name (a,b,c,d): a<RETURN>
Second Buffer Name (a,b,c,d): b<RETURN>
Number of Words?: 1024<RETURN>
=
>

>j<RETURN>
First Buffer Name (a,b,c,d): a<RETURN>
Second Buffer Name (a,b,c,d): b<RETURN>
Number of Words?: 1024<RETURN>
$a + 100(dec): FEFE(hex)
$b + 100(dec): FFFF(hex)
>
```

The error shows that buffer a word 100 is FEFE but buffer b word 100 is FFFF. (They are not equal).

k: AUTOMATIC TAPE TEST.

This command lets you start the automatic tape test. This test verifies the tape subsystem's ability to write and read by testing the following equipment:

1. Tape controller
   a. Read command
   b. Write command
   c. Write file mark command
   d. Search for file mark command
   e. High speed command
   f. Rewind command
2. Cables between the tape controller and the tape formatter.
3. Tape formatter and transport

NOTE

The Automatic Tape Test completes a write of the ENTIRE tape (BOT – EOT) before rewinding and reading.

CAUTION

Unload the System Tape. Mount a SCRATCH tape. This test erases any previous data. Unless a long test is required, we suggest a scratch tape about 100 ft long between BOT and EOT reflective markers.

```
k<RETURN>

TYPE OF DRIVES
**********
COMMAND COMMENT
**********
a Streamer
b Start-stop
> _<RETURN>
```

Enter a or b. TD needs to know which kind of tape transport is to be tested. There are two kinds of tape transports: Start–Stop and Streamer.

"Start–Stop" means a tape transport writes (reads) an 8k–byte block and will stop at the Inter Block Gap (IBG) to wait for the next instruction from the Tape Controller.

"Streamer" means a tape transport starts Writing (reading) an 8k–byte block and does not stop when the IBG is reached. If a "continuation" command is not received within the re–instruct window time, the transport will stop somewhere in the next 8k–byte block and rewind to the last IBG to wait for the next instruction from the Tape Controller.

NOTE

Some "streamers" may require the high speed tape controller (133–1226) in order to stream. However, if the maximum transfer rate of the transport is 160k–bytes or below, the standard tape controller (133–0172) will work just fine. Use DFU Configure mode to set the tape tuple as "T x 4". "x" represents the transport address, 0 – 7.

Table 1 lists both kinds of BL700 compatible tape transports.

Table 1

Manufacturer	Model	IPS	Density
STREAMERS			
Ampex	TMS–B	12.5/100	1600 P.E.
Cipher	F880 I	25/100	1600 P.E.
	F880 II	25/100	1600 P.E.
		50	3200 P.E.
Kennedy	6809	12.5/100	1600 P.E.
Megatape	MT–500		1600 P.E.

Table 1 (cont.)

Manufacturer	Model	IPS	Density
START - STOP			
CDC	92144	45	800 NRZI 1600 P.E.
IDT	TD1054	45	800 NRZI 1600 P.E.
Pertec	FT8640A-98	45	800 NRZI 1600 P.E.

The next two prompts asks for the address of the tape transport and microformatter. The BL700 Configure Tape Default address is 0. See addressing explanation below.

```
What transport do you want? (0,1,2 or 3): _<RETURN>
What formatter do you want?(0 or 1): _<RETURN>
```

When BL700s have multiple tape transports, the tape cables are daisy-chained from the TPC Interface Panel. Note this addressing process.

TPC uses a four-bit address. The microformatter address is the MSB: 00 or 01. The transport address are the LSB's: 00 through 11. Both are combined to form a tape address between 0 and 7. Set one formatter address at 0 and its transport address between 0 and 3. These transports become tape addresses 0 through 3. Set the second formatter address to 1 and its transport address between 0 and 3. The latter become Tape addresses 4 through 7. Check the manufacturer's Installation Manual to locate and set the correct addresses.

```
Do you want to continue with the test? (y or n): _<RETURN>
```

Respond yes to execute the streamer test. A no response brings in the standard tape test.

As the test starts, TPC rewinds the tape to BOT. If there are no errors, the tape controller issues a high speed command. The program generates a buffer of semi-random numbers to be written on the tape. Every time a new buffer is generated, different numbers are stored in the buffer. The tape controller is issued a command string causing it to write an 8K block of data on the tape. This command string is issued 10 times, each time with a new buffer of data. After the tenth time, a file mark is written. After this file mark, another ten 8K blocks of data is written followed by another file mark. This continues until the EOT mark is passed and the tape is rewound.

If errors occur during this write sequence without controller recovery, the error is displayed and logged in memory. The tape is rewound and the next pass is started. If the error is "drive not on-line" or "tape write protected", the error is displayed and the test is terminated.

If the write has no errors, the tape is rewound to BOT and the read phase is started. The tape controller is issued a command string that causes it to read an 8K block of data from the tape. The program verifies the data and continues. If it was not good, the following is displayed and a new pass is started.

```
GOOD DATA = XXXX BAD DATA = XXXX
```

If the data was good, the tape controller continues to read data until it has read ten 8K blocks and verified them. Now, it searches for a file mark. When the file mark is found, the controller continues to read ten 8K blocks of data and verify them. This pattern continues until EOT is found, and then the tape is rewound to BOT. Another pass is started, which consists of a write, rewind, and a read. If an error occurs during a read, it is displayed and logged in to be retrieved later.

If any errors occur on the following commands, the test is terminated.

1. Rewind command
2. High speed command
3. Write file mark command
4. Search for file mark command

If the tape system runs through one pass with no errors, the message below is printed:

```
Writing tape Pass 1 Error count = 0
Reading tape
```

If errors occur, these messages appear on the Console Terminal.

```
Writing tape Pass 2 Error count = 0
Reading tape
2901 timeout

Writing tape Pass 3 Error count = 1
2901 timeout

Writing tape Pass 4 Error count = 2
TPC interface correctable error
```

The test is terminated if more than 50 errors are encountered, or if you press <RETURN>.

l: SUMMARY OF ERRORS.

This command lets you retrieve errors occurring during the Automatic Tape Test. See the previous command on Automatic Tape Test.

```
l<RETURN>
Passes = 4, Errors = 3
Pass #2, Error => 2901 timeout
Pass #3, Error => 2901 timeout
Pass #4, Error => TPC interface correctable error
```

That series of errors represents the kind of message that might be displayed.

m: EXIT TO DBP BOOT.

This command exits TPC and returns to the DBP to execute the Reset Self-Diagnostics. Unless your Console Terminal is already set to 300 baud, you will be unable to read the Console report.

```
>m<RETURN>

Mirrored DBP Rev 33 (6MHz)
Mirrored DBP Rev 33 (10MHz)
Slot 0:1Meg mem
Slot 1:1Meg mem
Slot 2:t&c rev 3
Slot 5:serial chan rev 38
Slot 6:parallel chan rev 26
Slot 7:block mux rev 0
Slot 8:ethernet rev 0
Slot 9:tpc rev 107
Slot 11:da rev 37
Slot 13:dc rev 28
Slot 14:dc rev 28
Slot 15:dbp rev 30M (6MHz)
Slot 15:10MHz dbp rev 31M 256K RAM

BL700 - Filename:
```

TPC DIRECT COMMANDS

This menu contains commands that send Tape Controller commands directly to the TPC Command Port. Each command is defined below. Each command may be looped for "scoping by responding yes to "Loop on direct command?" TD does not permit direct commands to loop from the main menu. Stop looping with <RETURN>. The proper answer is presented but difficult to read since the screen scrolls to answer the <RETURN> as "ILLEGAL COMMAND".

```
TPC DIRECT COMMANDS
**************************
COMMAND      COMMENT
**************************
a            identify
b            set cont.  priority
c            abort
d            memory inter
e            loopback
f            return
>
```

a: IDENTIFY.

Identify yourself. This command causes the Tape Controller to post the Status Port with the Tape Controller Board ID code (Hex 5), the microcode revision level, and the Diagnostic Self-Check result. The self check codes, generated during power-on diagnostics are:

0 = good board
1 = an error occurred, but the board is now working
2 = problem with the 2901 system
3 = problem with internal-external registers
4 = problem with FIFO
5 = memory data error
6 = memory timeout
7 = off-board error interrupts

```
>a<RETURN>
Continue as specified (y or n)?: y<RETURN>
Do you want to loop on command (y or n)?: n<RETURN>
TPC IDENTIFY RESPONSE
ID = TPC
REV = 6A
Self Check = 0
>
```

b: SET CONTROLLER PRIORITY.

NOTE

This command is not useful in the field and is rarely used in house.

This command assigns the controller arbitration priority number to the controller. This number is used in the arbitration between multiple contending disk or tape controllers and must be issued prior to any command that results in bus arbitration (any indirect command). You must set the controller priority to one of four settings (0,1,2, or 3).

CAUTION

If other controllers are plugged in and the controller priority is changed to a value of an existing controller, the BL700 will trap. Refer to Appendix G for more information on trap messages.

```
b<RETURN>
Enter controller priority number: 2<RETURN>
Continue as specified (y or n)?: y<RETURN>
Do you want to loop on command (y or n)?: n<RETURN>

***DONE***
>
```

c: ABORT.

This command sends abort to the Tape Controller, stopping its current command execution.

```
>c<RETURN>
Continue as specified (y or n)?: y<RETURN>
Do you want to loop on command (y or n)?: n<RETURN>

(Press any key to continue).
```

d: MEMORY INTERFACE.

This command sends a memory transfer test command to the Tape Controller, testing memory transfer capabilities of the Tape Controller. The response to this command is one of two results.

PASS: The test was executed without error.
FAIL: The Tape Controller was unable to complete the test without error.

```
d<RETURN>
Continue as specified (y or n)?: y<RETURN>
Do you want to loop on command (y or n)?: n<RETURN>
PASS
***DONE***
>
```

e: LOOPBACK.

This command sends a loopback test command to the Tape Controller. This command tests the signals going to and coming from the Tape formatter. Run this test by installing the loopback connector (Britton Lee part number 133-0118-054) on the TPC top edge connectors or on the TPC Interface Panel. One of two results is printed on the Console Terminal.

PASS: This means that the tape interface is working correctly.
FAIL: This means that the test failed. An error code explains what failed. (The error codes are defined in Appendix G).

```
e<RETURN>
Continue as specified? (y or n): y<RETURN>
Loop on command? (y or n): n<RETURN>
PASS
>
```

f: RETURN.

This command returns to the TPC Command Menu.

```
g<RETURN>

TPC COMMAND MENU
**********************
COMMAND       COMMENT
**********************
a   direct
b   indirect
>
```

TPC INDIRECT COMMANDS

This command accesses the TPC Indirect Command menu. One or more Command Blocks (CB's) are set up in memory to diagnosticaly exercise a drive. The menu and descriptions follow the TPC Direct Command Menu descriptions.

```
TPC INDIRECT COMMANDS
**********************
COMMAND COMMENT
**********************
a read
b write
c poll
d special
e return
>
```

a: READ.

This command starts the program prompting for a read CB. The read CB, when executed by the tape controller, causes the tape transport to read a 2048 byte block of data from the tape. The data is transferred to the selected buffer.

```
a<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
What buffer do you want? (a,b,c or d): b<RETURN>
Continue to next CB? (y or n) y<RETURN>
>
```

b: WRITE.

This command starts the program prompting for a write command block. The write command block, when executed by the tape controller, causes the tape transport to write a 2048 byte block of data on the tape. The data is transferred from the selected and previously "written" buffer. The example starts with clearing the buffers, setting buffer "a" to value ABCD, and moving to the Indirect menu.

```
c<RETURN>
h<RETURN>
What Buffer do you want? (a,b,c or d): a<RETURN>
Starting word offset: 0<RETURN>
Amount of words: 1024<RETURN>
Word value (hex): ABCD<RETURN>
b<RETURN>
```

In the second part, write is invoked only once and a transport and formatter selected. Buffer a, previously set to ABCD is used and we will chain to another Command Block command.

```
b<RETURN>
Is there another write command after this one? (y or n): n<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
What buffer do you want? a<RETURN>
Continue to next CB? y<RETURN>
b<RETURN>
>
```

c: POLL.

This command starts the program prompting for a poll command block. The poll command retrieves information about the drive regarding its readiness to receive a command. This process is not complete until returning to the main menu and invoking execute. Then TD reports the status.

```
c<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
Continue to next CB? (y or n): n<RETURN>
e<RETURN>
d<RETURN>

Selected transport is on-line
Selected transport is file protected
Selected transport at load point
Selected transport not phase encoded
>
```

d: SPECIAL.

This command executes the TPC SPECIAL COMMAND menu. Upon completion of a SPECIAL COMMAND entry, TD reverts to the INDIRECT COMMAND menu. There is no "return" SPECIAL COMMAND.

```
d<RETURN>

TPC SPECIAL COMMAND
**********************
COMMAND COMMENT
**********************
a file sear.  for.
b file sear.  rev.
c rewind
d rewind & unload
e write file mark
f security erase
g sel.  high spd
>
```

e: RETURN.

This command returns the operator to the TPC COMMAND MENU.

```
e<RETURN>

TPC COMMAND MENU
**********************
COMMAND COMMENT
**********************
a direct
b indirect
```

TPC SPECIAL COMMANDS

Each of these commands prompts the operator for the information needed to setup each of the special indirect commands.

a: FILE SEARCH FORWARD.

This command starts the program prompting for a file search forward command block. This command searches a tape for a selected quantity of file marks. When the correct number is found, the tape is positioned right after the last file mark.

```
>a<RETURN>
How many filemarks do you want to look for?: 5<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
Continue to next CB? (y or n): y<RETURN>
>e<RETURN>
>d<RETURN>
***DONE***
>
```

"Done" status indicates that the transport did search and find the specified number of file marks and is currently positioned over the fifth file mark.

b: FILE SEARCH REVERSE.

This command starts the program prompting for a file search reverse command block. This command is the same as command a, except the tape motion is in reverse.

```
>b<RETURN>
How many filemarks do you want to look for?: 5<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
Continue to next CB? (y or n): y<RETURN>
>e<RETURN>
>d<RETURN>
***DONE***
>
```

c: REWIND.

This command starts the program prompting for a rewind command block. The rewind command causes the tape to be rewound to BOT. The Tape Controller expects rewind to complete no more than 2 minutes, 41 seconds after rewind is issued. Otherwise, a message will be printed indicating the tape controller timed out waiting for rewind to complete.

```
>c<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
Continue to next CB? (y or n): y<RETURN>
>e<RETURN>
>d<RETURN>
***DONE***
>
```

d: REWIND & UNLOAD.

This command starts the program prompting for a rewind and unload command block. This command is like command c, except that the tape is rewound completely so that it may be removed from the transport.

NOTE

Some transports may not support this command.

```
>d<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
Continue to next CB? (y or n): y<RETURN>
>e<RETURN>
>d<RETURN>
***DONE***
>
```

e: WRITE FILE MARK.

This command starts the program prompting for a write file mark command block. This command writes two file marks on tape and backspace over the second one. This can accomplish one of two things.

1. Logical end of tape, if no more writes occur.
2. End of file, if another write occurs, the second file mark is erased before the write begins.

```
e<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
Continue to next CB? (y or n): y<RETURN>
>e<RETURN>
>d<RETURN>
***DONE***
>
```

f: SECURITY ERASE.

This command starts the program prompting for a security erase command block. This command causes the entire tape to be erased from BOT to EOT. Because of this starting point, the tape must be at BOT. If it is not, the command is terminated with an error. Use the following formula for the time determination of the tape erasure. Length of tape times tape speed equals the time required to erase tape.

```
f<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
Continue to next CB? (y or n): y<RETURN>
>e<RETURN>
>d<RETURN>
***DONE***
>
```

g: SELECT HIGH SPEED.

This command starts the program prompting for a select high speed command block. This command should only be used for streamers operation at high speed. This command may have to be issued after every rewind. Some streamers' are automatically reset to low speed.

```
g<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
Continue to next CB? (y or n): y<RETURN>
>e<RETURN>
>d<RETURN>
***DONE***
>
```

The following is an example of a write/read test with compare results. The purpose of the test is to show the ability of the tape drive to write 2 blocks of data (each block is 2K bytes in length) from the tape controller, rewind and read them back, and compare the results.

NOTE

Before proceeding, ensure that the write ring is installed on the tape, the tape is loaded, online, and at BOT. If you haven't the time to work this exercise, refer to the menus and follow the operation.c<RETURN>

```
h<RETURN>
What Buffer do you want? (a,b,c or d): a<RETURN>
Starting word offset: 0<RETURN>
Amount of words: 1024<RETURN>
Word value (hex): ABCD<RETURN>
b<RETURN>
b<RETURN>
Is there another write command after this one? (y or n): y<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
What buffer do you want? a<RETURN>
Continue to next CB? y<RETURN>
b<RETURN>
Is there another write command after this one? (y or n): n<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
What buffer do you want? (a,b,c or d): a<RETURN>
Continue to next CB? (y or n): y<RETURN>
d<RETURN>
c<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN
Continue to next CB? y<RETURN>
a<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
What buffer do you want? (a,b,c or d): b<RETURN>
Continue to next CB? (y or n) y<RETURN>
a<RETURN>
What transport do you want? (0,1,2 or 3): 0<RETURN>
What formatter do you want? (0 or 1): 0<RETURN>
What buffer do you want? (a,b,c or d): c<RETURN>
Continue to next CB? (y or n) y<RETURN>
e<RETURN>
d<RETURN>
***DONE***
j<RETURN>
First buffer name (a,b,c or d): a<RETURN>
Second buffer name (a,b,c or d): b<RETURN>
Amount of words: 1024<RETURN>
=
j<RETURN>
First buffer name (a,b,c or d): a<RETURN>
Second buffer name (a,b,c or d): c<RETURN>
Amount of words: 1024<RETURN>
=
c<RETURN>
>
```

CHECK DATABASE (CKDB)

INTRODUCTION

Check Database (CKDB) is a utility that looks for data inconsistencies on a Britton Lee database server. It is primarily a debugging tool and is useful for Britton Lee Product Support and customers. Because it is intended for occasional use, it does not receive commands through the BL700's channels; instead, it receives commands through the console port as does Disk Format Utility (DFU).

CKDB's primary function is to verify the physical structure of databases. Many sources of inconsistencies are examined including malformed tuples, invalid indices, and incorrectly allocated data. You may check a single relation in one database, every relation in all databases, or a range of items in between.

All problems are flagged with a "severity" code and a short description of each error is given. An error summary is printed after the scan of the database has been completed. Usually, inconsistencies are reported and corrective action is not taken. However, some types of errors may be corrected if you make an explicit request. Corrections are not performed if there is a possibility that CKDB may have misdiagnosed the problem.

Other major CKDB functions aid in solving a problem once its diagnostic tools have been detected. For example, CKDB can read blocks from the BL700's disks. It can also perform simple, one-variable retrieves and read IDM tapes. These modes are intended to be used primarily by Britton Lee support and development personnel.

CKDB's remaining capabilities are not closely associated, except that they are useful for specific debugging purposes. These modes allow you to do the following:

Verify bad block remapping information on a disk.
Compare two databases for logical equivalence.
Check that databases on mirrored disks are identical or if they are not, which mirror is more consistent than the other.
List the databases on the BL700.
List the relations in a database.
List the disks on-line.
Read blocks from an IDM tape.

There are options that change how CKDB works. Some options affect many functions, while others affect only a single function. For instance, an option may allow a support engineer to obtain additional information about a relation that would not be useful to an end-user.

STARTING CKDB

Refer to Appendix C for Software Loading Procedures. Use one of these three entry lines to start CKDB.

```
BL700 - Filename: kernal ckdb<RETURN>
BL700 - Filename: load kernal ckdb<RETURN>
BL700 - Filename: loadtape kernal ckdb<RETURN>


KERNAL 39 7/23/85 16:14:56
Accessibledisks:
Name      Ctlr   Drive    Low      High       Status
'system'  0      0        1        73260      System
'user1 '  1      0        73260    146520
loading syscalls
loading ckdb
loadingckdb2
number of available dbins: 126
Free process pages: 83
CKDB Version 6 Wed May 8 13:22:55
Name of database:
```

CKDB PROMPTS

CKDB has three nested levels and three prompts. The highest level prompt is "Name of database:". You may return to this prompt from another prompt by typing <RETURN>. The other prompts will be introduced with their subject matter.

INITIAL OPERATION

When you enter the name of a database, CKDB examines every object in the database. The name and relid of each relation will be printed. If a relation has any indices, the type and keys of each index will be printed as well. As CKDB moves to each new task, it reports the object it is checking. After CKDB is finished with a database, it displays the number of pages, tuples, relations, and indices that were examined. If any errors occurred, the numbers of errors will also be printed. If you want to examine system, type "system" after the prompt. Below is a reasonable facsimile of what will be displayed for a system database:

```
Name of database:  system<RETURN>
Examining relation relation          1
   unique clustered index on         relid
Examining relation attribute         2
   unique clustered index on         relid attid
Examining relation indices           3
   unique clustered index on         relid indid
Examining relation protect           4
   unique clustered index on         relid user access
Examining relation query             5
   unique clustered index on         relid num seq
Examining relation crossref          6
          clustered index on         relid
Examining log transact               7
Examining log batch                  8
Examining relation host_users        9
   unique clustered index on         hid huid
Examining relation disk_usage        10
          clustered index on         relid low
Examining relation blockalloc        11
Examining relation databases         12
   unique clustered index on name
Examining relation users             13
          clustered index on         name
Examining relation lock              14
Examining relation description       15
          clustered index on         relid attid key
Examining relation disks             16
          clustered index on         name
Examining relation monitor           17
          clustered index on         seqno
Examining relation dbinstant         18
Examining relation configure         19
          clustered index on         type number
Examining relation devmonitor        20
          clustered index on         seqno type d3
Examining relation account           21
   unique clustered index on         hid huid
Examining relation login             22
   unique clustered index on         huname hid
Examining nonclustered index 1 on    huid hid
Examining file dumpload              373
Examining file kernal                3031
Examining file compile               3384
Examining file syscalls              6042
Examining file execute               6395
Examining file fileload              9053
Examining file recover               9406
Examining file dfu                   12064
Examining file listen                12417
Examining file dfu2                  15075
```

```
Examining file ckdb                    15428
Examining file wcs                     18086
Examining file ckdb2                   18439
Examining file front                   21097
Examining file memtest                 21450
Examining file qryproc                 24108
Examining file dd                      24461
Examining file support                 27119
Examining file td                      27472
Examining file sort                    30130
Examining relation jojo                30483

system: examined 468 pages, 1218 tuples, 43
relations, 38 indices
Name of database:
```

Relation jojo is an example of a user relation merged into a system database.

CKDB CONVENTIONS

CKDB has several rules that help give it consistent user interaction. Exceptions are described in the appropriate sections below.

1. CKDB commands are indicated by "/", followed by a lower case letter (for example: /q, /a, /x, and /d).
2. Options are a "-" followed by a single letter. (-q, -b, -t, and -v are examples).
3. Normal words are assumed to be the name of a database or a relation. (system, databases, configure, and jojo are examples).
4. Blanks (spaces) may be used freely where they do not cause a word or number to be broken. Their only function is to delimit tokens; beyond that, extra "white space" is ignored.
5. **Number Displays** CKDB outputs numbers in decimal and octal.

   Integer values are octal if a leading "0" is present.
   (0 121 456 is an example).
   All other numbers are decimal.

   An exception occurs in page dump mode. Page dump mode prints the bytes of data from the page in octal without the leading zero; it is not possible to display 16 bytes on a single 80-character line if the "0" is included.

   Your input to CKDB is assumed to be decimal except:

   ```
   Start an octal number with a "0".
   Start a hexadecimal number with a "#".
   ```

6. **Carriage Return** The <RETURN> key is interpreted by most sections as a quit command. In general, entering an empty line causes CKDB to "pop" back one level. For example:

   ```
   Relation to examine: <RETURN>
   Name of database:
   ```

   This feature can be used to correct errors on previous lines, as shown next.

   ```
   First database? wrongdbname<RETURN>
   Second database?<RETURN>
   First database? rightdbname<RETURN>
   Second database?<RETURN>
   First database?<RETURN>
   Name of database:
   ```

   But during a yes-or-no question, <RETURN> causes the question to be asked again.

   ```
   Really reboot?<RETURN>
   Please answer  y or n
   Really reboot? n<RETURN>
   ```

   However when the prompt says, "<RETURN> to continue", (dump and query modes), <RETURN> indicates that you would like to continue viewing more of the information that was just displayed.

At the top level prompt "Name of database:", <RETURN> is ignored.

```
Name of database:<RETURN>
Name of database:
```

7. **Unprintable ASCII Control characters** Data with values between octal 0 through 31 are displayed as "^ <char>", where <char> is obtained by adding 0100 to the byte. Thus, a null byte is printed as "^@". The other unprintable ASCII character (delete, 0177) is displayed via "^?". This can happen while CKDB displays relation or attribute names or when a character attribute of a tuple is displayed. CKDB strips off the uppermost (0200) character bit before sending it to the console.

8. **Interrupting a Command** When CKDB is running an operation, it may be interrupted by pressing the <SPACE> key. After a few seconds, CKDB stops and requests the next command. Wait until CKDB acknowledges the interrupt before entering the next command.

9. **The Current Database** CKDB has a "current" database concept. This refers to the database CKDB is currently reading. CKDB identifies the current database for you.

```
Name of database: -r db<RETURN>

Relation to examine: /r otherdb<RETURN>
relation  1   attribute  2    indices    3    protect 4
query     5   crossref   6    transact   7    batch   8
host_user 9   disk_usage 10   blockalloc 11   users   1
3descriptions 15
Current database now is db
Relation to examine:
```

OPTIONS

CKDB options allow you to customize the basic verification operation of a database. Options may be enabled or disabled at any time except during a yes-or-no question. All combinations of options are allowed; precedence between conflicting options resolves conflicting requests. For instance, the time it takes to print relation names and index fields on a slow terminal might be a limiting factor when the speed of CKDB is considered. In this case, it might be undesirable to print the names. An option has been provided which prevents them from being displayed.

OPTION LIST DISCUSSION

Option List

All options start with the character "-" and have a single character name. The list may be seen by typing "-?<RETURN>". CKDB always presents the option letters in lower case. Options are changed only by typing them on the Console keyboard.

Option Description

Each option is listed as a declarative statement. If you wish the condition AS STATED, type the option letter in lower case (TRUE). If you wish to negate, drop, or ignore the option, type the option letter in UPPER case (FALSE). Entering a letter in the same case twice does not cause the state to change. More than one option may be typed in a string, following the option "-".

Option True/False state

Each option has two states. CKDB displays the current state of each option when you type "-?<RETURN>". The next display shows the states when you first enter CKDB.

```
Name of database: -?<RETURN>
-b Begin a scan at a certain relid or relname.    False
-c Print pages as characters.                     False
-d Enter dump page mode on error.                 False
-e Echo terminal input.                           True
-f Automatically fix correctable.                 False
-i Examine a particular index.                    False
-l /c option - compare logged relations only.     False
-q Quiet. Print only error messages.              False
-r Examine a specific relation.                   False
-t Print pages as tuples.                         True
-v Print much information about each relation.    False

Name of database:
```

Examples:

Examine some particular relations in the system database. Examine all databases with the "quiet" option on and the "single relation" option disabled. (Notice the space between options and the command).

```
Name of database: -qR /a<RETURN>
```

The same actions could be performed by entering:

```
Name of database: -q<RETURN>
Name of database: -R<RETURN>
Name of database: /a<RETURN>
```

BOOLEAN VS NUMERIC OPTIONS.

Most of CKDB's options take only two possible values (true and false). Several of these boolean options can be set at the same time, starting the list with a single -, as shown above.

However, some options can have a numeric value as well as an implicit boolean one. If you follow the option name with a number, then the option is not only "true", but in addition, it has the numeric value specified. It is possible to set a numeric option to true but not give it an integer value. Turning a numeric option off removes any numeric value that it may have had.

OPTION DESCRIPTIONS.

The following option descriptions contain the meaning of the option, its letter option name, and its initial value.

-b BEGIN SCANNING AT A CERTAIN RELATION FALSE

When an entire database is checked, the relations are processed in ascending relid (Relation Identification) order. The "begin" option allows the scan to start somewhere in the middle of the database. After a database is named, CKDB examines this flag to see if it should ask for a starting point.

```
Name of database: system<RETURN>
Beginning relation: blockalloc<RETURN>
```

A relation may be specified in several ways:

1. The name of the relation. This may be ambiguous if there are two relations of the same name but with different owners.
2. The name, a colon ":" and the owner of the relation. This specifies the object uniquely.
3. The integer relid of the relation. This also identifies it uniquely. To leave this mode and return to the top-level prompt, enter <RETURN>.

-c DUMP PAGES AS CHARACTERS FALSE

The "character" option tells the dump page function of CKDB to display disk pages as printable ASCII characters. See the description of dump mode (/d) for further details.

-d ENTER DUMP MODE ON ERROR FALSE

This option lets disk pages be examined whenever an inconsistency is detected. After printing a short explanatory message, CKDB calls for the dump page function (/d). Data related to the error can then be examined to obtain the complete description of the problem. Upon returning from dump mode, CKDB continues from the point at which the error occurred.

-e ECHO TERMINAL INPUT TRUE

If a terminal is connected directly to the Console Port, you should probably enable echo. If the console port is connected to a host computer that does its own terminal echo, it may be desirable to turn echo off.

-f FIX CORRECTABLE ERRORS FALSE

There are many correctable errors that CKDB can safely correct. For instance, it can deallocate pages that belong to a relation that no longer exists.

There is one correctable error that CKDB does not print unless fixing is enabled. This error is the correction of the tuple and page counts in the Relation relation. It is common for these values to be inaccurate because they are guaranteed correct only when the BL700 is turned OFF and back to SAFE to bring all databases to a consistent state. CKDB does not consider it an error for tuple and page counts to be wrong, but it corrects them because it is convenient to do so. When one of these counts is updated, CKDB prints:

```
Changing tuple count from <old value> to <correct value>
```

or

```
Changing page count from <old value> to <correct value>
```

-i EXAMINE ONLY CERTAIN INDICES FALSE

When examining single relations, it is useful to examine a particular index, especially if one is suspected of being in error. By typing:

```
Name of database: -i 0<RETURN>
Name of database: system<RETURN>
```

CKDB checks only the clustered indices (index id 0) of all the relations in the system database. Other indices are not examined, although all of the remaining checks are still performed. Note that this is a numeric option and must be listed as the last option.

-l COMPARE ONLY LOGGED RELATIONS FALSE

This option is used only by the compare database mode function. Refer to the "/c" command for a complete description.

-q QUIET TRUE

This option prints error messages and user prompts only, when true is selected. It is useful when a slow device is attached to the Console.

It temporarily hides the "verbose" option "-v". When the "quiet" option is turned off, the state of the verbose option will be apparent. (See also Verbose option).

-r NAME SPECIFIC RELATIONS TO EXAMINE FALSE

Sometimes only a few relations need to be verified. With this option enabled, CKDB asks for the name of the relations to be checked, one by one. As with the "-b" option, a relation may be specified either by its numeric relid or by its name. If a name is chosen, it is possible that more than one relation will be verified because two users may have relations with the same name. Any number of relations may be examined one at a time. Entering an empty line returns CKDB back to the top level prompt. The allocation of each relation will be examined after the data and indices are verified.

If the database has more than 1000 relations in it, the system catalogs relation, attribute, and indices, and examines them automatically. CKDB avoids using the indices on these relations unless it knows that they are good. However, it uses them often enough that the examination would be slowed significantly if it could not take advantage of their indices.

-t PRINT PAGES AS TUPLES TRUE

The "tuple" option means that CKDB should display disk pages as tuples. See the dump mode (/d) description for further details.

-v VERBOSE FALSE

Print the name and relid of each relation and the numerical ID and status of each index. Enabling this flag causes more information from the relation and indices relations to be printed.

```
Relation to examine: -v relation<RETURN>
Examining relation relation      1


first < 01 : 014 : 100 > last < 01 : 014 : 100 > type S
status 0327 tuples 54 pages 7
2 empty data pages
unique clustered index on relid
status 0203 root < 01 : 014 : 103 > ituplen 5 card 1
No root
unique non-clustered index 1 on name owner
status 01 index 1 root < 01 : 014 : 103 > ituplen 21 card 1


Relation to examine: -v<RETURN>
Relation to examine: 1<RETURN>
Examining relation relation 1<RETURN>
unique clustered index on relid<RETURN>
unique non-clustered index 1 on name owner<RETURN>
Relation to examine:
```

CKDB COMMAND FUNCTIONS

There are several modes of CKDB which do not directly check the physical structure of a database, but nonetheless are useful features. All of these sub-functions are indicated by "/<letter>"; some of the commands optionally accept a database name after the function name. These functions may be called upon at almost any point where CKDB accepts input so that a call to one command function can be nested inside a call to another command function. Nested calls to the same function are NOT allowed.

```
Name of database?: ?<RETURN>
/a All databases
/c Compare databases
/d Dump page mode
/l List databases
/m Checks halves of a mirror
/q Execute queries
/r List relations
/s Scan disks
/t Tape mode
/x Exit from ckbd

Name of database?:
```

/a ALL DATABASES

This command will print the names of every database before it is examined. This is only valid from the top-level prompt "Name of database:". Every database on a BL700 may be verified by typing:

```
Name of database: /a<RETURN>
```

/c COMPARE DATABASES

For the verification of channel drivers and the dump, load and rollforward commands are useful to know if two databases are logically identical. Compare database mode is entered by typing "/c" after Name of database:. Database names are not employed at the top-level prompt.

```
Name of database: /c<RETURN>
First database ? db<RETURN>
Second database ? backup<RETURN>
relation        1
attribute       2
```

CKDB will then compare each relation or file in the first database with its counterpart in the second database, on a tuple by tuple or byte by byte basis.

It doesn't try to find the minimal subset of differences which would make the objects equal. A single tuple inserted into the beginning of one relation is all that is needed for CKDB to think that they are vastly different.

The "-r" and "-b" options may be used to compare only certain relations in the databases.

1. Comparing Relations

Tuples from each relation are retrieved and compared one at a time. The tids and attributes of tuples which differ are printed. CKDB stops checking a relation when it finds five mismatching tuples; it then prints a message and continues to the next relation.

```
Too many differences
```

2. Comparing Files

The comparison of files stops as soon as a single byte difference has been detected, as in:

```
<filename> differs at <offset> <byte1> <byte2>
```

3. Relations Which May be Compared

A relation or file in one database can be compared to its equivalent in another database only if both relations have:

the same name
the same owner
the same type and number of attributes

The relations do not need to have the same relid. If a relation exists only in the first database, CKDB will issue the message:

```
<name> not common to both databases
```

and continue to the next relation in the first database. If the relation does exist in both databases but the attributes are not identical:

```
<name> has different attributes in the second
database
```

NOTE

Relations which are only in the second database are silently ignored.

4. Relations Which Cannot be Compared

There are some objects which, even though they meet the above criteria, cannot usefully be compared.

System Catalogs
Some system relations, such as blockalloc and disk-usage, are not comparable. These relations depend upon the physical layout of the blocks on the disk and are guaranteed to be different across databases. CKDB also ignores the disk-dependent fields of the relation and indices relations (firstpg, lastpg, and rootpg) so they may be compared meaningfully.

Special Relations
The text of views, stored programs, and stored commands is stored in the system query relation. Equivalence of these objects is checked when the query relation is compared. Compare mode silently skips over these objects.

5. Options for Compare Mode Compare Only Logged Relations

The logged-only option makes CKDB skip relations which are not logged thus avoiding spurious error messages. If one database is the "-l False" output of a rollforward, and another is the original database, non-logged relations that were changed will not be identical. When a non-logged relation is encountered, a message will be printed. The comparison will continue to the next relation.

```
<relation> is not logged
```

Enter Dump Mode
Turning on the "-d" option indicates that CKDB should enter dump page mode (/d) when a pair of tuples do not match. After returning from dump mode the comparison will continue.

Begin and Specific Relation Options
The "begin" option "-b" can be used to start the comparison at a certain relation. This is useful in skipping over relations which have previously been found to match. Similarly the "single-relation" can be used to compare only specific relations.

/d DUMP PAGE MODE

This function allows a disk page to be looked at in variety of formats. You need to know how to tell CKDB which page to read. It prompts with:

```
Page number to read:
```

1. Specifying a Page

Every block on a BL700 is identified by a page number which can range from 0 to $2^{24}-1$. It can be specified in several ways:

a. A three byte page number
Each byte is given separately. Much of the database management code uses these three byte page numbers to identify a disk block. By default these numbers are interpreted as octal; you can specify hexadecimal by preceding each number with a "#". There is no way for them to be entered as three separate decimal numbers.

```
Page number to read: 0 51 227<RETURN>
Page number to read: 0 #21 #97<RETURN>
```

b. A single long page number
This one is decimal by default since many error messages (hard disk errors and syserrs) print out block numbers as a single long decimal integer. All of the following are equivalent to the previous example:

```
Page number to read: 020627<RETURN>
Page number to read: 8599<RETURN>
Page number to read: #2197<RETURN>
```

c. A four byte tuple identifier (tid)
The first three bytes of a tid is a page number; the last byte uniquely identifies the tuple on that page.

d. A physical block specification consisting of:
disk controller number (ranges from 0 to 3)
disk drive number (ranges from 0 to 15)
cylinder of the block
track of the block
sector of the block

```
Page number to read: c 0 5 47 6 18<RETURN>
```

e. These numbers default to base 10. This type of page identifier is printed by the low level disk error routines.

```
Disk hard err code xE000
cmd 3, ctlr 0, drive 1, cyl 35, head 7, sec 1
errcode 100, errstat 97, phyaddr 163800
cmd was retried and failed.
```

A "b" in front of the previous methods of page numbering. Occasionally, the information that you are interested in is not on the page itself but in its blockalloc tuple.

```
<RETURN> to continue: b 8599<RETURN>
```

If you accidentally entered dump mode, just type a <RETURN> to return back to the previous mode.

2. Checking Out the Page before Reading it

CKDB finds out several interesting things about a page just before reading it. If the page does not belong to the current database it will say so.

```
Page number to read: 0 23 303<RETURN>
<0 : 23 : 303>  belong to testdb  (dbid 8)
```

When the page to read is on a mirrored drive, CKDB will ask you from which drive it should read.

```
Page number to read : 0 23 303<RETURN>
<0 : 23 : 303> is mirrored on ctlr 0 drive 0 and ctlr 1
drive 0
Use first drive ? n<RETURN>
Using second drive
```

It reads the disk sector header of the page to see if it has been remapped to another page. If the page number on the page is not the same as the page that was specified, it prints:

```
Ask for <3 : 124 : 340>, got  <3 : 127 : 103>
```

It looks at the "mode" field of the page's blockalloc tuple for one of the following cases:

02	the page is free
020	the page was marked bad by DFU and presumably contains a media defect.
0104	a file data page
0220	a bad page which is remapped to another block
0204	a good page to which a bad block is remapped

If none of the allocation modes is found and the mode does not have the used (04) bit on, then the page is allocated incorrectly.

3. Display Format

The data from a page may be printed in a variety of formats. The first 14 bytes will always be printed as the internal page header (pageid, status, nextpg, nexttno, relid, freeoff, and level). The rest of the page may be displayed as octal, bytes, characters @, or tuples. By default, they are displayed as 12 rows of 16 octal bytes. Two options used only by dump mode determine how the data of a page should be printed.

PRINT AS CHARACTERS -c

If the "-c" option is set, bytes that are printable ASCII or EBCDIC characters will be shown as characters. Data that is outside the printable range of <SPACE> to "~" will be displayed in octal. Bytes that contain values between 0 and 7 will be preceded by a "\" to distinguish them from the digits "0" to "7" (values 060 - 067).

PRINT AS TUPLES -t

The "-t" tells CKDB to try to print the page as a sequence of tuples. The True tuple format is similar to that of the IDL parser. All the attributes of the relation are printed first, followed by a few tuples worth of data. To print tuples, CKDB must be able to get the attributes of the current page. This is quick and easy to do for data pages. However, it is not as efficient to find the attributes of a particular index page; CKDB may have to search all the indices of the relation in order to find out which index a particular page belongs to. It estimates how many index pages there may be; if it thinks that it will take a long time to search them all, it will ask whether you want to wait for it to find the index.

```
There are 23 indices to search
Continue ?
```

"@" characters are printed in the mode appropriate to the database from which the page was read. Thus, EBCDIC characters are translated to their ASCII equivalents before they are printed.

This is the "indid" attribute of the indices relation. The indid of a clustered index is always 0; non-clustered indices are numbered from 1 to 250. When a non-clustered index is examined, its indid is printed before the keys of the index, as in:

```
Examining relation relation 1
clustered index on relid
non-clustered index 1 on name owner
```

If for any reason CKDB is unable to find the attributes, the tuple area of the page will be displayed as characters or octal bytes. Attributes may be unavailable because the database has been destroyed, or the database is locked for a load or rollforward, or the relation has been destroyed.

If you elect to continue searching the indices, CKDB will print out the indid and attributes of the index when it is found. If you decide against waiting, the page is displayed as if the "-t" option

were not turned on. However, all is not lost. If you know which index this page belongs to, you can tell CKDB by specifying a decimal index identifier after the "-t", as in:

```
<RETURN> to continue: -t<RETURN>
```

Whether or not the attributes were found, the tuple number table at the end of a data page will be displayed as decimal offsets to tuples from the start of the page. The "tuple" option takes precedence over the "character" option.

4. Looking at a Page

Disk blocks on a BL700 contain 2048 bytes. It is not practical to print the whole page out at a time. Dump mode tries to print out about half a screenfull of data. After printing those 12 lines "#", CKDB will respond with:

```
<RETURN> to continue:
```

Type ? here to get the list of dump page mode sub-commands. These commands are local to dump page mode and are only recognized after you have read the page.

j<n>	jump to offset <n> in the page
f	follow the next page pointer in the page header
l	list next 10 page headers
p	find previous page
q	run a query on current page's relation
t<n>	go to tuple number <n> on this page
x	exit from this page
.[+ -]<n>	go forward or backward <n> pages
<pno>	exit from this page, read another

NOTE
Only six lines are displayed if the "quiet" option is set.

A command that causes another page to be read f,p, or <pno> may be followed by another command, such as:

```
<RETURN> to continue: f j 1000<RETURN>
```

to read the following page and then start displaying from the middle of it.

j <n> JUMP TO OFFSET <n> IN THE PAGE

The "jump" command allows data on the page to be seen in any order. The decimal number after the "j" (and optionally any blanks) should fall in the range 0 to 2047. If the offset given points into the middle of a tuple, CKDB will adjust the offset so the whole tuple will be printed. Otherwise, CKDB will round the address down to a 16 byte (Quadword) boundary. (Jumping to 37 will cause CKDB display from offset 32).

f FOLLOW NEXT PAGE POINTER IN THE PAGE HEADER

A page header contains a "next page" field. This page is logically related to the current page, and it is common to follow a chain of these pointers down a linked list of pages. The "f" may be followed by a count of the number of pages to read; by default it goes forward one page. The end of the list is marked by a next page of "<0 : 0 : 0>". CKDB prints this if it is told to go beyond the end.

```
<RETURN> to continue: f<RETURN>
On last page
<RETURN> to continue:
```

l LIST NEXT 10 PAGE HEADERS

Just as it is useful to follow the next page pointer, it is sometimes desirable to look at a list of page headers.

```
<RETURN> to continue: l<RETURN>
<0 : 3 : 5> nextno 012 freeoff 48 stat 021
<0 : 3 : 6> nextno 013 freeoff 66 stat 021
<RETURN> to continue:
```

CKDB prints up to 10 page headers. If the relid or level of the next page does not match the previous page, CKDB will note that and stop. It also indicates where there are more pages after the tenth, as shown in the following example:

```
<RETURN> to continue: 1<RETURN>
<0 : 3 : 17> nextno 012 freeoff 48 stat 021
<0 : 3 : 20> nextno 012 freeoff 50 stat 021
    .
    .
    .
<0 : 3 : 35> nextno 012 freeoff 46 stat 021
<0 : 3 : 36> nextno 013 freeoff 66 stat 021 ... more
<RETURN> to continue:
```

p FIND PREVIOUS PAGE

Disk pages are grouped together in singly-linked lists. Occasionally, it is desirable to go backwards in that list to find the page that points to the current one. Since there is no backwards pointer, there is no quick way to do this. The "p" command tells CKDB to search the entire database for a page which points to the current one. First it looks among other pages which are allocated to the same relation. If it does not find a "previous page" at this point, CKDB will ask:

```
Try unallocated pages? y<RETURN>
```

Enter "y" or yes and CKDB will look at all the other pages in the database. CKDB must use Blockalloc to restrict its scan of the disk. If the database was not opened, CKDB prints this message and stays at the current page. It does not try to find the previous page.

```
Database not opened.
```

q RUN A QUERY ON CURRENT PAGE'S RELATION

Dump mode allows you to run queries on the relation to which the current page belongs. You may restrict the queries to a subset of the relation: Start at the current page and go until the end of the data pages list. Refer to the "/q" mode description for further details.

t <n> GO TO TUPLE NUMBER <n> ON THIS PAGE

You can jump to a specific tuple on a data page with the "t<n>" command. Here <n> is the octal tuple number of the tuple that you want to look at.

```
<RETURN> to continue: t 27<RETURN>
tno       mode    check     stat     relid
152       027                        0
158       030                        0
164       031                        0
165       032                        0
166       033                        0
<RETURN> to continue:
```

.+/-<n> GO FORWARD OR BACKWARD <n> PAGES

This command takes the page number of the current page, adds or subtracts the number given after the ".", and goes to that page. It is not a duplicate of the "f" or "p" commands. They read along a linked list [+ -] <n> of pages; the command does not care whether the next higher block number is the next page of the current block. If you are currently on page 3150, subtracting 20 is equivalent to addressing 3130.

```
<RETURN> to continue: .-20<RETURN>
<RETURN> to continue: 3130<RETURN>
```

If the current page is on a mirrored disk, the next page will be read automatically from the same drive. CKDB will not ask you whether or not it should read from the first drive of the pair or the second drive.

x EXIT FROM THIS PAGE

To stop looking at this page, type:

```
<RETURN> to continue: -x<RETURN>
Page number to read:
```

CKDB will return to the top level of dump page mode and ask for the next page to scan. Once at the second level prompt, entering <RETURN> will move CKDB up to the top level prompt.

```
Page number to read: <RETURN>
Name of database:
```

Another way to exit from the page is to go to the end of the page and enter an empty line. The prompt changes here to inform you that you are about to "fall out" of dump mode.

```
2032  514  508  502  496  490  484  478  472
<RETURN> to leave page: <RETURN>
Page number to read:
```

/l LIST DATABASES

Print a list of all the databases that are on this BL700.

```
Name of database: /l<RETURN>
Databases on this BL700:

attfoo     jwinery       mfgdb        mike
mike2      sprtest       system       vino2        workdb
Name of database:
```

/m CHECKS HALVES OF A MIRROR

This command can be used on mirrored databases which "mirror compare" of DFU has found to be inconsistent. It can temporarily un-mirror and re-mirror a drive so that CKDB can subsequently be run on all possible combinations of disks. You can then select the best set of drives and "mirror copy" from them.

Finding the best set of drives is somewhat complicated. The steps involved are:

1. Determine which databases are on the suspected drives. If you do not remember where the databases are, you may query disk_usage of each database with the "/l" function.
2. Find out the controller and drive numbers of the accessible disks with the "/o" command.

```
Name of database: /o<RETURN>
Ctlr 0 drive 2 disk160a 45091-118350 814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 2 disk160a 45091-118350 814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 0 disk160b 118351-191610 814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 0 disk160b 118351-191610 814(823)M cyls 10 heads 9 sectors
Name of database:
```

3. Select the drive(s) that you wish to check first; it does not matter which disk you choose. In this example, we shall turn off the mirroring on the second disks of each pair, so that only the first one will be used. Notice that the "M" is not present in the second drive any more; this means that the disk is not currently mirrored.

```
Name of database: /m 1 2<RETURN>
Ctlr 0 drive 2 disk160a 45091-118350 814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 2 disk160a 45091-118350 814(823) cyls 10 heads 9 sectors
Ctlr 0 drive 0 disk160b 118351-191610 814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 0 disk160b 118351-191610 814(823) cyls 10 heads 9 sectors
Name of database:
```

4. Toggle the mirroring on the drives so that we get the desired set of disks. In this case:

```
Name of database: /m 00<RETURN>
Ctlr 0 drive 2 disk160a 45091-118350 814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 2 disk160a 45091-118350 814(823)M cyls 10 heads 9 sectors
Ctlr 1 drive 0 disk160b 118351-191610 814(823)M cyls 10 heads 9 sectors
Ctlr 0 drive 0 disk160b 118351-191610 815(823)M cyls 10 heads 9 sectors
Name of database:
```

Now CKDB will only see the data that is on the disks at ctlr 0/drive 2 and ctlr 1/drive 0. Notice that we turned off the mirroring of the third drive. Because the BL700 wants the drive that thinks it is mirrored to be first, CKDB exchanged the order of the third and fourth drives.

5. Check the databases or specific relations which reside on the blocks numbered 45091 to 191610. The "fix" option -f should be off, because CKDB may report some spurious errors if you didn't happen to chose the "correct" set of disks. You must run mirror copy mode of DFU after using this command and checking any relations or databases.

6. To find out which disk drives are connected to this BL700 use the /o command. It prints a list of both accessible and inaccessible disks with their drive characteristics and whether or not they are mirrored. A "foreign" disk that actually has a disk name may be an out-of-date mirror, or it simply might need to be merged into the system with DFU.

```
Name of database: /o<RETURN>
Ctlr 0 drive 2 disk160a 45091-118350 814(823) cyls 10 heads 9 sectors
Foreign disks
Ctlr 0 drive 0 disk80a 1-36450 810(823)M cyls 5 heads 9 sectors
Name of database:
```

The fields are (all numbers in decimal):

a. The disk controller number (0-3).

b. The disk drive number (0-15).

c. The name of the disk.

d. The lowest block number on the drive.

e. The highest block number on the drive.

f. The number of cylinders available for use by the database management code. On disks formatted without a remap zone (release 25 DFU or earlier) this is the actual number of cylinders.

g. The actual number of physical cylinders of the drive. Only remap zone disks have this field. If it is followed by an "M", the drive thinks that it is mirrored.

/q EXECUTE QUERIES

It would be a tedious job to find a particular tuple in a relation just by using dump mode, looking at tuples one at a time until an interesting one is found. Fortunately, CKDB can execute a subset of one variable retrieves. This is a sub-mode which, like "dump" mode, can be entered at any time.

Entering Query Mode

Entering query modes involves selecting the database to use; once inside the database, many different relations may be retrieved without leaving this mode. There are two ways to start querying:

On a regular command line where options and functions are recognized, type:

```
Name of database: /q [<database>]<RETURN>
Relation to query:
```

to enter query mode. The database name is optional; by default the current database is chosen. CKDB will ask for a database name if there is no current database.

While looking at a page in dump mode, the command:

```
<RETURN> to continue: q<RETURN>
```

may be used to try to run a query on the relation to which the current page belongs. If it is a data page, CKDB will ask whether the query should start or end on this page.

```
<RETURN> to continue: q<RETURN>
Start on this page? y<RETURN>
End on this page? n<RETURN>
1)  attname relop value:
```

Respond "no" to use the first (or last) page of the relation; enter "yes" to start (or end) on the current page; type <RETURN> to either question to return out of query mode.

In either case, the attributes of the relation must be obtainable. Both dump and query mode have the same restrictions about attributes; refer to the discussion about the "tuple" option, "t" for further information.

Selecting the Query Variable

Query mode only allows single variable queries. The relation to use is specified when query mode is entered:

```
Name of database: /q testdb<RETURN>
Relation to query: relation<RETURN>
1) attname relop value:
```

You can select another relation by entering an empty line at this point:

```
Name of database: /q testdb<RETURN>
Relation to query: relation<RETURN>
1) attname relop value: <RETURN>
Relation to query: attribute<RETURN>
```

Qualifications

CKDB can run queries of the form:

```
retrieve (x.att 1, x.att 2, ....)
where (x.a <<relop> const 1) and
                .
                .
                .
(x/b <relop> const 2) and
(x.c <relop> const 3) or
(x.c <relop> const 4) or
(x.d <relop> const 5) go
```

The general format allows a string of "and" clauses followed by a string of "or" clauses; either or both may be empty. All of the "and" clauses must qualify; at least one of the "or" clauses have to be satisfied. Once the relation has been selected, CKDB asks:

```
1) attname relop value:
```

This is the prompt for query mode. Either simple clauses (restrictions) or commands which modify the target list (projections) may be entered here. The order of any restrictions is immaterial. Type <RETURN> or go to execute the query.

AND Clauses

The attribute names (a, b, c, or d in the example above) should be one of the attributes of the relation. The relational operator may be one of the following:

```
= or ==  attribute equality
! or !=  attribute inequality
>         attribute > constant
>=        attribute >= constant
<         attribute < constant
<=        attribute <= constant
```

It must be separated from the surrounding tokens by at least one blank.

How the constant is entered depends upon the type of the attribute. One-, two-, and four-byte integers are assumed (by default) to be decimal. Character strings start at the first non-blank after the relop and continue until the length of the attribute has been exceeded, the end of the line has been reached, or a blank has been seen. A character string that contains embedded blanks may be entered as a double-quote delimited string. Binary, binary coded decimal (bcd), and floating point data must be entered as a list of octal numbers, each one being the next byte in the binary string.

For example:

```
attname relop value: relid >=21<RETURN>
attname relop value: type = T<RETURN>
attname relop value: firstpg > 0 16 262<RETURN>
```

searches for all transaction logs which start after page "0 : 16 : 262".

OR Clauses

These are differentiated from "AND" clauses by the first character of the line. If it is a "|", then the rest of the line is added to a list of or clauses, other-wise it is treated as an AND. The "|" must be separated from the attribute name by a blank or tab. The following query retrieves the stored commands or stored programs owned by BL700 user 23.

```
attname relop value: | type = P<RETURN>
attname relop value: | owner = 23<RETURN>
attname relop value: | type = C<RETURN>
```

Changing the Target List

Normally, all the attributes of the relation will be displayed. This can get a bit tiring especially on a slow console port. By typing:

```
1) attname relop value: only <<att1> <<att2> ....
```

The domains mentioned will be the only ones displayed. Enter:

```
1) attname relop value: only<RETURN>
```

to retrieve all the attributes again. Attributes can also be excluded from the target list by typing:

```
1) attname relop value: not  <<att1> <<att2> ....
```

This prevents the specified domains from being printed. Attributes which have been removed from the target list may be used in the qualification.

Count

The count command removes all the attributes from the target list. No tuples will be printed; however, it will display the number of qualifying tuples.

Correcting a Query

If a mistake has been made in the query, it can be reentered by typing:

```
3) attname relop value: reset<RETURN>
1) attname relop value:
```

and re-typing the query.

Queries from Dump Mode

Retrieves initiated from dump mode can be limited to a subset of the relation with the q command.

```
<RETURN> to continue: q<RETURN>
Start on this page? y<RETURN>
End on this page? n<RETURN>

<RETURN>to continue: q<RETURN>
Start on this page? y<RETURN>
End on this page? n<RETURN>
```

Starting and ending page numbers or tuple identifiers may be specified before the qualifications are given. If you give a page number, the scan starts at the first tuple on the page (or ends at the last tuple on the page). CKDB checks that the tids point to data tuples of the current relation before using them.

```
<RETURN>to continue: q<RETURN>
Start on this page? 0 244 53<RETURN>
End on this page? y<RETURN>
```

If the "q" was typed accidentally, just enter <RETURN> to return to dump mode.

```
<RETURN> to continue: q<RETURN>
Start on this page? <RETURN>
<RETURN> to continue:
```

/r LIST RELATIONS

Prints the names and relids of all the relations in a database. By default, CKDB uses the current database; another may be specified explicitly:

```
Name of database: /r workdb<RETURN>
relation  1   attribute  2    indices    3    protect
4query    5   crossref   6    transact   7    batch
8host_users   9              disk_usage 10
blockalloc    11             users      13
descriptions  15
Current database now is system
```

Initially "system" is the current database.

/s SCAN DISKS

This function reads each zone of every accessible disk, looking for fragments which are allocated to non-existent databases, as well as fragments which are disjoint from their databases.

```
Name of database: /s<RETURN>
Ctlr 0 drive 2 disk160a 45091-118350 814(823)M cyls
0 heads 9 sectors
407 zones
Zone 0
Zone 10
.
.
Zone 400
Ctlr 0 drive 0 disk160a 45091-118350 814(823)M cyls 10
heads 9 sectors
Zone 0
.<HR>
.
Zone 400
Name of database:
```

CAUTION

Scan disks uses disk_usage to determine whether a fragment is connected. If it is suspect, CKDB "/a" should be run on all the databases before using "/s" to look for missing regions.

Scanning a disk also checks all the bad block remapping information. The physical sector header of a bad block contains the alternate cylinder, head, and sector of the block that is used to store the data of the page. There are many consistency checks done to verify that nothing is wrong with the remapping information.

To scan only a single disk enter the drive name:

```
Name of database: /s disk160b<RETURN>
Ctlr 0 drive 2 disk160b 45091-118350 814(823) cyls 10 heads
9 sectors
407 zones
Zone 0
Zone 10
   .
   .
   Zone 400
   Name of database:
```

/t TAPE MODE

You can examine IDM tapes with CKDB's tape dump mode /t. It is similar to page dump mode except that it reads a tape drive instead of a disk. Its commands may be viewed with a "?".

```
Name of database: /t<RETURN>
Tape mode: ?<RETURN>
b move backwards a block
d display current block
f forward to a file mark
h search on page header
l list tape buffers
n go next block
r rewind tape
s seek to a block #
Tape mode:
```

By default, it uses tape transport #0. You may select an alternate tape drive when entering tape mode:

```
Name of database: /t 4<RETURN>
Tape mode:
```

Tape Cache

Tape mode keeps the last 200 buffers in memory because the Tape Controller does not support backspacing a single block. The cache makes some things easy. For instance, to look at the last few blocks of a file, read until the next file mark. The blocks will be in memory and you may look around in it at random. Whenever you request a page that might be either in the cache or on the tape, CKDB asks:

```
Look in cache?
```

If is does not find what it is looking for there, then it will ask:

```
Read tape?
```

before it tries to read the next block from the tape.

b MOVE BACKWARDS A BLOCK

This command does not backspace the tape; it looks in the cache for the block preceding the current one. If the page is not in the cache, CKDB will print this and stay at the current block.

```
Tape command: b<RETURN>
Cannot read backwards.
Tape command:
```

d DISPLAY CURRENT BLOCK

Tape mode always has a "current block" which is the last block that it read or looked at. Display enters a version of "page dump mode" which is very similar to that of the "/d" command. You may print out the page as either tuples, octal bytes, or characters. Only system relations may be printed as tuples because CKDB cannot tell what the attributes of any users relations on the tape might be.

A few of the regular dump mode commands are not available in tape mode; specifically, the ones which would read pages from the disk (l, f, and p).

f READ FORWARD TO A FILE MARK

Read forward does not use the "search to a file mark" command of the tape controller; it reads each block. This lets it put the blocks before the file mark into the tape cache so that you may look at them later.

h SEARCH FOR A CERTAIN PAGE HEADER

You may search the cache and/or the tape for a particular page with the h command. It can find a block which has a certain pageid, nextpg or relid.

l LIST BLOCKS IN TAPE CACHE

Prints out the blocks in the tape cache with their block offset from the start of the tape and, if the verbose option is set, the pageid's in the headers of each page.

n GO TO NEXT BLOCK

This is equivalent to "<RETURN> to continue: .+1".

r REWIND TO TAPE

Rewinds the tape but does not wait for it to complete. A second rewind command will wait for the first one to finish and then report:

```
tape caution error #2: at load point
```

s SEEK TO A CERTAIN BLOCK

Seek looks for a certain block number either in the cache or on the tape drive, or both. The number is a 2K block offset from the start of the tape.

/x EXIT FROM CKDB

This command allows an BL700 to be rebooted from inside CKDB; instead of having to be near the machine. Before the reboot is attempted, CKDB will ask for confirmation:

```
Really reboot ?  y<RETURN>
```

The message

```
turn front panel switch to off please .....
```

will appear on early production database processors in which this self-reset is not possible. In this case, it will be necessary to turn off the BL700 with the front panel keyswitch.

ERRORS

There are many errors that CKDB detects. Some of them are so minor that CKDB can just note the inconsistency and continue. Others are so severe that CKDB cannot reliably determine what action to take next. On these, it will stop the examination of the current object (database, relation, or index) and continue to the next one. All diagnostic messages are in one of the following formats:

```
* * *    Error in <relation name> <relid>
* * *    Harmless Error in <relation name> <relid>
* * *    Correctable Error in  <relation name> <relid>
* * *    Severe Error in <relation name> <relid>
* * *    Fatal Error in <relation name> <relid>
```

and are followed by a description of the error.

CKDB groups together strings of related errors. For instance, if a relation has been deleted from the relation relation but has not had its disk space freed, CKDB could issue several dozen messages complaining about it.

The first few errors will print and then note

```
* * *    26 more errors
```

1. Mild Errors

There are three mild types of errors that CKDB attempts to deal with: Plain, Harmless, and Correctable errors.

- A Plain error ("*** Error") is one which can easily be corrected by the user without any loss of data. This type of error is reported when an index on a relation has been found to be inconsistent. Simply recreating the index will usually correct the problem.
- Harmless errors are less severe than plain errors but cannot always be easily corrected by user intervention.
- A Correctable error is one that CKDB knows is safe to fix. If the X "fix" option is enabled, CKDB will correct it; if not it will print "Not corrected" and continue checking the relation.

2. Severe Errors

Severe errors usually cannot be corrected as easily as mild errors. They are not always correctable without losing any data. If the error is in a system relation, it may be necessary to load the database from the last dump.

3. Fatal Errors

A Fatal error means CKDB required some information that it could not obtain. The rest of the current relation or index will not be examined. It is possible for other errors to cause a milder error to be upgraded to a more severe one.

4. Summarizing Errors

CKDB remembers which relations and indices have errors so that problems may be summarized. Just as CKDB uses the unused "s1" field in blockalloc to verify disk allocation, some of the spare attributes in the relation and indices relations are used to record when and where errors are seen.

4.1. Remembering Errors of a Relation

Errors are recorded in the previously unused attribute "s1" in the relation. Each bit may be set independently of the others.

Bit Value	Meaning
001	fatal error in this relation
002	relation has a bad index
010	CKDB completed its examination of this relation
020	some type of error
040	severe error in this relation

4.2. Remembering the Errors in an Index

A bad index is marked by setting bit 3 (octal value 010) bit in the attribute of the tuple.

EXAMINATION SEQUENCE

Under normal circumstances, CKDB does its exhaustive verification process on the whole database. By setting the -r, -b, or -i options, this basic action can be adjusted to suit your needs. CKDB still goes through its basic examination sequence when one of these options is enabled, allowing for the obvious effect of the options.

1. Open a Database

To run CKDB, open the database you want to check. This is always done whether or not any of the above options are set.

1.1. Look For Incompleted Updates

Scan the transaction and back logs to determine whether there may be incomplete transactions in the database. Partially executed transactions can leave the database in an inconsistent state and cause CKDB to report spurious errors. If it is not in a "safe" condition, CKDB responds:

```
Recovery should be run on <database>
Go ahead ahead anyway?
```

Enter y to check the database, or n to return to the top-level prompt.

1.2. Check Blockalloc

CKDB depends heavily on blockalloc; it stores status information about each page in the "s1" attribute. To avoid generating spurious errors, it does a quick check of blockalloc. Examine the system catalogs blockalloc and disk_usage. CKDB uses these relations to detect inconsistently allocated pages. It must be able to trust them (at least partially) before it can proceed with the verification.

2. Check a Relation

For each relation check the data pages of the relation for malformed tuples. Also, make sure that the pages are correctly allocated to the relation.

For each index:

1. Print the type and keys of the index.
2. Read the root page of the index, then read the level immediately underneath it; continue until CKDB gets to the level of the data pages. This is a horizontal scan of each level of the index following the sibling pointers of the tree.
3. Search the index recursively. Starting from the root, check the current page, then the lesser children, then the greater children. This section verifies that the tree structure of the index is correct and determines whether the information in the index agrees with the data of the relation.
4. Scan the index again, looking for pages which were found in the horizontal search of each level, but were not seen in the vertical recursive pass. If any errors were found, record the type of problem in the "s1" attribute of this object's relation relation tuple.

3. End of Database Checks

3.1. Find Uselessly Allocated Pages

Look through blockalloc for incorrectly allocated disk blocks. Common (but harmless) errors are: page allocated to <relation> but used page allocated to non-existent relation

3.2. Find Useless Tuples in System Relations

Verify that the system catalogs are consistent with each other. Here, CKDB looks for attributes that refer to non-existent relations and other similar errors.

4. Summarize Errors

CKDB now prints the name of each object which has an error and the type of error. A complete listing of the possible errors, and a brief explanation of each error, is given in Appendix G.

REMOVAL AND REPLACEMENT

Removal and Replacement Procedures

REMOVAL AND REPLACEMENT PROCEDURE

INTRODUCTION

The purpose of this section is to guide you through the proper steps to remove and replace individual parts of the BL700. Remove and replace includes only those parts or modules that can be manipulated with your hands or with the aid of common hand tools. (See figures 6-1 & 6-2).

The structure of this section is presented in a sequential order. In order to move to the next step you must have successfully completed the previous step.

DISCONNECT AC POWER FROM SOURCE

OPENING THE BL700 (INSTALLED IN CABINET)

1. Pull the safety leg out and extend its full length. The leg is under the front of the cabinet on the center line.
2. Insert T-bar or small screwdriver into small hole near lower right corner of the Front Bezel. Pry to the left and pull the BL700 straight out until the chassis latches on the slide rails.

FRONT BEZEL ASSEMBLY

1. Open the BL700.
2. Remove the six 10-32 x 1/4" screws from the back of the bezel (three on each side).
3. Lay the bezel on a piece of foam or cloth to avoid scratches.

CONTROL PANEL:

1. Open the BL700.
2. Remove the Front Bezel.

   Disconnect the following cables from the Control Panel:

   control panel J4 (from DBP J4)
   control panel J5 (from Motherboard J5 or ribbon connector on upper power supply)
   control panel J9 (Molex connector from power harness)

3. Remove the four screws holding the Control Panel assembly mounting brackets into the chassis. The Control Panel assembly, with brackets attached, may now be removed from the chassis for repair or replacement.
4. Reassemble by reversing steps 1 through 3. Use caution to avoid pinching or damaging the cables to the Control Panel. (PCB)
5. After re-installing the Front Panel Assembly in the chassis, verify that the PC-mounted "FAULT OVERRIDE" toggle switch is in the OFF or RUN position.

POWER SUPPLY (multi):

(This procedure applies to the FCC Britton Lee database server). Refer to figure 6-1.

1. Remove the top most power supply (multi) by loosening the four 6-32 x 1/4" screws attached to the internal bulkhead.
2. Carefully remove the power supply from the mounting screws and slip out for easy access to wires.
3. Wires should be marked with their corresponding slots before removal. Refer to Appendix E for a list of cable and wire connections.
4. Remove all cables and wires.

5. Remove the power supply and replace with a Britton Lee part.
6. Adjust DC voltages. Refer to the Diagnostic section, Power Supply Troubleshooting.

POWER SUPPLY (Bulk + 5V FCC):

1. Remove the bulk power supply (+5 vdc) by loosening the four 6–32 x 1/4" mounting screws that are located on the internal bulkhead.
2. Slide the power supply out for easy access to the connection.
3. Refer to Appendix E for a list of cable and wire connections.
4. Disconnect all cables and wires.
5. Remove the power supply and replace with a Britton Lee power supply.
6. Adjust DC voltages. Refer to the Diagnostic section, Power Supply Troubleshooting.

TOP REAR PANEL

1. Open the chassis to the locked position.
2. Remove four Phillips screws from the top rear panel.
3. Lay the two piece panel aside.

CARD CAGE AREA SIX INCH FANS

1. Open the top rear panel.
2. Disconnect the logic cage fan power and ground wires.
3. Remove the four 6–32 x 3/4" screws from the BL700 back panel.
4. Remove the logic cage fan.
5. Replace with a BLI fan. Connect power and groundwires.
6. Verify that the airflow is outward.
7. Replace screws.

DEFECTIVE CABLES (internal)

Internal connecting cables are defined as all cables connecting the PCB to the interface panel located on the back of the BL700.

1. Replace a bad cable by tracing the cable internally to the back of the BL700 interface panel.
2. For easy access to the male end of the cable, remove the corresponding connector plate. The connector plates are attached by two 4–40 1/2" screws from the designated male end ribbon connector. Replace with BLI cable.

T1 STEPDOWN TRANSFORMER

The stepdown transformer is mounted on the back of the rear panel.

1. Remove the top rear panel for access to the transformer.
2. Verify that all power is OFF, then continue by disconnecting the male molex connector from the internal bulkhead on the back of the card cage area.
3. Disconnect the ground wire (green/yellow) from the transformer.
4. Remove the two 8–32 x 1/2" large screws (screws are inserted through the back of the BL700) and the two kepsnuts.
5. Replace the BLI transformer with the same.

THERMAL SENSOR

The thermal sensor is located directly behind the PCB card cage (slot 11).

1. Remove by unplugging the two position molex connector. Avoid damaging the female connector.
2. Cut loose strain relief cable tie and remove the two 6-32 x 1/2" screws from the bottom of the thermostat. These screws go through two 1/4" spacers and are secured by two 6-32 kepsnuts. Use caution to avoid dropping screws, nuts and spacers on to the Motherboard.
3. Reassemble by reversing the procedure.

LINE FILTER AC

The line filter is on the back of the power area (see the *BL700 Installation Manual*).

1. Verify that all power is OFF and the cord is disconnected.
2. Replace the line filter by removing the two 6-32 x 1/4" large flathead screws and the two 6-32 kepsnuts from the bottom of the BL700
3. Disconnect crimped connectors from the line filter and replace.
4. When reassembling, orient filter so the three tabs on one end are facing the rear of the unit.

(AC) POWER SWITCH / CIRCUIT BREAKER

The ac power switch/circuit breaker is located on the rear panel of the BL700.

1. First verify that all ac power is OFF and that the power cord is disconnected.
2. Label all wires connected to the switch with their respective locations.
3. Disconnect the eight wires from the ac switch. Use caution to avoid loosening crimp connectors (do not pull by the wires).
4. The ac switch can now be removed by depressing the four plastic lock mechanical tabs located on the top and the bottom of the switch.
5. Reassemble by reversing the procedure.

(AC) POWERCORD AND PLUG

1. Disconnect the power cord from the power source.
2. Replace the power cord by disconnecting the three wires from the line filter on the back of the ac power area.
3. Disconnect the end wire from the chassis ground block (see the *BL700 Installation Manual*).
4. Replace the plug by removing three screws from the plug.
5. Pull back outer cover to expose crimping screws. Unscrew and remove wire.
6. Replace the plug with same type (NEMA # L6-15P).
7. Reverse procedure to reassemble.

REPLACEMENT OF PRINTED CIRCUIT BOARDS *(WEAR STATIC GROUND STRAP)*

1. Turn OFF all dc power to boards.

CAUTION

Failure to turn OFF the power may result in a power surge that will damage the board.

2. Remove six 6-32 x 1/4" screws from top cover of BL700 for access to all PCBs.
3. Slots are numbered left to right (0-15). Remove a defective board by lifting the card ejectors and pulling the board STRAIGHT up and out.
4. Insert the replacement board in the PCB slot (component side toward the power area) and secure by clamping down the ejectors.

SERIAL DISTRIBUTION BOX

The RS-232 serial distribution box is located on the back of the interface panel.

1. Replace the distribution box by removing the four 6-32 x 1/4" large screws holding the assembly. Lift and pull carefully (be careful not to damage the ribbon cable).
2. Disconnect the 50 pin ribbon cable from the box and either replace it with the same type or upgrade it to a distribution rack.

REPLACING SERIAL CABLE

1. Remove the top cover from the unit.
2. Locate the serial channel board.
3. Disconnect the 50 pin ribbon cable and follow its internal path to the interface panel.
4. Disconnect the cable from the distribution box and replace.

REPLACING MOTHERBOARD

NOTE

Some BL700s contain insulating washers between the Motherboard and stand offs.

1. Disconnect ac power and remove the BL700 from the rack.
2. Remove the bezel assembly (refer to bezel removal).
3. Remove the access covers (see the *BL700 Installation Manual*).
4. Unplug the PCB's from the Motherboard (it is not necessary to disconnect the ribbon cables).
5. Disconnect the Front Panel ribbon cables and the molex connector.
6. Remove the bottom power supply (PS1, 5V (MM30)) from internal bulkhead (refer to removing power supply for instructions).
7. Before disconnecting the power supply wires, label all wires with their corresponding locations. Disconnect the power supply.
8. Mark all wires on the Motherboard before disconnecting. Disconnect the dc harness from the Motherboard.
9. Remove the three screws from the lower right side panel and the three screws from the lower left side panel. Remove the two nuts from the internal bulkhead (attaching the internal bulkhead to the Motherboard) and the two screws attaching the internal bulkhead to the back panel.
10. Lift the front of the BL700 approximately 15-20 degrees for access to twelve screws holding the Motherboard in place. Remove the screws, then, carefully remove the Motherboard for repair or replacement.
11. Reassemble by reversing the above procedure.

SERIAL DISTRIBUTION RACK

The serial distribution rack is mounted in the upper section of the cabinet.

1. Replace by moving four 10-32 x 1" screws that hold the rack into place.

CAUTION

Avoid damaging the ribbon cable.

2. Disconnect the ribbon cable from the back of the rack (if you are only replacing the ribbon cable, do the above (#1) and disconnect the cable from the serial I/O interface plate located on the back of the BL700).
3. Reverse the above procedure to reinstall.

MEMORY CHIPS

The console message:

```
"correctable memory error slot < n >, pg < z >, ic < xy >"
```

specifies the memory board's slot number and the location, by board matrix, of the defective Integrated Circuit (chip). If a memory chip is to be replaced, turn the power OFF and remove the memory storage board in slot "n" using the procedure in the previous sub-section.

After the board has been removed, replace the defective chip "xy", observing the following:

1. *DO NOT ATTEMPT CHIP REPLACEMENT* unless you are trained or qualified to perform this task.
2. The console message specifies the defective chip by its letter-number location on the memory board.

The board's memory chip sockets occupy rows labeled "D, E, F, H, J & K" and are grouped into two arrays with columns labeled "1-12" and "21-32". The earlier 0.5MB version of the board has only alternate rows populated with Integrated Circuit's.

CAUTION

Use proper handling procedures to avoid static damage to the memory Integrated Circuits.

3. Using an Integrated Circuit removal tool, carefully remove the defective memory chip.
4. Insert the new memory chip. Be sure pin 1 is properly oriented and all pins are fully inserted into the socket.

EXPLODED VIEW OF BL700 CHASSIS (FRONT)

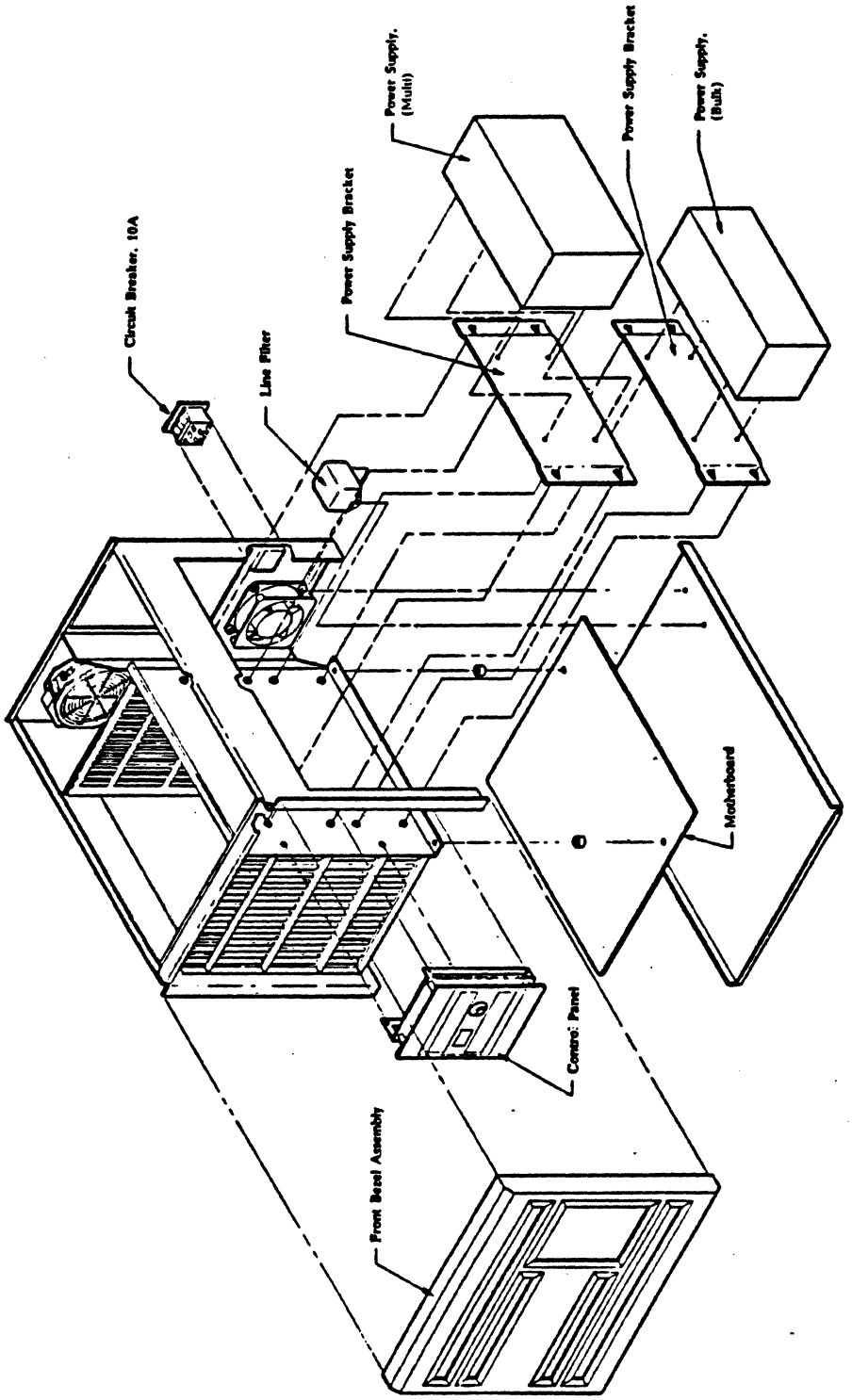
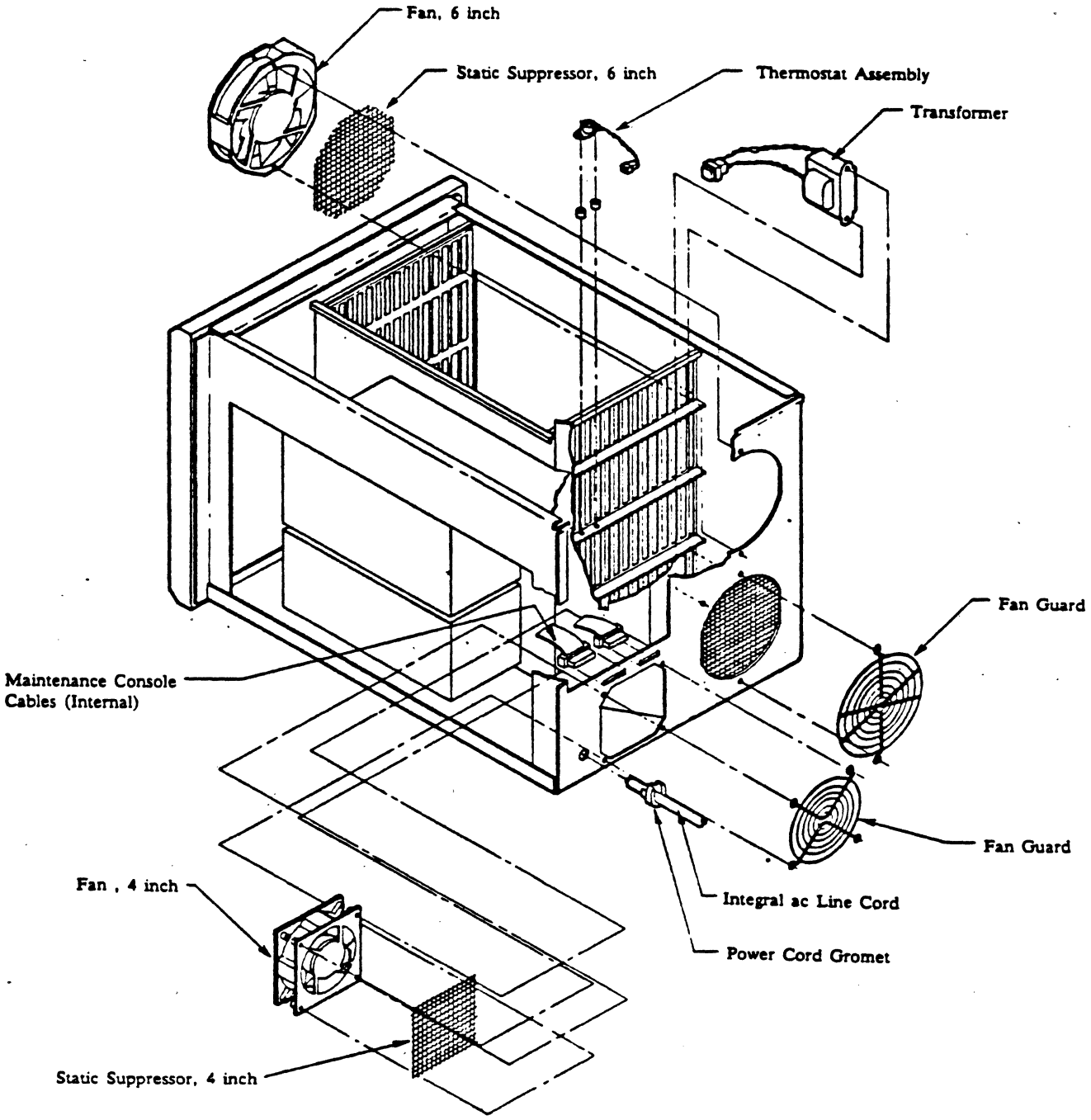


Figure 6-1: BL700 Chassis (Front)

Figure 6-2: BL700 Chassis (Rear)



APPENDIX A: CONSOLE COMMANDS

conport <baud> or maintport <baud>

list

listtape

load

load <filename>

loadtape

loadtape <filename>

loopback <slot number>

kernal

kernal ckdb or dfu

kernal fileload

kernal fileload -t

kernal <filename>

<return>

slots

td/dd

version

CONSOLE COMMANDS

INTRODUCTION

The commands listed below are entered at the console port or maintenance port after the prompt

```
BL700 - Filename
```

when the BL700 is powered up and in the MAINT mode. With early production systems, if the system is in the RUN or SAFE mode it must be turned OFF and then to MAINT before it will accept console commands.

The *load, list,* and *fileload* commands require a load device connected to the maintenance port. This device is typically a DSC-3 or IBM PC software loader, but may also be a host equipped for download.

The *loadtape, listtape,* and *fileload -t* commands require a BL700 tape controller and drive.

NOTE

This appendix applies to Database Processors (DBP) with firmware revision 27 or a later revision. The appropriate load device software is also required.

After typing a command, press the <return> key to enter it. The expression <filename> represents a file name.

The use of the maintenance and console ports is interchanged when performing remote diagnostics with a modem.

CONSOLE COMMANDS:

conport <baud> or maintport <baud>

These commands temporarily set the baud rate of the BL700's console port or maintenance port to the specified rate. The default rate (300 baud) is assumed when the BL700 is powered down or reset. The BL700 resets itself at the completion of fileload and most other programs.

Valid baud rates are:

300
600
1200
2400
3600
9600
19,200

If an invalid rate is specified, the console displays:

```
unknown baud rate <baud>
```

NOTE

Be sure the terminal baud rate matches that of the port to which it is attached. Since booting the system code sets the baud rate at 300, all console messages will be lost unless the console terminal is also set at 300 baud.

These commands cause a delay of about four seconds before returning the BL700 prompt, to allow for the corresponding baud rate change on the attached device.

list

This lists the files in the load device on the console. The console displays the messages:

```
Contents of load device:
   File 1  :  directory revision <rev> size <z>
   File 2  :  <filename> revision <rev> size <z>
   ...
   File <n>:  <filename> revision <rev> size <z>
```

listtape

This lists the contents of IDM tape at the console. The console displays the messages:

```
Contents of load device:
   File 1  :  directory revision <rev> size <z>
   File 2  :  <filename> revision <rev> size <z>
   ...
   File <n>:  <filename> revision <rev> size <z>
```

load

This command loads the first file (after the directory) from the associated storage medium into the BL700 main memory for execution by the DBP. The file can be loaded from either the DSC-3 or IBM PC load devices or host tape. The file will not be written onto the system disk. In this case, the baud rate of the appropriate port must first be established, using the *conport 9600* or *maintport 9600* command. The console displays the message:

```
Loading <filename>
```

This command is also used for loading system software, in which case the first file on the medium is *fileload*. The execution of the *fileload* file transfers the system software files to the system disk. See the messages listed under the *fileload* command for more information.

load <filename>

This command loads *<filename>* from the load device diskette or host tape into the DBP for execution. In this case, the baud rate of the appropriate port must first be established, using the *conport 9600* or *maintport 9600* command.

If *<filename>* is not found on the load device the console displays:

```
File <filename>: not on load device
```

loadtape

This command loads the first file (after the directory) from IDM tape into main memory for execution. The file is not written onto the system disk. The console displays the message:

```
Loading <filename>
```

loadtape is also used for loading system software, in which case the first file on the medium is *fileload*. The execution of the *fileload* file transfers the system software files to the system disk. See the messages listed under the *fileload* command for more information.

loadtape <filename>

This command loads *<filename>* from the IDM tape into main memory for execution. *<filename>* is not written onto the system disk. If *<filename>* is not found, the console displays the message:

```
File <filename>:  not on load device
```

loopback <slot number>

This command starts a loopback test on the board with the given slot number. A loopback connector must be installed for that board. This command is only appropriate for the RS-232 Host Interface (revision 37 or greater) and the tape controller. The console may display any of the following messages:

```
no slot specified
bad slot specified
Port <m> failed - code <n>
board not responding
BL700 - Filename:
```

The absence of any error message indicates successful completion of the test. The RS-232 Loopback connections are shown in Appendix E, Cable and Connector Details. If desired, the connections may be applied to only one port at time, in which case only one of the 8 ports will pass.

Loopback test fixtures may be obtained from Britton Lee for the BL700 Tape Interface (fixture part number 118-0540) and/or the RS-232 Host Interface (fixture part number 118-0541).

For more information on the fault messages see Appendix G, Error Messages and Codes.

kernal

This command boots up the BL700 as if the key had been turned to the RUN position.

CAUTION
Recovery will not be run if the key is turned from MAINT to OFF.

kernal ckdb or dfu

This command loads the file CKDB or DFU from system database into the BL700 memory in preparation for execution.

kernal fileload

This command loads specified files from the IBM PC or host RS-232 through the maintenance port into the system database.

kernal fileload -t

This command loads specified files from the BL700 tape controller into system database.

kernal <filename>

This command loads files that run under the kernal.

Example: kernal dfu
kernal ckdb

<return>

By pressing the carriage return key, the console lists the names and connections of all disks in the system. It also lists all the files on the system disk. The list may vary depending on the files included in each software release.

system disk:Ctlr x cable y drive name <name>
fileload
kernal
front
qryproc
support
sort
dumpload
compile
recover
wcs
listen

If there is no system disk, the console displays:

```
No system disk available
```

slots <n>

This command causes the BL700 to perform a self-test of all circuit boards n times. The test is done once if n is not specified. If n is minus 1 (-1), the test is repeated indefinitely. To terminate slots,

turn the BL700 off, then on again. The console displays a list that can include the following:

```
Slot <m>:  serial chan rev <n>
Slot <m>:  parallel chan rev <n>
Slot <m>:  t&c rev <n>
Slot <m>:  dc rev <n>
Slot <m>:  tpc rev <n>
Slot <m>:  dbp rev <n>
Slot <m>:  da rev <n>
Slot <m>:  256K Mem
Slot <m>:  512K Mem
Slot <m>:  1meg Mem
```

NOTE

In this self-test, a parallel channel revision level above 100 indicates a non-controller parallel channel.

New possibilities with DBP firmware revision 29 or above are as follows:

```
Slot <m>:  non-controller parallel chan rev <n>
Slot <m>:  ethernet chan rev <n>
Slot <m>:  block mux chan rev <n>
```

NOTE

The appearance of this list depends on the order in which the boards are inserted into the backplane. The list starts with the board in the lowest numbered slot. Any of the messages above can be followed by a fault message:

```
.....Fault code <x>
```

A fault code of 1 indicates that the board previously filed, but is now functional.

If the fault code is other than 1, the board is inoperable. The additional message appears:

```
***DEFECTIVE BOARD***
```

and the SERVICE indicator on the BL700's control panel comes on.

If the board being tested does not respond to the slots command, the following message appears:

```
board not responding
```

td/dd

This command loads file td/dd from system database into BL700 memory in preparation for execution. For a full description of the fault messages refer to Appendix E.

version

This command produces DBP firmware revision level.

APPENDIX B: PERIPHERAL DEVICE STATUS & REPORTS

Peripheral Device Status and Reports

APPENDIX B: PERIPHERAL DEVICE STATUS & REPORTS

INTRODUCTION

Some BLI supported peripheral devices have a type of fault indicator or some extended self diagnostics. The following is a list of status and diagnostics tools available for each device. Please consult the manufacturers manual for a detailed description of troubleshooting aids.

Manufacturer	Device	Model	Fault Indicators	Self Diagnostics
CDC	Disk	9775	Yes	Yes
CDC	Disk	9766	Yes	No
CDC	Disk	9762	Yes	No
CDC	Disk	9730-160	Yes	No
CDC	Disk	9715-160	Yes	No
CDC	Disk	9715-340	Yes	Yes
CDC	Disk	9710-80	Yes	No
CDS	Disk	306	Yes	No
CDS	Disk	380	Yes	No
CDS	Disk	513	Yes	No
CDS	Disk	315	Yes	No
Fujitsu	Disk	m2282	Yes	No
Fujitsu	Disk	m2283	Yes	No
Fujitsu	Disk	m2284	Yes	No
CDC	Tape	92181	Yes	Yes
CDC	Tape	92144	No	Yes
Cypher	Tape	F880	Yes	No
Megatape	Tape	MT500	No	Yes

APPENDIX C: SOFTWARE LOADING PROCEDURE

Software Loading Procedure

SOFTWARE LOADING PROCEDURE

INTRODUCTION

Reading a File from the System Disk

Locating a file on the system disk requires locating all controllers. Then, every disk on every controller is examined until the system disk is located.

DBP:

- Locates all controllers and assigns them a unique bus priority.
- Checks for the presence of drive numbers 0 through 15 for each controller.
- If a drive is present, reads the master block.
- If the master block is readable, reports its name and checks to see if it is a system disk.

If no system disk is found, a message is issued and control starts over at the "BL700 Filename" message. If the system disk is found, the system database is searched for the presence of the requested file. If the name of the requested file is "" (null), then a list of all files is printed. If the file is found and has the proper header, it is loaded into memory.

Load Command

The *load* command causes a program to be read from the dcs-3, IBM PC, IDM tape, or host tape. *load* with no other parameters causes the first file on the dcs-3, IBM PC, IDM tape, or host tape to be read. *load filename* causes *"filename"* to be read from the dcs-3, IBM PC, IDM tape, or host tape. An error message is issued if the filename, as typed, does not exist.

Loading procedures for the dsc-3, IBM PC, host tape, and IDM tape are explained below. The same information can be found in the *BL700 Installation Manual*.

LOADING FROM THE DSC-3

Introduction

The dsc-3 can be used to load BL700 Utilities and System Software into the BL700. For loading of system software, both the System Boot Diskette and the Maintenance Diskette are required. For loading of utilities, only the Maintenance Diskette (or other supplied diskette) is required.

The connections apply to the loading of either utilities or system software; however, the loading procedure given in this sub-section applies to the loading of system software only. A detailed description of the loading and use of the SysFormat and Format modes of DFU is given in the *BL700 Operation Manual*.

Connections

The dsc-3 must be connected to the BL700's maintenance port as shown in Figure C-1. Connect the BL700's maintenance port to the dsc-3's second (from the top) RS-232C port with the maintenance cable. (The full-modem maintenance cable is supplied with the dsc-3.) This port on the dsc-3 is internally jumpered as "DCE" (as defined in RS-232C specifications). Thus, the "null-modem" or "flip" type cable is not required.

SYSTEM SOFTWARE LOADING PROCEDURE (dsc-3)

1. Power up the dsc-3 and its disk drive(s).
2. Power up the BL700. Turn the function switch from OFF to SAFE. Wait for the SAFE indicator to light, then turn the function switch to MAINT. (If this is the initial installation of the BL700, the switch can be turned directly from OFF to MAINT.)
3. Insert the System Boot Diskette into the dsc-3's diskette drive, label up. Press the drive door down to lock it.

4. When the BL700's self-test is complete and the prompt

```
BL700 - Filename:
```

appears on the console, enter

```
load
```

at the console.

The kernal program will be loaded into memory from the system disk. The console will display:

```
loading kernal
```

The kernal program will then print:

```
kernal n: <day> <date> <time>
Accessible Disks:
    Name    Ctlr    Drive    Low    High    Status
`disk 1 ´   x       y        m      n       <status 1>
    .
    .
`disk n ´   x       y        m      n       <status n>
loading syscalls
```

The FILELOAD program is loaded into memory from the diskette and the console displays:

```
loading fileload
number of available dbins: x
Free process pages: y
```

5. When loaded, FILELOAD lists the programs on the diskette:

```
Fileload version n <date>
KERNAL    revision <m> size <n>
syscalls  revision <m> size <n>
fileload  revision <m> size <n>
    ...
    ...
```

FILELOAD then starts loading the system software onto the system disk. A message such as the following one appears at the console for each file copied to disk. This continues until all files are copied to the system disk.

```
fileload <filename>, size n, file y
creating idm file <filename>
file assigned relid z
  <filename> loaded
```

If a file is already present on the system disk, (e.g. when updating software) the following message appears for that file:

```
fileload <filename>, size n, file y
   <filename> loaded
```

6. When FILELOAD is finished, the BL700 resets itself, performs a self-test, and displays the prompt:

```
BL700 - Filename:
```

7. Be sure the red light in the dsc-3's diskette drive is OFF before removing the System Boot Diskette.

8. Insert the *Maintenance Diskette* and then enter:

```
kernal fileload
```

9. FILELOAD is loaded into memory from the system disk and lists the files on the diskette:

```
KERNAL n: <day> <date> <time>
Accessible Disks:
Name        Ctlr    Drive    Low     High     Status
`disk 1 ´   x       y        m       n        <status 1>
            .
            .
`disk n ´   x       y        m       n        <status n>
loading syscalls
loading fileload
number of available dbins: x
Free process pages: y
Fileload version n <date>
sort        revision <m> size <n>
dumpload    revision <m> size <n>
compile     revision <m> size <n>
execute     revision <m> size <n>
wcs         revision <m> size <n>
listen      revision <m> size <n>
ckdb        revision <m> size <n>
ckdb2       revision <m> size <n>
    ...
    ...
```

Fileload then copies the remainder of the system software onto the system disk. A message appears at the console for each file copied to disk. This continues until all the remaining system software is copied to the system disk.

10. When load is complete, the BL700 resets itself, performs a self-test, and displays the prompt:

```
BL700 - Filename:
```

The BL700's clock must be reset via the host after power-up.

LOADING FROM THE IBM PC

Introduction

The IBM PC can be used to load utilities and system software into the BL700. For loading of system software, the PC Idmboot Diskette and BL700 Software Diskettes A, B, and C are required.

The connections apply to the loading of either utilities or system software; however, the Loading Procedure given in this sub-section applies to the loading of system software only. A detailed description of the loading and use of the SysFormat and Format modes of DFU is given in the *BL700 Operation Manual*.

Connections

Use the PC/BL700 Interface Cable (part no. 113-1016) or equivalent null-modem cable to connect the IBM PC RS-232 asynchronous port, "com1:", to the BL700's maintenance port, shown in Figure C-2. Connect the cable to the BL700's console port if the BL700 has revision 29 or a later revision DBP firmware.

SYSTEM SOFTWARE LOADING PROCEDURE (IBM PC)

1. Boot your IBM PC system, following the normal procedures described in the DOS manual, using any DOS diskette.
2. When the boot procedure is complete, insert the PC Idmboot Diskette (5.25 inch), label up, into drive "A:".
3. Enter the word "idmboot", and press the carriage return (a key on the right, with an arrow pointing down and to the left). Enter "idmboot -2" if the BL700 has revision 28 (or earlier) DBP firmware. There will be a whirring noise from the IBM PC's diskette drive, and the

following message appears:

```
Britton-Lee IDMBOOT Utility - loads the IDM/RDBMS
software from the IBM PC diskette

This program can only be terminated by a user
interrupt

Reading drive A:, writing to BL700.
```

NOTE

If you have an IBM PC with two diskette drives, you may load software from one drive while running the IDMBOOT program from another. The IDMBOOT program always reads from the default drive "A:" unless you specify a different drive. Enter "idmboot B:" if the PC IDMBOOT Diskette is in drive "A:" and a BL700 Software Diskette is in drive "B:" while booting.

4. Power up the BL700. Turn the function switch from OFF to SAFE. Wait for the SAFE indicator to light, then turn the function switch to MAINT. (If this is the initial installation of the BL700, the switch can be turned directly from OFF to MAINT.)

5. After the idmboot prompt appears on the IBM PC, insert the BL700 Software Diskette *A* label up into the IBM PC drive. Press the drive door down to close and lock it.

NOTE

The *PC* BL700 Software Diskettes must be inserted alphabetically: *A, B,* then *C*. Loading them out of sequence will cause idmboot to fail.

6. When the BL700's self-test is complete and the prompt

```
BL700 - Filename:
```

appears on the BL700 console, enter

```
load
```

at the BL700 console. This causes file the kernal to be loaded from the diskette into the BL700 and executed. The BL700 console displays:

```
loading kernal
```

7. When loaded, the kernal displays:

```
KERNAL n: <day> <date> <time>
Accessible Disks:
   Name    Ctlr    Drive    Low    High    Status
`disk 1 ´   x       y        m      n       <status 1>
    .
    .
`disk n ´   x       y        m      n       <status n>
loading syscalls
loading fileload
number of available dbins: x
Free process pages: y
```

The kernal will then load FILELOAD and execute it. FILELOAD will display:

```
Fileload version n <date>
kernal    revision <m> size <n>

syscalls  revision <m> size <n>
fileload  revision <m> size <n>
   ...
   ...
```

FILELOAD then starts loading the system software onto the system disk. A message such as the following one appears at the BL700 console for each file copied to disk. This continues until all files are copied from diskette to the system disk.

```
fileload <filename>, size n, file y
creating idm file <filename>
file assigned relid n
   <filename> loaded
```

If a file is already present on the system disk, (e.g. when updating software) the following message appears for that file:

```
fileload <filename>, size n, file y
   <filename> loaded
```

1. When FILELOAD is finished with a BL700 Software Diskette, the BL700 resets itself, performs a self-test and displays the following prompt at the BL700 console.

   ```
   BL700 - Filename:
   ```

2. Remove *PC BL700 Software Diskette A* and insert *PC BL700 Software Diskette B*.
3. Type

   ```
   kernal fileload
   ```

   at the BL700 Console Terminal.

4. FILELOAD is loaded into memory from the system disk and lists the files on the diskette:

   ```
   KERNAL n: <day> <date> <time>
   Accessible Disks:
       Name    Ctlr    Drive    Low    High    Status
   `disk 1 ´   x       y        m      n       <status 1>
   .           .
   .           .
   `disk n ´   x       y        m      n       <status n>
   loading syscalls
   loading fileload
   number of available dbins: x
   Free process pages: y
   Fileload version n <date>
   front       revision <m> size <n>
   qryproc     revision <m> size <n>
   support     revision <m> size <n>
   sort        revision <m> size <n>
    ...
    ...
   ```

   FILELOAD then copies these system software files onto the system disk. A message appears at the console for each file copied to disk.

5. When load is complete, the BL700 resets itself, performs a self-test, and displays the following prompt at its console:

   ```
   BL700 - Filename:
   ```

6. Remove *PC BL700 Software Diskette B* and insert *PC BL700 Software Diskette C*.
7. Enter

   ```
   kernal fileload
   ```

   at the BL700 console terminal.

8. FILELOAD is loaded into memory from the system disk and lists the files on the diskette:

```
KERNAL n: <day> <date> <time>
Accessible Disks:
Name        Ctlr    Drive    Low    High    Status
`disk 1 ´   x       y        m      n       <status 1>
    .
    .
`disk n ´   x       y        m      n       <status n>
loading syscalls
loading fileload
number of available dbins: x
Free process pages: y
Fileload version n <date>
sort        revision <m> size <n>
dumpload    revision <m> size <n>
compile     revision <m> size <n>
execute     revision <m> size <n>
listen      revision <m> size <n>
    ...
    ...
```

FILELOAD then copies these system software files onto the system disk. A message appears at the console for each file copied to disk.

9. When load is complete, the BL700 resets itself, performs a self-test, and displays the following prompt at its console:

```
BL700 - Filename:
```

10. Remove *PC BL700 Software Diskette C*. Essential software is now loaded. CKDB and DFU are also loaded if space exists. Td, dd, and memtest will be loaded upon request.

CAUTION

Files, once loaded, cannot be subsequently removed from the system disk.

LOADING FROM IDM TAPE

Introduction

The IDM Tape Drive can be used to load BL700 Utilities and System Software into the BL700. For any such loading procedures, the IDMBOOT program must be present in the host, and the System Boot and Maintenance Tape is required on the IDM Tape Drive.

Most of the loading procedure given in this sub-section applies to the loading of system software only.

SYSTEM SOFTWARE LOADING PROCEDURE (IDM Tape)

1. The BL700 Tape Drive must be connected to the tape interface panel on the rear of the BL700.
2. Mount the System Boot and Maintenance Tape on tape drive 0.
3. Power up tape drive 0 and the disk drive(s). Power up the BL700. Turn the function switch from OFF to SAFE. Wait for the SAFE indicator to light, then turn the function switch to MAINT. (If this is the initial installation of the BL700, the switch can be turned directly from OFF to MAINT.)
4. Set the tape drive for 1600 bpi, and place it on-line.
5. Set the tape to the load point. The above steps apply equally to the loading of system software or utilities. The following steps apply only to the loading of system software.
6. When the BL700's self-test is complete and the prompt

```
BL700 - Filename:
```

appears on the console, enter

```
loadtape
```

The kernal is loaded into memory from the tape and the console displays:

```
loading kernal
```

7. When loaded, the kernal displays:

```
KERNAL n: <day> <date> <time>
Accessible Disks:
Name        Ctlr    Drive    Low    High    Status
`disk 1 ´   x       y        m      n       <status 1>
    .
    .
`disk n ´   x       y        m      n       <status n>
loading syscalls
loading fileload
```

The kernal then begins to load FILELOAD from the tape.

8. When loaded, FILELOAD lists the files on the magnetic tape:

```
number of available dbins: x
Free process pages: y
Fileload version n <date>
kernal    revision <m> size <n>
syscalls  revision <m> size <n>
fileload  revision <m> size <n>
dfu       revision <m> size <n>
dfu2      revision <m> size <n>
front     revision <m> size <n>
qryproc   revision <m> size <n>
support   revision <m> size <n>
sort      revision <m> size <n>
   ...
   ...
   ...
```

NOTE

Certain files are not part of the system software and are not copied to disk.

FILELOAD then starts loading the system software onto the system disk. A message appears at the console for each file copied to disk. This continues until all system software files are copied from the tape to the system disk.

```
fileload <filename>, size n, file y
creating idm file <filename>
file assigned relid z<filename> loaded
```

If the file is already present on the system disk, as in the case of a software update, the following message appears for that file:

```
fileload <filename>, size n, file y
   <filename> loaded
```

9. When the load completes, the BL700 resets itself, performs a self-test, and displays the prompt:

```
BL700 - Filename:
```

LOADING FROM HOST TAPE

Introduction

The host tape can be used to load utilities and system software into the BL700. For this, the IDMBOOT program must be present in the host, and the System Boot and Maintenance Tape must be on the host tape drive.

Figure C-3 depicts the loading of either utilities or system software; however, most of the loading procedure given in this sub-section applies to the loading of system software only.

Connections

The BL700's maintenance port must be connected to the host system as shown in Figure C-3.

The link must be a 9600 baud 8-bit raw data path with neither echo, parity checking, character transformation, or similar tty processing.

SYSTEM SOFTWARE LOADING PROCEDURE (Host Tape)

1. Power up the BL700 disk drive(s). Power up the BL700. Turn the function switch from OFF to SAFE. Wait for the SAFE indicator to light, then turn the function switch to MAINT. (If this is the initial installation of the BL700, the switch can be turned directly from OFF to MAINT.)
2. Mount the BL700 *System Boot and Maintenance* tape on the host, and be sure it is rewound. The drive should be on-line and set to 1600 bpi.
3. Execute the IDMBOOT program (supplied with IDM Host-resident Software) on the host system, indicating on the command line the serial device connection to the BL700 and the host tape device.
4. Reset the BL700. Wait for the message

   ```
   BL700 - Filename:
   ```

   to appear on the BL700 console before continuing.

   The above steps apply equally to the loading of system software and utilities. The following steps apply to the loading of system software only.

5. When the BL700's self-test is complete and the prompt

   ```
   BL700 - Filename:
   ```

   appears on the BL700 console, enter:

   ```
   load
   ```

   The kernal is loaded into memory from the tape and the BL700 console displays:

   ```
   loading kernal
   ```

6. When loaded, the kernal displays:

   ```
   KERNAL n: <day> <date> <time>
   Accessible Disks:
   Name       Ctlr    Drive    Low    High    Status
   `disk 1 '   x       y        m      n       <status 1>
   `disk 2 '   x       y        m      n       <status 1>
       .
       .
   `disk n '   x       y        m      n       <status n>
   loading syscalls
   loading fileload
   ```

7. The kernal then begins to load FILELOAD from the tape. When loaded, FILELOAD lists the files on the magnetic tape:

```
number of available dbins: x
Free process pages: y
Fileload version n <date>
kernal    revision <m> size <n>
syscalls  revision <m> size <n>
fileload  revision <m> size <n>
dfu       revision <m> size <n>
dfu2      revision <m> size <n>
front     revision <m> size <n>
qryproc   revision <m> size <n>
support   revision <m> size <n>
sort      revision <m> size <n>
   ...
   ...
```

NOTE

Certain files are not part of the system software and are not copied to disk.

FILELOAD then starts loading the system software onto the system disk. The following message appears at the BL700 console for each file copied to disk. This continues until all system software files are copied from the tape to the system disk.

```
fileload <filename>, size n, file y
creating idm file <filename>
file assigned relid z
   <filename> loaded
```

If the file is already present on the system disk, (e.g. when updating software) the following message appears on the BL700 console for that file:

```
fileload <filename>, size n, file y
   <filename> loaded
```

In addition, series of messages appear on the host terminal, allowing the operator to verify the progress of the load procedure.

8. At the completion of the load, the BL700 resets itself, performs a self-test, and displays the prompt:

```
BL700 - Filename:
```

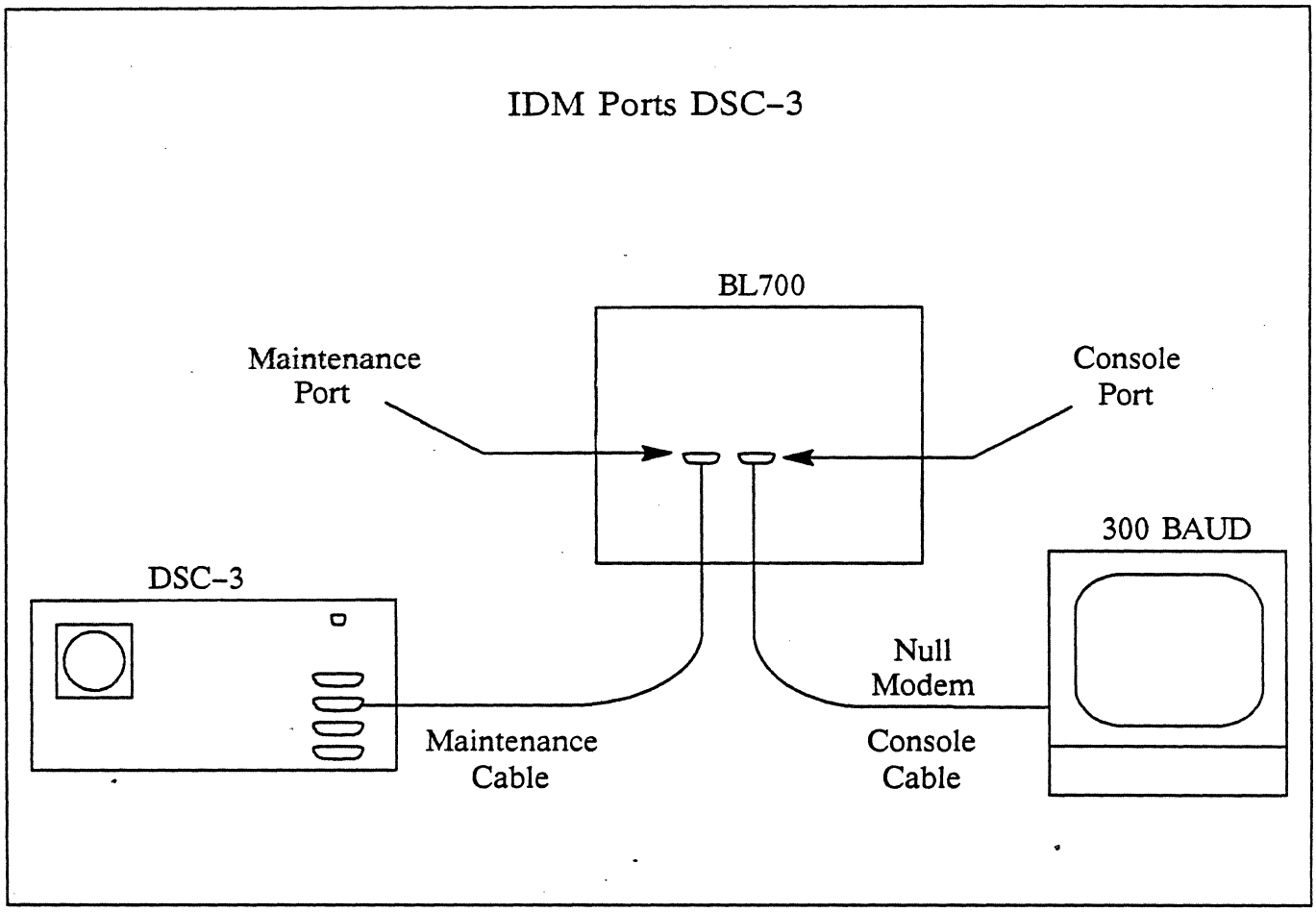


Figure C-1: BL700 Ports DSC-3

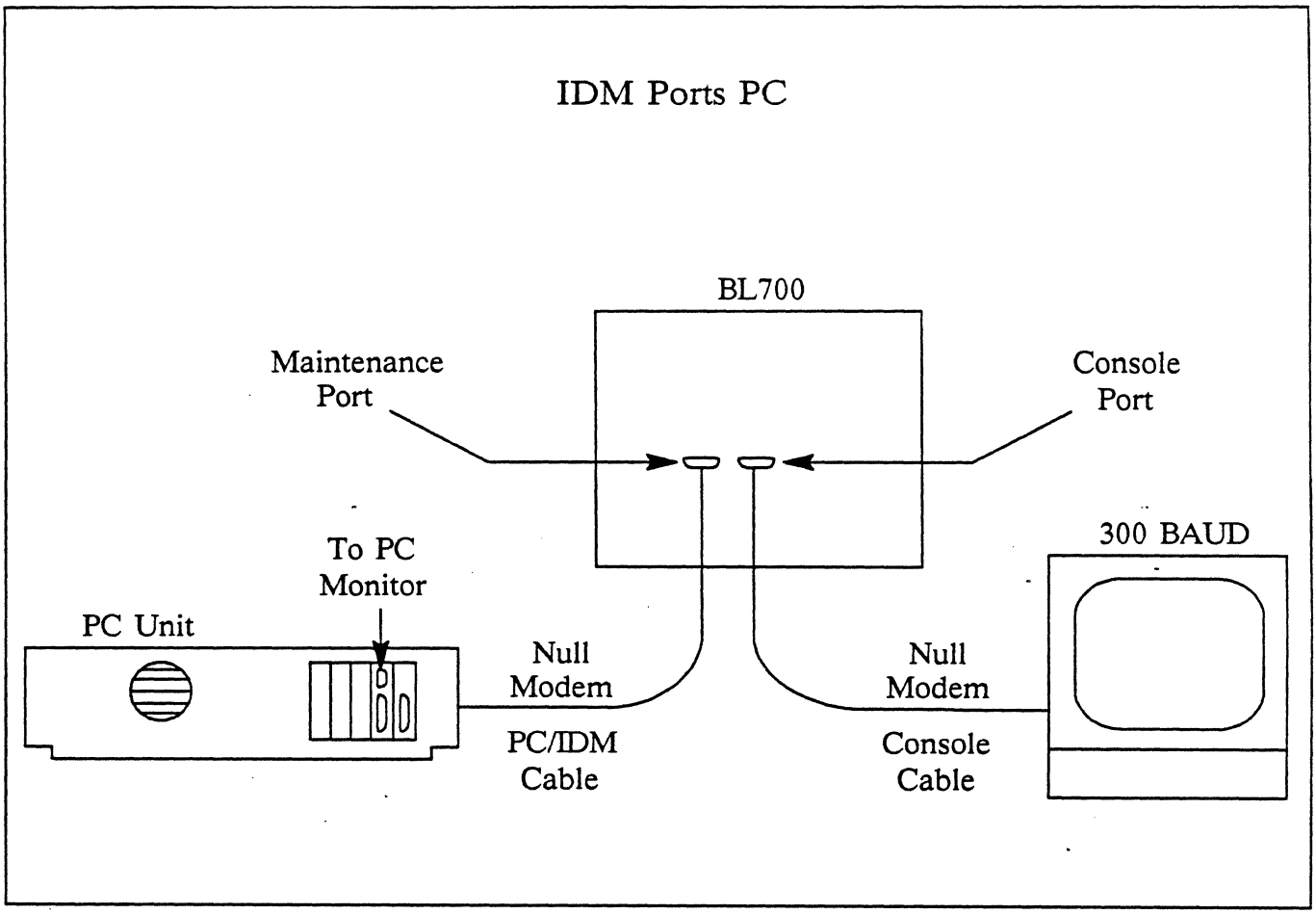


Figure C-2: BL700 Ports PC

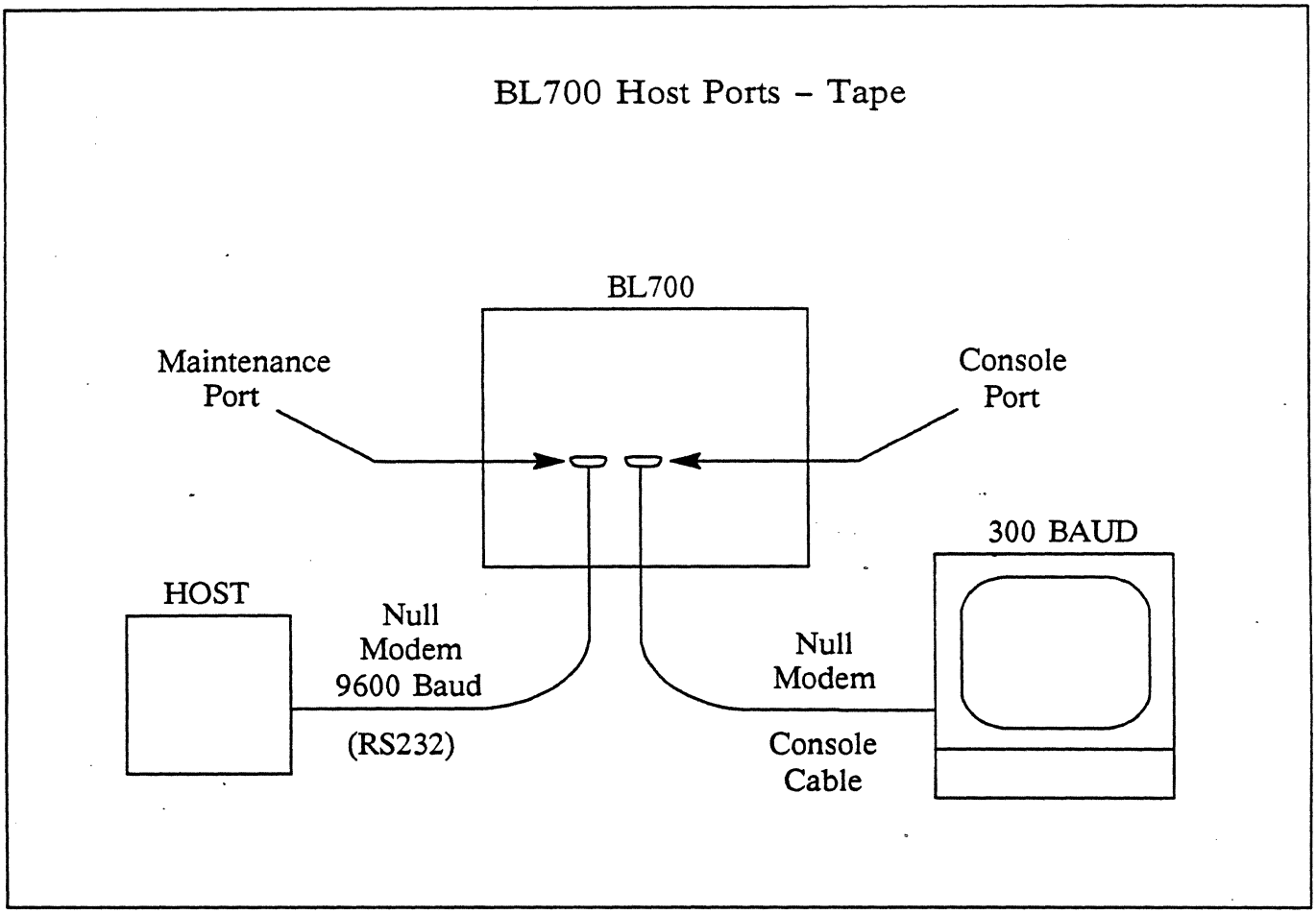


Figure C–3: BL700 Ports Host Tape

APPENDIX D: REPAIR PARTS LIST

IDM Board Parts Ordering Procedure

Repair Parts List

BL700 BOARD PARTS ORDERING PROCEDURE

GENERAL

This section contains the repair parts ordering procedure for the BL700 and a list of parts available for order.

This section is intended to assist you in replacing a failed board. If you plan to maintain a set of spare boards, contact your BLI Sales office for assistance.

BL700 BOARD PARTS ORDERING PROCEDURE

When ordering a repair part, please specify the part number and the BL700 serial number. The serial number is located on the rear panel below the fans.

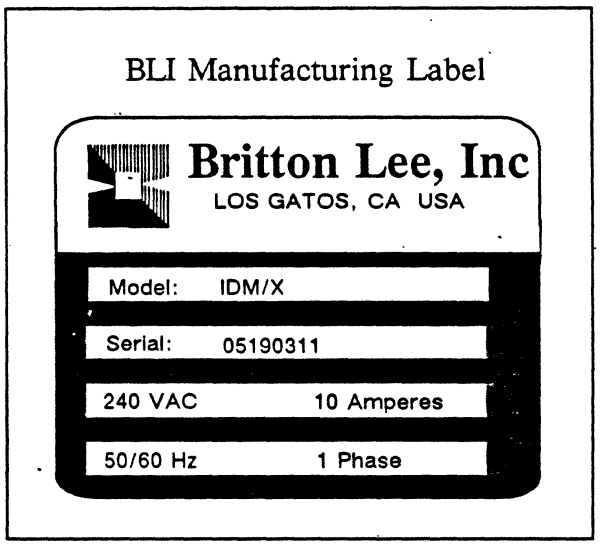


Figure D-1: BLI Manufacturing Label

The serial number is used to determine if your unit is still in warranty and to add information to its history file.

If the unit is out of warranty, we will need a Purchase Order number for the amount of repair. Please have the following information available when you call Britton Lee.

1. Name of person placing the order
2. Name of person receiving the part
3. Company name, shipping address (include suite, floor, room, etc.)
4. Billing address
5. Contact telephone number
6. Name of part to be replaced
7. Part number of your failed part
8. BL700 Serial Number
9. Purchase Order number if unit is out of warranty
10. Failure description
11. Engineering change order (ECO) number

All parts listed in the Repair Parts List are grouped by function. Certain parts may be ordered individually or as part of a complete assembly. Part numbers followed by the "@" symbol indicate that a single item within an assembly can be ordered. Part numbers preceded by the "#" symbol indicate a complete assembly.

Refer to the Repair Parts List to locate specific part numbers.

Use the replacement part shipping container and static bag to return the failed part to Britton Lee, Los Gatos, California. Please return the part immediately to receive credit. You will be invoiced for the full amount if the failed part is not received within 30 days.

REPAIR PARTS LIST

500 / 0*	500 / 1	500 / 2	500 / 1*	500 / 2*	BLI Part Number	Description
●	●		●		133-0001	Database Processor, 4Mhz
●	●		●		133-0719	Database Processor, 6 Mhz
●			●		133-0720	Database Processor, 6 Mhz
●	●	●	●	●	133-0752	Database Processor, 6 Mhz
	●	●	●	●	133-1180	Database Processor, 6 Mhz (Mirrored Disk)
	●	●	●	●	113-0069	DBP-Control Panel Internal Cable
	●	●	●	●	113-0070	DBP-Maintenance Internal Cable
	●	●	●	●	113-0070	DBP-Console Internal Cable
	●	●	●	●	113-0170	Console Port External Cable
	●	●	●	●	113-0156	Maintenance Port External Cable
		●		●	133-0007	Database Accelerator
●	●	●	●	●	133-0003	Memory Timing & Control
●	●	●	●	●	133-0188	1 Megabyte Memory
●	●	●	●	●	133-0006	Disk Controller
●	●	●	●	●	110-0098@	DC Interface Panel
●	●	●	●	●	113-0074@	DC Internal "A" Cable
●	●	●	●	●	113-0072@	DC Internal "B" Cable
●	●	●	●	●	#112-0727	DC Interface Panel & Cables Assy
●	●	●	●	●	113-1128-072	Disk Drive "A" Sh'ld Cable, 6'
●	●	●	●	●	113-1128-108	Disk Drive "A" Sh'ld Cable, 9'
●	●	●	●	●	113-1128-144	Disk Drive "A" Sh'ld Cable, 12'
●	●	●	●	●	113-1128-180	Disk Drive "A" Sh'ld Cable, 15'
●	●	●	●	●	113-1128-240	Disk Drive "A" Sh'ld Cable, 20'
●	●	●	●	●	429-0038-001	Disk Drive "A" Cable Terminator
●	●	●	●	●	113-1130-072	Disk Drive "B" Sh'ld Cable, 6'
●	●	●	●	●	113-1130-108	Disk Drive "B" Sh'ld Cable, 9'
●	●	●	●	●	113-1130-144	Disk Drive "B" Sh'ld Cable, 12'
●	●	●	●	●	113-1130-180	Disk Drive "B" Sh'ld Cable, 15'
●	●	●	●	●	113-1130-240	Disk Drive "B" Sh'ld Cable, 20'
●	●	●	●	●	133-0172	Tape Controller
●	●	●	●	●	133-1226	High Performance Tape Controller
●	●	●	●	●	110-0192@	TPC Interface Panel
●	●	●	●	●	113-0073@	TPC Read and Write Internal Cables
●	●	●	●	●	#112-0736	TPC Interface Panel & Cables Assy
●	●	●	●	●	113-1132	TPC Read & Write Ext Sh'ld Cables, 20'
●	●	●	●	●	113-1223-001	TPC Daisy Chain Cable set, 20'

* This indicates the Britton Lee database server's that conform to new FCC regulations (serial numbers greater than 05189311). These were first shipped November 1983.

500 / 0*	500 / 1	500 / 2	500 / 1*	500 / 2*	BLI Part Number	Description
●	●	●	●	●	133-0004	Parallel Channel I/O (BL700 Control)
●	●	●	●	●	133-0807	Parallel Channel I/O (Host Control)
●	●	●	●	●	110-0100@	PI/O Interface Panel
●	●	●	●	●	113-0071@	PI/O Internal Cable
●	●	●	●	●	#112-0726	PI/O Interface Panel & Cable Assy
●	●	●	●	●	428-0005	PI/O (DEC UNIBUS) N.I. card GPIB-2
●	●	●	●	●	428-0005-001	PI/O External Cable to N.I. card, 26'
●	●	●	●	●	133-0005	Serial RS-232 Asynchronous Channel I/O
●	●	●	●	●	112-0083@	SI/O Interface Distribution Panel (8 Ports)
●	●	●	●	●	113-0081@	SI/O RS232 Internal Cable
●	●	●	●	●	#112-0725	SI/O Interface Panel & Cable Assy
●	●	●	●	●	110-1213	SI/O Interface Distribution Panel
●	●	●	●	●	133-0191	SI/O Rear Panel Distribution Rack (16 ports)
●	●	●	●	●	113-1016	SI/O RS232 Sh'ld Cable for IBM PC
●	●	●	●	●	113-0169	SI/O RS232 Female-Female Adapter Cable
●	●	●	●	●	133-0812	Ethernet Channel I/O
●	●	●	●	●	112-1215@	EI/O Interface Panel
●	●	●	●	●	113-1147@	EI/O Internal Cable
●	●	●	●	●	428-1246@	EI/O Transceiver
●	●	●	●	●	428-0010@	EI/O Transceiver 50' Cable
●	●	●	●	●	205-1325	EI/O Interface Panel, Transceiver, & Cables Assembly
●	●	●	●	●	133-0811	Block Mux Channel I/O
●	●	●	●	●	112-1420	BMC Interface Panel
●	●	●	●	●	113-1166	BMC Internal Cable
●	●	●	●	●	113-1155	BMC External Cable
	●	●			133-0008	500/1/2 Motherboard
			●	●	133-1091	500/1/2 Motherboard
			●	●	402-9161-261	MB Resistor Pack
			●	●	402-9221-331	MB Resistor Pack
			●	●	113-1113	MB - CP Internal Cable
●					133-1089	500/0 Motherboard
●					402-9161-261	MB Resistor Pack
●					402-9221-331	MB Resistor Pack
●					113-1113	MB - CP Internal Cable
	●	●			113-0113	Removable AC Line Cord
●			●	●	113-1114	Integral AC Line Cord

* This indicates the Britton Lee database server's that conform to new FCC regulations (serial numbers greater than 05189311). These were first shipped November 1983.

500 / 0*	500 / 1	500 / 2	500 / 1*	500 / 2*	BLI Part Number	Description
	●	●			117-0127	Power Supply, Multi
			●	●	117-1098	Power Supply, Multi
	●	●	●	●	117-0130	Power Supply, Bulk
		●			117-1029	Power Supply, Booster
●					117-0231	Power Supply, Multi
	●	●			117-0051	Transformer (T1)
●			●	●	117-1106	Transformer (T1)
●			●	●	117-1112	Thermal Sensor
	●	●			423-0000	Line Filter
●			●	●	423-0003	Line Filter
●	●	●	●	●	470-0004	Fan, 4 inch
●			●	●	470-0007	Fan, 6 inch
	●	●			111-0055	Control Panel, 500/1/2
	●	●			113-0075	CP - PS Internal Cable
	●	●			113-0069	CP - DBP Internal Cable
			●	●	112-1116	Control Panel, 500/1/2
			●	●	113-0069	CP - DBP Internal Cable
			●	●	113-1113	CP - MB Internal Cable
●					112-1115	Control Panel, 500/0
●					113-0069	CP - DBP Internal Cable
●					113-1113	CP - MB Internal Cable
	●	●			412-0000-001	Circuit Breaker, 10A
●		●	●	●	412-0006-000	Circuit Breaker, 15A
	●	●			414-0100-025	Rear Panel Fuse, .25A
	●	●			414-1000-002	Fuse Holder and Lamp
●	●	●	●	●	118-0540	Tool, TPC Loopback Asm

* This indicates the Britton Lee database server's that conform to new FCC regulations (serial numbers greater than 05189311). These were first shipped November 1983.

APPENDIX E: WIRE LISTS

BL700 FCC Wire Connections (AC & DC)

FCC Motherboard Bus Bar

PS1,PS2 and J9 Wiring

Motherboard

BL700 FCC Motherboard Resistor Pack Schematic

FCC Motherboard Slot Wiring

Database Processor Connector Pinouts

Disk Controller Connector and Cable Details

Tape Controller Read and Write Cables

Serial Interface (RS-232) Pinouts J1

IEEE-488 Host Interface, GPIB Standard

Ethernet Connector Pinouts

Blockmux Board Connector Pin Assignment

Blockmux Interface BL700 Bus and Tag Cables

BL700 208V FCC Chassis

BL700 Chassis Wiring Options

BL700 FCC WIRE CONNECTIONS (AC & DC)

FROM	WIRE #	TO
MM30 +5V P/S POS SIDE	3A	MOTHERBOARD +5 BAR
MM30 +5V P/S POS SIDE	3B	MOTHERBOARD +5 BAR
MM30 +5V P/S MINUS SIDE	4A	MOTHERBOARD GND BAR
MM30 +5V P/S MINUS SIDE	4B	MOTHERBOARD GND BAR
INTERNAL BULKHEAD J-10 PIN 1	6	AC PWER SWITCH (A3) LOAD
INTERNAL BULKHEAD J-10 PIN 5	7	FRONT PANEL J-9 PIN 3
INTERNAL BULKHEAD J-10 PIN 4	8	FRONT PANEL J-9 PIN 8
INTERNAL BULKHEAD J-10 PIN 3	9	FRONT PANEL J-9 PIN 6
MM43-MULTI P/S TB2+S3 (PLUS)(5)	10	MOTHERBOARD GND BAR
MM30 -+5V P/S, TBI-S1 (NEG)(2)	11	MOTHERBOARD +5 SENSE (-5)
MM30 -+5V P/S, TBI-S1 (PLUS)	12	MOTHERBOARD +5 SENSE (+5)
MM43-MULTI P/S POS SIDE	14	MOTHERBOARD GND BAR
MM43-MULTI P/S TBI, +S1 (1)	15	MOTHERBOARD -5.2 SENSE (+5)
MM43-MULTI P/S TBI -S1 (2)	16	MOTHERBOARD -5.2 SENSE (-5)
MM43-MULTI P/S NEG SIDE	17	MOTHERBOARD -5.2 BAR
MM43-MULTI P/S TB2, +S2 (1)	18	MOTHERBOARD +12
MM43-MULTI P/S TB2, -V2 (3)	19	MOTHERBOARD GND BAR
MM43-MULTI P/S TB2, -V3	21	MOTHERBOARD -12
CHASSIS GND	22	MOTHERBOARD GND BAR
MM43-MULTI P/S GND	23	MM30 +5V P/S GND
MM43-MULTI P/S GND	24	CHASSIS GND BLOCK
MM30 +5V P/S TBI ON/OFF	25	FRONT PANEL J-9 PIN 2
MM43-MULTI P/S TBI, PF (40)	26	FRONT PANEL J-9 PIN 1
MM43-MULTI P/S TBI, ON/OFF (3)	27	FRONT PANEL J-9 PIN 4
MM43-MULTI P/S TBI, AC (6)	29	MM30 T5U P/S TBI AC (6)
AC SWITCH LOAD 1 (A1)	30	TERMINAL BLOCK SLOT 2
AC SWITCH LOAD 3 (C3)	31	TERMINAL BLOCK SLOT 1
TERMINAL BLOCK SLOT 2	32	FAN #1 AC POWER
MM30 +5V P/S TBI, ACC (7)	33	TERMINAL BLOCK SLOT 1
INTERNAL BLOCK J-10 PIN 2	34	TERMINAL BLOCK SLOT 4
FANS #3,2 AC POWER CORD	35	TERMINAL BLOCK SLOT 1
AC SWITCH LOAD 2 (A2)	36	TERMINAL BLOCK SLOT 3
FAN #1 AC POWER CORD	37	TERMINAL BLOCK SLOT 3
AC SWITCH R1 (B2)	38	AC SWITCH R1 (A2)
MM30 +5V P/S TBI, ACC (7)	39	TERMINAL BLOCK SLOT 4
MM43-MULTI P/S TBI, ACC (7)	40	MM30 +5V P/S TBI ACC (7)
FAN AC POWER #3, 2<Tab>	41	TERMINAL BLOCK SLOT 4
FAN #3 GND WIRE	42	T-1 GND WIRE
FAN #2 GND WIRE	44	F-1 GND WIRE
FAN #1 GND WIRE	45	CHASSIS GND
FAN #2 GND WIRE	46	CHASSIS GND BLOCK
CHASSIS GND BLOCK	47	GND SHIELD IN PLUG
CHASSIS GND BLOCK	48	LINE FILTER GND
LINE FILTER-LINE (BLUE)	49	LINE CORD/AC PLUG (BLUE)
LINE FILTER-LINE (BROWN)	50	LINE CORD/AC PLUG (BROWN)
AC SWITCH LINE 1 (C1)	51	LINE FILTER LOAD 1
AC SWITCH LINE 2 (C2)	52	LINE FILTER LOAD 2
AC SWITCH LOAD 1 (A1)	53	AC SWITCH (B1)
MM43-MULTI GND	54	CHASSIS GND BLOCK
MM30 +5V P/S GND	55	CHASSIS GND BLOCK
CHASSI	56	CHASSIS GND BLOCK
FRONT PANEL J-9 PIN 9	57	THERMAL SENSOR PIN 1
FRONT PANEL J-9 PIN 12	58	THERMAL SENSOR PIN 2

NOTE: Wire numbers 5, 13, 20, 28, and 43 are not present. Refer to figures for wire CONNECTIONS.

FCC MOTHERBOARD BUS BAR

CABLE	TO
<- 3A -> MM30 ->	+15V-PS1 - Pos side
<- 3B -> MM30 ->	+15V-PS1 - Pos side
<- 4A -> MM30 ->	+15V-PS1 - Neg side
<- 4B -> MM30 ->	+15V-PS1 - Neg side
<- 10 -> MM43 ->	Multi-PS2 - TB2, +53(5)
<- 11 -> MM30 ->	+15V-PS1 - TB1, -S1(2)
<- 12 -> MM30 ->	+ 5V-PS1 - TB1, +S1(1)
<- 14 -> MM43 ->	Multi-PS2 - Pos side
<- 15 -> MM43 ->	Multi-PS2 - TB1, +S1(1)
<- 16 -> MM43 ->	Multi-PS2 - TB2, -S2(2)
<- 17 -> MM43 ->	Multi-PS2 - Neg side
<- 18 -> MM43 ->	Multi-PS2 - TB2, +S2(1)
<- 19 -> MM43 ->	Multi-PS2 - TB2, +V2(3)
<- 21 -> MM43 ->	Multi-PS2 - TB2, -V3(7)
<- 22 -> Chassis GND	

MOTHERBOARD BUS

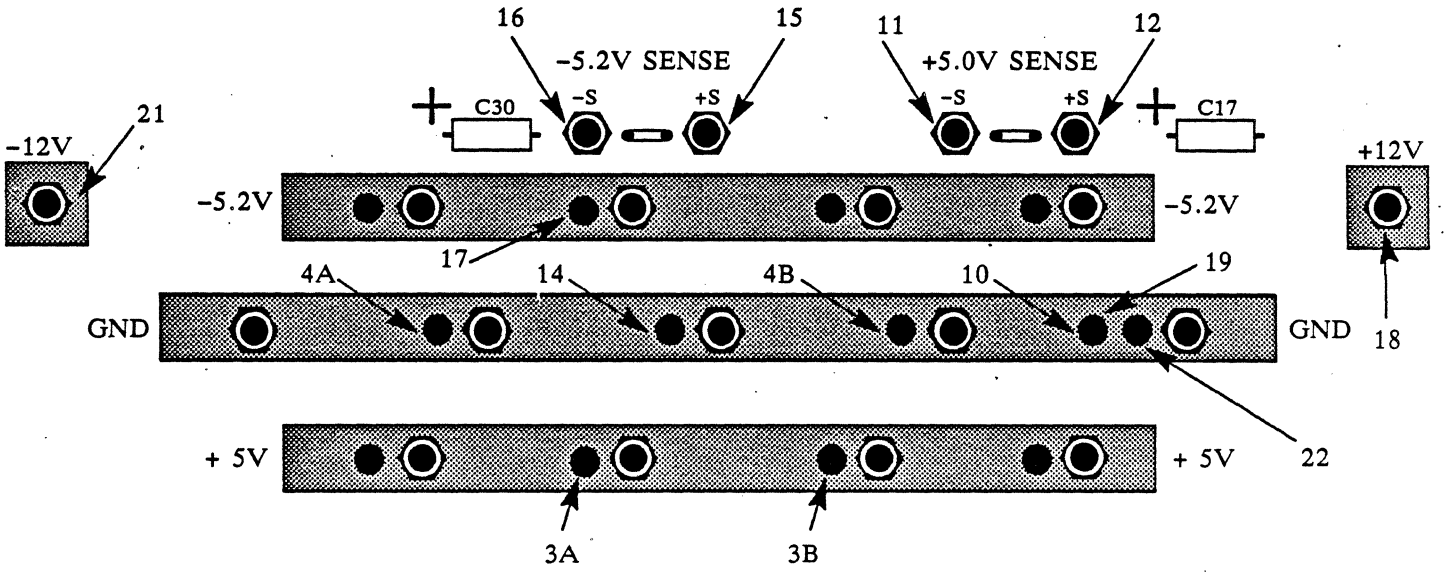
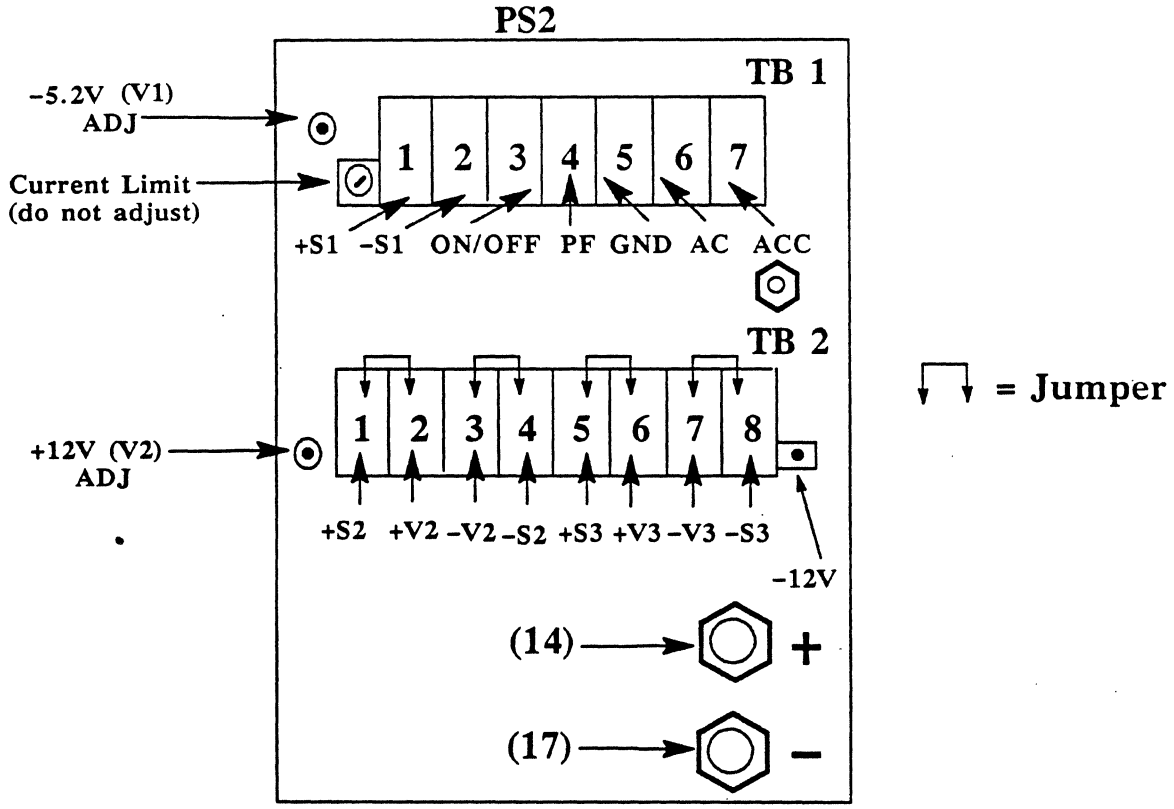


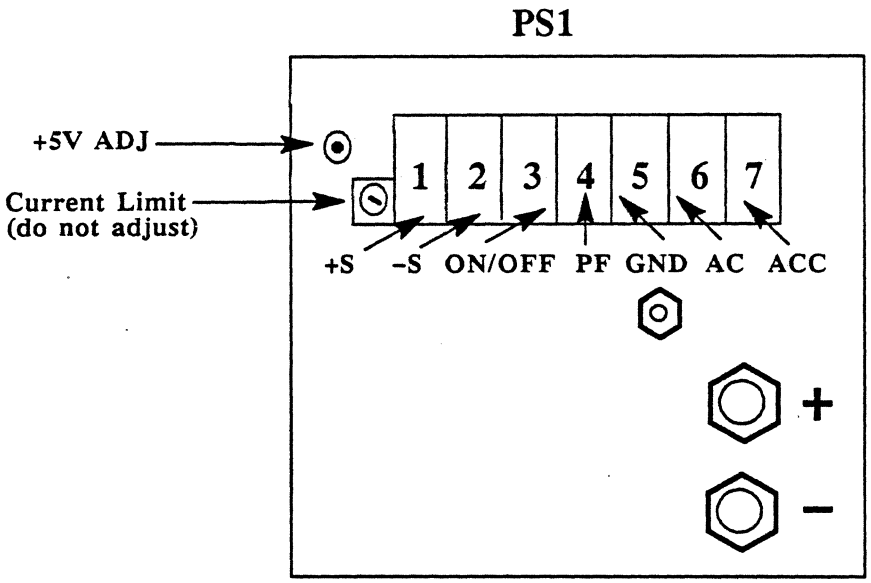
Figure E-1: Motherboard Bus

PS2, PS1 and J9 WIRING



PS2 Wiring
(Multi supplier MM43)

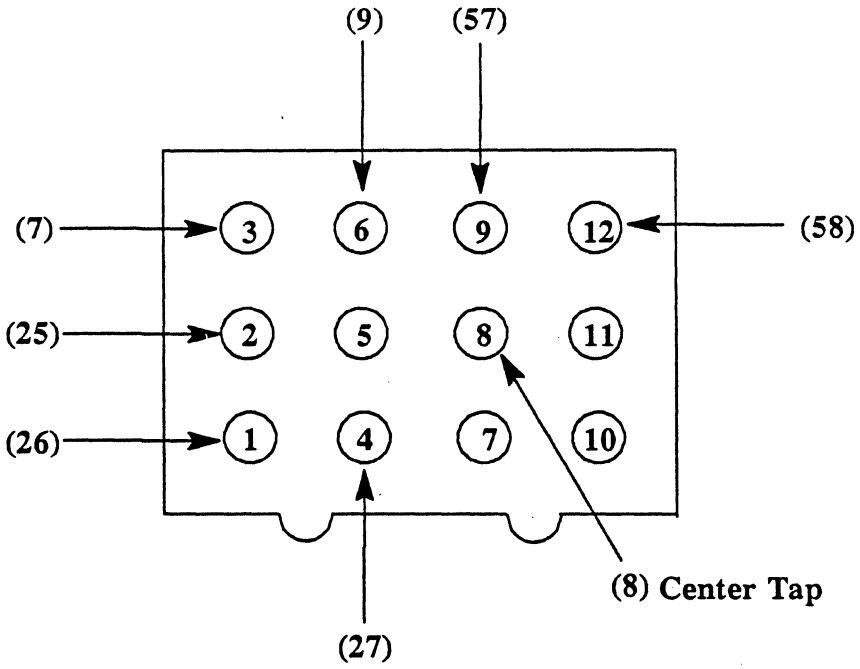
Wire #	From Multi-MM43-PS2	To
15	TB1 - +S1	Mthrbrd, -5.2 sense +
16	TB1 - -S1	Mthrbrd, -5.2 sense -
27	TB1 - ON/OFF	J9, Pin #4
26	TB1 - PF	J9, Pin #1
29	TB1 - AC	PS1, AC
40	TB1 - ACC	PS1, ACC
54	PS, GNB	Chassis GND Block
23	PS, GND	Chassis GND Block
24	PS, GND	Chassis GND Block
18	TB2, - +S2	Mthrbrd, +12
19	TB2, - V2	Mthrbrd, GND BAR
10	TB2, - +S3	Mthrbrd, GND BAR
21	TB2, - V3	Mthrbrd, - 12
14	V1, Pos.	Mthrbrd, GND BAR
17	V1, Neg.	Mthrbrd, - 5.2



PS1 Wiring
(+5V Supply, MM30)

Wire #	From	To
12	TB1, -S	Mthrbrd 5V Sense (+S)
11	TB1, -S	Mthrbrd 5V Sense (-S)
25	TB1, ON/OFF	J-9, Pin 2
39	TB1, AC	Term Block SL 4
29	TB1, AC	PS2, TB1, AC
40	TB1, ACC	PS2, TB1, ACC
33	TB1, ACC	Term Block SL1
23	PS1, GND	PS2, GND
55	PS1, GND	Chassis GND Block
3A	PS3, POS	Mthrbrd +5V BAR
3B	PS1, POS	Mthrbrd +5V BAR
4A	PS1, NEG	Mthrbrd GND BAR
4B	PS1, NEG	Mthrbrd GND BAR

J-9



J-9 Wiring
(Front Panel)

Wire #	From	To
26	J-9, Pin 1	PS2, TB1, PF
25	J-9, Pin 2	PS1, TB1, ON/OFF
7	J-9, Pin 3	J-10, Pin 5
27	J-9, Pin 4	PS2, TB1, ON/OFF
9	J-9, Pin 6	J-10, Pin 3
8	J-9, Pin 8	J-10, Pin 4
57	J-9, Pin 9	Thermal Sensor
58	J-9, Pin 12	Thermal Sensor

MOTHERBOARD

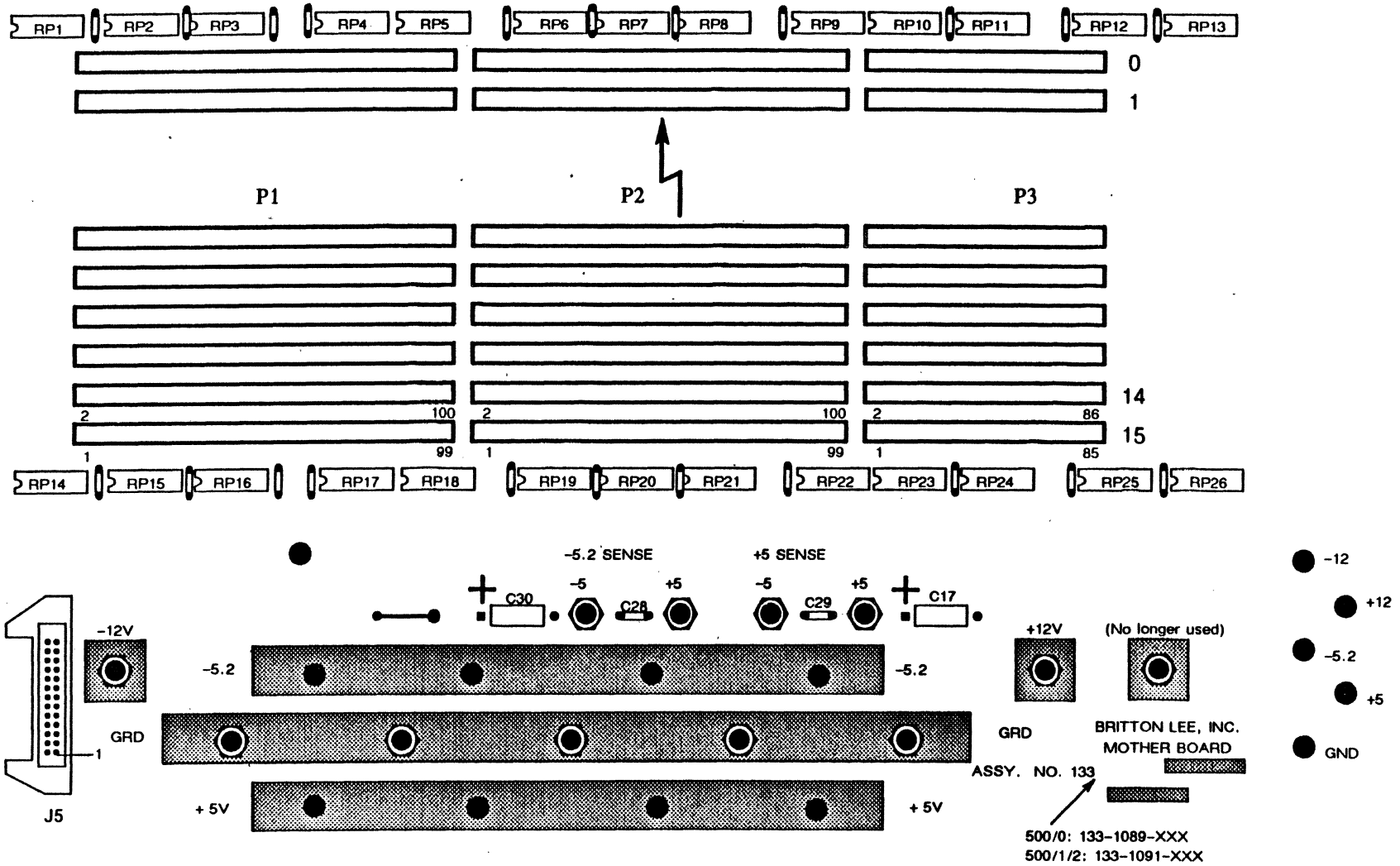


Figure E-2: Motherboard

BL700 FCC MOTHERBOARD RESISTOR PACK SCHEMATIC

NOTES:

1. Refer to pin listing below for all Power and Ground I/O pins.
2. Refer to "Terminator" columns below for resistor pack locations and pin numbers.
3. There are two types of 16 pin DIP R-Packs as follows:
   a. 220/330 ohm, pin 16=+5V, pin 8-GND
      14 resistor pairs per DIP.
   b. 160/260 ohm, pin 16=GND, pin 8=-5.2V
      14 resistor pairs per DIP.

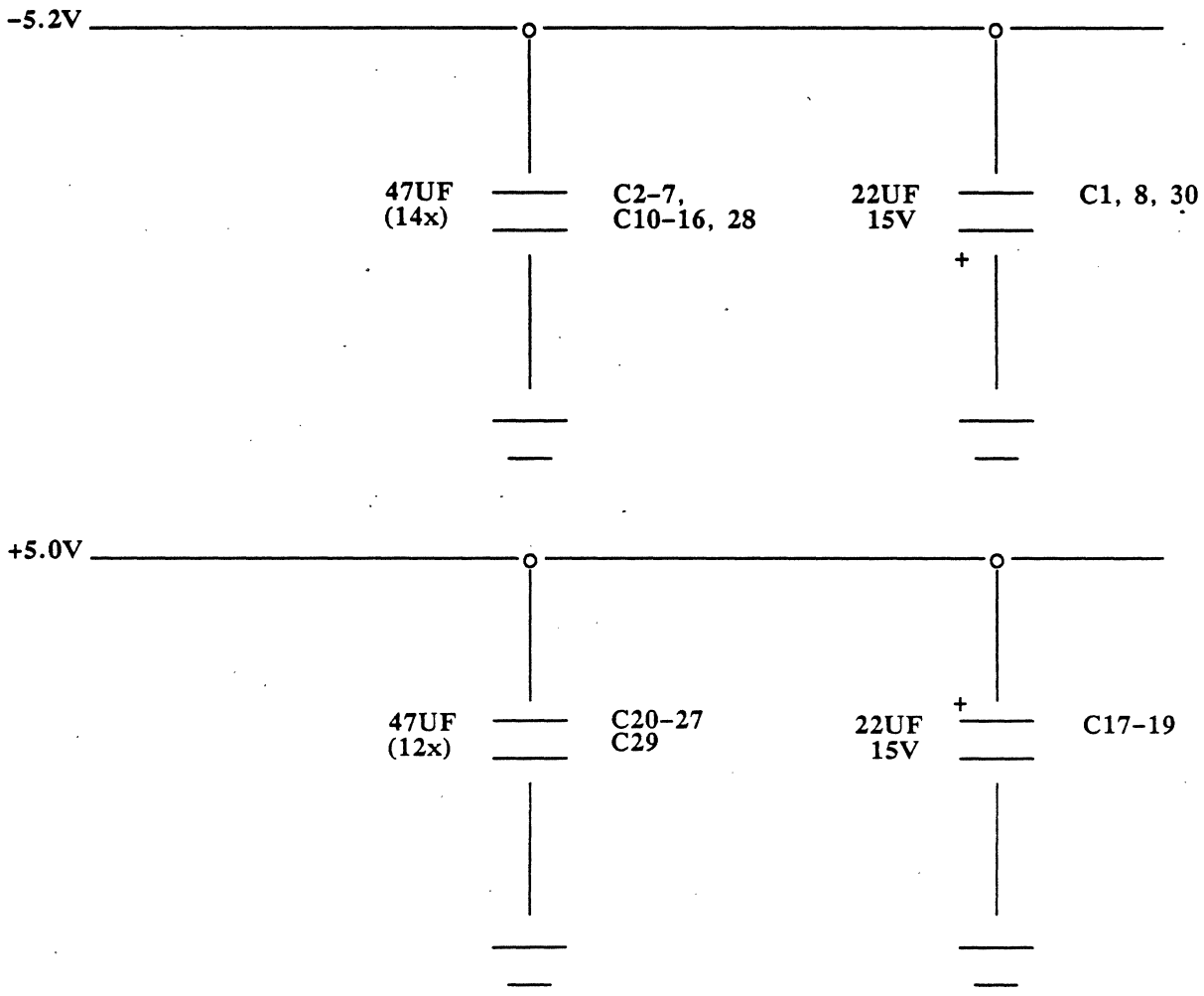


Figure E-3: Resistor Pack Schematic

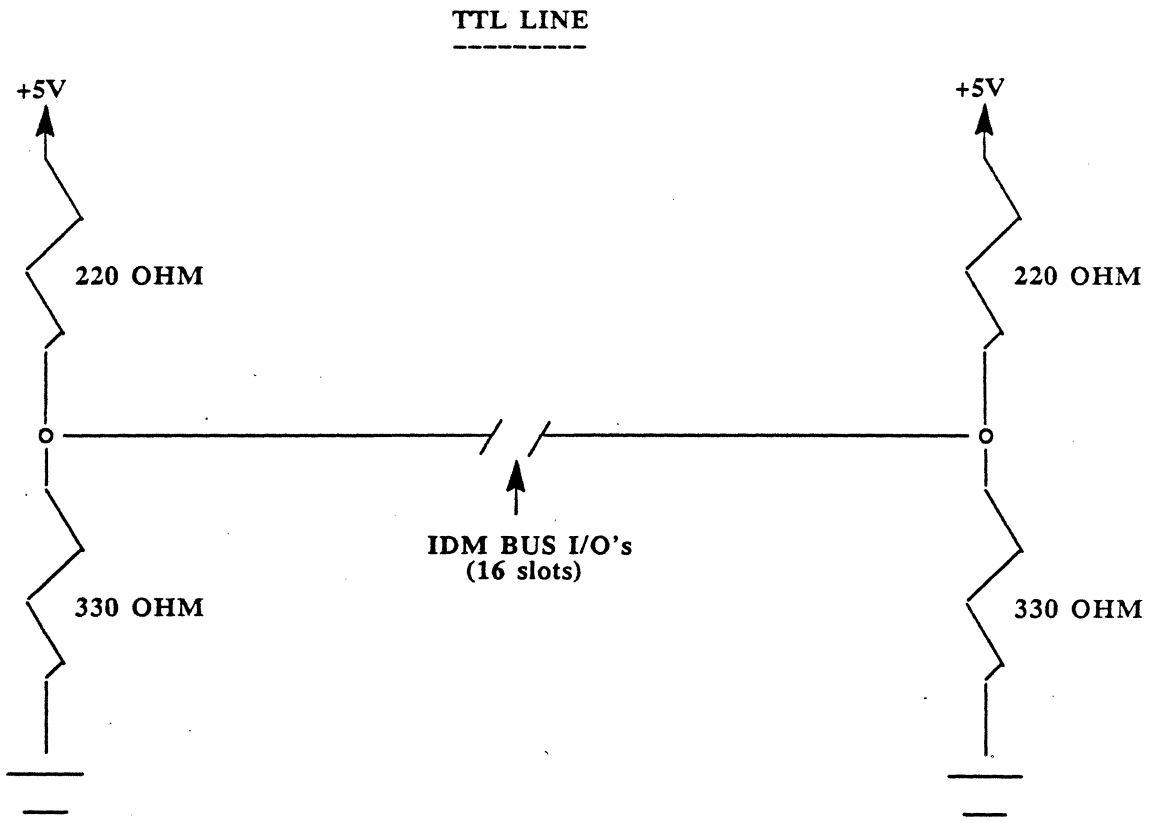


Figure E-4: TTL Line

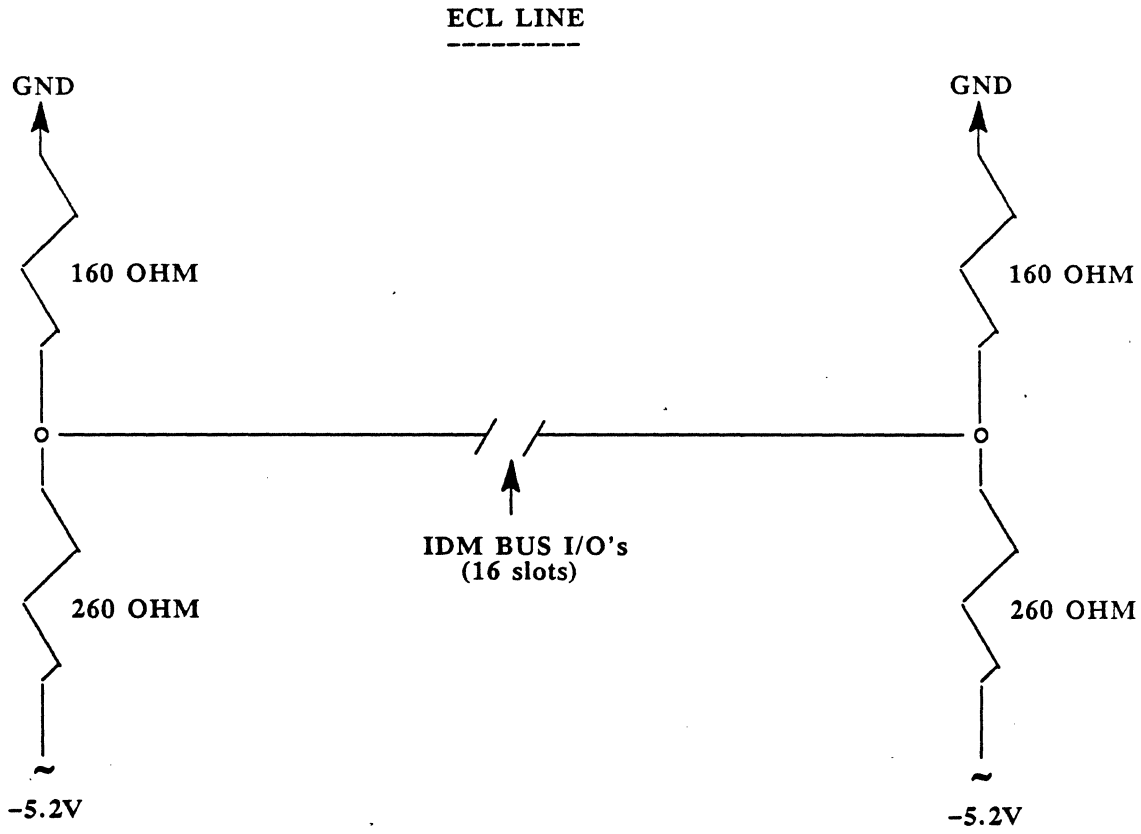


Figure E-5: ECL Line

FCC MOTHERBOARD SLOT WIRING

BL700 BUS PIN	SIGNAL NAME	SIGNAL TYPE	TERMINATORS	
J1-01	Ground	GND		
J1-02	Ground	GND		
J1-03	VCC (+5V)	PWR		
J1-04	VCC (+5V)	PWR		
J1-05	VCC (+5V)	PWR		
J1-06	VCC (+5V)	PWR		
J1-07	VCC (+5V)	PWR		
J1-08	VCC (+5V)	PWR		
J1-09	Ground	GND		
J1-10	Ground	GND		
J1-11	VDD (+12V)	PWR		
J1-12	VDD (+12V)	PWR		
J1-13	Ground	GND		
J1-14	Ground	GND		
J1-15	VFF (-12V)	PWR		
J1-16	VFF (-12V)	PWR		
J1-17	VEE (-5.2V)	PWR		
J1-18	VEE (-5.2V)	PWR		
J1-19	VEE (-5.2V)	PWR		
J1-20	VEE (-5.2V)	PWR		
J1-21	Ground	GND		
J1-22	Ground	GND		
J1-23	+IB03	ECL	RP15-01	RP01-01
J1-24	+IB02	ECL	RP15-15	RP01-15
J1-25	+IB01	ECL	RP15-02	RP01-02
J1-26	+IB00	ECL	RP15-14	RP01-14
J1-27	+IB06	ECL	RP15-03	RP01-03
J1-28	+IB07	ECL	RP15-13	RP01-13
J1-29	+IB04	ECL	RP15-04	RP01-04
J1-30	+IB05	ECL	RP15-12	RP01-12
J1-31	Ground	GND		
J1-32	Ground	GND		
J1-33	+IB19	ECL	RP15-05	RP01-05
J1-34	+IB18	ECL	RP15-11	RP01-11
J1-35	+IB17	ECL	RP15-10	RP01-06
J1-36	+IB16	ECL	RP15-06	RP01-10
J1-37	+IB22	ECL	RP15-09	RP01-07
J1-38	+IB23	ECL	RP15-07	RP01-09
J1-39	+IB20	ECL	RP14-09	RP03-09
J1-40	+IB21	ECL	RP16-01	RP02-15
J1-41	Ground	GND		
J1-42	Ground	GND		
J1-43	+IB35	ECL	RP16-02	RP02-14
J1-44	+IB34	ECL	RP16-15	RP02-13
J1-45	+IB33	ECL	RP16-03	RP02-01
J1-46	+IB32	ECL	RP16-14	RP02-12
J1-47	+IB38	ECL	RP16-04	RP02-02
J1-48	+IB39	ECL	RP16-13	RP02-11
J1-49	+IB36	ECL	RP16-05	RP02-03
J1-50	+IB37	ECL	RP16-12	RP02-10

BL700 BUS PIN	SIGNAL NAME	SIGNAL TYPE	TERMINATORS	
J1-51	Ground	GND		
J1-52	Ground	GND		
J1-53	+IB51	ECL	RP16-06	RP02-04
J1-54	+IB50	ECL	RP16-11	RP02-05
J1-55	+IB49	ECL	RP16-07	RP02-06
J1-56	+IB48	ECL	RP16-10	RP02-09
J1-57	+IB54	ECL	RP16-09	RP02-07
J1-58	+IB55	ECL	RP14-07	RP14-10
J1-59	+IB52	ECL	RP14-10	RP03-14
J1-60	+IB53	ECL	RP14-06	RP03-15
J1-61	Ground	GND		
J1-62	Ground	GND		
J1-63	+IB66	ECL	RP14-11	RP03-13
J1-64	+IB64	ECL	RP14-05	RP03-12
J1-65	+IB67	ECL	RP14-12	RP03-11
J1-66	+IB65	ECL	RP14-04	RP03-10
J1-67	Ground	GND		
J1-68	Ground	GND		
J1-69	SPARE	ECL	RP14-13	RP03-01
J1-70	SPARE	ECL	RP14-03	RP03-02
J1-71	SPARE	ECL	RP14-14	RP03-03
J1-72	SPARE	ECL	RP14-02	RP03-04
J1-73	SPARE	ECL	RP14-15	RP03-05
J1-74	SPARE	ECL	RP14-01	RP03-06
J1-75	SPARE	ECL	RP17-01	RP03-07
J1-76	+DMWS	ECL	RP17-15	RP04-02
J1-77	Ground	GND		
J1-78	Ground	GND		
J1-79	VEE (-5.2V)	PWR		
J1-80	VEE (-5.2V)	PWR		
J1-81	VEE (-5.2V)	PWR		
J1-82	VEE (-5.2V)	PWR		
J1-83	Ground	GND		
J1-84	Ground	GND		
J1-85	+MREQ	ECL	RP17-14	RP04-15
J1-86	+AACK	ECL	RP17-13	RP04-14
J1-87	+MACK	ECL	RP17-02	RP04-13
J1-88	+EREF	ECL	RP17-12	RP04-12
J1-89	Ground	GND		
J1-90	Ground	GND		
J1-91	+RSTB	ECL	RP17-03	RP04-11
J1-92	+WSTB	ECL	RP17-11	RP04-01
J1-93	SPARE	ECL	RP17-10	RP04-03
J1-94	+SMTH	ECL	RP17-09	RP04-04
J1-95	Ground	GND		
J1-96	Ground	GND		
J1-97	+IORG	ECL	RP17-04	RP04-05
J1-98	+IOAK	ECL	RP17-05	RP04-06
J1-99	+MPLD	ECL	RP17-06	RP04-07
J1-100	+MCLK	ECL	RP17-07	RP04-09

BL700 BUS PIN	SIGNAL NAME	SIGNAL TYPE	TERMINATORS	
J2-01	SLA0	TTL		
J2-02	SLA1	TTL		
J2-03	SLA2	TTL		
J2-04	SLA3	TTL		
J2-05	-HFLT	TTL	RP18-13	RP05-15
J2-06	-MRST	TTL	RP18-12	RP05-14
J2-07	-BREQ	TTL	RP18-11	RP05-13
J2-08	-BGNT	TTL	RP18-10	RP05-01
J2-09	Ground	GND		
J2-10	Ground	GND		
J2-11	-CREQ	TTL	RP18-09	RP05-02
J2-12	-CGNT	TTL	RP18-14	RP05-12
J2-13	-CARB	TTL	RP18-15	RP05-03
J2-14	-ACLO	TTL	RP18-01	RP05-11
J2-15	-CRB0	TTL	RP18-02	RP05-04
J2-16	-CRB1	TTL	RP18-03	RP05-10
J2-17	-CRB2	TTL	RP18-04	RP05-05
J2-18	-CRB3	TTL	RP18-07	RP05-09
J2-19	Ground	GND		
J2-20	Ground	GND		
J2-21	-CINT	TTL	RP18-06	RP05-06
J2-22	-CACK	TTL	RP18-05	RP05-07
J2-23	-BXFR	TTL	RP19-15	RP06-15
J2-24	-RSNC	TTL	RP19-01	RP06-14
J2-25	SPARE	TTL	RP19-14	RP06-01
J2-26	-BPER	TTL	RP19-02	RP06-02
J2-27	-MST1	TTL	RP19-13	RP06-13
J2-28	-MST2	TTL	RP19-03	RP06-03
J2-29	Ground	GND		
J2-30	Ground	GND		
J2-31	-MPDS	TTL	RP19-12	RP06-12
J2-32	-BYTE	TTL	RP19-11	RP06-04
J2-33	-WRITE	TTL	RP19-10	RP06-11
J2-34	-QUAD	TTL	RP19-09	RP06-05
J2-35	-DBP0	TTL	RP19-04	RP06-10
J2-36	-DBP1	TTL	RP19-05	RP06-06
J2-37	-DB00	TTL	RP19-06	RP06-09
J2-38	-DB01	TTL	RP19-07	RP06-07
J2-39	Ground	GND		
J2-40	Ground	GND		
J2-41	-DB02	TTL	RP20-01	RP07-13
J2-42	-DB03	TTL	RP20-15	RP07-14
J2-43	-DB04	TTL	RP20-02	RP07-15
J2-44	-DB05	TTL	RP20-14	RP07-12
J2-45	-DB06	TTL	RP20-13	RP07-11
J2-46	-DB07	TTL	RP20-03	RP07-10
J2-47	-DB08	TTL	RP20-04	RP07-09
J2-48	-DB09	TTL	RP20-12	RP07-01
J2-49	Ground	GND		
J2-50	Ground	GND		

BL700 BUS PIN	SIGNAL NAME	SIGNAL TYPE	TERMINATORS	
J2-51	-DB10	TTL	RP20-05	RP07-02
J2-52	-DB11	TTL	RP20-11	RP07-07
J2-53	-DB12	TTL	RP20-10	RP07-03
J2-54	-DB13	TTL	RP20-09	RP07-06
J2-55	-DB14	TTL	RP20-06	RP07-04
J2-56	-DB15	TTL	RP20-07	RP07-05
J2-57	-BAP0	TTL	RP21-01	RP08-15
J2-58	-BAP1	TTL	RP21-02	RP08-14
J2-59	Ground	GND		
J2-60	Ground	GND		
J2-61	-BAP2	TTL	RP21-03	RP08-13
J2-62	-BAP3	TTL	RP21-15	RP08-12
J2-63	-BA00	TTL	RP21-04	RP08-11
J2-64	-BA01	TTL	RP21-14	RP08-10
J2-65	-BA02	TTL	RP21-05	RP08-09
J2-66	-BA03	TTL	RP21-13	RP08-01
J2-67	-BA04	TTL	RP21-12	RP08-02
J2-68	-BA05	TTL	RP21-11	RP08-03
J2-69	Ground	GND		
J2-70	Ground	GND		
J2-71	-BA06	TTL	RP21-06	RP08-04
J2-72	-BA07	TTL	RP21-10	RP08-05
J2-73	-BA08	TTL	RP21-07	RP08-06
J2-74	-BA09	TTL	RP21-09	RP08-07
J2-75	-BA10	TTL	RP23-03	RP09-15
J2-76	-BA11	TTL	RP23-02	RP09-14
J2-77	-BA12	TTL	RP23-01	RP09-13
J2-78	-BA13	TTL	RP23-13	RP09-12
J2-79	Ground	GND		
J2-80	Ground	GND		
J2-81	-BA14	TTL	RP22-01	RP09-11
J2-82	-BA15	TTL	RP22-02	RP09-10
J2-83	-BA16	TTL	RP22-03	RP09-09
J2-84	-BA17	TTL	RP22-04	RP09-01
J2-85	-BA18	TTL	RP22-05	RP09-07
J2-86	-BA19	TTL	RP22-06	RP09-02
J2-87	-BA20	TTL	RP22-07	RP09-06
J2-88	-BA21	TTL	RP22-15	RP09-03
J2-89	Ground	GND		
J2-90	Ground	GND		
J2-91	-BA22	TTL	RP22-14	RP09-05
J2-92	-BA23	TTL	RP22-13	RP09-04
J2-93	-BA24	TTL	RP22-12	RP10-01
J2-94	-BA25	TTL	RP22-09	RP10-15
J2-95	-BA26	TTL	RP22-11	RP10-02
J2-96	-BA27	TTL	RP22-10	RP10-14
J2-97	-BA27	TTL	RP23-15	RP10-03
J2-98	SPARE	TTL	RP23-14	RP10-04
J2-99	Ground	GND		
J2-100	Ground	GND		

BL700 BUS PIN	SIGNAL NAME	SIGNAL TYPE	TERMINATORS	
J3-01	VCC (+5V)	PWR		
J3-02	VCC (+5V)	PWR		
J3-03	VCC (+5V)	PWR		
J3-04	VCC (+5V)	PWR		
J3-05	VCC (+5V)	PWR		
J3-06	VCC (+5V)	PWR		
J3-07	VCC (+5V)	PWR		
J3-08	VCC (+5V)	PWR		
J3-09	Ground	GND		
J3-10	Ground	GND		
J3-11	+RAO	TTL	RP23-12	RP10-13
J3-12	+RA1	TTL	RP23-11	RP10-05
J3-13	+RA2	TTL	RP23-10	RP10-12
J3-14	+RA3	TTL	RP23-04	RP10-06
J3-15	+RA4	TTL	RP23-09	RP10-11
J3-16	+RA5	TTL	RP23-05	RP10-07
J3-17	+RA6	TTL	RP23-06	RP10-10
J3-18	+RA7	TTL	RP23-07	RP10-09
J3-19	Ground	GND		
J3-20	Ground	GND		
J3-21	+IB71	ECL	RP24-01	RP11-15
J3-22	+IB69	ECL	RP24-02	RP11-14
J3-23	+IB70	ECL	RP24-03	RP11-13
J3-24	+IB68	ECL	RP24-15	RP11-12
J3-25	Ground	GND		
J3-26	Ground	GND		
J3-27	+IB59	ECL	RP24-14	RP11-11
J3-28	+IB58	ECL	RP24-04	RP11-10
J3-29	+IB57	ECL	RP24-13	RP11-01
J3-30	+IB56	ECL	RP24-05	RP11-09
J3-31	+IB62	ECL	RP24-12	RP12-15
J3-32	+IB63	ECL	RP24-06	RP12-14
J3-33	+IB60	ECL	RP24-11	RP12-13
J3-34	+IB61	ECL	RP24-07	RP12-12
J3-35	Ground	GND		
J3-36	Ground	GND		
J3-37	+IB43	ECL	RP24-10	RP12-11
J3-38	+IB42	ECL	RP24-09	RP12-10
J3-39	+IB41	ECL	RP26-06	RP12-06
J3-40	+IB40	ECL	RP25-07	RP12-07
J3-41	+IB46	ECL	RP25-07	RP12-09
J3-42	+IB47	ECL	RP25-01	RP13-15
J3-43	+IB44	ECL	RP25-02	RP13-13
J3-44	+IB45	ECL	RP25-03	RP13-14
J3-45	Ground	GND		
J3-46	Ground	GND		
J3-47	+IB27	ECL	RP25-04	RP11-02
J3-48	+IB26	ECL	RP25-05	RP11-03
J3-49	+IB25	ECL	RP26-05	RP11-04
J3-50	IB24	ECL	RP26-01	RP11-05

BL700 BUS PIN	SIGNAL NAME	SIGNAL TYPE	TERMINATORS	
J3-51	+IB30	ECL	RP25-15	RP12-05
J3-52	+IB31	ECL	RP25-14	RP11-06
J3-53	+IB28	ECL	RP25-13	RP12-04
J3-54	+IB29	ECL	RP26-02	RP11-07
J3-55	Ground	GND		
J3-56	Ground	GND		
J3-57	+IB11	ECL	RP25-12	RP12-03
J3-58	+IB10	ECL	RP26-03	RP12-02
J3-59	+IB09	ECL	RP25-11	RP12-01
J3-60	+IB08	ECL	RP26-04	RP13-01
J3-61	+IB14	ECL	RP25-10	RP13-02
J3-62	+IB15	ECL	RP25-09	RP13-03
J3-63	+IB12	ECL	RP26-15	RP13-04
J3-64	+IB13	ECL	RP26-14	RP13-05
J3-65	Ground	GND		
J3-66	Ground	GND		
J3-67	VEE (-5.2V)	PWR		
J3-68	VEE (-5.2V)	PWR		
J3-69	VEE (-5.2V)	PWR		
J3-70	VEE (-5.2V)	PWR		
J3-71	VEE (-5.2V)	PWR		
J3-72	VEE (-5.2V)	PWR		
J3-73	Ground	GND		
J3-74	Ground	GND		
J3-75	VDD (+12V)	PWR		
J3-76	VDD (+12V)	PWR		
J3-77	Ground	GND		
J3-78	Ground	GND		
J3-79	VCC (+5V)	PWR		
J3-80	VCC (+5V)	PWR		
J3-81	VCC (+5V)	PWR		
J3-82	VCC (+5V)	PWR		
J3-83	VCC (+5V)	PWR		
J3-84	VCC (+5V)	PWR		
J3-85	Ground	GND		
J3-86	Ground	GND		

BL700 BUS PIN	SIGNAL NAME	SIGNAL TYPE	TERMINATORS
J5-01	VDD (+12V)	PWR	
J5-02	Ground	GND	
J5-03	Ground	GND	
J5-04	N/C	-	
J5-05	VEE (-5.2V)	PWR	
J5-06	N/C	-	
J5-07	N/C	-	
J5-08	N/C	-	
J5-09	N/C	-	
J5-10	N/C	-	
J5-11	N/C	-	
J5-12	N/C	-	
J5-13	Ground	GND	
J5-14	N/C	-	
J5-15	VFF (-12V)	PWR	
J5-16	N/C	-	
J5-17	Ground	GND	
J5-18	N/C	-	
J5-19	N/C	-	
J5-20	N/C	-	
J5-21	N/C	-	
J5-22	N/C	-	
J5-23	N/C	-	
J5-24	N/C	-	
J5-25	+IDC	TTL	
J5-26	+Cable Open	TTL	

DATABASE PROCESSOR CONNECTOR PINOUTS

INTRODUCTION

The Database Processor has four connector locations. These connectors are J-4, J-5, J-6, and J-7. Connector J-7 is an in-house diagnostic connector only. It is not needed for installation or operation of the BL700. Below is a description of the signal pins, the signal origin of J4, J5 and J6 connectors and their respective cables.

CONNECTOR J-4 PINOUTS

This is a 20 pin cable that connects the DBP J4 to the Front Panel.

SIGNAL PIN	NAME	DESCRIPTION	SOURCE *
J4-01	+BACOK	BUFFERED AC OK	FP
J4-02	+B60HZ	BUFFERED 60 Hz	FP
J4-03	-BPONR	BUFFERED POWER ON RESET	FP
J4-04	+REMEM	REMOTE ENABLE	FP
J4-05	-BSW1	BUS SWITCH 1	FP
J4-06	-BSW2	BUS SWITCH 2	FP
J4-07	+LED1	NOT USED	DBP
J4-08	+LED0	NOT USED	DBP
J4-09	-PLFP	POWER LOSS FRONT PANEL	FP
J4-10	+LED3	SERVICE LIGHT	DBP
J4-11	+LED4	FAULT LIGHT	DBP
J4-12	+LED5	SAFE LIGHT	DBP
J4-13	+LED6	READY LIGHT	DBP
J4-14	-DCON	DC ON	DBP
J4-15	GND	GROUND	DBP
J4-16	GND	GROUND	DBP
J4-17		RESERVED	
J4-18	+5VDC	+ 5 VDC	DBP
J4-19	+5VDC	+ 5 VDC	DBP
J4-20		RESERVED	

*** SIGNAL SOURCE**

FP = FRONT PANEL
DBP = DATABASE PROCESSOR

CONSOLE PORT PINOUTS J-5

This is for the console port of the BL700. It is a 26 pin connector and it is connected to the interface panel by a three foot internal cable terminating in a RS232 female connector labeled console.

SIGNAL PIN	NAME	DESCRIPTION	SOURCE *
J5-01	GND	CHASSIS GROUND	DBP
J5-02	TXD	TRANSMIT DATA	DBP
J5-03	RXD	RECEIVE DATA	CT
J5-04	RTS	REQUEST TO SEND	DBP
J5-05	CTS	CLEAR TTO SEND	CT
J5-06	N/C	.....	---
J5-07	GND	SIGNAL GROUND	DBP
J5-08	DCD	DATA CARRIER DETECT	CT
J5-09	N/C	.....	---
|	|	.....	---
|	|	.....	---
J5-19	N/C	.....	---
J5-20	DTR	DATA TERMINAL READY	DBP
J5-21	N/C	.....	DBP
|	|	.....	---
J5-25	N/C	.....	---

*** SIGNAL SOURCE**

DBP = DATABASE PROCESSOR
CT = CONSOLE TERMINAL

CONSOLE TERMINAL CABLE

This cable (P/N 113-0170) is supplied by Britton Lee. It is a modified three conductor RS-232C null-modem cable for connecting a terminal to the console port of the BL700. The recommended maximum length of this cable is 50 feet (15 meters).

CONSOLE PORT SIGNAL	SIGNAL PIN	SIGNAL PIN	CONSOLE TERM SIGNAL
TXD	02 ——————>	03	RXD
GND	07 <——————>	07	GND
RXD	03 <——————	02	TXD
RTS	04————		
CTS	05 <————		
DTR	20 <————		
DCD	08————		

MAINTENANCE PORT J-6

This connector is for the maintenance port of the BL700. It has 26 pins and is connected to the interface panel by a three foot internal cable terminating in a RS232 female connector labeled maintenance.

SIGNAL PIN	NAME	DESCRIPTION	SOURCE *
J6-01	GND	CHASSIS GROUND	---
J6-02	TXD	TRANSMIT DATA	DBP
J6-03	RXD	RECEIVE DATA	OS*
J6-04	RTS	REQUEST TO SEND	DBP
J6-05	CTS	CLEAR TO SEND	OS*
J6-06	N/C		---
J6-07	GND	SIGNAL GROUND	DBP
J6-08	DCD	DATA CARRIER DETECT	OS*
J6-09	N/C	.....	---
		.....	---
		.....	---
J6-19	N/C	.....	---
J6-20	DTR	DATA TERMINAL READY	DBP
J6-21	N/C	.....	DBP
		.....	---
J6-25	N/C	.....	---

*** SIGNAL SOURCE**

DBP = DATABASE PROCESSOR
OS* = OUTSIDE SOURCE*
*Where the outside source is a loading device or a modem used for remote diagnostics.

MAINTENANCE CABLE

This cable is supplied by Britton Lee to customers who order the DSC-3 for software loading. It is a full-modem cable (P/N 113-0156).

ADAPTER CABLE

This is a female-to-female RS-232 cable for connecting customer equipment to RS-232 connectors (P/N 113-0169).

DISK CONTROLLER CONNECTOR AND CABLE DETAILS

INTRODUCTION

The index and sector pulses must be provided on the "A" cable. Cable "B" should be no longer than 46 feet (14 meters). This accounts for BL700 internal lengths. The "B" cable should be shielded and the "A" cable should be twisted pair. Disk drives must be configured for "daisy chain" option. The maximum length of the external daisy chained "A" cables should be no longer than 96 feet (29.25 meters) per disk controller.

"A" CABLE PINOUT

SIGNAL PIN -	SIGNAL PIN +	DESCRIPTION	SOURCE *
J4-01	J4-31	TAG 1	DC
J4-02	J4-32	TAG 2	DC
J4-03	J4-33	TAG 3	DC
J4-04	J4-34	BIT 0	DC
J4-05	J4-35	BIT 1	DC
J4-06	J4-36	BIT 2	DC
J4-07	J4-37	BIT 3	DC
J4-08	J4-38	BIT 4	DC
J4-09	J4-39	BIT 5	DC
J4-10	J4-40	BIT 6	DC
J4-11	J4-41	BIT 7	DC
J4-12	J4-42	BIT 8	DC
J4-13	J4-43	BIT 9	DC
J4-14	J4-44	OPEN CABLE DETECT	DC
J4-15	J4-45	FAULT	DD
J4-16	J4-46	SEEK ERROR	DD
J4-17	J4-47	ON CYLINDER	DD
J4-18	J4-48	INDEX	DD
J4-19	J4-49	UNIT READY	DD
J4-20	J4-50	NOT USED (ADR MARK FND)	-
J4-21	J4-51	BUSY	DD
J4-22	J4-52	UNIT SELECT TAG	DC
J4-23	J4-53	UNIT SELECT BIT 0 ("1")	DC
J4-24	J4-54	UNIT SELECT BIT 1 ("2")	DC
J4-25	J4-55	SECTOR	DD
J4-26	J4-56	UNIT SELECT BIT 2 ("4")	DC
J4-27	J4-57	UNIT SELECT BIT 3 ("8")	DC
J4-28	J4-58	WRITE PROTECTED	DD
J4-29	---	POWER SEQ PICK	DC
---	J4-59	POWER SEQ HOLD	DC
J4-30	J4-60	NOT USED	-

*** SIGNAL SOURCE**

DD = DISK DRIVE
DC = DISK CONTROLLER

"B" CABLE PINOUTS J5,J6,J7,J8.

SIGNAL PIN -	SIGNAL PIN +	DESCRIPTION	SOURCE *
J5-02	J5-14	SERVO CLOCK	DD
J5-03	J5-16	READ DATA	DD
J5-05	J5-17	READ CLOCK	DD
J5-06	J5-19	WRITE CLOCK	DC
J5-08	J5-20	WRITE DATA	DC
J5-09	J5-22	UNIT SELECTED	DD
J5-10	J5-23	SEEK END	DD
J5-12	J5-24	**RESERVED FOR INDEX	-
J5-13	J5-26	**RESERVED FOR SECTOR	-
J5-01,04,07,11,15,18,21,25			GND

**Index and sector pulses must be provided on "A" cable.

*** SIGNAL SOURCE**

DD = DISK DRIVE
DC = DISK CONTROLLER

TAPE CONTROLLER READ WRITE CABLES

WRITE CABLE PINOUTS J-1

SIGNAL PIN	GROUND PIN	NAME	DESCRIPTION	SOURCE *
J1-02	J1-01	IFBY	FORMATTER BUSY	TD
J1-04	J1-03	ILWD	LAST WORD	TPC
J1-06	J1-05	IW4	WRITE DATA 4	TPC
J1-08	J1-07	IGO	INITIATE COMMAND	TPC
J1-10	J1-09	IW0	WRITE DATA 0	TPC
J1-12	J1-11	IW1	WRITE DATA 1	TPC
J1-14	J1-13	--	(RESERVED)	TD
J1-16	J1-15	--	(RESERVED)	TPC
J1-18	J1-17	IREV	REVERSE	TPC
J1-20	J1-19	IREW	REWIND	TPC
J1-22	J1-21	IWP	WRITE DATA PARITY	TPC
J1-24	J1-23	IW7	WRITE DATA 7	TPC
J1-26	J1-25	IW3	WRITE DATA 3	TPC
J1-28	J1-27	IW6	WRITE DATA 6	TPC
J1-30	J1-29	IW2	WRITE DATA 2	TPC
J1-32	J1-31	IW5	WRITE DATA 5	TPC
J1-34	J1-33	IWRT	WRITE	TPC
J1-36	J1-35	--	(RESERVED)	TPC
J1-38	J1-37	IEDIT	EDIT	TPC
J1-40	J1-39	IERASE	ERASE	TPC
J1-42	J1-41	IWFM	WRITE FILE MARK	TPC
J1-44	J1-43	--	(RESERVED)	TPC
J1-46	J1-45	ITAD0	TRANSPORT ADDRESS 0	TPC
J1-48	J1-47	IR2	READ DATA 2	TD
J1-50	J1-49	IR3	READ DATA 3	TD

*** SIGNAL SOURCE**

TD = TAPE DRIVE
TPC = TAPE CONTROLLER

READ CABLE PINOUTS J-2

SIGNAL PIN	GROUND PIN	NAME	DESCRIPTION	SOURCE *
J2-01	---	IRP	READ DATA PARITY	TD
J2-02	---	IRO	READ DATA 0	TD
J2-03	---	IR1	READ DATA 1	TD
J2-04	---	ILDP	LOAD POINT	TD
J2-06	J2-05	IR4	READ DATA 4	TD
J2-08	J2-07	IR7	READ DATA 7	TD
J2-10	J2-09	IR6	READ DATA 6	TD
J2-12	J2-11	IHER	HARD ERROR	TD
J2-14	J2-13	IFMK	FILE MARK	TD
J2-16	J2-15	IDENT	IDENTIFICATION	TD
J2-18	J2-17	IFEN	FORMATER ENABLE	TPC
J2-20	J2-19	IR5	READ DATA 5	TD
J2-22	J2-21	IEOT	END OF TAPE	TD
J2-24	J2-23	IRWU	REWIND/UNLOAD	TPC
J2-26	J2-25	--	(RESERVED)	TD
J2-28	J2-27	IRDY	READY	TD
J2-30	J2-29	IRWD	REWINDING	TD
J2-32	J2-31	IFPT	FILE PROTECT	TD
J2-34	J2-33	IRSTR	READ STROBE	TD
J2-36	J2-35	IWSTR	WRITE STROBE	TD
J2-38	J2-37	IDBY	DATA BUSY	TD
J2-40	J2-39	ISPEED	HIGH-SPEED STATUS	TD
J2-42	J2-41	ICER	CORRECT ERROR	TD
J2-44	J2-43	IONL	ON LINE	TD
J2-46	J2-45	ITAD1	TRANSPORT ADDRESS 1	TPC
J2-48	J2-47	IFAD	FORMATTER ADDRESS	TPC
J2-50	J2-49	IHISP	HIGH SPEED SELECT	TPC

*** SIGNAL SOURCE**

TD = TAPE DRIVE
TPC = TAPE CONTROLLER

SERIAL INTERFACE (RS-232) PINOUTS J1

SIGNAL PIN	NAME	DESCRIPTION	SOURCE	PORT	PORT PIN
J4-01	TXD	TRANSMIT DATA	CH	0	02
J4-02	RTS	REQUEST TO SEND	CH	0	04
J4-03	DTR	DATA TERMINAL READY	CH	0	20
J4-26	RXD	RECEIVE DATA	HT	0	03
J4-27	CTS	CLEAR TO SEND	HT	0	05
J4-28	DCD	DATA CARRIER DETECT	HT	0	08
J4-25, 50	GND	GROUND	CH	0	07
J4-04	TXD	.....	CH	1	02
J4-05	RTS	.....	CH	1	04
J4-06	DTR	.....	CH	1	20
J4-29	RXD	.....	HT	1	03
J4-30	CTS	.....	HT	1	05
J4-31	DCD	.....	HT	1	08
J4-25,50	GND	.....	CH	1	07
J4-07	TXD	.....	CH	2	02
J4-08	RTS	.....	CH	2	04
J4-09	DTR	.....	CH	2	20
J4-32	RXD	.....	HT	2	03
J4-33	CTS	.....	HT	2	05
J4-34	DCD	.....	HT	2	08
J4-25,50	GND	.....	CH	2	07
J4-10	TXD	.....	CH	3	02
J4-11	RTS	.....	CH	3	04
J4-12	DTR	.....	CH	3	20
J4-35	RXD	.....	HT	3	03
J4-36	CTS	.....	HT	3	05
J4-37	DCD	.....	HT	3	08
J4-25,50	GND	.....	CH	3	07
J4-13	TXD	TRANSMIT DATA	CH	4	02
J4-14	RTS	REQUEST TO SEND	CH	4	04
J4-15	DTR	DATA TERMINAL READY	CH	4	20
J4-38	RXD	RECEIVE DATA	HT	4	03
J4-39	CTS	CLEAR TO SEND	HT	4	05
J4-40	DCD	DATA CARRIER DETECT	HT	4	08
J4-25,50	GND	GROUND	CH	4	07
J4-16	TXD	.....	CH	5	02
J4-17	RTS	.....	CH	5	04
J4-18	DTR	.....	CH	5	20
J4-41	RXD	.....	HT	5	03
J4-42	CTS	.....	HT	5	05
J4-43	DCD	.....	HT	5	08
J4-25,50	GND	.....	CH	5	07
J4-19	TXD	.....	CH	6	02
J4-20	RTS	.....	CH	6	04
J4-21	DTR	.....	CH	6	20
J4-44	RXD	.....	HT	6	03
J4-45	CTS	.....	HT	6	05
J4-46	DCD	.....	HT	6	08
J4-25,50	GND	.....	CH	6	07

SIGNAL PIN	NAME	DESCRIPTION	SOURCE	PORT	PORT PIN
J4-22	TXD	.....	CH	7	02
J4-23	RTS	.....	CH	7	04
J4-24	DTR	.....	CH	7	20
J4-47	RXD	.....	HT	7	03
J4-48	CTS	.....	HT	7	05
J4-49	DCD	.....	HT	7	08
J4-25,50	GND	.....	CH	7	07

*** SIGNAL SOURCE**

CH = CHANNEL (SIO)
HT = HOST TERMINAL

IEEE-488 HOST INTERFACE, GPIB STANDARD

SIGNAL PIN	NAME	DESCRIPTION	GPIB END
J5-01	DIO 1	DATA LINE (1)	01
J5-02	DIO 2	DATA LINE (2)	02
J5-03	DIO 3	DATA LINE (3)	03
J5-04	DIO 4	DATA LINE (4)	04
J5-05	EOI	END OR IDENTIFY	05
J5-06	DAV	DATA VALID	06
J5-07	NRFD	NOT READY FOR DATA	07
J5-08	NDAC	NOT DATA ACCEPTED	08
J5-09	IFC	INTERFACE CLEAR	09
J5-10	SRQ	SERVICE REQUEST	10
J5-11	ATN	ATTENTION	11
J5-12	SHIELD	SYS CONT GND ONLY	12
J5-14	DIO 5	DATA LINE (5)	13
J5-15	DIO 6	DATA LINE (6)	14
J5-16	DIO 7	DATA LINE (7)	15
J5-17	DIO 8	DATA LINE (8)	16
J5-18	REN	REMOTE ENABLE	17
J5-19	GND 6	GND FOR DAV	18
J5-20	GND 7	GND FOR NRFD	19
J5-21	GND 8	GND FOR NDAC	20
J5-22	GND 9	GND FOR IFC	21
J5-23	GND 10	GND FOR SRQ	22
J5-24	GND 11	GND FOR ATN	23
J5-25	GND	LOGIC GND	24

ETHERNET CONNECTOR PINOUTS

ETHERNET CONSOLE CABLE TO J1
(P/N 113-0170, external)
(P/N 113-0070, internal)

CONNECTOR J-1

SIGNAL PIN	NAME	DESCRIPTION
J1-01	N/C	.....
J1-02	TXDA	NOT USED ..
J1-03	RXDA	NOT USED ..
J1-04	RTSA	NOT USED ..
J1-05	CTSA	NOT USED ..
J1-06	DTRA	NOT USED ..
J1-07	GND	GROUND
J1-08	DCDA	NOT USED ..
J1-09	N/C	.....
J1-10	N/C	.....
J1-11	TXDB	TRANSMIT DATA
J1-12	RXDB	RECEIVE DATA
J1-13	RTSB	NOT USED ..
J1-14	CTSB	NOT USED ..
J1-15	DTRB	NOT USED ..
J1-16	GND	GROUND
J1-17	DCDB	NOT USED ..
J1-18	N/C	.....
|	|	.....
J1-26	N/C	.....

ETHERNET INTERNAL TRANSCEIVER CABLE TO J2
(P/N 133-1147)

"A" END		SIGNAL		"B" END
J2-01	<——	Collusion –	——>	09
J2-02	<——	Collusion +	——>	02
J2-03	<——	Receiver –	——>	12
J2-04	<——	Receiver +	——>	05
J2-05	<——	Transmitter –	——>	10
J2-06	<——	Transmitter +	——>	03
J2-07	<——	N/C	——>	07
J2-08	<——	N/C	——>	04
J2-09	<——	GND	——>	06
J2-10	<——	+12vdc	——>	13
		SHIELD	——>	01
		N/C	——>	08
		N/C	——>	11
		N/C	——>	14
		N/C	——>	15

BLOCKMUX BOARD CONNECTOR PIN ASSIGNMENT

J1	J1 SIGNAL	BLOCKMUX	J2	J2 SIGNAL
1	ground		1	ground
3	bus in 0		3	bus out 0
5	bus in 1		5	bus out 1
7	bus in 2		7	bus out 2
9	bus in 3		9	bus out 3
11	bus in 7		11	bus out 4
13	bus in 6		13	bus out 5
15	bus in 5		15	bus out 6
17	bus in 4		17	bus out 7
19	bus in p		19	bus out p
21	mark in		21	address out
23	status in		23	command out
25	address in		25	service out
27	service in		27	data out
29	data in		29	suppress out
31	disconnect in		31	operational out
33	operational in		33	hold out
35	request in		35	select out
37	select in		37	on line switch
39	enable in		39	flag
41	on line		41	+5 volts
43	ground		43	+5 volts
45	power on		45	+5 volts
47	+5 volts		47	+5 volts
49	+5 volts		49	+5 volts
even	ground		even	ground

BLOCK MUX INTERFACE
BL700 BUS AND TAG CABLES

(BLI) IBM Bus Cable Part number:
(BLI) IBM Tag Cable Part number:

Tag Cable
(J4 & J6 Daughters)

G = Ground

B02 G	G02 G
B03 Operational In	G03 Clock Out**
B04 G	G04 G
B05 Address In	G05 Metering In**
B06 Spare 1	G06 Spare 3
B07 G	G07 G
B08 Select In	G08 Data In
B09 G	G09 G
B10 Address Out	G10 Data Out
B11 +6V*	G11 +6V*
B12 Suppress Out	G12 Hold Out
B13 G	G13 G
D02 Spare 2	J02 Spare 4
D03 G	J03 G
D04 Status In	J04 Metering Out**
D05 G	J05 G
D06 Service In	J06 Request In
D07 G	J07 G
D08 G	J08 G
D09 Select Out	J09 Spare 5
D10 G	J10 G
D11 Command Out	J11 Disconnect In
D12 G	12 G
D13 Service Out	J13 Operational Out

Bus Cable
(J1 & J3 Daughters)

B02 G	G02 G
B03 Bus Out P	G03 Bus In P
B04 G	G04 G
B05 Bus Out 1	G05 Bus In 1
B06 Spare 1	G06 Spare 3
B07 G	G07 G
B08 Bus Out 3	G08 Bus In 3
B09 G	G09 G
B10 Bus Out 5	G10 Bus In 5
B11 +6V*	G11 +6V*
B12 Bus Out 7	G12 Bus In 7
B13 G	G13 G
D02 Spare 2	J02 Spare 4
D03 G	J03 G
D04 Bus Out 0	J04 Bus In 0
D05 G	J05 G
D06 Bus Out 2	J06 Bus In 2
D07 G	J07 G
D08 G	J08 G
D09 Bus Out 4	J09 Bus In 4
D10 G	J10 G
D11 Bus Out 6	J11 Bus In 6
D12 G	J12 G
D13 Mark Out	J13 Mark In

* = Used as +6V for Sperry SU 00039 interface.
Not used for IBM or Sperry FIPS.

** = Used on IBM interface.
Not used on any Sperry interface.

Figure E-6: BL700 208V FCC Chassis

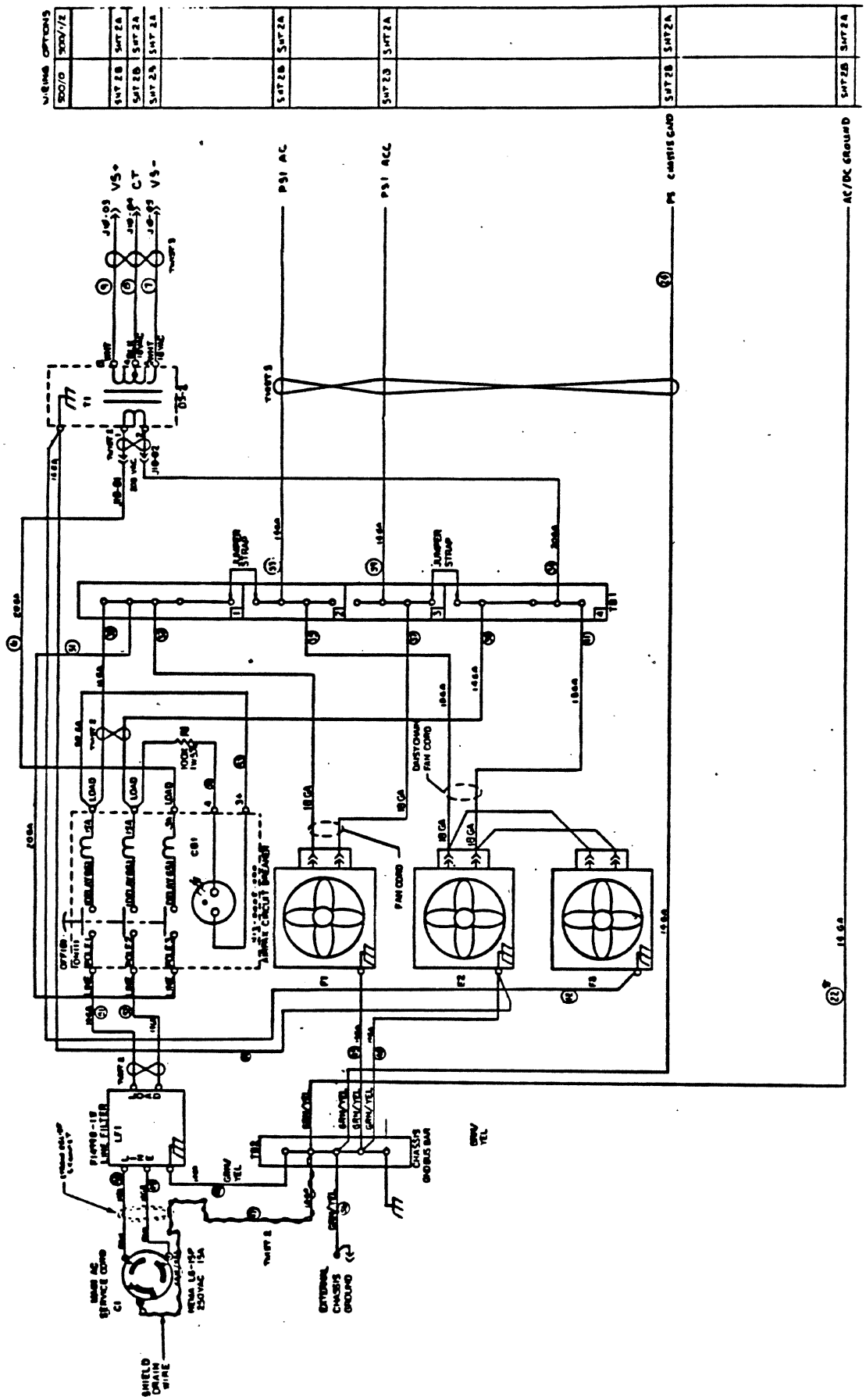


Figure E-7: BL700 Chassis Wiring Options

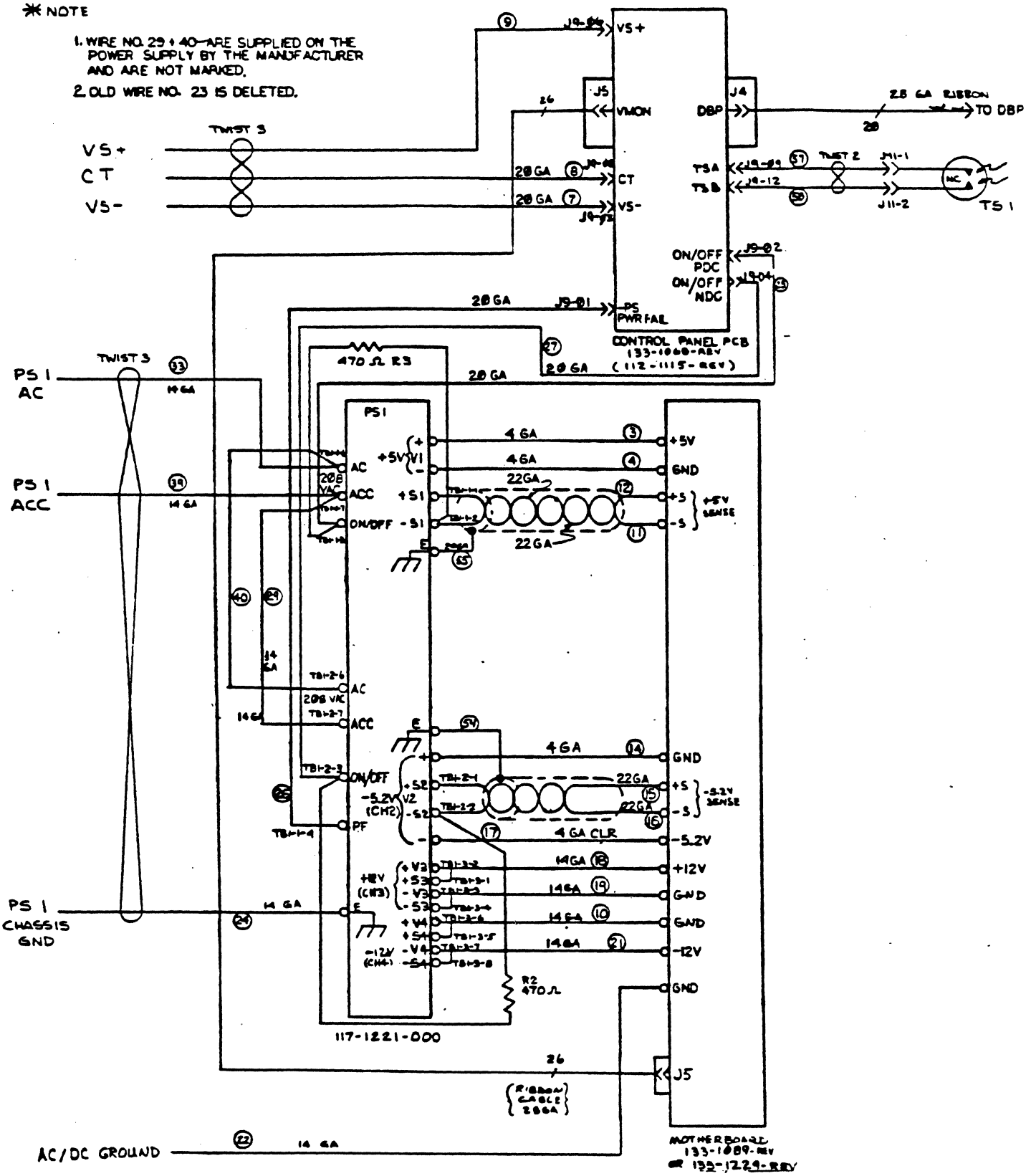
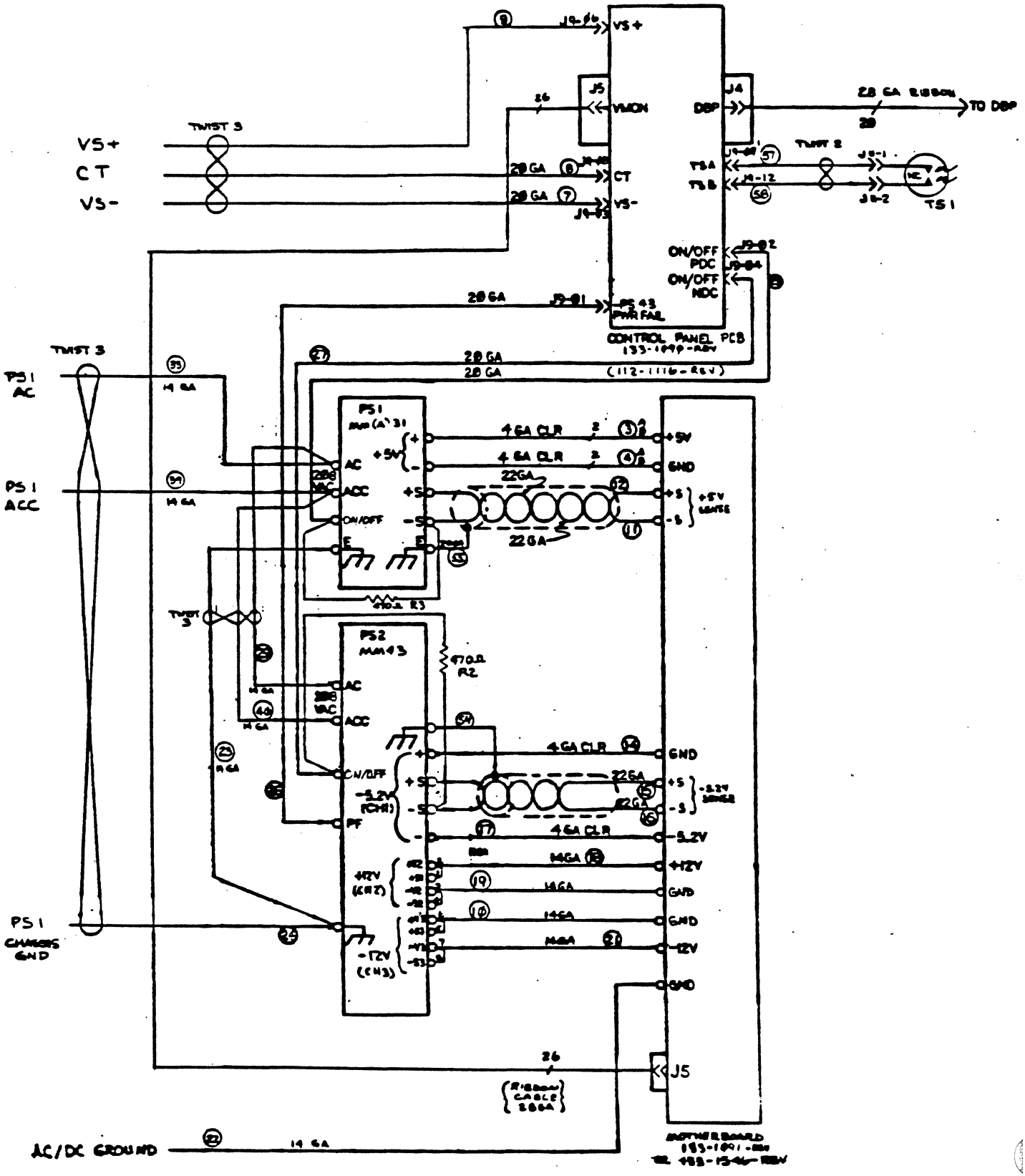


Figure E-8: BL700 Chassis Wiring Options



APPENDIX F: PRE-FCC DIFFERENCES

Back Interface Panel

PRE-FCC Areas

PRE-FCC AC/DC Wire List

Power Supplies

PRE-FCC Motherboard Slot Wiring

INTRODUCTION

This appendix explains the differences between pre-FCC and FCC Britton Lee database servers. There are no functional differences between pre-FCC and FCC Britton Lee database servers. Only the chassis was changed to meet FCC requirements. A pre-FCC Britton Lee database server can be upgraded to accommodate all levels of IDM/RDBMS software releases and the latest PCB changes.

Below is a list of pre-FCC and FCC differences covered in this section.

Chassis

1. Back Interface Panel
   a. fans
   b. power cord
   c. circuit breaker
   d. fuse (and holder)
   e. interface panel
2. Power supplies
   a. multi supply
   b. tm-11 supply
3. Wiring
   a. ac wiring
   b. dc wiring
4. Mother board
   a. two bolt line
   b. resistor packs
5. Control Panel
   a. connector plugs - non-interchangeable part
   b. thermal Sensor - only present on FCC model

BACK INTERFACE PANEL

1. There are four muffin fans that pull air through the front bezel, across the PC boards and the power supplies.
2. The standard ac power switch is an Airpax 10 amp circuit breaker. When upgrading a pre-Fcc Britton Lee database server, this circuit breaker will be replaced by a higher current handling 15 amp breaker.
3. There is a .25 amp safety fuse that is in-line to one of the input sides of the T1 transformer. This step-down transformer provides 14 to 18V ac to the multi power supply and control panel.
4. The ac receptacle is a high performance connector filter. It combines a low leadage EMI filter with an IEC (International Electrotechnical Commission) power line connector in on compact unit.

PRE-FCC REAR PANEL

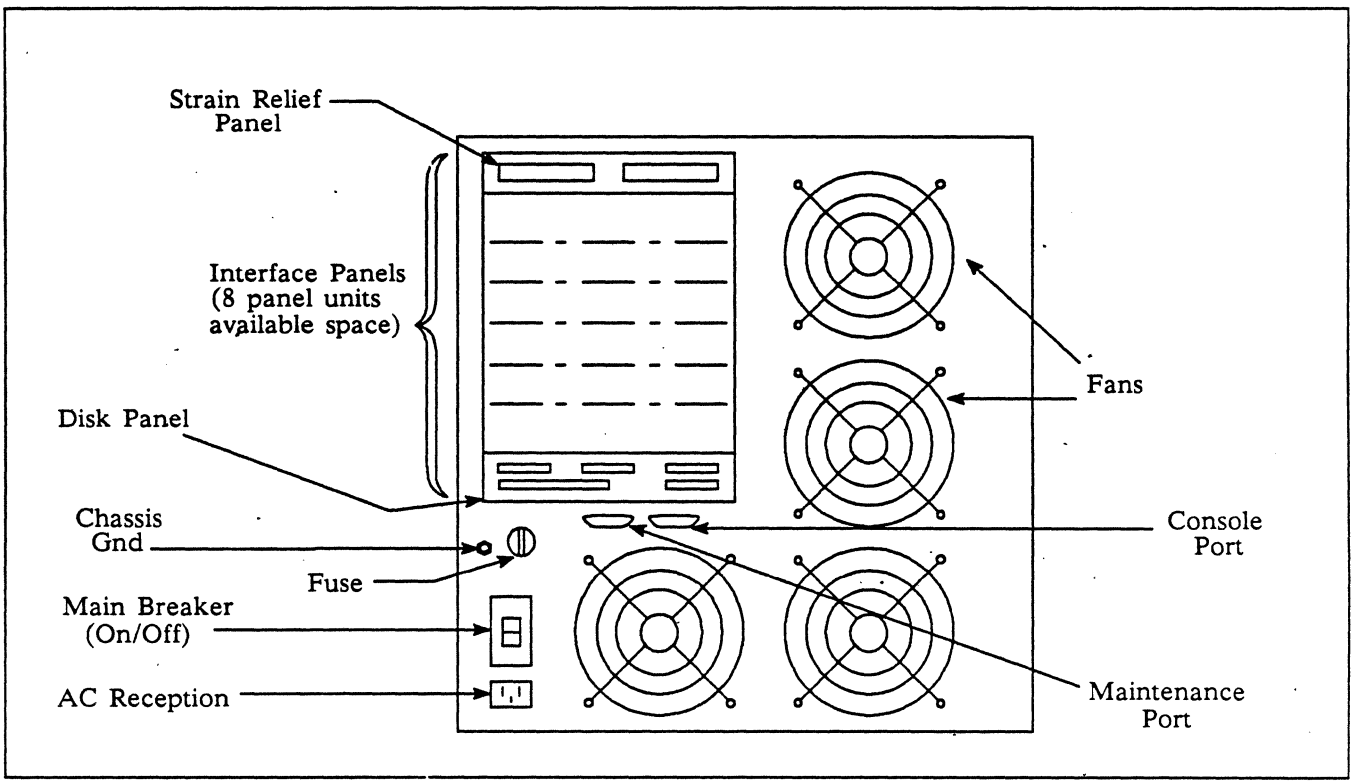


Figure F-1: Pre-FCC Rear Panel

FCC REAR PANEL

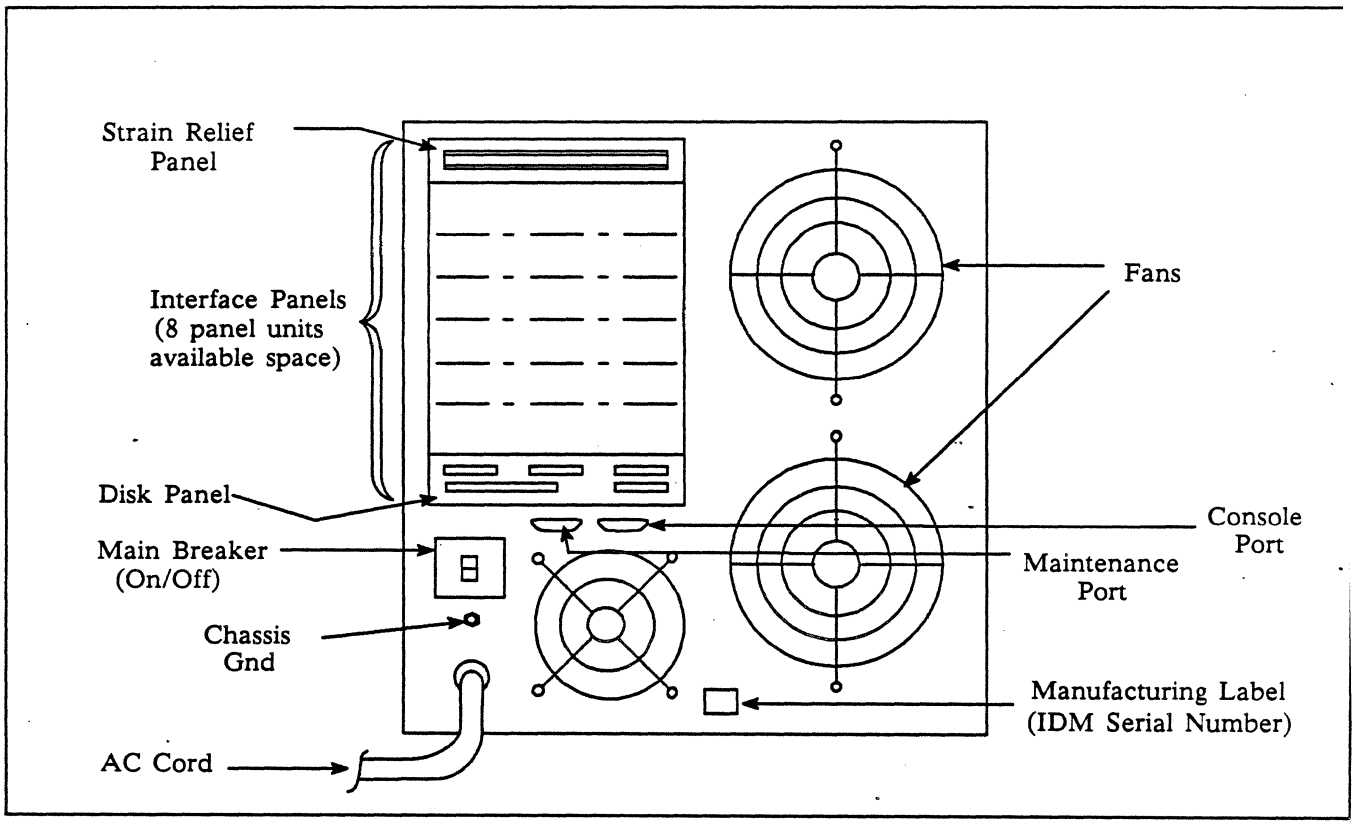
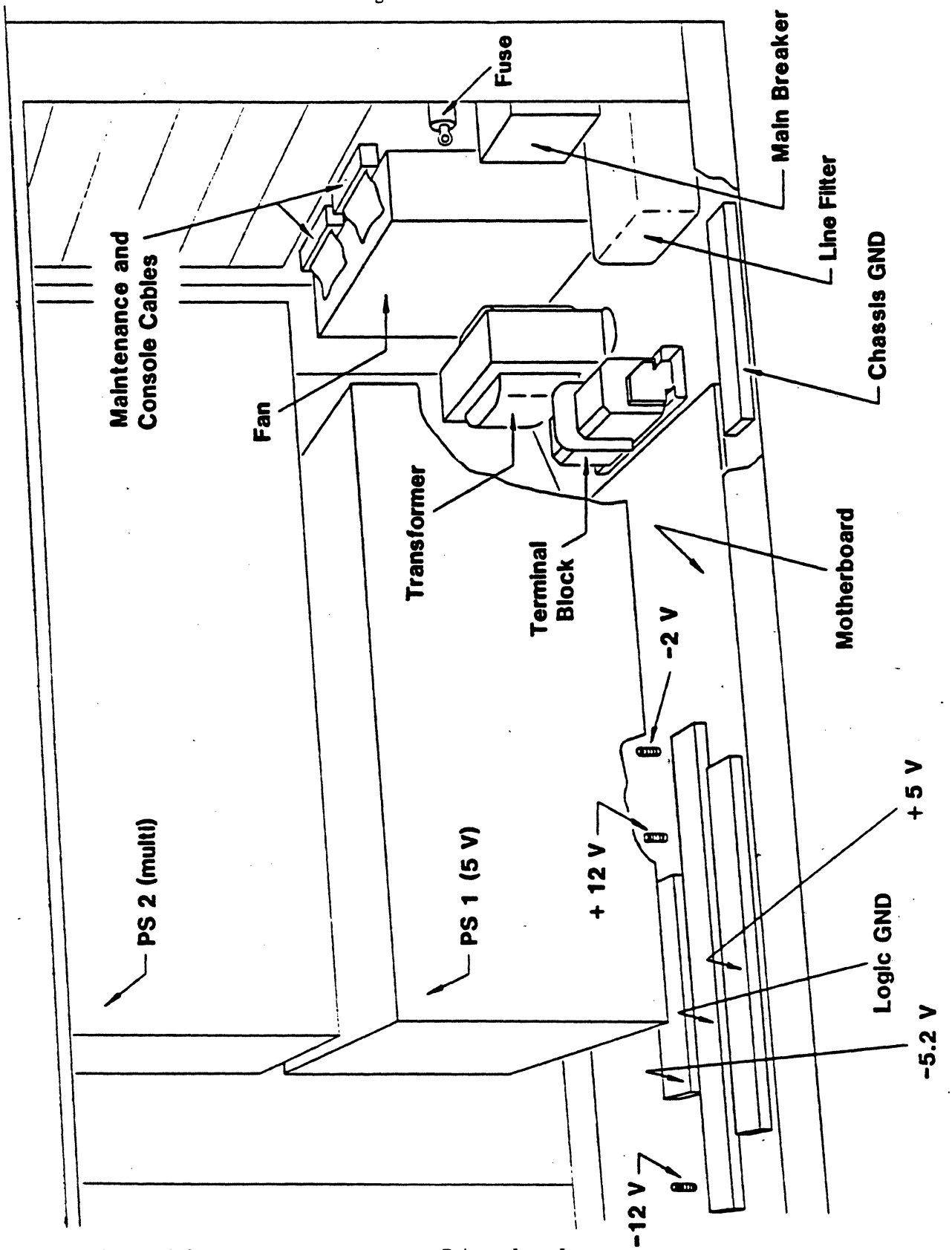


Figure F-2: FCC Rear Panel

PRE-FCC AREAS

PRE-FCC POWER AREA

Figure F-3: Pre-FCC Power Area



PRE-FCC ELECTRICAL CAUTION AREAS

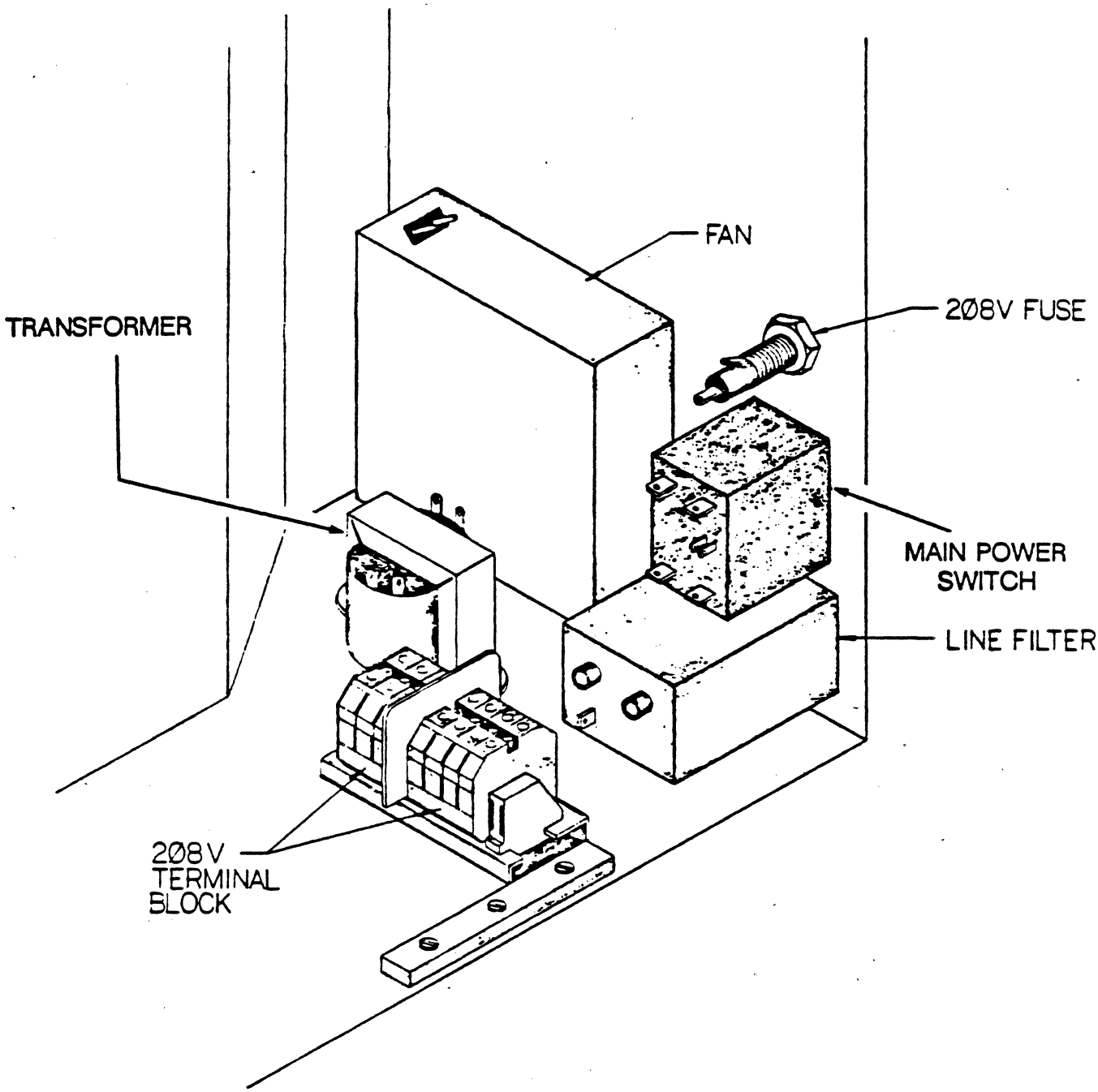


Figure F-4: Pre-FCC Electrical Caution Areas

PRE-FCC AC/DC WIRE LIST

WIRENUM	POINT A	SIGNAL	POINT B
0	TB2-AC(1)	AC	PCB FAN3 AC
0	TB2-AC(2)	AC	PCB FAN2 AC
0	TB2-AC(4)	AC	PCB FAN1 AC
0	TB2-ACC(5)	ACC	PCB FAN2 ACC
0	TB2-ACC(6)	ACC	PCB FAN3 ACC
0	TB2-ACC(8)	ACC	PCB FAN1 ACC
1	PLUG (L)	LOAD	CKTBKR LINE
2	CKTBKR LOAD	AC	TB1- AC(3)
3	CKTBKR LOAD	JUMPER	CKTBKR (3)
4	FUSE	AC	TB1-AC(4)
5	T1-1	AC	FUSE
6	PS2- AC(6)	AC	TB1-AC(9)
7	PS1- AC(6)	AC	TB1- AC(6)
8	TB1-AC(2)	AC	PS FAN AC
9	TBI-1	AC	TB2-AC(4)
10	PLUG (N)	LINE	CKTBKR LINE
11	CKTBKR LOAD	ACC	TB1-ACC(11)
12	PS1-ACC(7)	ACC	TBI-ACC(13)
14	PS2-ACC(7)	ACC	TB1-ACC(15)
15	T1-2	ACC	TB1-ACC(4)
16	TB1-ACC(10)	ACC	PS FAN4 ACC
17	TB1-ACC(7)	ACC	TB2-ACC(8)
18	T1-3	+VS	MOLEX-1
19	T1-4	CT	MOLEX-2
20	T1-5	-VS	MOLEX-3
21	PLUG (E)	GND	TB3-GND(2)
21	PLUG (E)	GND	TB3-GND(2)
22	T1-5	GND	TB3-GND(5)
23	PS1-GND	GND	TB3-GND(1)
24	PS2-GND	GND	TB3-GND(3)
25	MTB GND BAR	GND	TB3-GND(3)
26	PS FAN4 GND	GND	TB3-GND(4)
27	PCB FAN3 GND	GND	TB3-GND(4)
28	PCB FAN1 GND	GND	PCB FAN2 GND
29	PCB FAN2 GND	GND	PCB FAN3 GND
31	PS1-0N/OFF	ON/OFF	MOLEX-4
32	PS2- -S1(1)	-SENSE	MTB- -5.2V
33	PS2- +S1(2)	+SENSE	MTB-GND BAR
34	PS2- +VI	JUMPER	PS2- -V2(3)
35	PS2- +V1	GND	MTB-GND BAR
36	PS2- -V1	-5.2V	MTB- -5.2V
37	PS2- +V3(5)	JUMPER	PS2- -V2(3)
38	PS2- -V3(6)	-2V	MTB- -2V
39	PS2- +V2(2)	+12V	MTB- +12V
40	PS2- +V3(5)	JUMPER	PS2- +V4(7)
41	PS2- -V4(8)	-12V	MTB- -12V
42	PS1- +(POS)	+5V	MTB- 5V BAR
43	PS1- -(NEG)	GND	MTB-GND BAR
44	PS1- -S(2)	-SENSE	MTB-GND BAR
45	PS2- +S(1)	+SENSE	MTB- +5V
51	TM-11 -S	-5.2V	MTB- -5.2V
52	TM-11 +S	+SENSE	MTB- GND BAR
53	TM-11+-S GND	SHIELD	PS1-CHAS.GND
54	TM-11 GND	GND	PS1-GND
55	TM-11-ACC(2)	ACC	PS1-ACC(7)
56	TM-11-AC(1)	AC	PS1-AC(6)
57	TM-11 +V(6)	GND	PS2- +V1
58	TM-11 -V(5)	-5.2V	PS2- -V1
60	TM-11 ON/OFF	ON/OFF	PS2- ON/OFF

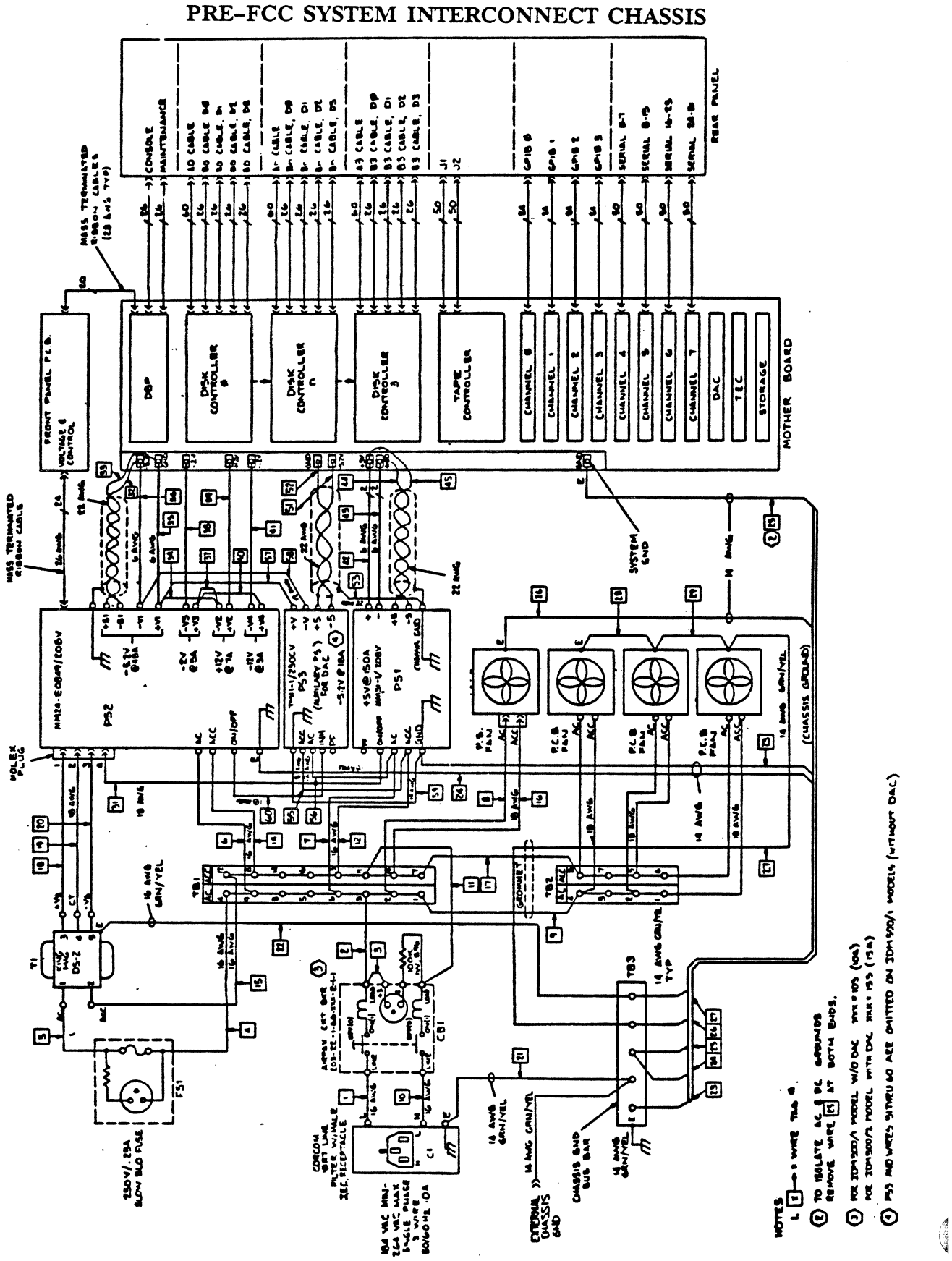


Figure F-5: Pre-FCC System Interconnect Chassis

POWER SUPPLIES

A fully configured pre-FCC Britton Lee database server may contain three power supplies PS1, PS2 (multi supply) and bulk supply.

NOTE

The input voltage of some power supplies may be 115V ac. Voltage input is specified on the front of the power supply.

PS2

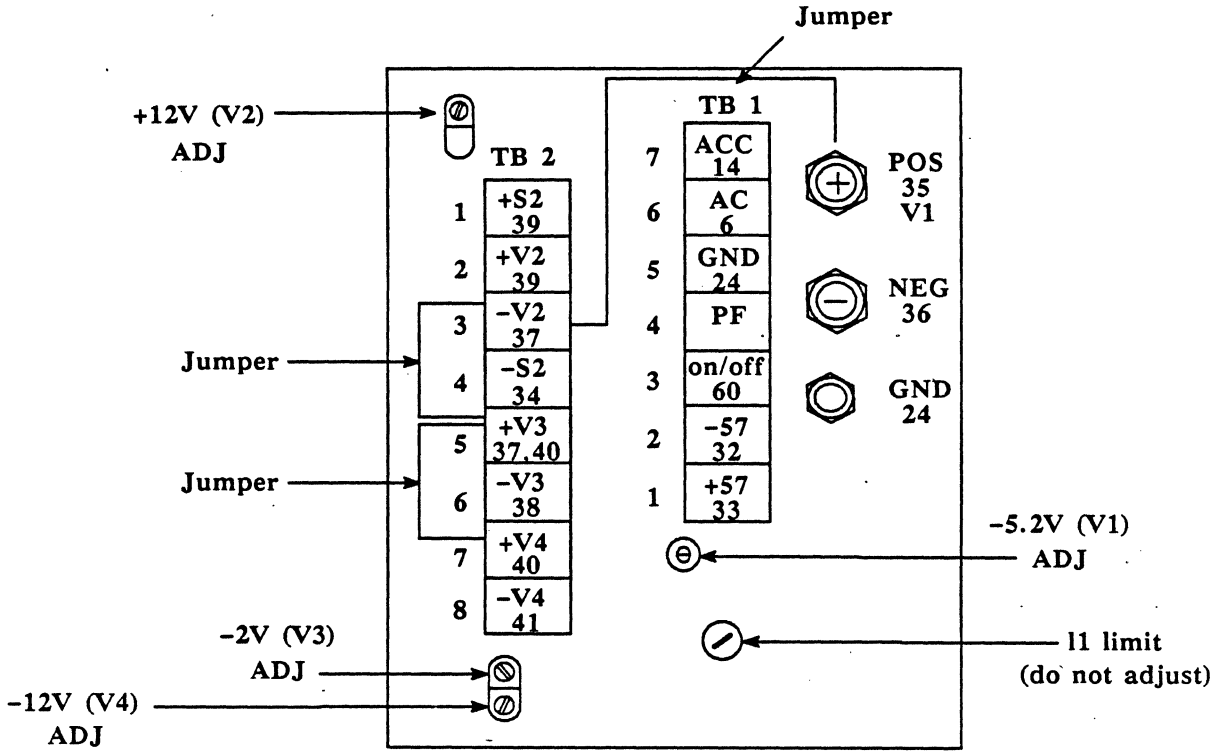


Figure F-6: PS2

PS1

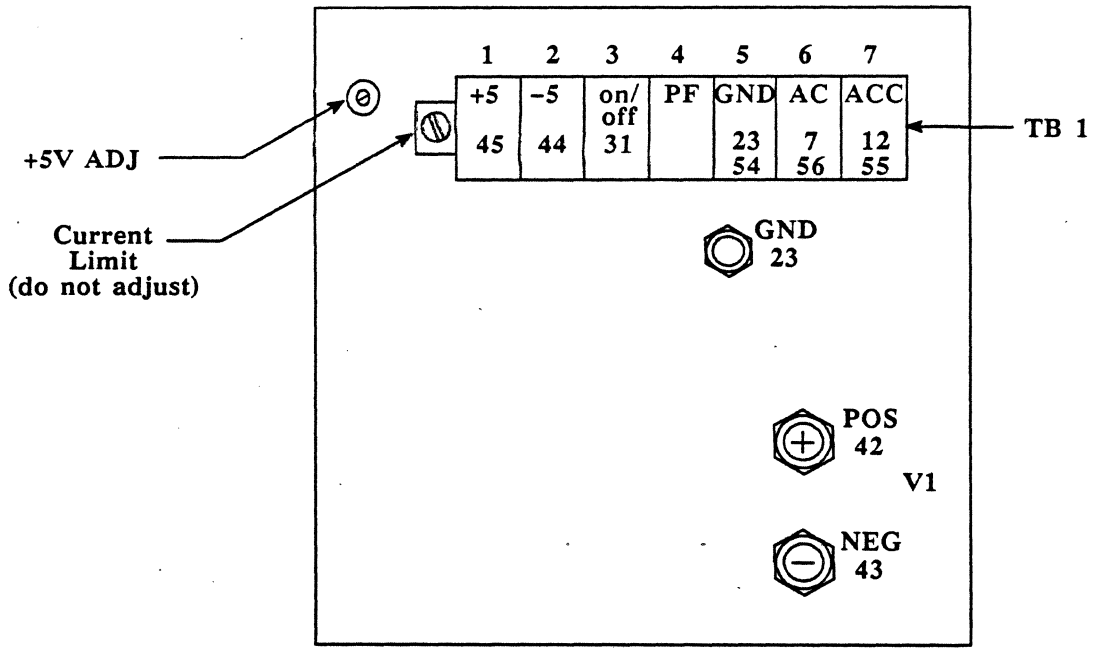


Figure F-7: PS1

PRE-FCC TM 11 WIRE POSITIONS

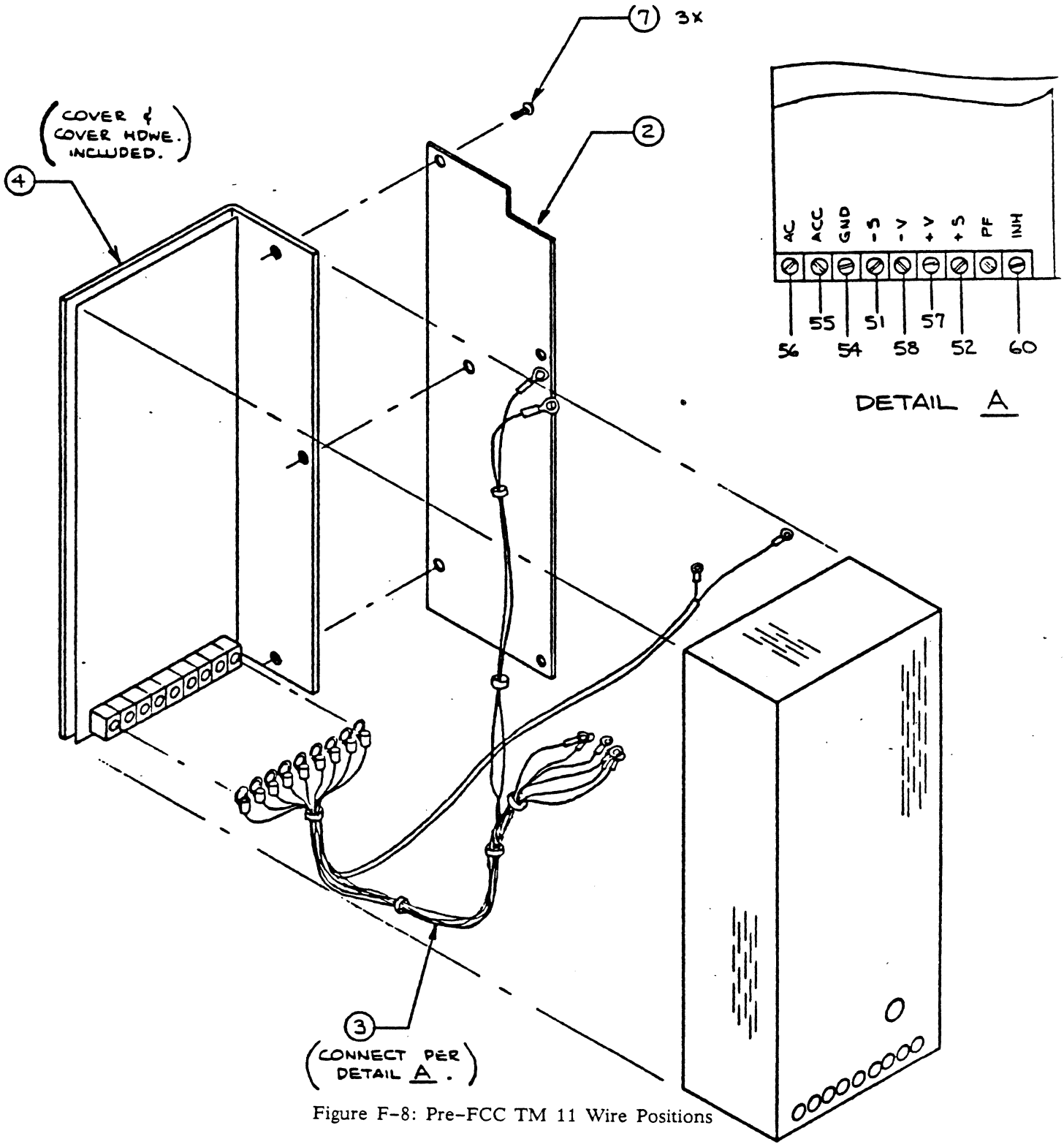


Figure F-8: Pre-FCC TM 11 Wire Positions

PRE-FCC TM 11 HOOK-UP

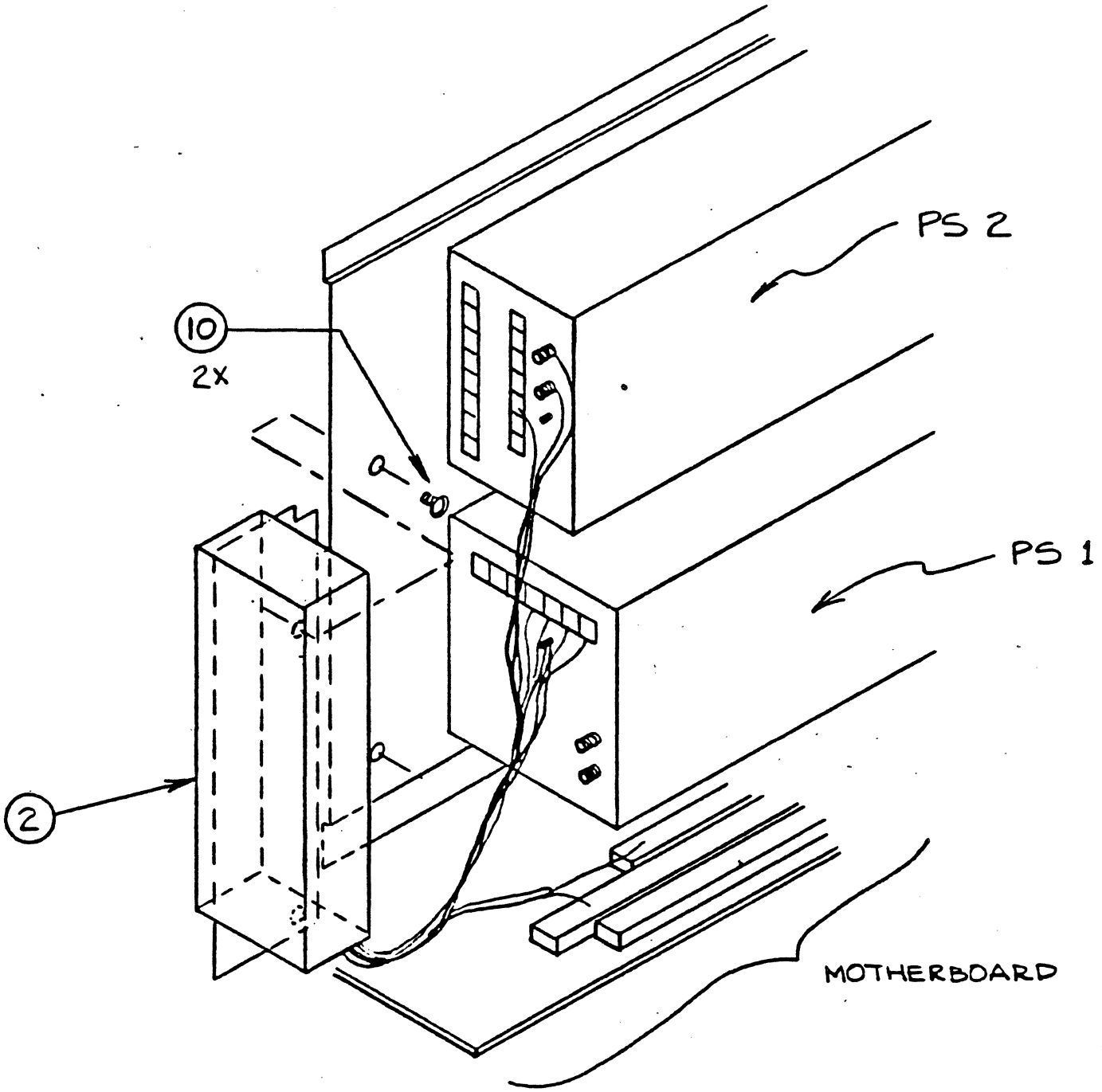


Figure F-9: Pre-FCC TM 11 Hook-Up

PRE-FCC TM 11 HOOK-UP

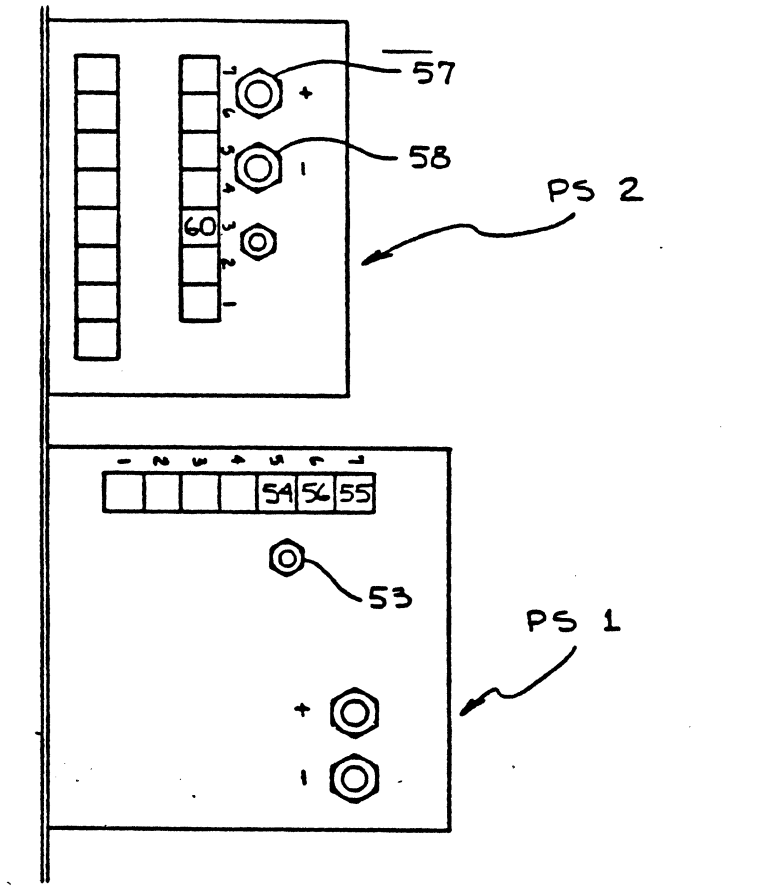
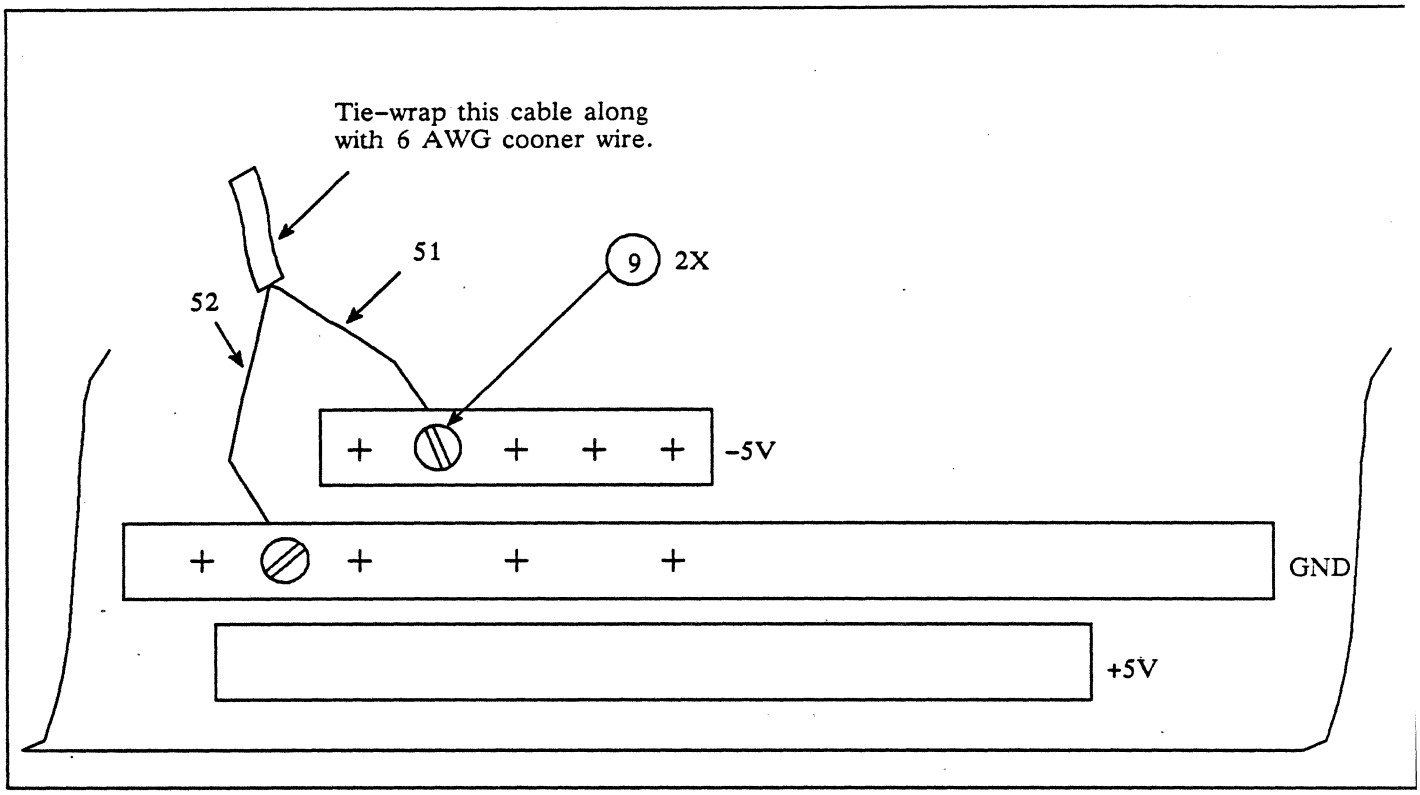


Figure F-10: Pre-FCC TM 11 Hook-up

PRE-FCC TM 11 HOOK-UP



MOTHERBOARD CONNECTIONS
(Detail B)

NOTES:
1. Numbers in circles are items on BOM (138-0553-001).
2. Numbers uncircled are wire numbers from DAC power supply harness.
3. This drawing documents Britton Lee part number 118-0553-001.

Figure F-11: Pre-FCC TM 11 Hook-Up

FRONT PANEL

The front panel of the pre-FCC models have the same functional objective. However, the FCC model has a design change and an added feature, therefore, the front panel is NOT interchangeable between models.

Pre-FCC	FCC
Indicator lights are POWER, READY, SAFE, SERVICE, PWR FAIL, and FAULT	PWR FAIL indicator was replaced by PS 1 FAIL and PS 2 FAIL.
There is no T FAULT indicator.	It has a T FAULT (temperature fault) indicator, which warns the operator if the temperature in the Britton Lee database server rises above the allowable level.

PRE-FCC FRONT PANEL

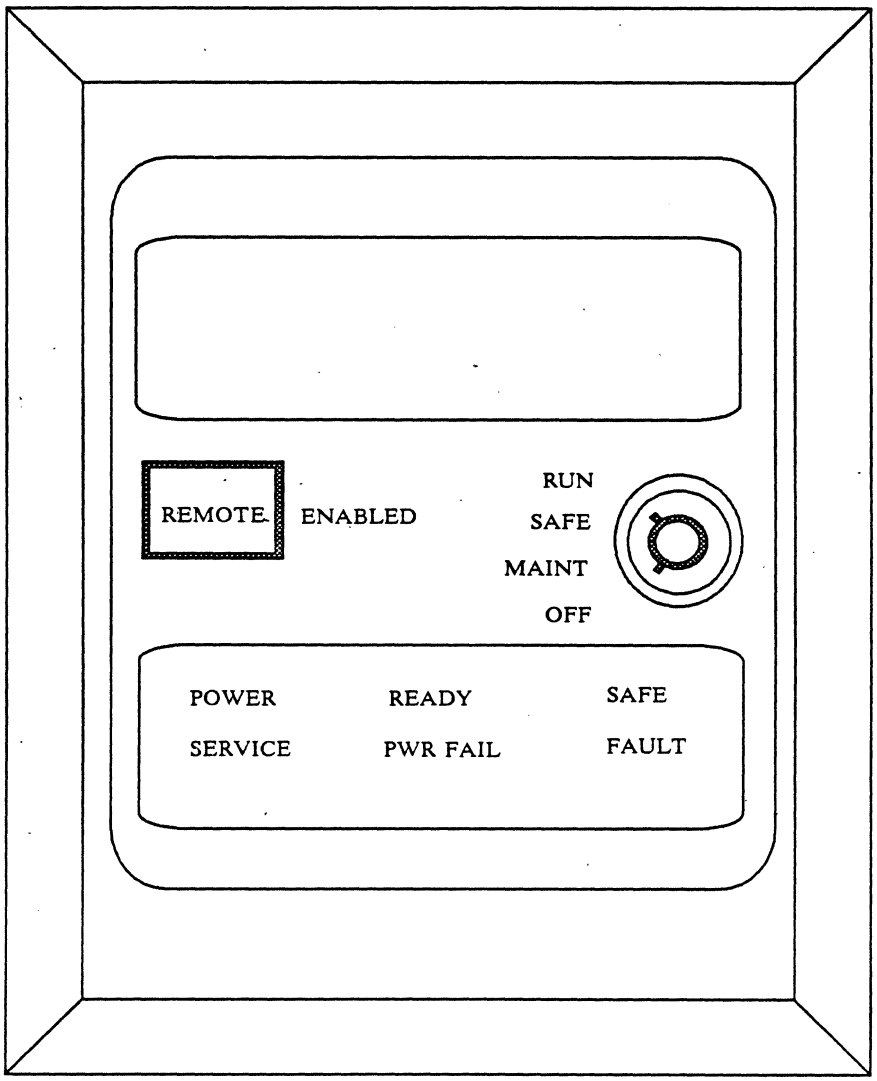


Figure F-12: Pre-FCC Front Panel

MOTHERBOARD

There are two versions of the BL700 motherboard, pre-FCC and FCC compliant. These boards are NOT interchangeable! The distinguishable differences are:

Pre-FCC	FCC
Resistor modules (RM) are solder mounted	RM's are socket mounted
ECL RM = 100 ohms	ECL RM = 160/260 ohms
Utilizes and has a test point for -2V dc (used as reference by ECL RM)	Eliminated
Must install a TM 11 power booster and adjust the -5.2V dc up to -5.35V dc if adding a data accelerator option.	Not needed -5.2V dc remains at -5.2V dc

Although there are marked differences between pre-FCC and FCC motherboards, these differences are transparent to the host computer and the users.

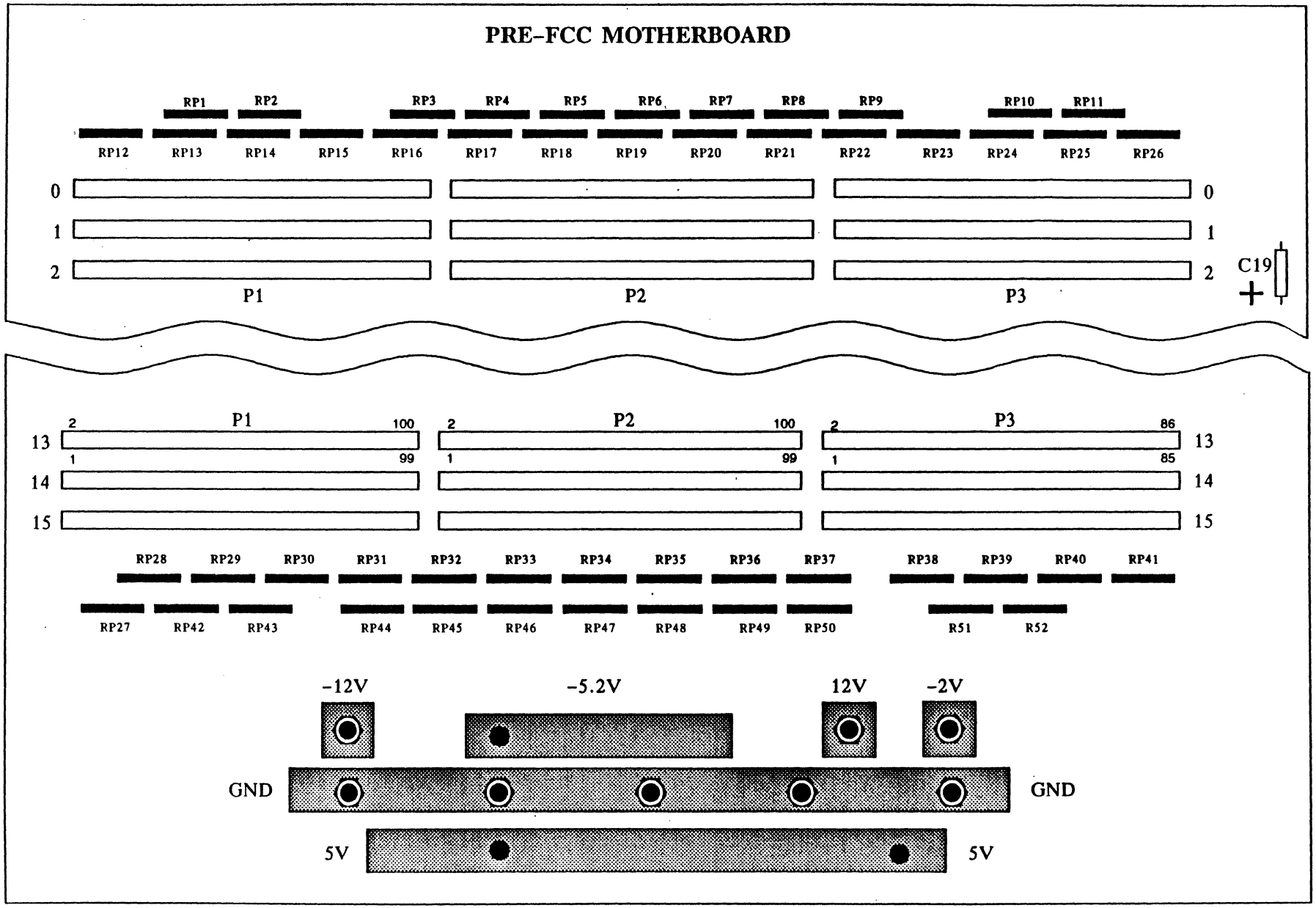


Figure F-13: Motherboard

PRE-FCC MOTHERBOARD SLOT WIRING
(Refer to Pre-FCC Motherboard Figure F-13)

BL700	SIGNAL	SIGNAL TYPE	TERMINATION	
P1-01	Ground	GND		
P1-02	Ground	GND		
P1-03	Vcc (+5v)	PWR		
P1-04	Vcc (+5v)	PWR		
P1-05	Vcc (+5v)	PWR		
P1-06	Vcc (+5v)	PWR		
P1-07	Vcc (+5v)	PWR		
P1-08	Vcc (+5v)	PWR		
P1-09	Ground	GND		
P1-10	Ground	GND		
P1-11	Vdd (+12v)	PWR		
P1-12	Vdd (+12v)	PWR		
P1-13	Ground	GND		
P1-14	Ground	GND		
P1-15	Vff (-12v)	PWR		
P1-16	Vff (-12v)	PWR		
P1-17	Vee (-5.2v)	PWR		
P1-18	Vee (-5.2v)	PWR		
P1-19	Vee (-5.2v)	PWR		
P1-20	Vee (-5.2v)	PWR		
P1-21	Ground	GND		
P1-22	Ground	GND		
P1-23	+IBO3	ECL	RP12-09	RP27-04
P1-24	+IB02	ECL	RP12-09	RP27-04
P1-25	+IB01	ECL	RP12-09	RP27-04
P1-26	+IB00	ECL	RP12-09	RP27-04
P1-27	+IB06	ECL	RP12-09	RP27-04
P1-28	+IB07	ECL	RP12-09	RP27-04
P1-29	+IB04	ECL	RP12-09	RP27-04
P1-30	+IB05	ECL	RP12-09	RP27-04
P1-31	Ground	GND		
P1-32	Ground	GND		
P1-33	+IB19	ECL	RP12-09	RP27-04
P1-34	+IB18	ECL	RP12-09	RP27-04
P1-35	+IB17	ECL	RP12-09	RP27-04
P1-36	+IB16	ECL	RP12-09	RP27-04
P1-37	+IB22	ECL	RP12-09	RP27-04
P1-38	+IB23	ECL	RP12-09	RP27-04
P1-39	+IB20	ECL	RP12-09	RP27-04
P1-40	+IB21	ECL	RP12-09	RP27-04
P1-41	Ground	GND		
P1-42	Ground	GND		
P1-43	+IB35	ECL	RP12-09	RP27-04
P1-44	+IB34	ECL	RP12-09	RP27-04
P1-45	+IB33	ECL	RP12-09	RP27-04
P1-46	+IB32	ECL	RP12-09	RP27-04
P1-47	+IB38	ECL	RP12-09	RP27-04
P1-48	+IB39	ECL	RP12-09	RP27-04
P1-49	+IB36	ECL	RP12-09	RP27-04
P1-50	+IB37	ECL	RP12-09	RP27-04
P1-51	Ground	GND		
P1-52	Ground	GND		

BL700	SIGNAL	SIGNAL TYPE	TERMINATION	
P1-53	+IB51	ECL	RP14-03	RP42-09
P1-54	+IB50	ECL	RP02-05	RP29-06
P1-55	+IB50	ECL	RP14-04	RP42-10
P1-56	+IB50	ECL	RP14-05	RP29-07
P1-57	+IB50	ECL	RP14-06	RP29-08
P1-58	+IB50		RP02-06	RP43-02
P1-59	+IB50	ECL	RP14-07	RP29-09
P1-60	+IB50	ECL	RP02-07	RP43-03
P1-61	Ground	GND		
P1-62	Ground	GND		
P1-63	+IB50	ECL	RP12-09	RP27-04
P1-64	+IB50	ECL	RP12-09	RP27-04
P1-65	+IB50	ECL	RP12-09	RP27-04
P1-66	+IB50	ECL	RP12-09	RP27-04
P1-67	Ground	GND		
P1-68	Ground	GND		
P1-69	SPARE	ECL	RP12-09	RP27-04
P1-70	SPARE	ECL	RP12-09	RP27-04
P1-71	SPARE	ECL	RP12-09	RP27-04
P1-72	SPARE	ECL	RP12-09	RP27-04
P1-73	SPARE	ECL	RP12-09	RP27-04
P1-74	SPARE	ECL	RP12-09	RP27-04
P1-75	+ENDO	ECL	RP12-09	RP27-04
P1-76	+IMWS	ECL	RP12-09	RP27-04
P1-77	Ground	GND		
P1-78	Ground	GND		
P1-79	Vee (-5.2v)	PWR		
P1-80	Vee (-5.2v)	PWR		
P1-81	Vee (-5.2v)	PWR		
P1-82	Vee (-5.2v)	PWR		
P1-83	Ground	GND		
P1-84	Ground	GND		
P1-85	+MREQ	ECL	RP12-09	RP27-04
P1-86	+AACK	ECL	RP12-09	RP27-04
P1-87	+MACK	ECL	RP12-09	RP27-04
P1-88	+EREF	ECL	RP12-09	RP27-04
P1-89	Ground	GND		
P1-90	Ground	GND		
P1-91	+RSTB	ECL	1209	RP27-04
P1-92	+WSTB	ECL	1209	RP27-04
P1-93	SPARE	ECL	1209	RP27-04
P1-94	+SMTH	ECL	1209	RP27-04
P1-95	Ground	GND		
P1-96	Ground	GND		
P1-97	+IORC	ECL	RP12-09	RP27-04
P1-98	+IOAK	ECL	RP12-09	RP27-04
P1-99	+MPLD	ECL	RP12-09	RP27-04
P1-100	+MCLK	ECL	RP12-09	RP27-04

BL700	SIGNAL	SIGNAL TYPE	TERMINATION	
P2-01	SLAO	ECL	RP12-09	SP27-04
P2-02	SLAO	ECL	RP12-09	SP27-04
P2-03	SLAO	ECL	RP12-09	SP27-04
P2-04	SLAO	ECL	RP12-09	SP27-04
P2-05	-HFLT	ECL	RP12-09	SP27-04
P2-06	-MRST	ECL	RP12-09	SP27-04
P2-07	-BREQ	ECL	RP12-09	SP27-04
P2-08	-BGNT	ECL	RP12-09	SP27-04
P2-09	Ground	GND		
P2-10	Ground	GND		
P2-11	-CREQ	ECL	RP12-09	SP27-04
P2-12	-CGNT	ECL	RP12-09	SP27-04
P2-13	-CARB	ECL	RP12-09	SP27-04
P2-14	-ACLO	ECL	RP12-09	SP27-04
P2-15	-CRB0	ECL	RP12-09	SP27-04
P2-16	-CRB1	ECL	RP12-09	SP27-04
P2-17	-CRB2	ECL	RP12-09	SP27-04
P2-18	-CRB3	ECL	RP12-09	SP27-04
P2-19	Ground	GND		
P2-20	Ground	GND		
P2-21	-CINT	ECL	RP12-09	SP27-04
P2-22	-CACK	ECL	RP12-09	SP27-04
P2-23	-BXFR	ECL	RP12-09	SP27-04
P2-24	-RSNC	ECL	RP12-09	SP27-04
P2-25	SPARE	ECL	RP12-09	SP27-04
P2-26	-BPER	ECL	RP12-09	SP27-04
P2-27	-MST1	ECL	RP12-09	SP27-04
P2-28	-MST2	TTL	RP18-05	RP33-05
P2-29	Ground	GND		
P2-30	Ground	GND		
P2-31	-MPDS	TTL	RP05-06	RP46-06
P2-32	-BYTE	TTL	RP18-06	RP33-06
P2-33	-WRITE	TTL	RP05-07	RP46-07
P2-34	-QUAD	TTL	RP18-07	RP33-07
P2-35	-DBP0	TTL	RP03-05	RP44-05
P2-36	-DBP0	TTL	RP03-04	RP44-04
P2-37	-DB00	TTL	RP03-04	RP44-03
P2-38	-DB01	TTL	RP03-02	RP44-02
P2-39	Ground	GND		
P2-40	Ground	GND		
P2-41	-DB02	ECL	RP12-09	SP27-04
P2-42	-DB03	ECL	RP12-09	SP27-04
P2-43	-DB04	ECL	RP12-09	SP27-04
P2-44	-DB05	ECL	RP12-09	SP27-04
P2-45	-DB06	ECL	RP12-09	SP27-04
P2-46	-DB07	ECL	RP12-09	SP27-04
P2-47	-DB08	ECL	RP12-09	SP27-04
P2-48	-DB09	ECL	RP12-09	SP27-04
P2-49	Ground	GND		
P2-50	Ground	GND		
P2-51	-DB10	ECL	RP12-09	SP27-04
P2-52	-DB11	ECL	RP12-09	SP27-04
P2-53	-DB12	ECL	RP12-09	SP27-04
P2-54	-DB13	ECL	RP12-09	SP27-04
P2-55	-DB14	ECL	RP12-09	SP27-04
P2-56	-DB15	ECL	RP12-09	SP27-04
P2-57	-BAP0	ECL	RP12-09	SP27-04
P2-58	-BAP1	ECL	RP12-09	SP27-04
P2-59	Ground	GND		
P2-60	Ground	GND		
P2-61	-BAP2	ECL	RP12-09	SP27-04

BL700	SIGNAL	SIGNAL TYPE	TERMINATION	
P2-62	-BAP3	ECL	RP12-09	SP27-04
P2-63	-BA00	ECL	RP12-09	SP27-04
P2-64	-BA01	ECL	RP12-09	SP27-04
P2-65	-BA02	ECL	RP12-09	SP27-04
P2-66	-BA03	ECL	RP12-09	SP27-04
P2-67	-BA04	ECL	RP12-09	SP27-04
P2-68	-BA05	ECL	RP12-09	SP27-04
P2-69	Ground	GND		
P2-70	Ground	GND		
P2-71	-BA06	ECL	RP12-09	SP27-04
P2-72	-BA07	ECL	RP12-09	SP27-04
P2-73	-BA08	ECL	RP12-09	SP27-04
P2-74	-BA09	ECL	RP12-09	SP27-04
P2-75	-BA10	ECL	RP12-09	SP27-04
P2-76	-BA11	ECL	RP12-09	SP27-04
P2-77	-BA12	ECL	RP12-09	SP27-04
P2-78	-BA13	ECL	RP12-09	SP27-04
P2-79	Ground	GND		
P2-80	Ground	GND		
P2-81	-BA14	ECL	RP12-09	SP27-04
P2-82	-BA15	ECL	RP12-09	SP27-04
P2-83	-BA16	ECL	RP12-09	SP27-04
P2-84	-BA17	ECL	RP12-09	SP27-04
P2-85	-BA18	ECL	RP12-09	SP27-04
P2-86	-BA19	ECL	RP12-09	SP27-04
P2-87	-BA20	ECL	RP12-09	SP27-04
P2-88	-BA21	ECL	RP12-09	SP27-04
P2-89	Ground	GND		
P2-90	Ground	GND		
P2-91	-BA22	ECL	RP12-09	SP27-04
P2-92	-BA23	ECL	RP12-09	SP27-04
P2-93	-BA24	ECL	RP12-09	SP27-04
P2-94	-BA25	ECL	RP12-09	SP27-04
P2-95	-BA26	ECL	RP12-09	SP27-04
P2-96	-BA27	ECL	RP12-09	SP27-04
P2-97	-BA28	ECL	RP12-09	SP27-04
P2-98	SPARE	ECL	RP12-09	SP27-04
P2-99	Ground	GND		
P2-100	Ground	GND		

BL700	SIGNAL	SIGNAL TYPE	TERMINATION	
P3-01	Vcc (+5v)	PWR		
P3-02	Vcc (+5v)	PWR		
P3-03	Vcc (+5v)	PWR		
P3-04	Vcc (+5v)	PWR		
P3-05	Vcc (+5v)	PWR		
P3-06	Vcc (+5v)	PWR		
P3-07	Vcc (+5v)	PWR		
P3-08	Vcc (+5v)	PWR		
P3-09	Ground	GND		
P3-10	Ground	GND		
P3-11	+RA0	ECL	RP12-09	SP27-04
P3-12	+RA1	ECL	RP12-09	SP27-04
P3-13	+RA2	ECL	RP12-09	SP27-04
P3-14	+RA3	ECL	RP12-09	SP27-04
P3-15	+RA4	ECL	RP12-09	SP27-04
P3-16	+RA5	ECL	RP12-09	SP27-04
P3-17	+RA6	ECL	RP12-09	SP27-04
P3-18	+RA7	ECL	RP12-09	SP27-04
P3-19	Ground	GND		
P3-20	Ground	GND		
P3-21	+IB71	ECL	RP12-09	SP27-04
P3-22	+IB69	ECL	RP12-09	SP27-04
P3-23	+IB70	ECL	RP12-09	SP27-04
P3-24	+IB68	ECL	RP12-09	SP27-04
P3-25	Ground	GND		
P3-26	Ground	GND		
P3-27	+IB59	ECL	RP12-09	SP27-04
P3-28	+IB58	ECL	RP12-09	SP27-04
P3-29	+IB57	ECL	RP12-09	SP27-04
P3-30	+IB56	ECL	RP12-09	SP27-04
P3-31	+IB62	ECL	RP12-09	SP27-04
P3-32	+IB63	ECL	RP12-09	SP27-04
P3-33	+IB60	ECL	RP12-09	SP27-04
P3-34	+IB61	ECL	RP12-09	SP27-04
P3-35	Ground	GND		
P3-36	Ground	GND		
P3-37	+IB43	ECL	RP12-09	SP27-04
P3-38	+IB42	ECL	RP12-09	SP27-04
P3-39	+IB41	ECL	RP12-09	SP27-04
P3-40	+IB40	ECL	RP12-09	SP27-04
P3-41	+IB46	ECL	RP12-09	SP27-04
P3-42	+IB47	ECL	RP12-09	SP27-04
P3-43	+IB44	ECL	RP12-09	SP27-04
P3-44	+IB45	ECL	RP12-09	SP27-04
P3-45	Ground	GND		
P3-46	Ground	GND		
P3-47	+IB27	ECL	RP12-09	SP27-04
P3-48	+IB26	ECL	RP12-09	SP27-04
P3-49	+IB25	ECL	RP12-09	SP27-04
P3-50	+IB24	ECL	RP12-09	SP27-04
P3-51	+IB30	ECL	RP12-09	SP27-04
P3-52	+IB31	ECL	RP12-09	SP27-04
P3-53	+IB28	ECL	RP12-09	SP27-04
P3-54	+IB29	ECL	RP12-09	SP27-04
P3-55	Ground	GND		
P3-56	Ground	GND		
P3-57	+IB11	ECL	RP12-09	SP27-04
P3-58	+IB10	ECL	RP12-09	SP27-04
P3-59	+IB09	ECL	RP12-09	SP27-04
P3-60	+IB08	ECL	RP12-09	SP27-04
P3-61	+IB14	ECL	RP12-09	SP27-04
P3-62	+IB15	ECL	RP12-09	SP27-04

BL700	SIGNAL	SIGNAL TYPE	TERMINATION	
P3-63	+IB12	ECL	RP12-09	SP27-04
P3-64	+IB13	ECL	RP12-09	SP27-04
P3-65	Ground	GND		
P3-66	Ground	GND		
P3-67	Vee (-5.2v)	PWR		
P3-68	Vee (-5.2v)	PWR		
P3-69	Vee (-5.2v)	PWR		
P3-70	Vee (-5.2v)	PWR		
P3-71	Vee (-5.2v)	PWR		
P3-72	Vee (-5.2v)	PWR		
P3-73	Ground	GND		
P3-74	Ground	GND		
P3-75	Vdd (+12v)	PWR		
P3-76	Vdd (+12v)	PWR		
P3-77	Ground	GND		
P3-78	Ground	GND		
P3-79	Vcc(+5v)	PWR		
P3-80	Vcc(+5v)	PWR		
P3-81	Vcc(+5v)	PWR		
P3-82	Vcc(+5v)	PWR		
P3-83	Vcc(+5v)	PWR		
P3-84	Vcc(+5v)	PWR		
P3-85	Ground	GND		
P3-86	Ground	GND		

APPENDIX G: ERROR CODES

Trap Messages

Disk Error Codes

Tape Error Codes

Data Accelerator Errors

Channel Syserr Messages

Tape Controller Loopback Error Codes

Channel Syserrs for Blockmux

Channel Syserrs for Ethernet

Disk Formatting Utility Error Messages

CKDB Error Messages

TRAP MESSAGES

The word "trap" means error interrupt. Error interrupts are associated with the Zilog Z8002 mpu. Unrecoverable errors will cause a trap message.

This is an example of a blank trap message.

1. trap___**2.** sp ____,**3.** pgm_,**4.** trapid =____,**5.** fcw =____,
6. pc =____,**7.** virt addr =____ **8.** slot _,**9.** phypage ___,
10. I/O addr_____,**11.** busmaster_____,**12.** err:_____.

The numbers indicated in bold are not present in a real trap message. These numbers are intended to help you reference an explanation of each component in a trap message. Item 10, I/O addr is only present as a result of an I/O error.

1. TRAP: A sequence number of traps that have occurred on this fault.

2. SP: Stack Pointer. This is the Z8000 stack pointer before the trap occurred. The stack is either system (system information) or normal (application program information). The stack pointers are part of the 6P register.

3. PGM: BL700 program number (program active at time of error).

0:front	4:dumpload
1:qryproc	5:compile
2:support	6:execute
3:sort	7:recovery

4. TRAP ID: Trap id defines the bus master and error which is later given on the trap.

5. FCW: Z8002 Flag and Control Word (see Zilog Z8002 Data Book)

6. PC: Program Counter.
Shows within 3 instructions what was being executed at the time of the failure. The PC is important in many instances for BLI troubleshooting.

7. VIRT ADDR: Virtual Address.
The BL700 addresses memory in two modes:
Virtual and Physical. Virtual Addressing uses the DBP Memory Management Unit (MMU). Channels, DBP's, and DAC's only address memory in virtual mode. Disk and tape controllers address memory in both virtual mode (command fetch and status port update) and physical mode (DMA transfers to and from memory).

8. SLOT: The BL700 slot number being addressed at the time of failure, however not necessarily the defective board. The slot number is the failing board if the trap is an uncorrectable memory error.

9. PHY PAGE: Physical page (address) in memory.
The DBP's memory management unit (MMU) maps virtual addresses into physical references within memory. However, disk and tape controllers transfer data in DMA to and from memory in physical mode.

10. I/O ADDR:
Present only if the error occurred during an I/O operation for board communication on the bus. See I/O timeout for more information.

11. BUS MASTER:
The Bus Master is the board accessing memory (either virtual or physical) at the time of failure. It does not indicate the failing board in many situations, yet can be useful in determining the cause.

EXAMPLE:

```
busmaster; DBP text, err: HFLT with a pgm=4
```

Pgm 4 is the BL700 program DUMP, used to dump databases. If the dumping was through an BL700 tape controller, the error may have happened when the tape controller or disk controller was reading a virtual address from its command port or updating the status port when a command is completed. In this example, TPC, DC (where the database was being dumped from), DBP, and Timing and Control (supplies bus timing) are under suspicion. If the dump was through an BL700 channel, that channel would also be a suspect.

12. ERR:
Item 12 is in "abbreviated" English to help you find out what is wrong and how to correct it. The error statement is listed in the left column, the error description is listed in the middle column and the probable causes are in the right column. The probable cause is given in the order of suspected failure. Probable cause numbers are defined following this list.

ERROR	DESCRIPTION	PROBABLE CAUSE
ABORT ACCESS	Memory reference to a portion of an MMU window marked abort access.	11, 9
READ ONLY	There was an attempt to WRITE a READ ONLY portion of an MMU window.	
STACK LIMIT	A selected MMU window's stack has grown into the lower 512 bytes of a memory block marked "low trap".	
MEMORY TIMEOUT	A memory request to a slot timed out. Either the slot missed the request or the slot's acknowledgment was missed by the requestor. A memory request could have been to an empty slot or a board with no memory. Cards that generate a memory request: DBP DC TPC DAC Cards that generate a memory acknowledge: Channels T&C	5, 6, 1, 9, 13 11, 12, 9
I/O TIMEOUT	BL700 boards communicate to each other through I/O request and acknowledge. Assume the same scenario as memory timeout, above. The trap will contain an I/O address. The most significant nibble is the slot requesting or acknowledging (possible failing board). Cards that generate an I/O request: DBP DAC Cards that can generate an I/O acknowledge: T&C DC TPC Channel DAC	14, 5, 6, 3, 4 1, 2, 9

ERROR	DESCRIPTION	PROBABLE CAUSE
READ DATA PARITY	The DBP detected that the 16 bit databus had a parity error.	13, 12, 1, 5, 6 7, 9
HFLT	Hardware fault. Catch all error for many different hardware problems on several BL700 Boards. 7, 2, 9 The pgm, pc, busmaster and what was running at the time may be useful for Britton Lee to determine the cause.	13, 12, 4, 3, 1
ODD WORD	An attempt was made to access memory on an odd word boundary. Memory can only be accessed at even word intervals.	11, 1, 2, 7, 9
UNCORRECTABLE MEMORY ERROR	The timing and control board detected that a memory card caused an error. Channels have memory and may cause uncorrectable memory errors. In both cases, the slot number within the trap indicates the failing board.	10, 8, 5, 6, 7 1, 13, 9
WRITE DATA PARITY	A parity error was detected by a board receiving the 16 bit data bus during a memory write cycle or an I/O write cycle.	13, 12, 1, 5, 6, 7, 3, 4, 9
ADDRESS PARITY	An address parity error was detected on the 24 bit address buss either during a memory or I/O cycle.	13, 12, 1, 5, 6, 3, 4, 9

Probable Causes

1. DBP (database processor).
2. DA (data accelerator).
3. DC (disk controller).
4. TPC (tape controller).
5. SIO (serial channel).
6. PIO (parallel channel). If the failure occurs with just the DBP, T&C and one memory card installed, suspect one of them as being at fault.
7. T&C (timing and control).
8. MEM (memory).
9. DC power supply voltage out of tolerance.
10. Use slot number in trap message to determine what board to change first.
11. Probable software inconsistency: get as many details as possible of what was running at time of failure, release of IDM/RDBMS software running etc. Report the problem to Britton Lee Support.
12. Capture what the BL700 was doing at the time of failure. Example: failed during a dump or load to IDM tape, might indicate that the bad board is the TPC, DC, DBP or T&C.
13. If the trap occurred during "slots" test, turn the BL700 off and remove all unnecessary cards except the DBP, T&C and one memory card. Repeat the "slots" test by turning the Control Panel switch to the MAINT position. Allow the test to complete, signified by the "BL700 - Filename" prompt being printed on the console. If the slots test completes without error several times, turn the BL700 off and insert one of the cards previously removed and follow the same procedure until the trap reoccurs. The most recently installed card is most likely the failing card.

BUS MASTER	DESCRIPTION	POSSIBLE BUS MASTER
Bus masters:	Board that was referencing memory	DBP, DAC, DC, TPC
DBP data:	DBP data reference to memory	
DBP text:	DBP text reference (commands)	
DA:	Data Accelerator (virtual)	
DC xq:	disk controller physical memory reference if x= 0–3 tape controller physical memory reference if x= 4	
DC xk:	disk controller memory reference virtual mode(command fetch or status port update) if x= 0–3 tape controller virtual memory reference if x=4	

Note: The "x" in DC is the disk or tape controller priority, assigned during slots. The priority is asserted by the controller wishing to become bus master to transfer data either to or from memory (physical mode) or fetch commands and post status of finished commands in virtual mode. The disk controller closest to slot 0 is assigned priority "0". The next lowest slot DC is assigned priority "1". The maximum number of disk controllers per BL700 is 4, so the range of x is 0–3. The idm tape controller is always assigned priority 4 regardless of slot position.

* The Z8002 processor uses a custom designed Memory Management Unit(MMU) to extend its capabilities of memory addressing from 64 KB to 16 MB of internal memory. The BL700 is limited to 6 MB of internal memory. This is done by way of MMU registers divided into 4 sets of 32 windows of 16 bits each. Each window is then mapped to a 24 bit physical address within BL700 main memory.

DISK ERROR CODES

The following list describes the possible 'soft' and 'hard' disk error messages.
This is an example of a blank disk error message.

1. Disk _____2. error code_____
3. cmd _____4. cntlr _____5. drive _____6. cyl _____7. head _____8. sec _____
9. errcode _____,10. errstat_____,11. phyaddr _____

The numbers indicated in bold are not present in a real disk error message. These numbers are intended to help you reference an explanation of each component in a disk error message.

1. DISK: hard err Error was unrecoverable
   soft err An error was detected and recovered through command retries

2. ERROR CODE: Hex XXXX (This is the command block return code).

Bit	15	Command Block completed
Bit	14	Unrecoverable error
Bit	13	ECC error
Bit	12	Correctable data
Bit	11	Not used, bit = 0
Bit	10	Seek error
Bit	9	Seek error or alternate used
Bit	8	Drive fault
Bit	7	Retry failed
Bit	6	Inhibit retry
Bit	5	Inhibit ecc
Bit	4	Retry counter
Bit	3:0	Subsectors processed counter

3. CMD: The following are the disk controller commands:

COMMAND CODE	DESCRIPTION
0	Invalid
1	Seek
2	Write
3	Read
4	No-op
5	Copyp
6	Copyv
7	Format track
8	Format sector
9	Write header
10	Read header
11	Diagnostic write
12	Null
13	Verify
14	Verify format
15	Invalid

4. CNTLR: Controller (0–3) on which drive is reporting the error
5. DRIVE: Drive logical address reporting error
6. CYL: Cylinder on which error occurred
7. HEAD: Head on which error occurred
8. SEC: Sector on which error occurred
9. ERRCODE: See disk error code

10. ERRSTAT: Retry counter

11. PHYADDR: Physical memory address

SOFT (RECOVERABLE) AND HARD (NONRECOVERABLE) DISK READ ERRORS

Disk read errors are reported to the BL700 console port. The errors reported contain the disk controller (0–3), the drive number, and a four-digit return code with the associated cylinder head and sector. There are three variations of disk read errors, each with an associated degree of severity:

ERROR	DESCRIPTION	ACTION
Disk Soft Error A008, Error Code 0	The A008 is the return code. The "A" indicates that one of the eight subsectors that make up a sector had five or fewer bits in error when it was first read (correctable syndrome). The command was retried and read correctly on the second attempt (zero syndrome). The "8" indicates that all eight, 256-byte subsectors were processed.	1. Remap if repeatable.
Disk Soft Error B008, Error Code 0	The "B" indicates that a soft read error occurred. The read was re-executed and was still soft on the second read. (Both reads had the same correctable syndrome.) The ECC circuitry on the disk controller was enabled and the subsector was fixed in memory. The "8" indicates that all eight, 256-byte subsectors were processed eventually.	1. Remap with DFU scan.
Disk Hard Error E00X, Error Code 100	The "E" indicates that a subsector within the sector read had more than five bits in error during each of the sixty retries (uncorrectable syndrome). The disk controller's ECC circuitry could not correct the data with any certainty. The sector in this case cannot be recovered. The "X" is a decimal number, 0–8, indicating how many subsectors were processed before the unreadable subsector was encountered.	1. Destroy affected database (this deallocates the bad sector). 2. Use DFU scan to remap defective sector. 3. Load latest back-up and rollforward.

SOFT AND HARD DISK ERRORS

ERROR CODE	DESCRIPTION	ACTION
50	Wrong head in sector header. The header field in the specified sector contains a head address different from the one selected by the controller.	*1.* Check disk drive. *2.* Check "A" cable (Seek problem). *3.* Media flaw. Refer to error code 100. 4. Reformat disk *5.* Check BL700 voltages *6.* Replace disk controller.
51	Wrong cylinder in sector header. The header field in the specified sector contains a cylinder address different from the one selected by the controller.	*1.* Check disk drive for possible media flaw. Bad heads. Select card in disk drive. *2.* Replace "A" cable (Refer to error code 100). 3. Reformat disk *4.* Check BL700 voltages. *5.* Replace disk controller.
52	Interrupt other than timeout. During a controller command sequence, an interrupt occurred. Since the controller can only test for a TIMEOUT interrupt, all other interrupts are lumped into this error. The command sequence is retried. HFLT, data or address parity error or the disk controller was granted bus mastership without requesting it.	*1.* Check sector setting on drive. *2.* Replace "A" and "B" cables, timing & control. *3.* Check BL700 voltages.
53	Timeout from 9513 (disk controller). An event expected by the controller did not occur, or timed out. This can occur during disk or BL700 memory transfer operations.	*1.* Wrong sector setting on drive. If correct (2B), reset sector switches) *2.* Check "A" cable. *3.* Not sensing index and sector on "A" cable. Verify that index and sector are optioned on "A" cable. *4.* Disk being read is not in BL700 format. *5.* Check cables and terminator. *6.* Replace disk controller.
54	Drive selection error. More than one drive (or no drive) responded to a select sequence.	*1.* Bad or loose "A" and "B" cables. *2.* More than one drive with same address on the same disk controller. *3.* Drive problem.

ERROR CODE	DESCRIPTION	ACTION
55	Drive did not go off-cylinder After a seek sequence was issued to the drive, the controller did not detect the drive going off-cylinder.	*1.* Replace disk drive. *2.* Check "A" cable. *3.* Check disk controller internal cables.
56	Drive is write-protected. A WRITE or FORMAT command was issued to a drive that is write-protected.	*1.* Check write protect switch on drive and possible secondary switch located inside drive. *2.* Replace disk controller.
80	SMD fault. A FAULT condition was detected on the specified drive during a disk operation.	*1.* Check disk drive for fault indication and interrogate type of fault (refer to drive's operation manual). *2.* The command code in disk error message could be helpful in determining problem. *3.* Replace disk controller. *4.* Disk drive problem.
81	Drive not ready. The "A" cable signal "READY" from a disk was found to be false during selection.	*1.* Check drive and verify that it is operational, fix if necessary. 2. The controller selected a drive, then found it not ready.
82	SMD seek-error. Indicates the drive had some major problem during the seek operation and could not finish the seek.	*1.* Check cylinder, head, sector from the disk error message to verify that it is within disk drive range. 2. Out of range: bad disk controller or cable. 3. In range: drive problem, refer to drive's operation manual.
83	Drive not on-cylinder. The selected drive was found to be off-cylinder after a select sequence.	*1.* Bad or loose cables. *2.* Bad "A" cable and/or terminator. *3.* Check disk drive *4.* Disk controller internal cabling.

ERROR CODE	DESCRIPTION	ACTION
84	Seek incomplete. A seek issued to a drive was never completed. No error was indicated by the drive.	*1.* Bad or loose cables. *2.* Bad "A" cable and/or terminator. *3.* Check disk drive. Verify fault set. *4.* Disk controller and internal cables.
85	Unexpected bad status. This is a catch-all error indicating that a previously good drive status was bad at a latter time during the disk operation.	*1.* Bad or loose cables. *2.* Check "A" cable and/or terminator. *3.* Check disk drive. *4.* Disk controller internal cables.
86	No drive selected. The controller timed out waiting for a seek to complete and discovered that no drive was selected.	*1.* Check unit select. *2.* Drive could be in Customer Engineering mode. *3.* Check "B" cable. *4.* Replace disk controller.
100	Permanent read error. This indicates that the specified sector is unreadable (has an error greater than 5 bits). All retries have been used.	*1.* Destroy affected database (deallocates a bad sector). *2.* Use DFU scan mode to remap defective sector. *3.* Load latest back-up and rollforward. *4.* Disk drive problem. *5.* Media flaw (check disk drive flow chart). *6.* Noisy "B" cable or the cable exceeds 14.02m (46').
101	ECC error in verify. During a VERIFY command, the specified sector was found to have a data error. Since this error is only seen during a format sequence, no retries are attempted, and the sector is mapped out.	*1.* Media flaw found only when formatting. This is a normal error that occurs during a format. *2.* A faulty ground connection between the drive and the BL700. 3. Check sector setting on drive.

ERROR CODE	DESCRIPTION	ACTION
102	Software tried to wrap FFFFFF. During a data transfer to or from BL700 memory, the addresses wrapped from FFFFFF (hex) to 000000 (hex).	*1.* Check disk controller.
103 (Refer	ac is low in BL700 A Power Fail condition during a write was detected in the BL700, causing the controller to abort the current disk operation.	*1.* Check that ac input to BL700 is within spec. install manual contains specs.) *2.* Power line disturbances. *3.* Check output of T-1 transformer (between 14 and 28V ac). *4.* Check J4 cable to DBP. *5.* Replace database processor. *6.* Replace disk controller. *7.* Replace control panel.
104	Misaligned parameter in command block. The low byte of the BL700 memory address passed to the controller in the command block (CB) was found to be nonzero.	*1.* Replace disk controller 2. Call Britton Lee.
105	Address sync error This error is similar to the above, except that the problem was detected after the transfer of one or more subsectors to or from BL700 memory.	*1.* Replace disk controller. *2.* Replace database processor. *3.* Replace timing & control.
106	Illegal command code. An illegal command was passed to the controller in the CB.	*1.* Replace database processor. *2.* Replace disk controller. *3.* Replace timing & control.
107	Sector flagged bad. The header in the target sector indicates that it is bad and cannot be read. This sector flag is set at format time, and the sector should not have been specified.	*1.* Call Britton Lee.

ERROR CODE	DESCRIPTION	ACTION
108	Checksum error in verify During a VERIFY command, the sector checksum did not match the checksum of the buffer specified in the CB.	*1.* Call Britton Lee.
109	Cannot find sector **This error has two meanings:** During a FORMAT TRACK command, it indicates that the controller detected the INDEX pulse before the final sector was formatted. Otherwise, it indicates that the controller could not locate the sector specified in the Command Block.	*1.* Possible media flaw. *2.* Check sector setting on drive. *3.* Replace disk controller. *4.* Replace timing & control.
110	Invalid sector flag The target sector had a non-zero flag byte, but it is not flagged bad or assigned an alternate address.	*1.* Disk drive media flaw. 2. Replace disk controller.
111	No seekend after RTZ This error may occur during a retry sequence for another disk-related error. It indicates that the controller initiated a disk RECALIBRATION sequence which was never completed.	*1.* Disk drive problem. *2.* Possible cable problem. *3.* Replace disk controller.

TAPE ERROR CODES

When required, the BL700 will respond to any tape command involving a tape hardware error. The first part of the error code that is returned is masked and reported in text. For example, 112 Tape Unavailable, would be reported as "Tape Unavailable:". Following the colon (:), an 'error code' number appears. The following is a list of the 'error code' numbers, their description and the suggested course of action, for each of the error codes reported with a message in text.

112 Tape Unavailable

ERROR CODE	DESCRIPTION	ACTION
0 (tpc)	Selected drive not at load point. (The Executed command expected the tape drive to be at load point. The interface signal is ILDP and is true when the BOT marker is sensed.)	*1.* Manually rewind tape to BOT. *2.* Reset tape drive by unloading tape and powering tape unit *OFF/ON*. *3.* Tape used does not contain a BOT marker try another tape. *4.* Tape Controller is not sensing the load point signal from the formatter. Run loopback on tpc to verify interface is good. *5.* Cables are bad. Switch cables, JP1 becomes JP2, JP2 becomes JP1. A different error indicates bad cable. *6.* Replace tape drive. *7.* Replace tpc. *8.* Check internal cables (cables inside the BL700) and external cables for broken pins or loose connections.
1	Selected drive is in write protect. (The interface signal IFPT (file protect) was true. Either the write ring is missing on the tape or the drive is write protected by an over-ride switch.)	*1.* Install a write enable ring on tape. *2.* Disable write protect switch on tape drive. *3.* Verify write sensing device is functioning on tape drive. *4.* Cables are bad. Switch JP1 and JP2 cables. If error is different – bad cable. Replace cable. *5.* Run loopback test on tape interface. *6.* Replace tape drive. *7.* Replace tpc. *8.* Verify cables are not loose or have broken pins.

ERROR CODE	DESCRIPTION	ACTION
2	Selected drive not online. (The tape controller did not sense the interface signal IONL (online) from the tape drive.) *** See note.**	*1.* Verify online switch on tape drive is enabled. *2.* Verify tape has loaded successfully. *3.* Bad cable. Switch JP1 and JP2. If error is different, replace cables. *6.* Replace tpc. *7.* Replace tape drive.
3	Timeout on previously initiated rewind. (The tpc issued a rewind and never recieved the interface signals back that the drive completed the rewind. A completed rewind will be signaled by IRWD false and IRDY true.)	*1.* tpc timeout. Tape is longer than 2400 feet. Tape cannot exceed 2400 feet. *2.* Run loopback test to verify interface is good. *3.* Bad cable. Switch JP1 and JP2 cable. If error is different replace cables. *4.* Replace tape drive. *5.* Replace tpc.

* "Selected drive not online" may be the response to "loadtape". The BL700 will look for formatter address 0 (zero) and transport 0 (zero) for either of the two commands.

113 Tape Hard Errors

ERROR CODE	DESCRIPTION	ACTION
0	Timeout (The tpc's 2901 processors timed out waiting for results). **WARNING:** If a tape is being written for the first time and the "do not write name" option is specified, hard error timeout will occur. This will also happen if high speed or high density is selected on a drive that does not support it.	*1.* Use known good tape. *2.* Clean tape drive head. *3.* Run self-test on tape drive. *4.* Run 'td' diagnostics on BL700. *5.* Have DBA interrogate the configure relation to verify that high speed option is not set. (If drive does not support high speed option). *6.* Bad cable. Switch JP1 and JP2. If error is different replace cable. 7. Run loopback test to verify interface is functioning. *8.* Look for broken pins or loose connectors on I/O connectors. *9.* Replace tape drive. *10.* Replace tpc.
1	Illegal command. (The DBP wrote an address to the command port of the tpc.The address was used to fetch a command block (CB) from memory. The command in the CB was one that the tpc could not execute).	*1.* Check dc voltage on BL700. *2.* Loop on slots command. *3.* Replace tpc. *4.* Replace dbp. *5.* Replace T&C.
2	Tape interface parity error. (The tpc detected that the data parity bit did not coincide with the byte read). **WARNING:** If the I/O cables are on the wrong connectors this error will be seen. Switch cables at connectors on one end.	*1.* Verify cables are installed correctly. *2.* Bad cable. Switch JP1 and JP2. If error is different replace cable. *3.* Run loopback test to verify tpc interface. *4.* Replace tape drive. *5.* Replace tpc. *6.* Replace I/O cables.

ERROR CODE	DESCRIPTION	ACTION
3	Tape formatter hard error. (This is the interface signal IHER from the formatter. When true it means that the formatter detected an error which could not be corrected.) A) False preamble detected B) False postamble detected C) Multi channel drop out D) Parity error without channel drop out E) Loss of data envelope prior to postamble F) Excessive skew **WARNING:** If the drive is set to the wrong density for the tape mounted, then tape hard error will occur.	*1.* Clean tape drive head. *2.* Verify density setting correct. *3.* Use known good tape. *4.* Run 'td' diagnostics to verify drive can read write. *5.* Run drive self test. *6.* Replace tpc. *7.* Replace tape drive.
4	Could not write file mark. (The formatter did not signal the interface with IFMF, "file mark found" line after writing a file mark.)	*1.* Clean tape drive head. *2.* Bad cable. Switch JP1 and JP2 cables. If error is different replace cables. *3.* Run loopback test on tpc interface. *4.* Run 'td' diagnostics on tape drive. *5.* Replace tape drive. *6.* Replace tpc. *7.* Replace cables.
5	Data over run. A) During a write, the tpc sent a byte before the formatter had accepted the previous byte using the IWSTR signal "write strobe". B) During a read the formatter sent a byte before the tape controller accepted the previous byte using the IRSTR signal "read strobe". C) FIFO problem (tpc).	*1.* Clean tape drive head. *2.* Use known good tape. *3.* Verify tape drive EOT latched option is *ON*. *4.* Replace tape drive. *5.* Replace tpc. *6.* Check BL700 voltages. *7.* Replace DBP.

ERROR CODE	DESCRIPTION	ACTION
6	Specified number of file marks not found. The tape controller was told to search for a specified number of file marks. Either the drive encountered the EOT marker while searching before the count was equal or double file marks were encountered indicating logical EOT.	*1*. This error can only occur during 'td' (tape diagnostics) and is usually caused by specifying more file marks than exist on the tape. *2*. Swap tape. *3*. Swap tape drive. *4*. Swap tape controller. *5*. Swap cables.
7	BL700 software detected tape controller timeout. The BL700 software issued a command to the tpc, no response was sensed within a timeout period.	*1*. Check BL700 voltages. *2*. Replace tpc. *3*. Replace DBP.

114 Tape Caution Errors

ERROR CODE	DESCRIPTION	ACTION
0	File mark found on a read but no data. (A read command was executed, instead of detecting data the formatter detected a file mark and set the interface signal IFMK true "file mark found".)	*1.* Clean tape drive head. *2.* Use known good tape. *3.* Run drive self-test. *4.* Run auto-test on 'td' diagnostics. *5.* Run loopback test on tpc interface. *6.* Replace tape drive. *7.* Replace tpc.
1 - 4:	Not Used.	
5 is predetermined by the	Tape record smaller than detected. (Data block size BL700 as 8K bytes between IBG's (interblock gaps) the BL700 buffers are 2k bytes. To read a tape block, 4 read commands are executed in succession, in this case databusy went false indicating that an IBG (interblock gap) was detected.	*1.* Tape being read was not written on a IDM Tape drive. Use known good tape. *2.* Run tape drive self-test. *3.* Run auto test on 'td'. diagnostics. *4.* Run loopback test on tpc interface. *5.* Replace tape drive. *6.* Replace tpc.
6	Tape record is longer than expected. (After executing 4 reads, an interblock gap should be sensed (databusy goes false) in this case the formatter detected more data.)	*1.* Same as errorcode 5.

DATA ACCELERATOR (DAC) ERRORS

INTRODUCTION

Before replacing any boards always make the following checks in the sequence listed below:

1. Check the BL700 voltages
   a. Pre-FCC Britton Lee database server's with TM11's set -5.2 to -5.35
2. Verify DBP is rev 27 or higher
3. Verify Disk Controller is 26 or higher
4. Verify Timing and Control is rev 3 or higher
5. Verify software is rev 25 or higher
6. Verify DAC is in slot 11

The following errors are detected by the PROM:

ERROR	DESCRIPTION	ACTION
8001	DAC is busy, not an error. The DAC status register busy bit was set. While the DAC is busy with a command it keeps a value of 8001 in the status register, the Database Processor (DBP) continuously polls this port waiting for the value to change.	*1.* No user action required
8011	Illegal direct command received. A direct command other than: IDENT, DUMP REG, I/O DIAG, MEM DIAG, CONTINUE was received. Direct commands are those which only return status to the input register, the DAC can execute them without any additional information, if the command contains parameters it is defined by the upper bits of the direct command.	*1.* Replace DAC *2.* Replace DBP *3.* Replace T & C
8013	I/O diagnostic error. During a slots command the DAC will send a set priority command to the first disk controller on the bus (I/O write). It will then do an I/O read to the disk controller and compare the results.	*1.* Replace DAC *2.* Replace DBP *3.* Replace T & C
8015#	Memory diagnostic error K mode (word mode transfer)	*1.* Replace DAC *2.* Replace T & C *3.* Check voltages
#	The Memory Diagnostic command will transfer ten words in K, Q, B mode from a general description register (R register) to memory. The data is then transferred to the input buffer and compared to the original data in the general registers.	
8017#	Memory diagnostic error Q mode (quad mode transfer)	*1.* Replace DAC *2.* Replace T & C *3.* Check voltages
8019#	Memory diagnostic error B mode (byte mode transfer)	*1.* Replace DAC *2.* Replace T & C *3.* Check voltages
801B	An illegal indirect command was detected at the DAC command port.	*1.* Replace DAC *2.* Replace T & C *3.* Check voltages
801D	The DAC thinks that some board did an I/O write to its command port while the busy bit was set.	*1.* Replace DAC

ERROR	DESCRIPTION	ACTION
801F	Interrupt fault error. Bus parity either on the DAC A, B, D bus or the DAC saw a BL700 bus parity error.	*1.* Replace DAC *2.* Replace T & C *3.* Replace DBP
841F*	The DAC detected an uncorrectable memory error. The signal on bus is MST2.	*1.* Replace 1 memory board at a time *2.* Replace T & C *3.* Replace DAC *4.* Replace DBP

* Bits 10–14 may be set in any combination.

* Bit 15 = 1.

EXAMPLE: Error code E01F = bit 15, 14, 13, on error code descriptions for 801F, C01F, A01F = E01F.

881F*	Bus parity error. The signal on the BL700 bus is: BPER	*1.* Replace DAC
901F*	Parity error on opcode bus (internal to the DAC) and then reads it.	*1.* Replace DAC
A01F	*Internal register address parity error (DAC RFB bus)	*1.* Replace DAC *2.* Check Voltages
C01F	*Internal register address parity error (DAC RFA bus)	*1.* Replace DAC *2.* Check Voltages
8021	Diagnostic on the DAC which writes and reads the MMU window failed.	*1.* Replace DAC
8023	The DAC could not detect a disk controller to do an I/O write to (set priority) during diagnostic startup.	*1.* Replace Disk Controller *2.* Replace DAC *3.* Replace T & C

The following errors are detected by Writable Control Storage (WCS):

8001	Same as PROM detected error 8001	*1.* Replace DAC
8003	An unknown indirect command was issued to the DAC. The command value is actually the virtual address of a command block which describes the function. The DBP issues a command to the DAC. This command is actually a location in main BL700 memory where the command can be found. The DAC went to this location in memory, fetched the contents through the Timing and Control (T & C) and found that the command in the location was an invalid, indirect command.	*1.* Replace DAC *2.* Replace T & C *3.* Replace DBP
8005	A tuple was found to have a bad size, either 0, negative, or very large.	*1.* Run CKDB *2.* Replace DAC

ERROR	DESCRIPTION	ACTION
8007	Buffer Search (Bsearch) was given a bad mode. Bsearch performs an index search on a single page and returns the buffer pointer of the examined page.	*1.* Run CKDB *2.* Replace DAC
8009	Buffer Compare (Bcmp) got a bad type. The Bcmp routine was passed a type which was not: CHAR, INT1, INT2, INT4, BCD, BCDFLT,FLT4, FLT8, BIN. Bcmp compares two scalar values of the same type. The return value indicates: >, <, or zero.	*1.* Run CKDB *2.* Replace DBP *3.* Replace DAC
800B	The count parameter of one of the move routines (Buffermove (Bmove), Backmove) was greater than 2047. Bmove moves a specified number of bytes between two virtual addresses. Backmove moves a specified number of bytes from a high address to a low address.	*1.* Run CKDB *2.* Replace DAC
800D	Bcdcount to bcdwrite was too small.This is an internal processing error of the Binary Coded Decimal (BCD arithmetic).	*1.* Run CKDB *2.* Replace DAC
800F	The shift function was given more than 17 bytes. This is an internal processing error of the BCD arithmetic.	*1.* Run CKDB *2.* Replace DAC
8101	First argument to a BCD routine had a length = zero. The DAC arithmetic was given a bad parameter.	*1.* Run CKDB *2.* Replace DAC

CHANNEL SYSERR MESSAGES

INTRODUCTION

When a channel syserr occurs the BL700 will print the following message on the BL700 console:

```
Channel syserr board n, syserr#.  p1.  p2.  p3.  p4
```

The parameters p1 through p4 are possible parameters to the syserr. The DBP will always print them, but they are not always useful. The board number , syserr number, and parameters are all printed in decimal. The following table can be used to determine what the syserr number means. Many of the syserrs refer to internal consistency checks in the software, while others may show a hardware problem. Syserrs referencing the DBP are possibly hardware related, involve the communication between the DBP and the channel board and may indicate a problem in either board. To isolate the problem first check dc voltages, then replace boards in the following sequence; channel I/O, then DBP.

ERROR	DESCRIPTION
1:	Process p2 tried to allocate less than 0 (p1) bytes.
2:	Process p2 tried to allocate a buffer out of something other than input or output space (p1).
3:	Process p2 tried to free something that is not a buffer (p1).
4:	Process p3 tried to free a buffer already free at address p1 that is p2 bytes big.
5:	Dbpsmproc, chansmproc, or retagedproc tried to allocate a shared memory buffer not in shared memory (p1).
8:	Hostproc woken with a message from some process other than canproc, dbpsmproc or chansmproc (p1).
10:	Received an unknown command from the DBP (p1).
11:	Bad serial interrupt, port not in range 0/-8 (p1).
12:	Got a transmitter interrupt from sio chip when not expecting one. A hardware problem.
13:	Got a special interrupt on sio chip for an unknown reason.
14:	Got a bad sio interrupt (p1).
15:	Process p4 tried to wakeup an illegal process (p2) waiting on event p1.
17:	Serial channel tried to send an asynchronous packet at the wrong time (p1).
18:	Chansmproc got a message from some process other than 1 of the 8 hostprocs (p1).
19:	Chansmproc got a command (p1) to do something other then send input to dbp.

ERROR	DESCRIPTION
20:	DBP sent an unknown response (p1) to input command from chansmproc.
21:	The DBP sent output to a channel host process not in the 3–11 range (p1).
22:	Dbpsmproc got an unknown dbp command (p1).
23:	Tried to write less than 0/ bytes (p1) over serial interface.
24:	Tried to read less than 0/ bytes (p1) from the serial interface.
27:	Got an interrupt that dma read from shared memory completed prematurely.
28:	Got an interrupt that dma write to shared memory completed prematurely.
29:	Got a parity error during a dma for one of the ports 4–7.
30:	Got a parity error during a dma for one of the ports 0/–3.
31:	Got a serial interrupt on the parallel channel.
32:	Dma onto the gpib got a parity error.
33:	Got a serial dma termination interrupt on a parallel channel.
34:	Got a serial dma termination interrupt on a parallel channel.
35:	Tried to write less than 0/ bytes (p1).
36:	Tried to read less than 0/bytes (p1).
37:	The DBP did not send an inptspace command to chansmproc before sending another command (p1).
40:	Dma read from shared memory got a parity error.
41:	DBP sent a cancel dbin acknowledge for a cancel on dbin 0/ even though this is illegal.
42:	Parallel channel told to do a loopback test.

ERROR	DESCRIPTION
46:	Shared memory process woken up even though the DBP has not yet responded.
47:	DBP sent more commands than the channel could process. (Only in Rev. 16 parallel channels).
48:	Chansmproc got a holdindbin command from the DBP even though the kernal said it would not send one.
49:	Got a needinpt command even though the kernal said it would not send one.
52:	Dma write to shared got a parity error.
53:	Retagedproc was sent a message from some process other than agedproc (p1).
54:	Tried to aged out p1 bytes data for a dbin (p2) that does not exist.
56:	Process p1 tried to free a message header that is already free.
57:	DBP sent bad response (p1) to retagedproc to acknowledge aged results.
58:	DBP sent bad response (p1) to retagedproc to acknowledge aged results.
60:	Canproc received a bad response (p1) from DBP to either a cancel, identify, or need result command.
62:	DBP sent bad response (p1) to retagedproc to acknowledge aged results.
64:	DBP reported to canproc that an internal cancel was illegal.
67:	Canproc received a message from some process other than hostproc (p1).
79:	Could not find specified result block in result list.
88:	Retagedproc has a bad pointer pointing to result to be aged out (p4).
90:	Could not find results for the given pid (p1, p2) in the given result list.
91:	Could not find results for the given pid (p1, p2) in the given result list.

ERROR	DESCRIPTION
96:	Shared memory process woken without getting a DBP command.
102:	DBP sent a final acknowledge to a cancel even though the channel was not waiting for one.
104:	Process p2 tried to free a shared memory buffer (p1) not in shared memory.
105:	Routine called by process p2 with the channel host process out of range (p3), should be between 3–11. If p2 has the value 2 this is probably a hardware problem involving the DBP.

TAPE CONTROLLER LOOPBACK ERROR CODES

Refer to the Tape Drive Operations Manual for a description of the standard interface signals.

ERROR	ERROR TEXT	DESCRIPTION
08	driving signal (IGO) receiving signal (IONL)	IONL should have been low, but when it was checked it was high.
09	driving signal (IGO) receiving signal (IONL)	IONL should have been high, but when it was checked it was low.
10	driving signal (ILWD) receiving signal (IRDY)	IRDY should have been low, but when it was checked it was high.
11	driving signal (ILWD) receiving signal (IRDY)	IRDY should have been high, but when it was checked it was low.
12	driving signal (IFEN) receiving signal (IFBY)	IFBY should have been low, but when it was checked it was high.
13	driving signal (IFEN) receiving signal (IFBY)	IFBY should have been high, but when it was checked it was low.
14	driving signal (IFAD) receiving signal (IFPT)	IFPT should have been low, but when it was checked it was high.
15	driving signal (IFAD) receiving signal (IFPT)	IFPT should have been high, but when it was checked it was low.
16	driving signal (ITAD0) receiving signal (ISPEED)	ISPEED should have been low, but when it was checked it was high.
17	driving signal (ITAD0) receiving signal (ISPEED)	ISPEED should have been high, but when it was checked it was low.
18	driving signal (ITAD1) receiving signal (ILDP)	ILDP should have been low, but when it was checked it was high.
19	driving signal (ITAD1) receiving signal (ILDP)	ILDP should have been high, but when it was checked it was low.
20	driving signal (IHISP) receiving signals (IDBY and IRWD)	IRWD should have been low, but when it was checked it was high.
21	driving signal (IHISP) receiving signal (IDBY and IRWD)	IDBY should have been low, but when it was checked it was high.
22	driving signal (IHISP) receiving signal (IDBY and IRWD)	IRWD should have been high, but when it was checked it was low.
23	driving signal (IHISP) receiving signal (IDBY and IRWD)	IDBY should have been high, but when it was checked it was low.
ERROR	ERROR TEXT	DESCRIPTION
24	driving signal (IRWU) receiving signal (IDENT)	IDENT should have been low, but when it was checked it was high.
25	driving signal (IRWU) receiving signal (IDENT)	IDENT should have been high, but when it was checked it was low.

26	driving signal (IREW) receiving signal (IFMK)	IFMK should have been low, but when it was checked it was high. RTPORT clears IFMK.
27	driving signal (IREW) receiving signal (IFMK)	IFMK should have been high, but when it was checked it was low.
28	driving signal (IERASE) receiving signal (IHER)	IHER should have been high, but when it was checked it was low. RTPORT clears IHER.
29	driving signal (IERASE) receiving signal (IHER)	IHER should have been low, but when it was checked it was high.
30	driving signal (IEDIT) receiving signal (ICER)	ICER should have been high, but when it was checked it was low. RTPORT clears ICER.
31	driving signal (IEDIT) receiving signal (ICER)	ICER should have been low, but when it was checked it was high.
32	driving signal (IWFM) receiving signal (IEOT)	IEOT should have been high, but when it was checked it was low. RTPORT clears IEOT.
33	driving signal (IWFM) receiving signal (IEOT)	IEOT should have been low, but when it was checked it was high.
34	driving signal (IWRT) receiving signal (IWSTR)	WRTLATCH should have been high, but when it was checked it was low.
35	driving signal (IWRT) receiving signal (IWSTR)	ORUR should have been high, but when it was checked it was low.
36	driving signal (IWRT) receiving signal (IWSTR)	WRTLATCH should have been low, but when it was checked it was low.
37	driving signal (IWRT) receiving signal (IWSTR)	ORUR should have been low, but when it was checked it was high.
38	driving signal (IREV) receiving signal (IRSTR)	RDLATCH should have been high, but when it was checked it was low.
39	driving signal (IREV) receiving signal (IRSTR)	ORUR should have been high, but when it was checked it was low.
40	driving signal (IREV) receiving signal (IRSTR)	RDLATCH should have been low, but when it was checked it was high.
41	driving signal (IREV) receiving signal (IRSTR)	ORUR should have been low, but when it was checked it was high.
ERROR	**ERROR TEXT**	**DESCRIPTION**
42	driving signals (IW0–IW7) receiving signals (IR0–IR7)	The TPC was trying to write to memory via the k–mode port. The cycle was started, but MACK was not received by the TPC.
43	driving signals (IW0–IW7) receiving signals (IR0–IR7)	The TPC was trying to read from memory via the fifo using Q–mode. The cycle was started, but MACK was not received by the TPC.

44	driving signals (IW0–IW7) receiving signals (IR0–IR7)	The data that was transferred to the fifo from memory was routed through the tape write and read ports and back into the fifo. It was then verified and wrong.

CHANNEL SYSERRS FOR BLOCKMUX

INTRODUCTION

These are CHANNEL SYSERRS for the Blockmux. The code number is on the left and the description to their right. Blockmux errors may indicate a bad channel board or defective DBP. If BL700 voltages are low or out of tolerance an error will also occur. To isolate the problem, first check voltages, then replace boards one at a time in the following sequence: channel board, DBP, and Timing and Control board.

ERROR	DESCRIPTION
1	Process &2 requested &1 bytes which is <= 0.
2	Process &2 requested illegal buffer pool number &1.
3	Process &2 tried to free buffer at &1 which is not in any buffer pool.
4	Process &2 tried to free buffer with bad buffer header, bufp = &1, size = &3.
9	Too many DBP commands.
10	Received AGEDDONE, NOAGEDSPACE, or AGEDSPACE command, command = &1.
15	Wakeup process not valid, event = &1, proc = &2, switch = &3, Curproc = &4.
20	Unrecognized response &1 from DBP.
22	Bad command from DBP &1.
37	Response &1 was not INPTSPACE after being told NOSPACE.
41	CANDACK for non-zero dbin but dbin = 0.
42	Loop back test not supported.
43	Attempt to move (using bmove) more than 2048 bytes.
46	DBP interrupt but no response in queue.
60	Response &1 was not CNCLDONE.
96	DBP interrupt but no command in queue.
102	Unexpected CANCEL response &1.
2001	Channel stopped data transfer on device &1, command &2, device flag &3.
2002	Data transfer complete but Operational In is down, device &1, command &2 device flag &3.

ERROR	DESCRIPTION
2003	Parity error on DMA.
2004	Single bit DRAM error.
2005	Parity error in static RAM.
2006	Illegal dbin pid = &1, dbin = &2.
2007	NEEDBUF while Output pending pid = &1, dbin = &2.
2008	OUTPUT with no Output pending, pid = &1, dbin = &2.
2009	Tried to remove a CANCELHOST msg, process = &1, pid = &1, dbin = &2.
2010	Zero length data packet on sendlist, send list pointer = &2.
2011	Tried to send results for which no request was pending, dptr = &1, msg = &2.
2013	No Pidtbl entry for OUTPUT, pid = &2, dbin = &3.
2014	No Pidtbl entry for NEEDBUF, pid = &2, dbin = &3.
2020	Bank number &2 out of range.
2021	Bank number &2 out of range.
2022	No Pidtbl entry for CANDACK, pid = &2, dbin = &3.

CHANNEL SYSERRS FOR ETHERNET

INTRODUCTION

Ethernet errors may indicate a bad channel board or defective DBP. If BL700 voltages are low or out of tolerance an error will also occur. To isolate the problem, check voltages, replace channel board and then DBP.

ERROR	DESCRIPTION
1	Process %2 called alloc with a bytecount (%1) <=0.
2	Process %2 called alloc with wrong buffer pool argument (%1).
3	Process %2 tried to free a buffer (%1) not IN, OUT, or BOTH space.
4	Process %3 tried to free a buffer that was not allocated.
5	Found a msghdr in free list that was used.
6	Tried to free a msghdr that was already free.
7	Tried to remove the wrong msghdr from TRANSMITPROC'S process table message queue.
100	Response (%2) to command (%1) sent from canproc was not CNCLDONE.
200	Chansmproc received some command (%1) from hostoutproc other than INPUT or SPOON.
203	Channel expected inptspace but got (%1) instead.
204	Channel received (%1) instead of DONE, HOLDIN, or NOINSPACE.
300	Process active but there was no command found to process.
301	Process active but there was no command found to process.
302	Kernal sent an unknown command to the channel (%1).
303	Kernal thought the channel tried to age something out.
400	DBP gave the channel an illegal pid.
401	DBP gave the channel an illegal port (%1).
402	DBP gave the channel an unexpected candack.
403	DBP gave the channel a regular candack for dpin 0.
404	DBP gave the channel an illegal pid.
405	DBP gave the channel an unknown command (%1).

ERROR	DESCRIPTION
500	Hostoutproc received a message it did not know how to process.
501	Could not find any message waiting for allocation.
502	Could not find any message waiting for allocation.
600	Hostproc could not find the packet it was supposed to process.
601	Socket table (%1) firstpak and lastpak are inconsistent.
602	Socket table (%1) firstpak and lastpak are inconsistent.
603	Garbage found socket table (%2) queue.
604	Free msghdr found in socket table queue.
605	Socket table queue is inconsistent.
700	Host table contained bad information in hosttype field.
802	Received a Z8000 parity error.
803	Received an unexpected bad transmit interrupt.
804	Received an unexpected good transmit interrupt.
900	Wakeup called by process %4 with bad process (%2).
1000	Boutptr is pointing to an illegal buffer.
1001	Packet is free to 0 length.
1002	Realfree called with nothing to free.
1100	The packet we are sending is not an XNS packet.
1101	Packet was sent 32 times with a hardware error.
1102	This packet does not have an allocation, it should not be sent.
1201	Badop code interrupt.
1202	Privileged instruction trap.
1203	Segmentation trap.
1200	Got DBP non-maskable interrupts too quickly.
1300	Count > 4096.
1400	Data should be in the socket queue, but it is empty.

DISK FORMATTING UTILITY (DFU) ERROR MESSAGES

Disk formatting error messages appear on the console during the disk formatting process in SysFormat mode.

ERROR	DESCRIPTION
There is no controller n.	The BL700 has no such disk controller. The disk controller with the lowest slot number is called controller 0. The next controller with a higher slot number is controller 1, etc.
No drive x on controller y.	Controller y exists, but there is no drive x on it.
Disk "X" is the system disk.	You have attempted to format the disk that contains the IDM/RDBMS code and the system database.
Disk "X" is already formatted. Use Unformat before formatting this drive.	You have attempted to format a drive that is already in the correct BL700 format. It may contain valuable data.
The name you have entered is not unique. Please enter a unique name.	Either the relation, disks, already contains a tuple with that name or another disk on line has that name.
Not enough space for that disk. Please enter all data again.	The maximum number of blocks in an BL700 system ($2^{24}-1$) would be exceeded if this disk is formatted. Try again with fewer cylinders.
Not enough room for another subdivision. Enlarging subdivision to cylinder n.	There is not enough room left on the disk for another subdivision of minimum size. This one will be enlarged to fill the rest of the disk.
Track cannot be formatted.	A hard error occurred while writing the formatting pattern onto a disk track.
Cannot remap n on original zone x.	The zone on which the block is from is full. The program will look in other zones for a free block to remap to.
WARNING: Hard error re-writing header of block n.	Remap failure.

ERROR	DESCRIPTION
WARNING: Hard read error on remapped header of block n.	Remap failure.
Can not continue formatting	Any fatal error generates this message with a description of the cause.
WARNING: Hard read error checking cylinder x.	A hard read error occurred while reading a block during the cylinder selection check.
Error in cyl select: cmd blk: cyl=x, head=y, sector=z sec hdr: cyl=x, head=y, sector=z	Possible drive fault. The block address in the sector header differs from that in the controller command block.
WARNING: Hard read error checking head x.	A hard read error occurred while reading a block during the head selection check.
Error in head select: cmd blk: cyl=x, head=y, sector=z sec hdr: cyl=x, head=y, sector=z	Possible drive fault. The information in the sector header differs from that in the controller command block.
Block already mapped out.	Block was mapped out earlier.
File page n has already been remapped. File pages may only be remapped once.	File pages have no page header and may only be remapped once.
Could not read block n.	The block requested cannot be read. It will be marked bad.
WARNING: Hard read error onblockalloc n. Skipping zone n.	Program attempted to reassign a block on a zone other than the original, and the blockalloc page is unreadable.

ERROR	DESCRIPTION
Could not find a free block on any zone with dbid n.	Remapping fails if no free blocks within the original database are found.
Please respond with "yes" or "no".	"yes", "no", "y", and "n" are the only valid responses to the prompt.
Please enter the requested information.	The operator has responded incorrectly to a prompt.
The name must be 12 characters or less.	The disk or subdivision name entered was too long.
The number must be between <min> and <max>.	An illegal number was entered.
The number must contain only digits.	The number contained illegal characters.

CKDB ERROR MESSAGES

CKDB ERROR MESSAGE FORMAT

CKDB error messages are illustrated with a shorthand to specify the type of data appearing in a message. This is the table of abbreviations and their respective meanings.

%d	A decimal number is printed; it never has a leading 0.
%o	An octal number is printed; it always has a leading 0.
%P	Print a virtual block number as three octal bytes separated by < and >: < 23 : 130 : 363 >.
%H	Print out a disk block address as either its virtual number (ala %P) or its physical disk block address as in cyl %d Head %d sector %d.

1. The error message tape controller and disk drive or tape transport is referenced. If the block is in the bad block remapping area of a drive, then it is printed with the physical address because blocks in the remap area do not have a virtual block number.
2. Correctable Messages. The "fix" option (-f) enables you to take corrective action when a correctable error is detected. Run CKDB to see if any fixable errors have occurred, set -f and then run it again to have the corrections applied.
3. Error Message Presentation. Each error message is presented in three parts as illustrated below:

ERROR	DESCRIPTION	ACTION
ERROR is the message received on the Console Terminal.	DESCRIPTION describes what the error message means. Not all error messages need a description.	ACTION shows what to do if recovery is possible. If you see an error that does not list a correction technique, call **BLI** before taking drastic action.

1. CORRECTABLE ERRORS

ERROR	DESCRIPTION	ACTION
Allocation state bit of %P is incorrect	The allocation state bit (010 in blockalloc. mode) is used during the create index process to distinguish the "old" index and data pages from the new index and data pages being sorted. After the create has finished, the ALSTATE bits of all the data pages should be the same as the equivalent bit (02000) in relation.status.	CKDB corrects this by setting the ALSTATE bit of an index or data page to the corresponding value in the relation or indices' relations. This may not be correct if it actually is the relation relation which is incorrect.
Tuple number is incorrect: %P offset %d has %o, should be %o	This error is specific to the blockalloc relation because its tuples never move about on a page. Thus, they always have the same structure; a simple array of 6-byte records in ascending order of tuple number. For instance, the blockalloc tuple at offset 158 must always be tuple number 030 no matter what the configuration of the drive or how it was formatted.	The tno of the tuple is changed to be its "index" into the array of blockalloc tuples that should be on the page.

ERROR	DESCRIPTION	ACTION
Relstat says it is one-zone, but it isn't	The one-zone bit (040000) in relation.status says that all of the data and index pages on the relation or file are in the same zone. However, while scanning the relation, CKDB reads pages of the object that were not in that zone. The only effect of this error is that if the object is destroyed then some pages will not be deallocated.	It is corrected by turning off the one-zone bit. This will increase the time it takes to destroy this relation, but it will correctly free all of its pages.
Lowzone is %P, should be %P Highzone is %P, should be %P	The allocation code maintains the lowest and highest zones which are used by an object. If the zone ranges are incorrect when the object is destroyed, the pages outside the known range will not be deallocated and would not be re-used.	If CKDB reads a page which is lower than relation.lowzone, or higher than relation. highzone, it can fix the error by extending the range as appropriate.
Root %P has a next page	The root of an index cannot have a "horizontal" page after it. If it does, it should not be the root page; there should be another level above it that points to both pages.	The corrective action clears the next page pointer. The index should be remade.
Last data page %P thinks it has an overflow page	The last data page of a relation has the "has-overflow" bit (04) set in its page status. This bit implies that there are other pages after the current one that have the same clustered index key values.	You can correct it by turning off the "has-overflow" bit.
Lastpage in rel-rel wrong; is %P should be %P	The value in relation.lastpg is incorrect. When CKDB read this page, it noticed that it was not at the end of data page list but there were more pages after it. Relation.lastpg only matters for relations that do not have a clustered index. Appends to "heaps" are made faster by maintaining this pointer to the end of the data pages so that new pages may be added quickly.	CKDB fixes this by setting relation.lastpg to the last data page.
Firstpage in rel-rel wrong; is %P should be %P	The first data page that CKDB found by going through the clustered index was not the page specified in relation.firstpg.	It is fixed by setting relation.firstpg to the first data page accessible through the index.
%P forgot about its overflow page %P	The first page does not have the "has-overflow bit " on, but the second page does have the "has-overflow" bit. This is detected while checking the clustered index.	CKDB fixes its setting the "has-overflow" bit in the first page's status.
Extra index tuples after %d on %P point after last data %P	This error occurs during the recursive scan of an index. CKDB has seen the last of the data pages, but there are still more index tuples to be looked at. The data pages that the "extra" index tuples point to are linked back into the main data page list.	If other errors still show up, then retrieve the data through the index and recreate the relation.

ERROR	DESCRIPTION	ACTION
%P allocated to non-existentrelid %d	The indicated page is allocated to a relation which no longer exists.	The page may be deallocated. If there were any errors in the system catalogues, CKDB will ask for confirmation before freeing the page.

CAUTION

A system error could prevent CKDB from finding an object that exists.

%P allocated to %.12s but not used	The indicated page is allocated to the relation or file but CKDB did not have to read the page during its examination of the object.	
%P refers missing tuple %o	The tuple number table on the specified page says there is a tuple of a certain number on the page, but CKDB did not find the tuple during its scan of the page.	This is fixed by zeroing out the tuple number table entry, effectively completing the deletion of the tuple.
Nexttno of %P is %o, should be %o	The size of the tuple number table on the given page is incorrect. The correct value is the second number given.	
Tnofree bit of %P is %o, should be %o	The "tnofree" bit in the page status is wrong. It either says that there is a free slot in the tuple number table though the table is full or it says that the table is full when there is an unused tuple number.	It is corrected by making psat agree with the tuple number table.
%H is free, should be marked bad	A bad block was encountered during the scan disks mode (/s) that was marked free in blockalloc.	

NOTE

This can only occur on disks that were formatted with DFU before release 30, i.e., disks without a remap area.

%P allocated to non-existent database %d	During the scan disks mode (/s) CKDB found a zone that was allocated to a non-existent database which used to have the given dbid. This may be caused (in a multi-disk system) by: 1. Taking a drive off-line 2. Destroying a database that was on the drive 3. Putting the drive back on-line	It is fixed by freeing the zone.

ERROR	DESCRIPTION	ACTION
%P says tuple %o is at %d, should be %d	There are two ways to find a tuple on a page. If you have tid, then the tuple number indices into the tuple number table which is on each data page. This gives the offset from the start of the page to the tuple. The other way to find a tuple is to start at the beginning of the page and do either a linear or a binary search down the page until you find it. This error means that the tuple number table says the tuple is at the first given offset, but the tuple is actually at the second one. It is more likely for the second offset to be correct.	

2. HARMLESS ERRORS

Small tuples were not found on %P, but it should have them	The header of each page contains a copy of the relation.status bit which says whether or not a relation has small tuples (value of 020). The page header's small tuple flag did not agree with the one in the relation relation.	
Small tuples were found on %P<< but it should not have them	This is a harmless error if the tuples are of fixed length; however, it is severe if the relation has any compressed attributes. **CAUTION:** If this error seems to be occurring on every data page, it is usually the relation tuple that is wrong. You can calculate the correct value for the bit by looking at relation.maxwidth or the appropriate tuples in attribute and summing up the sizes.	It may be corrected by copying the small-tuple bit value from the relation relation to the page.
Tuple %T refers to non-existent relid %d	At the end of a full database scan (neither the -r nor the -b option was specified), CKDB looks at the tuples in the attribute, indices, protect, query, crossref, and descriptions relation. If it finds any tuples which contain a relid which is not in the relation relation it prints this message.	
%P allocated to %.12s but not used	The page is allocated to the indicated relid, but CKDB did not have to fetch the page during its examination of the relation. This is harmless because the only side-effect is that it wastes space; besides this, the relation is consistent.	
Index page %P has tnofree set	The "tnofree" bit was set in the page header of an index page. This bit is only meaningful on data pages, so it is an interesting, if harmless, condition if it is set on an index page.	

ERROR	DESCRIPTION
%H is allocated in blockalloc but not used by the disk	A blockalloc tuple in the remap region of a disk says that the page is allocated for remapping, but the physical disk header itself indicates that it is not currently being used. This could be changed to a correctable error if it happens frequently enough that it is bothersome. The main undesirable side-effect is that the page is wasted and cannot be used if a block goes bad in the "normal" region of the disk.
It should be redefined	The size of the query buffer can be changed by a K tuple in the configure relation. A stored command or stored program cannot run if the query buffer at the time of execution is smaller than the query buffer at the time of the query is defined. If you tried to use the indicated query, the BL700 would respond with error 192 (redefine the stored command/program). Views are not dependent upon the size of the query buffer; they do not become out-of-date.
%H is not remapped to %H	On a remap-zone disk, a bad page points to a good page in the remap area. In addition, the good page points back to the bad page in the normal region of the disk; they both point to each other.

3. USER CORRECTABLE ERRORS

These problems can be corrected by IDL or Copy commands. Usually no data is lost by one of these errors. Many are fixed by one of the following procedures, as noted in the ACTION column.

Remake Indices.
Create the clustered index without destroying it first. This destroys any non-clustered indices as well. You will have to create them again.

Remake Index.
Destroy the index specified in the error message and create it again. It is not sufficient to simply recreate the index with the "with recreate" option.

Copy.
Copy out the relation destroy it, and copy it back in again. Stored commands, stored programs, and views defined on the relation will have to be destroyed and re-created.

Copy Through Index.
A simple copyout will not work, since it follows the data pages list. It will not retrieve the data which is accessible only through the index. Do a retrieve, but send the appropriate PLAN options so that the index specified in the error message will be used.

ERROR	DESCRIPTION	ACTION
Tuple %T is inconsistent mode %o relid %d	When checking the blockalloc relation at the start of a database examination, CKDB found an allocation tuple which contains inconsistent information. Either the mode attribute said that the page was free and the relid said that the page was used, or the mode said the page was used and relid was zero, indicating that the page should be free.	COPY
Index page %P is linked vertically but not horizontally	The index page indicated is not correctly linked into the b-tree structure of the index. An index has child pointers which point down in the tree, towards the data pages; associated pointers point sideways to other index pages at the same level above the data pages. CKDB detected a page that was only accessible via the horizontal pointers, but not the vertical pointers.	
%P was linked horizontally but not vertically	Self-evident.	Remake Indices
[Data|Index|File] page %P is referenced more than once	The page is used in two different contexts in the relation. It might be used as both an index and a data page.	Index Remake Indices Data or File Probably at least a page's worth of data has been lost. COPY
[Data|Index] page %P is from a file	The "file data page" (0100) bit should be on in blockalloc.mode if the page is a file data page. A page belonging to a normal relation, a transaction log, or the clustered index on a file should never have this bit set. It is most likely that the bit was accidentally turned on in blockalloc.	Index Remake Index Data COPY
[Data|Index] page %P is both an index and a data page	The page is used both as a data page and as part of an index.	COPY
Data page %P is not in data list	CKDB could get to the page only through an index, though all data pages should be in the data pages list. The index points to more tuples than the data.	COPY THROUGH Index
Disk_usage tuple %D to %D is a subset of the fragment from %D to %D	The tuples in disk_usage should correspond to the range of sequential blockalloc pages. This error means that disk_usage and blockalloc disagree on the size of this particular fragment. Disk usage thinks that it is smaller than it actually is in blockalloc. A dump of this database might not be loadable due to the conflicting information.	This probably should be a severe error!

ERROR	DESCRIPTION	ACTION
Found %D data tuples in index; should have %D tuples	The number of tuples retrievable through the index was not the same as the number that CKDB saw when it was searching the data pages. If the index finds more tuples than are in the data, you would have at least one error message "Data page %P not in data page list". That is the actual error.	If the index has fewer tuples, Remake Indices.
Lesser key @%d in %P is larger than first key in child index page%P	The index is no longer sorted correctly.	Remake Index
Greater key @%d in %P is smaller than last key in child index page%P	The index is no longer sorted correctly.	Remake Index.
Index tuple @%d on %P points to overflow page %P	Clustered index tuples should only point to main (not overflow) data pages. The index is not consistent and will not retrieve all the tuples. This can only occur on a clustered index.	Remake Indices
Index tuple @%d on %P points to %P should point to %P	When CKDB reads "i1", it remembers the data page that the next index tuple "i2" should point to. In this case, A says that i2 should point to B. However, i2 actually points to C.	Clustered index only Remake Indices
Data %P thinks %P pstat %o is overflow	The first page has the P_HASOVER bit (04) on in its page status. Such a page should always be followed by a P_ISOVER (010) page which may also have the P_HASOVER bit set.	Clustered index only Remake Indices
Extra data page %P after end of clustered index	The last index tuple should point to the last data page. However, the next page pointer in the data page header pointed to another main data page.	Clustered index only Remake indices
Index tuple @%d on %P points out a data page to data tuple	The index tuple points to a data page that was not seen when CKDB read the data pages list. This probably means that there are tuples missing from the data pages which are only accessible through a nonclustered index.	Non clustered index only COPY THROUGH Index
Index tuple @%d on %P has no data tuple %T	Each nonclustered index tuple should point to a data tuple which has the same attribute values as the key fields of the index. CKDB could not find the desired tuple; there was no tuple with that tuple number on that data page.	Non-clustered index only Remake Index.

ERROR	DESCRIPTION	ACTION
Index tuple @%d on %P does not match its data tuple %T	Each nonclustered index tuple should point to a data tuple that has the same attribute values as the key fields of the index. This error means that the data tuple did exist but its values did not match.	Non-clustered index only Remake Index
Rel-rel says maximum relation width is %d; should be %d	The maximum width in relation.maxwidth is not the same as the width computed by adding up the space needed by each attribute of the relation.	COPY
Rel-rel says maximum relation width is %d; this is larger than BL700 maximum %d	Somehow a relation was created which has a maximum tuple size which is larger than the limit that the BL700 supports.	COPY the relation, but either split the relation up into two or more relations or reduce the lengths of compressed attributes.
Rel-rel says it has [big|small] tuples	One of the bits in relation.status (020) indicates whether or not the maximum tuple width is less than 256 bytes. If this bit does not agree with the value of maxwidth that CKDB computed, this error is printed.	COPY
Rel-rel says tuplen is %d; should be %d	The field relation.tuplen was not the same as the value that CKDB computed using the information in the attribute relation.	COPY
Empty index page %P on level %d	There are not supposed to be any empty index pages in the upper levels of an index.	
Duplicate keys in unique/clustered index on %P	Index tuples which contained the same key values were found in a clustered or unique non-clustered index. Duplicate keys are only allowed in non-unique, non-clustered indices. This error is not an error in the data tuples; it is complaining about the actual index tuples that make up the B-tree structure of the index.	Remake Index
Identical tuples on %P	A pair of completely identical index tuples (including tids) were found. The offending tuples are printed after the error message.	Remake Index
Tids of identical keys out of order on %P	In a non-unique, non-clustered index tuples which have identical key values should be sorted according to the tid of the data to which the index tuples refer.	Non-clustered only Remake Index
Index keys out of order on %P	The tuples of an index are supposed to be sorted according to its keys. The offending tuples are printed after the error message.	
Freeoff points in middle of last tuple @%d on %P	The header of the indicated index page is incorrect. It says that the end of the useful index tuple space on the page is actually in the middle of the last index tuple.	Remake Index

ERROR	DESCRIPTION	ACTION
Data tuples @%d and @%d on %P have the same keys	A relation which has a unique clustered index has two data tuples with the same keys. This is a violation of the uniqueness criterion.	Remake Indices
Duplicate tuples @%d and @%d on %P	CKDB found two adjacent identical data tuples in a relation which has a clustered index.	Remake Indices
Data tuples out of order @%d and @%d on %P	Data tuples are supposed to be sorted according to the keys of the clustered index, if any exist. The BL700 may not be able to find tuples that are improperly positioned.	Remake Indices
Tuple @%d on overflow page %P does not match previous tuple @%d on %P	All tuples on overflow pages must have the same values in their key fields. These attributes are the key fields of the clustered index.	Remake Indices
Found data tuple @%d on %P which is smaller than index tup @%d on %P	The parent index tuple points to a child data tuple which is smaller than it. The data tuple may not be found when searching through the parent index.	
Found data tuple @%d on%P which is larger than index tup @%d on %P	The tuple after the parent index tuple points to a child tuple which is larger than it. The data tuple may not be found when searching through that index.	Remake Index
Leaf & data set in %P	Two bits were set in the indicated page's header that are never supposed to be on at the same time. The leaf bit says the page is in the bottom page of a nonclustered index. The data bit says the page is in the data pages list of the relation.	Remake Indices
%P is not linked into %.12s	This message is only printed by the disks scanning function of CKDB (/s). It means that the indicated zone thinks it is allocated to a particular database, but the database's disk_usage relation does not contain that zone.	
%P on level %d, should be %d	CKDB found the indicated page to be on the first numbered level, but it should have been on the second given level. Data pages are always level zero; index page levels vary from 0 to 10. This error can be encountered almost any place in CKDB.	
Index page %P links level %d to %d	One of the horizontal index page pointers points to a page that is on a different level.	Remake Indices

ERROR	DESCRIPTION	ACTION
Index status bit %o not compatible with index %d	There are two ways that identify an index as being clustered or nonclustered. One is the 02 bit in the indices.stat attribute, which says that it is a clustered index. Also, clustered indices always have an index id of 0. This error occurs when the status bit disagrees with the index id; e.g. the bit is 2 but the index id is nonzero.	Remake Indices

4. SEVERE ERROR MESSAGES

Relation tuple expected it not to have a [non]clustered index, but one was found	A pair of bits in relation.status indicate whether the relation has any clustered (02) or non-clustered (04) indices. These errors mean that the indices that should have been there were not found in the indices relation or there were unexpected indices that relation.status did not mention.	
Relation tuple expected it to have a [non]clustered index but one was not found	Destroy the indices COPY	
Small tuples were not found on%P, but it should have them	The header of each page contains a copy of the relation.status bit which says whether or not a relation has small tuples (value of 020). The page header's small tuple flag did not agree with the one in the relation relation. **CAUTION:** If numerous page headers are diagnosed as being wrong, it probably is the relation relation which is incorrect.	It is corrected by copying the relation relation value to the page.
%P allocated to %d	The indicated page is allocated to the relid given in the message and not to the relation that is using the page. This check is performed before the page is actually read. It may or may not contain the correct relid.	

ERROR	DESCRIPTION
Allocation error:%P is bad, mode %o	The blockalloc tuple of the indicated page says it is a bad page. This means that the page probably is not consistently readable, so it should not be used.
Allocation error:%P is free, mode %o	The blockalloc tuple of the indicated page shows it is not allocated to any relation. Thus, a process which wanted another page might take this one, since it says it is available. Once this page is re-used, a "syserr: bufrd" may occur.
Allocation error:%P is not used, mode %o	The blockalloc tuple of the indicated page says it is not used. The page is neither free, used, nor involved in remapping. This condition is illegal whether or not the page is being used by a relation.
Allocation error:%P is (used)-remapped, mode %o	The blockalloc tuple of the indicated page says it is an alternate page used in remapping. However, there are not supposed to be any pointers to the alternate "target" page, only to the original bad page.
Allocation error:%P is demanded, mode %o	The blockalloc tuple of the indicated page says it has been "hard" allocated. This is the "demand" allocation state that has not been implemented in the BL700.
File page %P is allocated to %d	A data page from a file is allocated to another relation.
File page %P is not allocated as file data	Each data page of a file should be allocated in blockalloc. The mode attribute should contain the file data page (0100) bit.
Disk_usage tuple low %D high %D does not match blockalloc	Before examining a database, CKDB reads blockalloc and constructs a fragment table. It should correspond to tuples in disk_usage with relid = -32768.
[Data|Index] tuple %T has an unsorted attribute table	A relation with compressed fields has an attribute offset table at the beginning of each tuple. It contains the position and lengths of each of the variable length fields. The BL700 requires that this table be sorted in nondescending order.
Freeoff %d on %P	Freeoff is one of the elements of the page header. It contains the offset from the start of the page to the end of the used tuple space on the page. This error occurs when the offset points off the page. It is either less than the size of the page header or greater than the size of the page. It probably should be a fatal error.

ERROR	DESCRIPTION
Data area of %P extends into tuple table freeoff %d	Freeoff marks the end of the used tuple space on a page. This error means that the two ways of finding the amount of used space on a page disagree.
Tuple %o is used both @%d and @%d on %P	Each data tuple on an BL700 is uniquely identified by combination of the page number that it is on and its tuple number on that page. The indicated page had two different tuples which claimed to have the same tuple number. CKDB cannot determine which one, if either, is correct, so it skips the rest of the relation.
Size of data tuple @%d on %P is %d	The indicated data tuple had an illegal size; either negative or larger than the largest allowable tuple size for the relation (relation.maxwidth).
%P contains a tuple @%d marked as deleted	The tuple space of a page and the tuple number table of the page disagree on which tuples are present. The tuple number table indicates a deleted tuple which is still present on the page.
%P has tuple %o @%d >=nexttno %o	The tuple space of a page contains a tuple number which is higher than the largest tuple number that the page header knows about.
%P has large index tuples	An index page always contains small <256 byte) tuples. The indicated page header did not have the small-tuple bit on.
Can't read disk header of %H	CKDB could not read the physical sector header of the indicated cylinder/head/sector. The IDM/RDBMS code needs to be able to read the physical sector header of every page on a drive, even though the actual data portion of a page may be unreadable.
%H is remapped on the disk but not in blockalloc	During scan disks mode (/s), CKDB found that the physical sector header says the page is remapped but blockalloc does not. This can cause the page to appear to be free and be re-allocated which may eventually generate a "syserr:bufrd."
%H is not mapped into the reserved region but to %H	During scan disks mode (/s), a page was found to be incorrectly remapped. The page might seem to be "free" and could be re-used by another relation. It is also possible that load database could fail. This error only occurs on disks formatted by release 30 DFU or later.

ERROR	DESCRIPTION
%H badly mapped into remap region at %H	During scan disks mode (/s), the remap pointer in the remap zone points to a bad page (in the virtual region of the drive) which does not point back to the remap zone page. (A points to B but B does not point to A, though A does point into the remap region.) This error only occurs on disks formatted by release 30 DFU or later.
Can't read blockalloc page %H	The indicated page has a hard disk error. CKDB cannot check the pages in this zone.
Expected relid 11 on %H but found relid %d	CKDB could read a blockalloc page but it did not have the correct relid on it.
Can't read disk header for remap page %H	The physical sector header of the page was not readable.
%H is remapped into the remap region	A page in the remap region of a drive is remapped to another page in the remap area. This is not allowed; bad pages in the remap area are marked bad and are unused.
%H is remapped off the drive	The physical sector header of the page contains an alternate cylinder address which is larger than the number of cylinders on the drive.
%H erroneously says %H is bad	The first page (which is in the remap zone) points to the second page in the virtual area of the disk, but the second page is not remapped at.
Blockalloc thinks that %P is remapped	The blockalloc tuple of a page thinks it is remapped (mode has the 020 (bad) and 0200 (remapped) bits on), but the physical sector header says the page is not remapped.
%P mode of bad & remapped on a remap-zone drive	Drives that have been formatted with release 30 or greater DFU should have neither the bad (020) nor the remap (0200) bits on any page's blockalloc tuple.
%P remapped from database %d to %d	On a drive that was formatted with a version of DFU before release 30, a page was remapped from one database to another. Only blockalloc can be remapped across databases.

ERROR	DESCRIPTION
%P remapped to %P which isn't used and remapped	On a drive that was formatted with a version of DFU before release 30, a page was remapped to another page but the blockalloc tuple of the second page did not say that it was used in remapping.
%P remapped to already remapped page %P	On a drive that was formatted with a version of DFU before release 30, a page was remapped to another page which was already being remapped to by yet a third page.
%P remapped to relid %d on %P	On a drive that was formatted with a version of DFU before release 30, a page was remapped to another page which was allocated to a different relation. Both the original (bad) page and the alternate (good) page should have the same relid in their corresponding blockalloc tuples.

5. FATAL ERROR MESSAGES

%o blockalloc tuple for %P	The blockalloc tuple for the indicated page did not exist on the page that should have contained it. This usually means the page was trashed.
%P belongs to %.12s dbid %d	The page did not belong to the database CKDB was checking. The zone in which the indicated page belongs is allocated to the given dbid. A database id of zero means that the zone is free.
%P not in blockalloc and disk_usage	The indicated blockalloc page said that it belongs to the correct database but neither disk_usage nor the real blockalloc admit to knowing about it.
Loop @ %P	CKDB discovered a loop in the linked list of data pages for blockalloc. Because CKDB needs to use blockalloc for recording state information about each page, it cannot continue the examination of the database. Prior to the error message, CKDB prints out the 50 fragments it already found.
%P remapped to non-existent page %P	The first page in the message is allegedly remapped to the second page which is not on-line. This is a fatal error because pages can only be remapped to a different cylinder/head/sector combination on the same drive. For this error to occur, the cylinder number in the physical disk sector header must be larger than the number of cylinders on the drive.

ERROR	DESCRIPTION
%P in blockalloc remapped to %P relid %d	A blockalloc page is remapped to a page which is not allocated to blockalloc but to relid %d. This message occurs when the page is remapped to another database which is only legal for blockalloc pages.
Loop in data pages:%P points to %P	One of the pages in the data pages list of a relation points back to an earlier page in the list. CKDB can do nothing more with this relation.
Loop in rel-rel	There were more than 32,767 tuples in the relation relation; CKDB diagnoses this as a loop in its data pages.
It has no attributes	A relation or log has no tuples in the attribute relation. CKDB can do nothing with it.
No root page for nonclustered index	It is possible for clustered indices to not have a root index page if there is still one data page. However, nonclustered indices must always have a root page. CKDB skips this index.
Empty page %P	An empty page was found. This is a fatal error because there are no child page pointers to follow. CKDB can go no further. There is another message similar to this one in sort.c. CKDB skips the rest of this index.
It has no query text	Stored commands, stored programs, and views do not have any data in pages on disk. Their contents are stored in the system query relation. Attempting to run the query will result in a syserr "readqry: impty cmd" in the main IDM/RDBMS code. Destroy the object and redefine it so the syserr will be prevented.
Data page %P points to index page %P	While reading the data pages list, CKDB found an index page. The variant form of this message means that the first page of the relation (relation.firstpg) was wrong. CKDB can do nothing more with this relation.
%P is not a data page	CKDB was verifying the clustered index and the data pages of a relation and found an index page where it expected to find a data page. This means that what was supposed to be the bottom level of the index actually turned out to have another index page below it. This is caused by a random child pointer that happens to point into one of the indices of the relation.

ERROR	DESCRIPTION
Found data page %P in index	While reading the pages of an index, CKDB found a data page.
Can't read blockalloc page %P	Dump page mode (/d) tries to find the blockalloc tuple of each page that is read so that it can print out any interesting information about it. This message is printed if there is a hard disk error on the page or if the page is so trashed that the appropriate blockalloc tuple cannot be found. The original page requested will still be read, though its allocation information will not be available.
More than %d attributes	A relation had more than 250 tuples in the attribute relation. Since 250 is the maximum number of attributes a relation may have, CKDB does not handle storing any more.
No attribute %d	CKDB could not find the indicated attid in its memory array of attributes for the current relation.
No more descriptors	CKDB wanted to open another relation but it could not because it had too many already open. This may be caused by excessive nesting of commands, such as running (/r) from inside a (/q) from inside (/d) under certain exceptional error conditions. It does not imply a fatal error in the Database; just reboot and run CKDB again.
Can't continue	In several places CKDB scans all of blockalloc to look for certain information. It cannot do this if there is a fatal error in blockalloc.
%P is remapped to a non-existent block	The blockalloc tuple of the page to which the indicated page is remapped cannot be found. Either an unreadable or trashed blockalloc page may be the cause. It is also possible that the page is remapped off the drive which must be caused by an excessively large cylinder number in the physical block's remap header.
Asked for %P, got %P	The first page number was read, but the data had the page number of the second page. This can be caused by a remapping error or by a page being written out to the wrong location.
%P has wrong relid %d	The indicated page should have belonged to the current relation, but the relid on the page says that it belongs to another. The main IDM/RDBMS code would get "syserr:bufrd" if it tried to read this page.

ERROR	DESCRIPTION
Tid %T in an index	A tid (which can only identify a data tuple) pointed to an index range.
Bad tno %o, pg %P	The tuple number component of a tid was larger than the maximum valid tuple number for the indicated page.
Bad offset %d, tno % opg %P	The tuple number table of the indicated tid pointed off the page.
Index tuple @%d on %P has size %d	The indicated index tuple is smaller than the minimum allowed for that index.
Data tuple @%d on %P has size %d	The indicated data tuple is smaller than the minimum allowed for that relation.
Data tuple @%d on %P has size %d > max size %d	The indicated data tuple is larger than the maximum allowed for that relation (relation.maxwidth).
%P not on-line	CKDB wanted to read the page, but there are no accessible disks which contain it.
tnochk:tno %o on %P	The tuple number component of a tid was larger than the maximum valid tuple number for the indicated page.
tnochk:off %d on %P	The tuple number table of the indicated tid pointed off the page.
tuple extends beyond end of page	While looking at a page in either (/d) or (/t) mode, CKDB found a tuple which was smaller than the maximum size of tuples for the relation but which started so far into the page that it overflowed onto the next page.