AUTOPSY STUDENT REFERENCE MANUAL

PRIME COMPUTER INC.

PRIME PARK

NATICK, MA. 01760

CREDITS

Course Developer. . . . . .Nick DiFabio

Table of Contents

1 STUDENT REFERENCE MANUAL OUTLINE

INTRODUCTION TO AUTOPSY (PE-T-484)

- INITIALIZING AUTOPSY
- EXAMINING DUMPS USING AUTOPSY COMMANDS
- INTERPRETING AUTOPSY DATA (PRIMOS INTERNALS GUIDE)

PRIMOS DEBUGGING USING AUTOPSY

- PRIMOS SUBROUTINE DEFINITIONS
- USING PRIMOS SOURCE LISTINGS
- SYSTEM CONFIGURATION INFORMATION
- CRASH DUMP DEBUGGING APPROACH

TROUBLESHOOTING FLOWCHARTS

- MACHINE CHECK HALTS
- MISSING MEMORY MODULE HALT - MMOD_
- SYSTEM HANGS
- LABELED HALTS - BOOT0/PAGES_/IPAGF/PGMPA_

INTRODUCTION TO DOC

- DOC USER'S GUIDE
- DOC SITE ADMINISTRATOR'S GUIDE

PRIMOS DEBUGGING USING AUTOPSY

- PRIMOS SUBROUTINE DEFINITIONS

- USING PRIMOS SOURCE LISTINGS

- SYSTEM CONFIGURATION INFORMATION

- CRASH DUMP DEBUGGING APPROACH

2 PRIMOS SUBROUTINE DEFINITIONS

2.1 KS-ROUTINES

ABSW.PLP	Routine to read ABBRSW in FIGCOM for Ring 3.
ACCOM$.PLP	Access cominput info. In PUDCOM for Ring 3 procedure.
AD$PAR.PLP	Parse the ADDISK/SHUTDN command line.
AD_CMD.PLP	Process the ADDISK command.
AINIT.FTN	Cold Start initialization (part 1).
ALIPQC.PLP	Process ASYNC line config. Changes for LYNX (ICS)/HAWK.
AMINIT.PMA	Initializes AMLC controller(s).
AMLC$.FTN	Process internal command AMLC.
AMLC$.PLP	Process internal command AMLC.
AMLDIM.PMA	Processes AMLC input and output.
APROTO.PLP	Select protocol for an ASYNC line.
ASNDE$FTN	Assign disk and other peripheral devices except magtape.
ASNLN$.PLP	Assign and unassign ASYNC lines.
ASNMT$.PLP	Assign magtape drive units.
ASRDIM.PMA	Clock driven ASR driver (Option-A).
ASSUR$.PLP	ALLOW A USER PROCESS TO ASSURE IT HAS A CERTAIN AMOUNT OF CPU TIME LEFT.
ASYEND.PMA	LOCATES END OF NEW ASYNC SUPPORT MODULES.
ASYINI.FTN	INITIALIZE ASYNC FUNCTIONALITY ON NEW COMMUNICATION CONTROLLERS.
ASYIPQ.FTN	PERFORM ROIPQN INITIALIZATION ON BEHALF OF CONTROLLER ASY PROCESS.
ASYNDM.PMA	PROCESS ASYNC I/O FOR NEW COMMUNICATION CONTROLLERS.
ASYNOK.PMA	INFORM THE ASYNC DIM THAT A CONTROLLER IS OK TO BE USED.
ATSH1.PLP	LINK TO A SEGMENT SET UP BY MKSH1$
ATSHR$.PLP	ATTACH TO A SEGMENT ALLOCATED BY GTSHR$
AU$CUR.PLP	ACCESS CURRENT LOG ENTRY FOR A GIVEN USER.
AU$DRN.PLP	SHUT DOWN AN AUSLOG PHANTOM.
AU$GET.PLP	RETURN COPY OF CURRENT LOG BUFFERS FOR LOG UTILITY.
AU$START.PLP	START UP AUSLOG UTILITY PHANTOM.
AU$STAT.PLP	SHOW CURRENT STATUS OF AUSLOG PHANTOM.
AU$TSK.PLP	ASSEMBLES AUSLOG LOGIN/LOGOUT MESSAGE TYPES BEFORE LOGIN.
AU$WRT.PLP	WRITE TO AUSLOG LOG FILE & WAIT FOR A DATE BUFFER.
AUSCOM.PMA	AUSLOG COMMON
AUSLOG.PLP	AUSLOG BUFFER MANAGER ROUTINE.
AU_ALLOW.PLP	IS CALLER CONSOLE, ADMIN, OR USER WITH SAME NAME AS USER 'n'.
BADDSK.PLP	CHECK FOR LEGAL PRIMOS DISK NUMBER.
BADGAT.PMA	BAD GATE HANDLER.
BCKUPB.PLP	BACK UP RETURN PB FOR RING 0 RESTART.
BFGETR.PMA	BUFFERING PACKAGE USED BY MPCDIM, VERDIM.
BINIT.FTN	COLD START INITIALIZATION (PART 2).
BIT_SUBS.PMA	BIT MANIPULATION ROUTINES FOR PLP ASSISTANCE.
BRCONV.PMA	CONVERT BAUD RATE SELECT ENCODING FROM AMLC TO LYNX (ICS) FORMAT.
BREAK$.PMA	MANAGE QUIT INHIBIT COUNTERS FOR ALL RINGS.

BRPDIM.FTN	PAPER TAPE PUNCH DIM.
BTPCC.PLP	BOOTS A SINGLE PROGRAMMABLE COMMUNICATION CONTROLLER.
BTPCC$.PLP	BOOTS ALL PROGRAMMABLE COMMUNICATION CONTROLLERS.
C1IN$.PLP	SINGLE CHARACTER COMMAND INPUT
C1IN.PLP	SINGLE CHARACTER INPUT.
CCPAT.PMA	DECLARATION OF COMM CONTROLLER PHYSICAL ATTRIBUTE TABLE.
CCPTIX.PLP	RETURN THE INDEX INTO THE CCPAT TABLE FOR A GIVEN DEVICE ADDRESS.
CHANGE_UID.P	CHANGE PROCESS'S UNIQUE ID.
CHAP.FTN	PROCESS CHAP COMMAND FOR SETTING PROCESS PRIRORITIES AND TIMESLICE VALUES.
CHG$PW.PLP	CHANGE THE USERS LOGIN PASSWORD.
CHG$SA.PLP	CHANGE SYSTEM ADMINISTRATOR.
CHGPRI.PLP	CHANG A PROCESS'S PRIORITY LEVEL
CHKABT.PMA	HACK MODULE TO CHECK FOR ABORTS STILL IN THE PCB AND PROCESS THEM.
CINIT.FTN	COLD START CONFIGURATION.
CMC$ST.PLP	LIST COMMUNICATIONS CONTROLLER STATUS.
CMREA$.FTN	OLD STYLE COMMAND LINE PARSER.
CNEQV.PMA	NAMEQV-COMEQV COMPARE ASCII NAMES.
CNFLCT.FTN	CHECK FOR CONFLICTING PRIMOS PARTITIONS.
COMINI.PLP	INITIALIZE THE COMMS SUBSYSTEM AT COLD/WARM START.
COMMS0.PMA	STATIC SEG 0 ALLOCATIONS FOR COMMS CONTROLLERS.
CPS$.PLP	CROSS PROCESS SIGNALING SEND SIGNAL ROUTINE.
CPS$CA.PLP	CROSS PROCESS SIGNALING CLEAR A USER FROM ALL ACL.
CPS$CN.PLP	CROSS PROCESS SIGNALING CONTROL ROUTINE.
CPS$CU.PLP	CROSS PROCESS SIGNALING CLEAR A USERS USER SIGNALED LIST.
CPS$DF.PLP	CROSS PROCESS SIGNALING DEFER SIGNAL ROUTINE.
CPS$IN.PLP	CROSS PROCESS SIGNALING INITIALIZATION ROUTINE.
CPS$NA.PLP	CROSS PROCESS SIGNALING NAME ROUTINE.
CPS$RC.PLP	CROSS PROCESS SIGNALING SIGNAL RECEIVED ROUTINE.
CPS$RG.PLP	CROSS PROCESS SIGNALING REGISTRATION ROUTINE.
CPS$SN.PLP	CROSS PROCESS SIGNALING WHO SIGNALED ROUTINE.
CPS$ST.PLP	CROSS PROCESS SIGNALING STATUS ROUTINE.
CPUID$.PMA	RETURN THE CPU ID AND MICROCODE REVISION NUMBERS.
CRDDIM.PMA	CARD READER DRIVER.
CSTAK$.PLP	MANIPULATE/EXAMINE THE CALLING PROCESS'S CONCEALED STACK.
DATE$.PLP	RETURN THE STANDARD (FS) FORMAT DATE AND TIME.
DELAY.PMA	SET SLOPE OF DELAY CURVE FOR TERMINAL.
DEVCHK.FTN	CHECK EXTERNAL DEVICE ASSIGNMENT.
DISKIO.PMA	DISK I/O FOR PRIMOS.
DMQSET.FTN	SET-UP DMQ CONTROL BLOCKS AND BUFFERS.
DOSSUB.FTN	COMMAND LINE PROCESSOR FOR PRIMOS IV.
DROPD_D.PLP	INVOKE THE DROP DTR COMMAND FROM RING 3.
DRPDTR.PLP	DROP THE AMLC OR ICS LINE DTR FOR A DESIRED USER.
DSKCHN.PMA	DISK CONTROLLER CHANNEL PROGRAMS.
DSKEQV.FTN	CHECK FOR SAME PARTITION OR OVERLAPPING PARTITIONS.
DUPLX$.FTN	SET/RETURN TERMINAL CONFIGURATION WORD.

ENCRYPT$.PLP	ENCRYPT A USER'S LOGIN PASSWORD AS IRREVERSIBLY AS PRACTICABLE.
EPF_PROFILE.PLP	ROUTINE TO RETRIEVE EPF RELATED DATA FROM USER PROFILE.
ERKL$$.FTN	SET ERASE AND KILL CHARACTERS FOR USER.
ERRCOM.PMA	STANDARD ERROR MASSAGE TABLE.
ERRPR$.FTN	PRINT SYSTEM ERROR MESSAGE.
ERRRTN.FTN	ERROR RETURN HANDLER FOR PRIMOS IV.
ERTXT$.PLP	RETURN THE TEXT OF A SPECIFIED ERROR CODE.
EXTLOG.PLP	RESTORE THE EXTERNAL LOGIN/LOGOUT PROGRAM.
FATAL$.PMA	FATAL PROCESS ERROR.
FBT.PMA	DEFINE BUFFER AVAILABILITY TABLE FOR ASSIGNED ASYNC LINES.
FILPAG.PMA	FILL PAGE WITH ZEROS.
FIND_SEG.PLP	RETURN A VECTOR OF FREE SEGMENT NUMBERS.
FORKW.PLP	FORK SEMAPHORE DATA ABSTRACTIONS.
FRK$CP.PLP	ADDRESS COPY ROUTINE FOR FORKED PROCESSES.
GATE_HTB.PMA	GATE HTB TABLE.
GATE_INIT.PLP	INITIALIZE GATE SEGMENT.
GATE_TABLE_HA	RING 0 GATES ENTRIES FOR PRIMOS IV.
GCHAR.PMA	GET CHAR FROM ARRAY, STEP CHAR PTR.
GEM$PB.PLP	A GATE ROUTINE TO CALL PROBE TO MONITOR RING 3 ACTIVITIES.
GEM$R3.PLP	A GATE ROUTINE TO CALL PROBE TO MONITOR RING 3 ACTIVITIES.
GEM$ST.PLP	CONTROL PROCEDURE FOR GENERAL EVENT MONITOR (GEM).
GEM$WT.PLP	GATE ROUTINE TO WAIT FOR AND DUMP GENERAL EVENT MONITOR AND BUFFERS.
GEMCOM.PMA	COMMON DEFINITIONS FOR GENERAL EVENT MONITOR (GEM).
GETAT$.PLP	READS SYSTEM DEFAULTS AND PASSES THEM TO EDIT PROFILE.
GETID.PMA	INITIALIZE CONTROLLER AND FETCH ID GIVEN A DEVICE ADDRESS.
GETSO.PLP	THESE RETURNS MANAGE THE ALLOCATION OF SEG 0 FOR IPQN BASED DEVICES.
GETSEG.FTN	ALLOCATE A PAGE MAP FOR A NEW SEGMENT FOR SPECIFIED USER.
GETSN$.PLP	RETURN A VECTOR OF ALLOCATED SEGMENT NUMBERS.
GET_PCCIV.PLP	ROUTINES TO MANAGE PHANTOM INTERRUPT CODE DYNAMICALLY FOR PROGRAM COMMUNICATION DEVICES.
GET_SANAME.PLP	READ SA NAME FROM SAD INTO SUPCOM.
GMETR$.PLP	GET METERING DATA OF VARIOUS SORTS.
GPGREC.FTN	ALLOCATE A PAGING DEVICE INDEX.
GPIDIM.PMA	INTERRUPT PROCESS FOR T$GPPI INTERFACE.
GTCHAN.PLP	GET A DMA OR A DMC CHANNEL.
GTSHR$.PLP	DEFINE AND MAP A DTAR2 SEGMENT ONTO A DTAR0 SEGMENT.
GTWNDO.PMA	ROUTINE TO ALLOCATE SEG0 WINDOWS FOR MAPPED I/O.
HWSTAT.PLP	PERFORM A STATUS HARDWARE COMMAND.
ICS2TCT.PMA	GATE TO ALLOW OTA AND INA FROM ICS2 MONITOR TO CONT.
ICSCFG.PLP	CHECK FOR INCONSISTANCIES IN THE ICS CONFIGURATION.
ICSFP.PLP	INITIALIZES FREE POOL FOR NEW OMMUNICATIONS CONTROLLERS.
IN$LO.PLP	RETURN STATE OF PPMD.IN GRACE PERIOD.
INIT$U.PLP	INITIALIZE A NEW USER.
INSON$.PLP	INITIALIZES A STATIC ON ON-UNIT LISTS.
IOQ$SY.PMA	IOA$ CALL FOR SYSTEM CONSOLE.

IOWIRE.PMA	WIRE/UNWIRE PAGES FOR PERFORMING I/O.
IOWNDW.PMA	OPEN MAPPED I/O WINDOWS.
IPC$C.PLP	CLOSE A IPC MAILBOX USING THE MBX ID SPECIFIED.
IPC$CA.PLP	CLOSE ALL MAILBOXES THE CURRENT USER OWNS.
IPC$CM.PLP	CHANGE MAILBOX ACCESS MODE FROM READ/WRITE TO SPECIFIED MODE.
IPC$GU.PLP	GET THE DESIRED MAILBOX USER ID SPECIFIED BY KEY.
IPC$NC.PLP	CLOSE A IPC MAILBOX WITH NOTIFICATIONS USING THE MBX ID SPECIFIED.
IPC$SO.PLP	OPEN AN IPC MAILBOX FOR SPECIFIED ACCESS USING PATHNAME FOR ACL.
IPC$R.PLP	RECEIVE A MASSAGE FROM SPECIFIED IPC MAILBOX WAITING.
IPC$RA.PLP	RECEIVE A MESSAGE FROM ANY IPC MAILBOX OWNED BY THE USER.
IPC$SA.PLP	SEND A MESSAGE TO ANY IPC USER ATTACH TO SPRCIFIED MAILBOX.
IPC$SB.PLP	SEND A MESSAGE TO ALL IPC USERS ATTACHED TO SPECFIED MAILBOX.
IPC$SS.PLP	SEND A MESSAGE TO A SPECIFIED IPC USER.
IPC$ST.PLP	RETURN VARIOUS IPC STATUSES DETERMINED BY USER SPECIFIED KEY.
IPC_ACKM.PLP	ACKNOWLEDGE A SPECIFIED MAILBOX MESSAGE.
IPC_CKAC.PLP	CHECK ACCESS TO A MAILBOX BY A SPECIFIED KEY FOR A SPECIFIED MAILBOX USER ID.
IPC_CMBX.PLP	CLOSE A MAILBOX FOR THE SPECIFIED MAILBOX USER ID IN THE LOCAL DATABASE.
IPC_CNFY.PLP	CLOSE A MAILBOX WITH NOTIFICATION.
IPC_DB.PMA	DEFINE STATIC STORAGE FOR THE IPC MECHANISM.
IPC_FATAL.PLP	SEND A FATAL ERROR MESSAGE TO THE SYSTEM CONSOLE.
IPC_GIDP.PLP	GET MAILBOX USER ID (MBX UCTL) POINTER.
IPC_GUID.PLP	GET A SPECIFIED USER'S MAILBOX USER ID POINTER.
IPC_GUNM.PLP	GET NEXT MESSAGE FOR SPECIFIED RECEIVER.
IPC_NFYR.PLP	INTERRUPT A SPECIFIED IPC USER BY MAILBOX USER ID.
IPC_SALL.PLP	SEND A MESSAGE OF A SPECIFIED TYPE TO ALL USERS OF A SPECIFIED MAILBOX.
IPQBL.PMA	THE CHEAP PROCESS TO HANDLE BUFFER SERVICE FOR THE INTELLIGENT CONTROLLERS.
IPQBSP.PLP	CHEAP PROCESS TO DO BUFFER SERVICE FOR THE INTELLIGENT CONTROLLERS.
IPQCS.PLP	ROIPQNM INITIALIZATION AND DELETION ROUTINES FOR THE BASIC STRUCTURES.
IPQDEF.PMA	IPQNM COMMON DEFINITIONS.
IPQEND.PLP	THIS MARKS THE END POINT OF THE WIRED CODE FOR ROIPQNM.
IPQICP.PLP	PROCESS TO HANDLE INTERRUPTS FOR THE INTELLIGENT COMMUNICATION CONTROLLERS.
IPQNM.PMA	QUEUE HANDLING ROUTINES FOR INTELLIGENT CONTROLLER PRODUCTS.
IPQPI.PMA	HANDLES INTERRUPTS FOR THE COMMUNICATIONS CONTROLLERS FOR ROIPQNM.

JOB$O.PLP	OPERATES ON BATCH QUEUE CONTROL FILE IN A SECURE MANNER.
LCDEL.PLP	PROCEDURE TO DELETE A LOGICAL CONNECTION FOR THE IPQNM. ROUTINES
LCINT$.PLP	SUBROUTINE TO INITIALIZE A LOGICAL CONNECTION FOR THE IPQNM ROUTINES.
LGINI$.PLP	TURN ON AND OFF OS AND NETWORK LOGGING.
LIMIT$.PLP	SET/READ CPU, REALTIME, AND LOGIN TIME LIMITS.
LISTEN.PLP	RING ZERO (LOGGED OUT) LISTENER.
LOCKPG.FTN	WIRE AN AREA OF THE VIRTUAL MEMORY.
LOGABT.PLP	HANDLE LOGOUT PROCESS ABORTS (FORCED AND TIMEOUTS).
LOGEV1.PMA	FIRST-LEVEL EVENT LOGGING.
LOGEV2.FTN	SECOND-LEVEL EVENT LOGGER.
LOGIN$.PLP	RING ZERO LOGIN COMMAND PROCESSOR.
LOGO$$.FTN	SUBROUTINE TO LOGOUT A USER OR USERS.
LOGO$CP.PLP	LOGGED OUT COMMAND PROCESSOR.
LOGOCMT_.PMA	LOGGED OUT COMMAND TABLE.
LOGOCM_.PLP	DECIDE WHETHER A GIVEN COMMAND IS A VALID LOG OUT.
LOGOUT.PLP	LOGOUT INTERFACE (R3 TO R0) AND MESSAGE SENDER.
LOG_INIT.PLP	RESET PARAMETERS AFTER LOGOUT OR BEFORE LOGIN.
LON$C.PLP	CLOSES A USER'S LOGOUT NOTIFICATION MESSAGE QUEUE.
LON$O.PLP	LOGOUT NOTIFICATION REVEIVER MESSAGE QUEUE OPENER.
LON$S.PLP	LOGOUT NOTIFICATION PHANTOM MESSAGE SEND MODULE.
LOV$SW.PLP	ROUTINE TO READ LOGOVR IN FIGCOM FOR RING 3.
LO_CLEAN.PLP	CLEAN UP AFTER EXTERNAL LOGOUT OR LOGIN ERROR.
LO_FATAL.PLP	MAIN LOGOUT PROCESSOR, CALLED BY LOGOUT AND FATALS.
LUDEV$.PLP	LIST A USER'S ASSIGNED DEVICE.
MAPIO.PMA	LOCK AND MAP (AND UNLOCK) USER BUFERRS INTO SEGMENT 0.
MAPSEG.FTN	MAPS A SEGMENT ALREADY DEFINED IN DTAR 0 TO ANY OTHER SEGMENT.
MEMDAT.PMA	DEFINE MEMDAT (MEMORY USEAG DATABASE) COMMON AREA.
MESSAG.FTN	HANDLE MESSAGE COMMAND.
MESSG$.FTN	HANDLE MESSAGE COMMAND.
MGSET$.FTN	SETS MSG RCV STATE FOR USER.
MINABT.FTN	HANDLE 1 MINUTE PROCESS ABORT.
MKSHL$.PLP	ALLOCATE PURE DTAR2 SHARED SEGMENT.
MMAP.PMA	MEMORY MAP DATABASE FOR PRIMOS MEMORY MANAGEMENT.
MOVE$.PMA	DATA MOVEMENT SUBROUTINES.
MOVSEG.PLP	COPY A CURRENT USER'S SEGMENT FROM ANY OTHER SEGMENT.
MOVUTU.FTN	MOVE WORDS FROM ONE USER'S VIRTUAL ADDRESS SPACE TO ANOTHER USER'S VIRTUAL ADDRESS SPACE.
MP2DIM.PMA	DRIVES LINE PRINTER, CARD READER, CARD PUNCH VIA MPC#2.
MPCDIM.PMA	DRIVES LINE PRINTER, CARD READER, CARD PUNCH VIA MPC.
MSG$.FTN	SEND A MESSAGE TO A USER ON ARBRITARY NODE.
MSG$ST.FTN	RETURN MESSAGE STATUS TO CALLER.
MSGCOM.PMA	MESSAGE COMMON.
MSGOUT.PLP	MESSAGE FACILITY – OUTPUT MESSAGE TO USER.
MTDIM.PMA	DRIVES MAG TAPE VIA MPC.

N1LOCK.PMA	LOCKING ROUTINES FOR PRIMOS.
NCCFPD.PMA	COMMON AREA FOR IMCS FREE POOL ID.
NLKCOM.PMA	NON-WIRED COMMON.
NLOGIN.PLP	MAIN LOGIN ROUTINE FOR NORMAL USERS.
NS4_NTFY.PLP	ROUTINE TO NOTIFY NS4 USRSEM IF PROCESS IS WAITNG OR ABOUT TO WAIT ON A SEMAPHORE.
OERRTN.FTN	OLD STYLE ERROR HANDLING.
ORGO.PMA	SETS LOADER WDNO TO ZERO.
PABAORT.FTN	HANDLE PROCESS ABORT CONDITIONS (NEE SCHED).
PAG$FS.PLP	PAGE TO/FROM THE FILE SYSTEM (1040 WORD-RECORD DEVICES).
PAGINI.FTN	PRIMOS PAGING MECHANISM COLD START INITIALIZATION.
PAGTUR.FTN	TURN PAGE(S) IN RESPONCE TO A PAGE FAULT.
PBDIOS.PMA	PAPER TAPE READER, PUNCH, PRINTER I/O RELATED ROUTINES.
PBH$ON.PLP	PB HISTOGRAM FACILITY STARTUP/ACCESS ENTRIES.
PBTABL.PMA	DATA AREA FOR PB HISTOGRAM.
PCBINI.FTN	PCB INITIALIZATION FOR COLD START.
PCBPTR_.PLP	RETURN POINTER TO A SPECIFIED USER'S PCB.
PCC$HT.PLP	BREAKS IPQCS LINKS TO THE PROGRAMABLE CONTROLLERS.
PCC$RA.PLP	REINITIALIZES ASYNCHRONOUS SERVICES: LYNX AND AMLC.
PCC$URS.PLP	REINITIALIZATION FOR THE PROGRAMMABLE CONTROLLER SYNCHRONOUS SERVICE.
PCC$WM.PLP	WARM-STARTS THE PROGRAMMABLE CONTROLLERS.
PCCBS.PLP	"pccbs" LOADS AN EXECUTABLE FILE INTO A PROGRAMMABLE CONTROLLER.
PCCDLL.PMA	DEFINES AN AREA IN SEG 0 FOR PROGRAMMABLE CONTROLLER DOWN LOADING.
PCCSO.PMA	DEFINES AN AREA IN SEG 0 FOR PROGRAMMABLE CONT. DATABASES.
PCC_DCL.PMA	DEFINES SOME STORAGE AREAS FOR PROGRAMMABLE COMM. CONT.
PGFSTK.PMA	PUDCOM AND PAGE FAULT STACK FOR USER 1.
PGMAPA.PMA	ROUTINES TO RETRIEVE SDW AND PAGE MAP POINTERS.
PGMAPS.PMA	START OF ALL THE PAGE MAPS IN THE SYSTEM.
PHLOGIN.PLP	LOGIN A PHANTOM USER.
PHTTYREQ.PLP	FORCE A PHANTOM TO LOGOUT AFTER AN ILLEGAL REQUEST.
PHYSAD.PMA	THIS CONVERTS VIRTUAL TO PHYSICAL ADDRESSES (DTAR 0,1) FOR IPQNM.
PID$CK.PLP	VALIDATES A PROCESS'S UNIQUE ID.
PID$GET.PLP	GET THE PID OF THE CURRENT PROCESS.
PIO.PMA	ROUTINES TO CONSTRUCT AND PERFORM PIO INSTRUCTIONS.
PMPRIM.PMA	PAGE MAP PRIMITIVES FOR USE IN ACCESSING PRIMOS PAGE MAPS.
PMSG$.FTN	PRINT INTER USER MESSAGE.
PRERR.FTN	PRINT NAME AND/OR MESSAGE FROM USER'S ERRVEC.
PRI$RV.PLP	RETURNS THE PRIMOS REV. STAMP OF THE CURRENTLY RUNNING OPERATING SYSTEM.
PRJID$.PLP	RETURN PROJECT ID OF CURRENT USER.
PRN$ST.FTN	PRINT SYSTEM STATUS ON USER TERMINAL.
PROBE.PLP	GENERAL EVENT MONITOR PROBE ROUTINE - WRITES RECORDS INTO BUFFER.
PROBE_UTILS.PMA	PMA UTILITIES FOR THE PROBE ROUTINE OF GENERAL EVENT MONITOR (GEM).

PTRAP.FTN	RESTRICTED MODE TRAP HANDLER.
PTRDIM.FTN	PAPER TAPE READER DIM.
R0BASE.PMA	GET A POINTER TO THE FIRST FRAME ON THE RING0 STACK.
R0FALT.PMA	RING0 FAULT HANDLER, RING 0 UTILITY SUBROUTINES.
R0UII.PMA	SPECIAL (QUICK,SMALL STACK FRAME) UII F.I.M. FOR RING0.
R3CALL.PMA	RING 3 CALL TABLE.
REMLI$.FTN	PROCESS THE REMLIN COMMAND.
REPLY$.FTN	OPERATOR/USER COMMUNICATION FACILITY.
REQLCD.PMA	SIGNALS THAT CHEAP PROCESSES HAVE REQUESTED LOGICAL CONNECT DELETION.
RMKSHL.PLP	REMOVE A SHARED PURE DTAR2 SEGMENT FROM HE WORLD.
RMSGD$.FTN	RETURNS CONTENTS OF PER USER MESSAGE BUFFER TO CALLER.
RMSHL.PMA	DETACH A SHARER FROM A PURE DTAR2 SHARED SEGMENT.
RMSHR.PLP	DETACH A SHARER FROM A DTAR0/DTAR2 SHARED SEGMENT.
RSEGAC$.PLP	FUNCTION WHICH RETURNS PER RING ACCESS TO THE SEGMENT IF THE SEGMENT IS IN USE.
RSEGACU.PLP	FUNCTION WHICH RETURNS PER RING ACCESS TO ANY USER'S SEGMENT IF THE SEGMENT EXISTS.
RTIME$.PMA	RETURN REAL-TIME AS 48-BIT VALUE IN PIC COUNTS.
RTNSG$.PLP	RETURNS ON SEGMENT OR ALL PRIVATE SEGMENTS IN A USER'S PROCESS.
RTNSG2.PMA	INTERLUDE TO RTNSG3.
RTNSG3.FTN	DOES THE DIRTY WORK OF RELEASING A SEGMENT.
RWREC.PLP	HANDLE READ AND WRITE REQUESTS FOR ASSIGNED DISKS.
S$ATR.PLP	READS SYSTEM DEFAULTS FROM THE SAD AND PUTS THEM INTO EPFCOM.
S$ATRB.PLP	SETS UP DEFAULT ATTRIBUTES (IN MEMORY COPY) FOR SYSTEM.
S$ATRG.PLP	RANGE CHECK FOR ATTRIBUTES.
SAL_SYST.PLP	SYSTEM CLASS STORAGE ALLOCATOR.
SANAM$.PLP	RETURN THE NAME OF THE SYSTEM ADMINISTRATOR.
SCH$RD.PLP	SCHEDULAR VARIABLE READ SUBROUTINE.
SCH$ST.PLP	SCHEDULAR VARIABLE SET SUBROUTINE.
SCHAR.PMA	STORE CHARACTER INTO ARRAY, STEP CHARACTER POINTER.
SCHED.PMA	PRIMOS IV SCHEDULING ROUTINES.
SEG0.PMA	SEGMENT 0 MODULE.
SEG14.PMA	SEGMENT 14 MODULE.
SEG4.PMA	SEGMENT 4 MODULE.
SEGAC$.PLP	SUBROUTINE TO SET SEGMENT ACCESS.
SEM$CA.PLP	NAMED SEMAPHORE ROUTINE TO CLOSE ALL SEMAPHORES AT LOGOUT TIME.
SEM$CL.PLP	NAMED SEMAPHORE ROUTINE TO CLOSE AN OPEN SEMAPHORE.
SEM$DR.PLP	NAMED SEMAPHORE ROUTINE TO DRAIN A SEMAPHORE.
SEM$NF.PLP	NAMED SEMAPHORE ROUTINE TO NOTIFY A SEMAPHORE.
SEM$OP.PLP	NAMED SEMAPHORE ROUTINE TO OPEN A SEMAPHORE ASSOCIATED WITH A FILENAME.
SEM$OU.PLP	NAMED SEMAPHORE ROUTINE TO OPEN AND INITIALIZE A SEMAPHORE.
SEM$ST.PLPNAMED	SEMAPHORE ROUTINE TO REPORT STATUS OF SEMAPHORES.

SEM$TN.PLPNEMAED	SEMAPHORE ROUTINE TO SET A TIMER FOR A SEMAPHORE.
SEM$TS.PLP	NAMED SEMAPHORE ROUTINE TO TEST A VALUE OF A SEMAPHORE.
SEM$TW.PLP	NAMED SEMAPHORE ROUTINE TO WAIT ON A SEMAPHORE AND TIMER.
SEM$WT.PLP	NAMED SEMAPHORE TO WAIT ON A SEMAPHORE.
SEMUTL.PLP	NAMED SEMAPHORE UTILITY ROUTINES.
SEMVQA.PLP	NAMED SEMAPHORE ROUTINE TO ADD A PROCESS TO A VIRTUAL SEMAPHORE QUEUE.
SEMVQR.PLP	NAMED SEMAPHORE ROUTINE TO REMOVE A RANDOM PROCESS.
SEMVQS.PLP	NAMED SEMAPHORE ROUTINE TO REMOVE TOP PROCESSES FROM VIRTUAL SEMAPHORE QUEUE.
SETACC.PLP	SUBROUTINE TO SET SEGMENT ACCESS.
SETASD.PMA	SETUP AUTO SPEED DETECT PROTOCOL FOR A GIVEN LINE.
SETCPU.PMA	LOCK/UNLOCK PROCESS TO NASTER CPU.
SET_INFO.PLP	SET AND CHECK VALUES OF INFO STATUS FOR PRIME INFORMATION.
SFR_SYST.PLP	FREES SPACE FROM SYSTEM CLASS STORAGE.
SGINFO.PLP	RETURN INFORMATION ABOUT A SEGMENT.
SHARESEG.PMA	DATA FOR SNA SHARED SEGMENT UTILITY.
SHRLIB.FTN	INSTALL SHARED LIBRARY (RESTRICTED TO USER <SUSR>).
SHUTDN.FTN	SHUTDOWN COMMAND PROCESSING FOR PRIMOS IV.
SH_CMD.PLP	PROCESS THE SHUTDOWN COMMAND.
SID$GT.PLP	GET SPAWNER'S ID
SISSH0.PLP	REPORTS ON WHETHER A SEGMENT IS BEING SHARED OR NOT.
SISSH1.PLP	REPORTS ON WHETHER A SEGMENT IS SHARED OR NOT.
SISSH2.PLP	REPORTS ON WHETHER A SEGMENT IS SHARED OR NOT.
SISSH3.PLP	REPORTS ON WHETHER A SEGMENT IS SHARED OR NOT.
SMSG$.FTN	SEND A MESSAGE TO A USER ON AN ARBITRARY NODE (USER CALLABLE).
SNAP$0.PLP	SNAP A DYNAMIC LINK INTO RING0 (i.e. A GATE).
SNDBLK.PLP	SEND AN ASYNC CONTROL BLOCK TO A NEW COMM CONT.
SORO$.PLP	INVOKES LIST OF RING0 STATIC ON-UNITS.
SPAWN$.PLP	SPAWN A NEW PROCESS WITH ATTRIBUTES PARTIALLY SPECIFIED BY SPAWNER.
SRPHAN.PLP	APPLY SUFFIX SEARCH CONVENTIONS FOR PHANTOM LOGINS.
SRWREC.FTN	SVC HANDLER FOR RREC, WREC SVC.
STKINI.FTN	INITIALIZATION OF RING 0 STACK SEGMENTS.
STNOU.PMA	SVC-PCL INTERLUDES TO TNOU, TNOUA
SUPSTK.PMA	UNWIRED RING 0 STACK FOR USER 1.
SUSR$.PLP	Returns whether or not caller is user 1.
SVCAL$.PMA	MISCELLANEOUS SUPERVISOR ENTRIES.
SV CALPAR.PLP	Do all the validation for a system variable setting.
SW$ABT.PLP	Handle Software Interrupt Process aborts for the current process.
SW$AD.PLP	Routine to cause a ring 0 routine to be restartable if a so abort is deferred.
SW$INT.PLP	Software Interrupt Enable Control Module.
SW$MKRCS.PLP	Makes A Reverse Critical Section.
SW$ON.PLP	Turns On The Specified Software Interrupts For Ring 3
SW$ROOFF.PLP	Turns Off Specified Software Interrupts For Ring 0
SW$RAOF.PLP	Reads And Then Turns Off All Present Interrupts For Ring 3

SW$RST.PLP	Reset Ring 0 Software Interrupt Enable Mechanism.
S_ADD_0.PLP	Adds an entry to DTAR0/1 shared segment table.
S_ADD_2.PLP	Adds a new entry to pure DTAR2 shared segment table.
S_CNJN_2.PLP	Returns truth value of "Pure DTAR2 uid exists and is attachable"
S_FULL_0.PLP	Returns true if and only if the DTAR0/1 share tables are f
S_FULL_2.PLP	Returns truth value of "Share table for DTAR2-only segs is full"
S_FULL_X.PLP	Returns true if an only if the DTAR0/1 share table s table is full.
S_GETKEY.PLP	Attempts to retrieve first two words from specified file
S_INDEX0.PLP	Returns truth value of "Index of specified uid in most rec
S_JOIN_0.PLP	Adds a new attacher to a given uid in the DTAR0/1 shared table.
S_JOIN_2.PLP	Attach an entry in pure DTAR2 shared segment table.
T$AMLC.PLP	Raw data mover for amlc lines.
T$CMPC.FTN	I/O TO CARD READER/PUNCH VIA MPC.
T$GPPLPLP	General purpose parallel interface routine.
T$GS.PMA	DRIVER FOR VECTOR GENERAL GRAPHICS TERMINALS
T$LMPC.FTN	LINE PRINTER OUTPUT VIA MPC
T$MG.PMA	DRIVER FOR SOC-MEGRAPHIC 7000 INTERFACE
T$PMPC.FTN	CARD PUNCH I/O VIA MPC
T$TM.PMA	PRIMOS DIRECT-CALL HANDLER FOR TAG MONITOR
T$VG.FTN	VERSATEC-GOULD PLOTTER I/O
TA$.FTN	Obsolete tree attach (processes register settings).
TDUMPC.PMA	Define the symbol TDUMPC and cause seg to allocate space.
TERM$L.PLP	SET/RESET TERMINAL PARAMETERS FOR USE WITH THE INFORMATION PRODUCT.
TFLADJ.PLP	Adjust size of tfliob buffers
TFLIO$.PMA	LOGICAL I/O BUFFERING ROUTINES.
TI$MSG.PLP	Print a message summarizing connect, cpu, and I/O time utilization.
TIMDAT.PMA	DATE AND TIME CONVERSION ROUTINES.
TMAIN.PMA	CLOCK PROCESS, RING 0 UTILITY SUBRS.
TODEC.PLP	Print decimal or octal integer on any user's terminal.
TP$CON.PLP	Terminal-Process connect amlc line
TP$DIS.PLP	Terminal-Process disconnect for amlc lines

TPIOS.FTN	PAGE TURNING INTERLUDE TO DISK I/O.
TPLOGO.PLP	Do TP logout cleanup
TPUT_SAV.PMA	DEFINE STORAGE AREA FOR SAVED USER TYPES FOR TP
TRMBUF$G.PLP	Get the number of the terminal parent's I/O buffer and uid
TRMPID$$.PLP	Maintains all process's terminal parent id attributes.
TTY$IN.PLP	Gate to check if there are any characters in the tty input buffer for user.
TTY$RS.PLP	Routine to clear a process's I/O buffers.
TTYPER.PMA	Typers (terminal output routines).
TUTIL$.PMA	RANDOM SUBROUTINES
UID$BT.PLP	Generate unique id as a bit string.
UID$CH.PLP	Generate a unique identifier as a character string.
ULOKPG.FTN	UNWIRE AN AREA OF THE VIRTUAL MEMORY.
UNO$GT.PLP	Get the id's associated with this user.
USER$.FTN	Retreive ring0 data.
USNMT$.PLP	Unassign magnetic tape drive units.
USRA$$.FTN	Process the USRASR command.
UTIL$.PMA	UTILITY SUBROUTINES FOR FORTRAN PROGRAMS
UTYPE$.PLP	Function to return type of user (normal, remote, phantom)
VERDIM.PMA	PRIMOS 4 DRIVER FOR SOC INTERFACE
WAITIN.PMA	WAIT WITH PROCESS EXCHANGE INHIBITTED.
WARMST.PMA	IS A WARM STARTABLE HALT ROUTINE.
WIRSTK.FTN	Procedure to wire the page fault stack for a process.
WRL$.PLP	ESTABLISH WHICH RINGS STACK OF STATIC ON UNITS TO BE ACCESS
WRMABT.FTN	HANDLE WARM START PROCESS ABORT.
XTDISO.PLP	Extend the allocation of seg 0 for IPQNM.

2.2 FS-ROUTINES

AC$CAT.PLP	Place an object into an access category.
AC$DFT.PLP	Protect an object with default access rights.
AC$LST.PLP	Return the contents of an ACL in logical format.
AC$RVT.PLP	Revert an ACL directory to password protection.
AC$SET.PLP	Create an ACL.
ACC_CHK.PLP	Handle access checking for access-setting routines.
ACDECODE.PLP	Decode a physical ACL entry into a logical one.
ACENCODE.PLP	Encode logical <id>:<access> pair into physical ACL entry
ACLSEG.PMA	ACL system databases.
AC_CLEAN.PLP	Common cleanup for ACL gates.
AC_DELPA.PLP	Delete a priority ACL for a specified logical device.
AC_NEWPA.PLP	Add a new priority ACL to the specified LDEV.
ADD_ENT.PLP	Add a new entry to a directory.
ADD_REC.PLP	Extend a file.
ALC_REC.PLP	Allocate record(s) for new directory entry.
AT$.PLP	Attach to the specified pathname.
AT$ABS.PLP	Attach to a top-level directory on a specified partition.
AT$ANY.PLP	Do an attach scan.
AT$HOM.PLP	Set current attach point to be same as home.
AT$INV.PLP	Invalidates specified attach point(s).
AT$OR.PLP	Set home and/or current attach points to be same as initia
AT$REL.PLP	Attach relative to the current attach point.
AT$TMP.PLP	Save or restore the current attach point.
ATCH$$.PLP	Writearound for new attach modules.
ATLIST.PLP	Do a local attach scan on a specified list of disks.
AT_ADREM.PLP	Set unit table entry for attach point just gone remote.
AT_BADPW.PLP	Signals BAD PASSWORD$ for attach routines.
AT_CLEAN.PLP	Common cleanup for attach modules.
AT_UNREM.PLP	Invalidate remote attach point(s).
AT_VALPAR.PLP	Validate key and directory name for AT$ routines.
CALACS.PLP	Calculate accesses available on a named object.
CALACS.PLP	Calculate accesses.
CAT$DL.PLP	Delete an access category.
CH$MOD.PLP	Change the open mode of an open file.
CL$FNR.PLP	Close a file by name and return a bit varying indicating closed units.
CLO$FN.PLP	Close an open file by name.
CLO$FU.PLP	Close an open file by unit.
CLOSE.PLP	Close a file by name (BRA/device number) or unit.
CNAM$$.PLP	Change the name of a file system object.
CO$GET.PLP	Retreive ring0 data for invoking CLOSE and COMOUTPUT command in ring3.
COMI$$.PLP	COMINPUT COMMAND AND SVC HANDLING
COMO$$.PLP	COMOUT-PUT COMMAND/SVC HANDLER – SWITCH COMMAND OUTPUT ON/OFF
COPY_AP.PLP	Copy one attach point to another, handling hashing and quotas.
COPY_UTE.PLP	Copy one unit table entry to another.
DAMDATA.PLP	Calculate the first data record for a DAM type file.

DEL_ENT.PLP	Remove a directory entry.
DIR$CR.PLP	Create a directory.
DIR$RD.PLP	Read physical directory entries.
DOPO$$.PLP	Do positioning on an open file.
DOSUFFIX.PLP	Scan a directory for a (possibly) suffixed entry name.

EMPTY_CK.PLP	Make sure the object whose BRA is passed may be deleted.
ENTINDIR.PLP	Attach to directory, return entry name in it.
FIL$DL.PLP	Delete a file or directory.
FIL_CR.PLP	Create a named file.
FIL_EX.PLP	Check existence of a named ufd entry.
FIL_OP.PLP	Open a named file.
FIND_ENT.PLP	Find an entry in the directory specified by the unit table entry.
FIND_HOLE.PLP	Find the first available hole of the required size in a directory.
FINFO$.PLP	Return information about specified file unit.
FORCEW.PLP	Force a file to be written to disk.
FREE_REC.PLP	Free a file's records when it is deleted.
FSAHSH.PLP	Add a unit table entry to file system and/or ACL hash threads.
FSHASH.PMA	Calculate the hash index for the unit table.
FSUHSH.PLP	Remove a unit table entry from file system and/or ACL hash threads.
GETID$.PLP	Returns a user's complete ID (user id plus group ids).
GETQB.PLP	Return a pointer to the directory/quota block & increment use count.
GETREC.PLP	Get a free record in a logical partition by searching RAT.
GETUN.PLP	Get a unit table entry from the system-wide pool.
GET_LDEV.PLP	Convert partition name to logical device number.
GOODUNIT.PLP	Check the validity of a unit number.
GPAS$$.PLP	Read passwords on named directory.
GPATH$.PLP	Return a pathname given a unit, attach point or segment number.
GSG_RA.PLP	Return segdir entry number by matching BRA in record LOCAT by caller.
GUF_RA.PLP	Return directory entry by matching BRA in dir defined by current LOCATE buf.
HASH_TBLS.PMA	Miscellaneous filesystem hash tables.
ISACL$.PLP	Indicates whether specified unit is an ACL directory.
KICKQB.PLP	Increment quota block use count for a subtree.
LDISK$.plp	Return a list of disk names.
LDSKU$.PLP	List all users using a given partition.
LOCATE.PMA	PRIMOS FILE SYSTEM ASSOCIATIVE BUFFERING.
LSMCOM.PMA	Table containing added disk information.
LUDSK$.PLP	Return a list of all disks in use by a given user.
LUID$.PLP	Description: Return a unique ID consist of the ldev and BRA
M2SMA$.PLP	Return unit number in slave given unit in master.
MARKUT.PLP	Mark unit table when a disk error occurs.
MKUTEPTR.PLP	Return a pointer to the unit table entry of the given unit.
MOVNAM.PMA	Move names between two fields
NAMEQ$.FTN	COMPARE TWO NAMES FOR EQUIV (RET TRUE IF SAME)
NCLBIT.PLP	Turn on/off the no-close bit for a file unit.
NEWDAM.PLP	Add record to new partition dam file.

NEW_ACL.PLP	Process addition of a new ACL to a directory
OPENFILE.PLP	Open a file (possibly allocating a unit) and return the unit
OPEN_CHK.PLP	Check to see whether or not a file unit is open.
PA$DEL.PLP	Delete a priority ACL.
PAR$RV.PLP	Returns the partition rev. stamp of a named disk partition
PK2LDV.PLP	Convert disk pack name, remote system name into an LDEV
PRWF$$.PLP	Moves data to and from files; also does positioning of fi
Q$READ.PLP	Read quota information for current directory.
Q$SET.PLP	Set quota fields on specified directory.
Q_TRWK.PLP	Count records used in a subtree.
Q_UPDT.PLP	Update directory headers with quota information.
R/W_ENT.PLP	Read or write the directory entry at the specified position
RA2PTH.PLP	Return PATHNAME : <disk name>tree name based on BRA and L
RDEN$$.PLP	Writearound for RDEN$$ gate.
RDLIN$.PLP	Read a line from a file.
RDLN$X.PMA	SUBROUTINE TO EXPAND LINE READ FROM FILE.
REMSHT.PLP	Shutdown all remote disks on the system.
REST$$.FTN	Restore memory-image R-mode run file ("SAVE" file).
RTNQB.PLP	Return Quota Block.
RTNREC.PLP	Return specified record to logical device's free list.
RTNUN.PLP	Return a unit table entry to the global pool.
RVKID$.PLP	Revokes indices AGTIDX into Active Group Table for given user.
RWLKCK.PLP	Check unit tables for conflict with specified file, open desired, and r/w lock setting.
SATR$$.PLP	Set attributes for specified file.
SAVE$$.FTN	Save memory image
SEMSEG.PMA	NAMED SEMAPHORE DATA AREA
SETID$.PLP	Adds a group into the specified user's Active Group List.
SET_DTM.PLP	Set date/time modified of specified file entry to current date/time.

SET_OR.PLP	Set initial attach point (origin).
SET_QMOD.PLP	Set modified bit in a quota directory block.
SGD$DL.PLP	Delete a segment directory entry.
SGD$EX.PLP	Check the existence of a segment directory entry.
SGD$OP.PLP	Open a segment directory entry.
SGDR$$.PLP	Manipulate segment directory (open status demanded).
SGD_RE.PLP	Segment directory read entry.
SGD_WE.PLP	Segment directory write entry.
SPAS$$.PLP	Set passwords on current directory.
SRCH$$.FTN	Open, close, delete, change access on, check existence of files.
SYS_OPEN.PLP	Open a directory on the system unit or some othe unit.
TEXTOK.PMA	TESTS FOR A VALID 6-CHARACTER FILE NAME
TRUNC$.PLP	Truncate a file at the Unit Table Entry's rel wordno & rel recno.
TRWRAT.PLP	Startup/shutdown a filesystem partition.
UKCKQB.PLP	Decrement quota block use count for a subtree.
UNITS$.PLP	Get the current unit number bounds.
UTALOC.PLP	Allocate a unit table for a user.
UTDALC.PLP	Deallocate a users unit table.
VINIT$.PLP	Subroutine to initiate a VMFA segment.
WTLIN$.PLP	Write a line to a file.
WTLN$C.PMA	SUBROUTINE TO COMPRESS LINE WRITTEN TO FILE.

2.3 R3S-ROUTINES

$CALLS.FTN	Interludes to old style calls
ABBREV.PLP	This is the internal command for abbreviations.
AB_FILE_.PLP	This is the routine to handle file I/O for abbreviations
AB_GET_.PLP	Get next whole token from command line, processing abbreviations.
AB_PCS_.PLP	This is the routine to expand abbrevs.
AC$CHG.PLP	Modifies the contents of an existing ACL
AC$LIK.PLP	Set ACL on one file to be like that on another.
AC$PAR.PLP	Parse an access control list.
ADD_REMID_.PLP	Process the add remote id command.
ALC$RA.PLP	Allocate space in process class storage for return function data.
ALOC$$.PMA	ALLOCATE STORAGE ON THE STACK (FREE ONLY BY PRTN).
AL$$RA.PLP	Allocate space and set return data for return function.
APPEND.PMA	APPEND — CONCATENTATE TO VARING STRING
APSFX$.PLP	Append a suffix onto a pathname according to file naming standards
AREA_MAN.PLP	This is a general PL/I Area Manager.
ASTRSK$.PLP	Command
ATCH_.PLP	Invoke the ATTCH command from ring3.
BIN$SR.PLP	Do a binary search using pointers in a single segment.
BINARY_.PLP	BINARY Command.
CACHE_POP.PLP	POP an entry from the per-level stack of program EPFs.
CACHE_PUSH.PLP	POP an entry from the per-level stack of program EPFs.
CH$FX1.PMA	CHARACTER TO FIXED BIN (15, 0) AND FIXED BIN (31, 0) CONVERT
CH$OC2.PMA	CHARACTER (OCTAL) TO FIXED BIN (31, 0) CONVERTER.
CHANGE_PW.PLP	Command to allow a user to change his/her login password
CKDYN$.PLP	Check the existence of a Dynamic Entrypoint.
CL$GET.PLP	Gets A Command Line Into User's Buffer
CL$GET_EV.PLP	Command Loop GET Entry Variable.
CL$PAR.PLP	Parse string according to basic "command line" rules.
CL$PIX.PLP	Parse command line according to a picture specifier.
CL$SET_EV.PLP	Command Loop SET Entry Variable.
CLOSE_.PLP	Check cmdl syntax and call SRCH$$ to close file units.
CLOS_ALL.PLP	Closes All Of A User's Open File Units.
CLRLV_.PLP	Clear the existing level.
CMD_	
POST_INVK.PLP	Routine to perform post-program invocation initialization
PRE_INVK.PLP	Routine to perform pre-program invocation initialization.
CNAME_.PLP	Invoke the CNAME command from RING3_Via GATE CNAM$$.
CNIN$.PLP	Reads A Specified Number Of Characters From Command Input Device
CNSIG$.PLP	Set continue sw on in most recent fault frame.
COMANL.PLP	Writearound To CL$GET.
COMLV$.PLP	Call a new command level.
COMO$.PLP	COMOUTPUT Command.

COND_CALLS.PMA	ADDITIONAL ENTRY POINTS FOR THE CONDITION MECHANISM.
CP$.PLP	Invoke the user's currently specified command processor.
CP_ITER.PLP	Command language iteration, wildcard, treewalk, eqname processor
CRAWL_.PLP	Perform a "crawlout" from an inner ring, and rejoin signl
fim .	
CREATE_.PLP	Process the CREATE (directory) command.
CRFIN_.PMA	CRAWLOUT "FAULT INTERCEPTOR" TO REINVOKE SIGNL$ IN THE OUT
RING.	
DB$MOD.PLP	Set/Reset debugger-mode switch and static on-unit.

DBG_.PLP	Internal command writearound to the DBG external command
DCOD_ITR.PLP	Decode command language extended feature token type.
DEF_GV.PLP	Command to define global variables file to command env.
DELAY_.PLP	Invoke the DELAY command from ring3.
DELETE_VAR.PLP	Delete global variables
DELSEG_.PLP	Process the DELSEG command.
DET$GET.PLP	Get msg from a diagnostic Error Table.
DF_UNIT_.PLP	System Standard Default On-Unit (includes PL/I runtime support).
DUMPS_.PLP	Dump stack in a pretty format.
EDIT_ACC_.PLP	Process the edit access command.
EDIT_CL.PMA	EDIT COMMAND LINE TO REMOVE EXPLICIT NULL STRINGS.
ELIGTS_.PLP	Set the scheduler variable ELIGTS.
ENDPAGE_.PLP	PL/I runtime support for ENDPAGE condition (called from DF_UNIT_)
EPF$ALLC.PLP	EPF linkage allocation routine.
EPF$CPF.PLP	Get command processor flags from an epf.
EPF$DEL.PLP	Terminate an epf invocation.
EPF$INFO.PLP	Return info about a desired epf file.
EPF$INIT.PLP	EPF static data initialization routine.
EPF$INVK.PLP	Routine to start the execution of an EPF.
EPF$MAP.PLP	Routine which maps an EPF file to virtual memory.
EPF$RELC.PLP	EPF Relative Pointer relocation routine.
EPF$RELC.PMA	Relocate EPF relitive pointers.
EPF$RUN.PLP	Run an EPF : Executable Program Format file
EPF_ERR.PLP	Routine to print diagnostic error messages to a user's
EPF_NW.PLP	Push volatile EPF smt data for program and library EPFs.
EPF_NWA.PLP	Push volatile EPF smt data for ALL program and library E
EPF_RL.PLP	Pop volatile EPF smt data for program and library EPFs.
EPF_RLA.PLP	Pop volatile EPF smt data for program and library EPFs.
EPF_SRCH.PLP	This routine searches an EPF library to resolve a faulte entrypoint.
EQUAL$.PLP	Generate name from an object (source) name and a pattern
EQUAL$.PLP	Append pathname generated from equalname to a given stri
ERRSET.PMA	ERRSET INTERLUDE FOR SEGMENTED
EX$CLR.PLP	Disable the signalling of the EXIT$ condition upon progr termination.
EX$RD.PLP	Return the value of the TRANSMIT EXIT command environment counter.

EX$SET.PLP	Enable the signalling of the EXIT$ condition upon program termination.
EXIT.PLP	Exit from Static Mode, and return to Recursive Mode.
FATAL.PMA	GENERATE FATAL PROCESS ERROR.
FILL$A.FTN	FILL ARRAY WITH LITERAL
FIND$BKT.PLP	Search a PRIMOS standard hash table for a bucket address
FINDPROC.PMA	FIND NAME AND ADDR FOR DF UNIT PL/I CONDITION MESSAGES
FIND_EPFS.PLP	Routine to generate lists of epfs for a process.
FIND_UID.PLP	Find a <user id> in a validation file.
FNCHK$.PMA	Check the string passed for validity as a file system na
FNDCF$.PLP	Find most recent condition frame.
FNDLOW.PLP	Finds lowest timer of either type.
FNONU$.PLP	Find onunit in specified stack frame.
FRE$RA.PLP	De-allocate space used for return information for command functions.
GATES.PMA	GATES table.
GET_EPF_ PATHNAME.PLP	Routine to retrieve the full pathname of EPF.
GET_FR.PMA	Get the field address registers and floating point regi
GS_FAC.PMA	GET/SET FLOATING ACCUMULATOR FROM A FAULT FRAME REGISTER
GT$PAR.PLP	Parse string according to four types of characters.
GV$GET.PLP	Get the value of a global variable
GV$SET.PLP	Set the value of a global variable
G_FACVAL.PLP	Procedure to get the value of an offending fac.
HASH UID.PLP	Hash a <user id>.
I$GCLB.PLP	Get CLDATA.EXIT LB and CLDATA.EXIT SB for INFORMATION S
ICE$.PLP	Initialize Command Environment
ICMTB_.PMA	INTERNAL (OLD AND NEW) COMMAND TABLE.
IDCHK$.PLP	Check a (user or project) id for legality.
INFIM_.PMA	CRAWLOUT "FAULT INTERCEPTOR" FOR INIT$3 (INITIALIZE RIN ENVIRONMENT).
INIT$3.PLP	Initialize the ring 3 environment, and make sure that t External Login is run
INIT$P.PLP	Invoke initial routine (cominput, CPL, EPF, etc.) at lo
INPUT$.PLP	INPUT Command.
INTCM_.PLP	Fetch local command table entry if any, else check system table.
INVKSM_.PLP	Invoke (or restore) static mode program image.
IOA$.PMA	INTERLUDE TO CALL THE IOA$ FORMATTER. (IOA$, IOA$RS, IO
IOAFM$.FTN	FORMATTING PACKAGE FOR IOA$.
IOAGA$.PMA	IOAGA$- GET ARGUMENT ROUTINE FOR IOAFM$, PRIMOS4 08/08/77
IOAGD$.PMA	This module does an unsigned long divide.
IS$EPFUS.PLP	Routine to determine if a given file is an EPF which is use.
ISF_EPF.PLP	Determine if my grandfather is an EPF or a static-mode
IS_EPFEX.PLP	Routine to check the existance of an EPF run file and op an EPF.

IS_EPFMP.PLP	Routine to determine if an epf file is mapped to memory.
IS_EPFSG.PLP	Perform a mapping between a list of segment numbers and file in memory.
ITR_WLDC.PLP	Perform command language Wildcard Iteration.
ITR_WLDT.PLP	Perform command language Treewalk Iteration.
KTRAN$.PMA	PERFORM A KEY TRANSFORMATION ON AN ENTRY POINT NAME.
LIBTBL.PMA	LIBRARY TABLES.
LIST$CMD.PLP	List to terminal ring3, internal mini-level commands specified by input arguments.
LIST$EN.PLP	This routine returns all the library entrynames in a lib EPF.
LISTEN_.PLP	Primos command loop standard Listener module.
LIST_ACC_.PLP	Process the list access command.
LIST_ACL.PLP	Print the contents of an ACL on the terminal.
LIST_CMD.PLP	List out the current mini-level internal commands and what their options are.
LIST_EN.PLP	Displays a list of entry names within a library EPF.
LIST_EPF.PLP	This displays information about EPFs.
LIST_GROUP.PLP	List the user's active and/or inactive groups.
LIST_LIM.PLP	Routine to retrieve and print EPF related data from user profile.
LIST_PA_.PLP	Process the List priority access command.
LIST_QUOTA.PLP	Process the LIST QUOTA command.
LIST_REMID .PLP	List one or all ID's used by this user on remote nodes.
LIST_SEGMENT.PLP	This routine parses command, prints the segments in user private, dynamic and static address spaces which are in
LIST_SRL.PLP	Command to Print Search List(s).
LIST_VAR.PLP	List global varialbes and their values.
LN_EPF.PLP	Process an EPF library search rule to resolve a faulted reference.
LN_ISR.PLP	Open the dynamic linking search list for a process.
LN_LEG.PLP	Determine whether an attempted dynamic link is valid.
LN_SLIB.PLP	Search dynamic linking name space in order to resolve entrypoint.
LN_STAT.PMA	Search shared, static-mode library list to resolve fault reference.
LOGOUT_.PLP	Logout command processor.
LON$CN.PLP	Perform Logout Notification control through SW$INT.
LON$PR.PLP	Print phantom logout notification message
LON_.PLP	Logout Notification Command

MAKE_LIST.PLP	Makes a sorted linked list from the passed list.
MAXSCH_.PLP	Set the scheduler variable MAXSCH.
MESSAG_.PLP	Interface for the message facility
MIN$CP.PLP	Mini-level Command Processor
MISSIN.PMA	HANDLE MISSING ARGS IN V-MODE.
MKON$F.PLP	Fortran Interface (PCL call) to make an on-unit in calle frame.
MKONU$.PMA	MAKE AN ON-UNIT IN THE CALLER'S STACK FRAME.
MKSON$.PLP	Make a static on-unit for either ring.
MOVWDS.PMA	DATA MOVEMENT SUBROUTINES.
NEWLV$.PLP	Module to create a new level within the command environment
OCALLS.FTN	OLD PRIMOS SUBROUTINES CALLS
ONDISP.PLP	Display onunit data in a specific frame.
OPEN_.PLP	OPEN Command.
OPN$SR.PLP	Open using Search List.
OPN$SRSF.PLP	Open file using search rules and suffixes.
ORIGIN_.PLP	Command to return to initial attach point.
P$EPAGE.PLP	Write end of page text to a PL/I file for PL/I runtime support.
P$EXCPT.PLP	PL1 Condition Exception Handler.
P$KEY.PMA	PL/I ONKEY BUILTIN FUNCTION
PASSWD_.PLP	Set owner/non-owner passwords for current password direc
PHANTOM_.PLP	PHANTOM Command.
PL1$NL.PLP	Nonlocal goto processor for PL/I (and any other block-structured language).
PM$.PLP	Post Mortem command.
PRERR_.PLP	PRERR Command
PREVSB_.PLP	Find previous stack frame, given ptr to current.
PRTN_.PMA	VARIOUS PLAVOURS OF "RETURN" FOR USE BY THE UNWIND ROUT
PWCHK$.PLP	Check a password for legality.
R3ENTS.PMA	R3ENTS table.
R3FALT.PMA	RING 3 FAULT CATCHER.
RAISE_.PLP	Search the stack for an onunit for a specific condition, invoke it.
RD$CE_DP.PLP	Return to the caller the current depth of the command env. program session.
RDTK$$.PLP	Writearound to rdtk$p for use by static mode programs.
RDTK$P.FTN	READ NEXT TOKEN FROM COMMAND LINE

RDTKN$.FTN	USER CALLABLE ENTRY FOR RDTK$$ (OLD STYLE)
RDY__.PLP	Set user's ready message mode(s).
READY$.PLP	Print the ready (or error or warning) message.
REENT__.PLP	Signal the condition REENTER$ for subsystem reentry.
REMEPF$.PLP	Delete (Remove) An EPF From A User's Address Space
REMEPF__.PLP	Remove An EPF From A User's Address Space
REM__PA__.PLP	Process the Remove priority access command.
REM__REMID__.PLP	Process the REMOVE REMOTE ID command.
RESTO__.PLP	Internal command "restore": load memory image of SM pr
RESU$$.PMA	WRITEAROUND FOR RESU$$ CALL.
RING3.LOAD	RING THREE LOAD DATA FILE
RING3_ENTRY_ TABLE_HASH	All-rings direct call entrypoint definitions.
RLSLV$.PLP	Module to restore a level within the command environment
RLSTK__.PLP	Generate the Listener Order "release stack".
RMODE__.PLP	Return into Static Mode program, as defined by an "rvec
RPL$.PLP	Routine to replace One File With Another File.
RPL$CN.PLP	Change the name of an open epf file.
RSTERM.PLP	Command interface to reset terminal I/O buffer(s).
RVONU$.PLP	Revert an onunit in caller's or given activation.
RVSON$.PLP	Remove static on-unit.
SAL__HEAP.PLP	Heap Storage Allocator.
SAL__ISEG.PLP	Initialize A New Segment For Storage Allocation
SAL__LEVEL.PLP	Program Class Storage Allocator
SAL__PROC.PLP	Process Class Storage Allocator
SAL__USER.PLP	USER Program Class Storage Allocator
SAL__VRFY.PLP	Verifies Storage Class Key
SAVE$.PLP	Save a portion of memory as a file.
SCIT.PMA	Storage Class Information Table
SEARCH_CASELESS_ HASH_TABLE$.PLP	Caselessly search a PRIMOS standard hash table.
SEARCH_HASH_ TABLE$.PLP	Search a PRIMOS standard hash table.
SETRC$.PLP	Set Static Mode error code.
SETREG.PMA	SETREG, GETREG -- SET, RETRIEVE REGS IN SVEC
SET__ACC__.PLP	Process the set access command.
SET__PA__.PLP	Process the Set priority access command.
SET__QUOTA.PLP	Command to change quota or create a quota directory.
SET__SRL.PLP	Command to Set Search List.
SET__VAR.PLP	Internal command equivalent of &set var CPL directive
SFR__CFSC.PLP	Completely Free Storage Class
SFR__HEAP.PLP	Heap Storage Deallocator.
SFR__LEVL.PLP	Frees Space From level Class Storage
SFR__PROC.PLP	Frees Space From Process Class Storage
SFR__USER.PLP	Frees Space From User Program Class Storage
SHUTDN__.PLP	Process the SHUTDN command.
SIGNL$.PLP	Signal a specific condition.
SMT__QAD.PLP	Thread an entry to the head of the per-process queue of EPFs.

SMT_QFR.PLP	Unthread an entry from the smt list for active EPSs.
SNAP$3.PMA	Snap link to a ring three (all-ring callable).
SOR3$.PLP	Invoke ring 3 static on-unit.
SOUR3_.PLP	Find static on-unit list for ring 3.
SR$ADDB.PLP	Add a search rule to a list before an existing rule.
SR$COPYL.PLP	Copy all locator values from old list to same rules in new list.
SR$CREAT.PLP	Create search list specified by name and "open" it.
SR$DEL.PLP	Delete search list specified by name.
SR$FINDR.PLP	Find a specific rule in a given search list.
SR$FR_LS.PLP	Return to free pool the storage used by a search list.
SR$HEADP.PLP	Get/Set Search List Head Pointer for this process.
SR$LIST.PLP	Return a list of all search list names in this process.
SR$NEXTR.PLP	Fetch the next search rule from a given search list.
SR$OPEN.PLP	Find a search list specified by name and "open" it.
SR$PARSE.PLP	Parse a string search rule into a type and a text.
SR$READ.PLP	Return a list of all search rules of a given search list printable.
SR$REM.PLP	Remove a search rule from a list.
SR$SETL.PLP	Set the locator value in a given search rule.
SR$TEMPL.PLP	Process a search list template file.
SR$UPDT.PLP	Install (update) a new copy of a possibly existing search list.
SRSFX$.PLP	Perform tree search, with or without suffix standard.
SRVEC_.PLP	Set Static Mode "rvec" from a fault frame.
SS$ERR.PLP	Used by subsystems to declare that they have run into an
START_.PLP	Internal command "start": restart recursive or static mo
STD$CP.PLP	Standard Command Processor.
STK_EX.PLP	Handle auto static extension.
STR$AL.PLP	Interlude To User Program Class Storage Allocator
TR$AS.PLO	Subsystem Process Class Storage Allocator
STR$FR.PLP	Interlude to User Program Class Storage Deallocator
STR$FS.PLP	Frees Space From Subsystem Process Class Storage
SWFIM_.PMA	Ring 3 QUIT FIM-Invoke QUIT Condition In Ring 3.
TALOC.PLP	Allocate large storage area
TEMP$A.FTN	OPEN UNIQUE TEMPORARY FILE ON CURRENT UFD
TEXTO$.PLP	Check a character string for validity as a filename.
TIME_.PLP	Process the TIME command.
TM$ABS.PLP	Sets timer for time of day.
TM$ASS.PLP	Assigns virtual timers.
TM$ONU.PLP	Virtual timer static on unit.
TM$RD.PLP	Reads time remaining on virtual timers.
TM$RLS.PLP	Releases an assigned timer.
TM$SET.PLP	Sets virtual timers.
TNCHK$.PLP	Checks a character string for being a legal treename.
TSRC$$.FTN	OPENS FILE WITH SPECIFIED TREENAME
TYPE.PLP	Type text at a user's terminal.
UNWIND_.PLP	Prepare the stack for nonlocal-goto-induced unwinding.
USERS$.PLP	USERS Command
VLIST.PMA	VLIST
WILD$.PLP	Match wildcard name.
XIS.PMA	XIS UNIMPLEMENTED INSTRUCTION EMULATOR

2.4 CPLS-ROUTINES

AFTER_AF.PLP	'after' active function for CPL.
ALLOC_VAR.PLP	Allocate an extension area for variables
ATTRB_AF.PLP	Get certain file attributes (command function).
BEFORE_AF.PLP	'before' active function for CPL>
CALC.PLP	Evaluate arithmetic and logical expressions for CPL.
CH$HZ2.PMA	CHARACTER (HEX) TO FIXED BIN(31,0) CONVERTER.
CND_INFO_AF.PLP	condition info a.f.: retrieve selection cond. info.
COM_ABRV.PLP	Interlude to invoke command abbreviation processor.
CPL.PLP	Interface CPL interpreter to command level.
CPL_.PLP	Command Procedure Language Interpreter.
CPL_ET_.PLP	Return pointer to CPL Error Table pathname.
CV$DQS.PLP	Convert FS format date/time to quadseconds since Jan. 1,
\CVDTB.PLP	Convert Date from ASCII to Binary (file system) format.
CV$FDA.PLP	Standard fs date-time-mod converted to format mm/dd/yy dow
CV$QSD.PLP	Convert quadseconds since January 1, 1901 to date.
DATE_AF.PLP	Date Command (Function).
DIR$LS.PLP	Write-around to the routine DIR$SE.
DIR$SE.PLP	Retrieve info about selected entries in a given directory
DIRSER.PLP	Remote interlude to DIR$SE.
DIR_AF.PLP	'dir' active function for CPL.
ENTRY_AF.PLP	'entry' active function for CPL.
EVAL_ AF.PLP	Active function evaluator for CPL.
AN_EXPR.PLP	Evaluate an expression containing variable references and command functions
EXT_VBL.PLP	Evaluate character string containing local/global variable refs.
EXISTS_AF.PLP	EXISTS command function for CPL.
EXTR$A.PLP	Extract pathname components.
EXT_ VBL_MAN.PLP	External Variable Manager for Primos Command Loop.
FROM_DEC.PLP	Convert a decimal integer to an integer in a given base less than 17.
GET_EXPR.PLP	Accumulate the next expression from the current line.
GET_LINE.PLP	Get a new logical line from file on cpl unit
GET_REPLY.PLP	Fetch a yes/no/null/next reply from command input stream.
GET_TOKEN.PLP	Get next token from CPL program
GET_VAR_AF.PLP	Get var command function for CPL.
GVPATH_AF.PLP	Return pathname of current global variable file.
GV_PRT_.PLP	Get pointer to global variable area.
HEX_AF.PLP	Convert hexadecimal integer to decimal integer
ICPL_.PLP	Invoke CPL interpreter on given file, processing suffix.
ID_CHECK.PLP	Check a given string to see if it is a valid command var identifier.
INDEX_AF.PLP	'index' active function for CPL
LENGTH_AF.PLP	'length' active function for CPL.
MOD_AF.PLP	Implement mod function for CPL.
NULL_AF.PLP	'null' active function for CPL.
OCTAL_AF.PLP	Convert octal integer to decimal integer

OPEN$B.PLP	Open a branch by tree name (nonstandard)
OPEN_	
FILE_AF.PLP	Open file command function for CPL.
PATHN_AF.PLP	Pathname command function for CPL.
QUERY_AF.PLP	Query command function - get yes/no answer.
QUOTE_.PLP	Perform a quote operation on a given string.
QUOTE_AF.PLP	Perform quote operation for CPL active function.
READ_	
FILE_AF.PLP	Read file command function for CPL.
RESCAN_AF.PLP	Rescan command function for CPL.
RESPONSE_AF.PLP	Response command function - get textual answer.
SEARCH_AF.PLP	'search' active function for CPL.
SET_A_VAR.PLP	Set local and global user variables.
SIZE$B.PLP	Return the size of a branch in WORDS.
SUBSTR_AF.PLP	'substr' active function for CPL.
SUBST_AF.PLP	Substitute command (function): substitute s3 for s2.
TEST_EQUALS.PLP	Test expression equality for CPL.
TO_HEX_AF.PLP	Convert a decimal integer to a hexadecimal integer.
TO_OCT_AF.PLP	Convert a decimal integer to a octal integer.
TRANSL_AF.PLP	'translate' active function for CPL.
TRIM_AF.PLP	'trim' active function for CPL.
UNQUOTE_AF.PLP	Perform unquote active function for CPL.
VBL_MAN.PLP	Variable manager for subsystems allowing dynamacally allocated string vars.
VERIFY_AF.PLP	'verify' active function for CPL.
WILD_AF.PLP	"wild" command function – get list of files by wild name.
WR_FILE_AF.PLP	Write file for CPL.

2.5 NS-ROUTINES

ALCADR.PLP	Allocate and initialize (to all zeros) an address entry.
ALCHCB.PLP	Allocate and initialize (to all zeros) a host control block.
ALCMYL.PLP	Allocate and initialize my node's line definition table entry.
ALCNAM.PLP	Allocate and initialize (to all zeros) a name table entry.
ALCPDN.PLP	Allocate and initialize (to all zeros) a PDN control block.
ALCPTA.PLP	Allocate and initialize (to all zeros) a source address cha link.
ALCPTH.PLP	Allocate and initialize (to all zeros) a path control block.
ALCRNG.PLP	Allocate and initialize a ring line definition table entry.
ALCSLC.PLP	Allocate and initialize an SMLC line definition table entry.
ALLOC.PMA	ALLOCATES SPACE FOR TEMPS ON THE FLY FOR SLAVES
CALLIT.PMA	GIVEN A PCL NAME AND IT'S ARGS, THIS SUBR MAKES THE DYNT A CALLS_IT.
CFGSLC.PLP	Configure an SMLC line definition table entry.
CIRLOG.PLP	STUFFS CIRCULAR BUFFER FOR DEBUG OF NPX
CKNDNM.PLP	Subroutine to check the validity of node name on the name table.
CKSLID.PLP	Subroutine to check the validity of the SLAVID.
COMDEF.PMA	Network common definitions
EXTRAC.PLP	EXTRACTS A SPECIFIC SPARE DATA FIELD FROM A REQ OR RESP MESSAGE
FNSID$.PLP	Search the DIFNS id structure for the id for a given node.
Find_Addr.Plp	Look for an addr block in the network databases.
Find_Name.Plp	Look for a name block in the network databases.
Find_PDN.Plp	Look for a pdn block in the network databases.
GETVCIX.PLP	GETS AN INDEX INTO THE VCDATA FOR THIS USER
GNUSR$.PLP	Gets the network process' user number.
HDLCER.PLP	REPORT INTERNAL ERROR IN NETWORK SYNC SOFTWARE AND DISABLE LINE.
ICSCC.PLP	Routine to process ICS1 code works and control block receive by X.25 level II.
ICSSAV.PMA	Buffers for ICS1 interrupt status and counters.
INIPNC.PLP	Initialize the Ring's cold start timer and line def timers
ISREM$.PLP	Return information on remoteness of a filesystem object.
LKFA.PMA	LOCKFA
LKTA.PMA	LOCKTA
MOVB.PMA	MOVES N BYTES FROM SRC 32 BIT POINTER TO DST POINTER
N$AADR.Plp	Add a node "addr block" to the network database.
N$AHCB.PLP	Add an HCB block and a linedef block to the database.
N$ANAM.Plp	Add a node "name block" to the network database.
N$APDN.Plp	Add a "pdn block" to the network database.
N$APTH.Plp	Add a "path block" to the network database.
N$ASAD.Plp	Add an address to a source address chain
N$INIT.PLP	Initialize all the network databases.
N$IPDN.Plp	Fill the PDN table with known pdn values.
N$LALL.PLP	GATHERS STATISTICS FOR ALL PRIMENET SYNCHRONOUS LINES.
N$LCFG.PLP	GATHERS CONFIGURATION STATISTICS FOR ONE PRIMENET SYNCHRONOUS LINE.

N$LDYN.PLP	GATHERS DYNAMIC STATICS FOR ONE PRIMENET SYNCHRONOUS LINE
N$LOGO.FTN	TELL NETWORK TO SEND FORCED LOGOUT MESSAGE TO REMOTE USER PROCESS
N$NETS.PLP	Do final network configuration and setup.
N$RTRC.PLP	ROUTINE TO GATHER PNC STATICS DATA.
N$RTRC.PLP	Turn network ring tracing on/off.
N$SPME.Plp	Add all the "myself specific" data to the network database.
N$VALL.PLP	GATHERS DATA FOR ALL VIRTUAL CIRCUITS
N$VONE.PLP	GATHERS STATISTICS FOR ONE VIRTUAL CIRCUIT
NBKDEF.PMA	NETWORK NEW BLOCK AND QUEUE DEFINITIONS
NBKINI.FTN	ROUTINE TO INITIALIZE NETWORK BLOCKS AND QUEUES
NCMSUB.FTN	Initiates a HDX Primenet link.
NETABT.FTN	Main "work" loop for network process.
NETCM$.FTN	Handles 'NET' commands for HDX operator interface.
NETDMP.PMA	USED TO TRACE ILLOGICAL SYSTEM FAILURES DURING PRIMOS OPERATION.
NETDWN.PLP	Shuts down networks.
NETEV1.PMA	FIRST-LEVEL EVENT LOGGER (PCL-ABLE VERSION).
NETEV2.FTN	SECOND-LEVEL EVENT LOGGER
NETFIG.PLP	Building ring 0 new-network-configurator databases from old NETCFG
NETMAP.PLP	Subroutine to manage segment mapping for networks.
NETPRC.PLP	NETWORK PROCESS RUNNING IN RING 0
NETRTN.PLP	Subroutine to invalidate network cache on RTNSEG
NETSET.PLP	Checks authorization of user starting network and init network segments.
NETSGS.PMA	COMMON DEFINITION FOR NETWORK MAPPED DATA MOVEMENT SUB-ROUTINE.
NETUTU.PLP	Subroutine to copy from Networks to user space.
NICSOF.PLP	Deconfigure an ICS network line.
NICSON.PLP	Configure an ICS network line.
NNITL.PMA	ALL THAT'S LEFT HERE IS A HALD (FOR FORTRAN STOPS).
NPX$RL.PLP	CALLED BY SLAVE CK TO RETRIEVE THE ENTRY POINT OF ANY HANDLER.
NPX$SL.PLP	CALLED BY SLAVE TO SOTRE ITS ANY HANDLER IN RING 0 DATA BASE.
NPXDNT.PMA	NPXDNT - THE DYNT TO GET NPXPRC DEFINED FOR R$CALL.
NPXON.PLP	Start up NPX slaves.
NPXPRC.FTN	THE RING 0 CALLS TO SUPPORT NPX (ANALOGOUS TO FAMSVC, FAMPR).
NSLCOF.FTN	Subroutine to turn off a network synchronous line.
NSLCON.FTN	Subroutine to configure a network MDLC line.
NSLDN.PLP	BRING ALL RUNNING NETWORK LINKS DOWN.
NSLSTP.PLP	STOP A SYNCHRONOUS LINE FOR PRIMENET.
NSLSTR.PLP	START UP A SYNC LINE FOR PRIMENET.
NSLUP.PLP	START UP ALL CONFIGURED SYNCHRONOUS NETWORK LINKS.
NTINIT.FTN	initialize the network.
NTWMAB.PLP	Warm start code executed by the network process.
PDNDEF.PMA	BLOCK DATA FOR DEFAULT PDN TABLE DEFINITIONS.
PNCDIM.PMA	HARDWARE INTERFACE FOR PRIMENET NODE CONTROLLER (FORMERLLY FARNET).

PRFTMR.PLP	Timer routine for Level II Protocol for Ring Network.
PRHDLC.PLP	Routine to implement X.25 Level II Protocal.
PRHLOG.FTN	Subroutine to enter data into NETREC.
PROALM.PMA	Indicate protocol required and notify network server process
R$ALO1.PLP	This routine increment the ALOCNT by 1.
R$ALOC.PLP	ALLOCATES A VCIX SLOT FOR NODE=XRNODE...IF THE SLOT EXISTS INCREMENTS AN ALLOCATION COUNT.
R$BGIN.PLP	The user callable interface to NPX for synchronous and asynchronousRECALL.
R$CALL.PLP	THE USER CALLABLE INTERFACE TO NPX TO MAKE REMOTE PROCEDURE CALLS
R$CKNT.PLP	Subroutine to check the validity of the supplied node name.
R$CKVC.PLP	CALLED BY LOGABT TO CHECK NPX VIRTUAL CIRCUIT, IF ACTIVE. CLEAR IT.
R$END.PLP	The Asynchronous Remote Procedure Call-eND, check slave's task.
R$MYNM.PLP	Return name of local node.
R$NAME.PLP	Convert node number to node name.
R$NODN.PLP	Add systemname to the node name table (NDNTBL) and return pointer.
R$RLS.PLP	Decrements a pernode allocation count for NPX, if count r the slave is released.
R$SLID.PLP	Subroutine to convert node name to slave id if the VC is secured.
R$SYSN.PLP	Subroutine to return the system name for a given SLAVID.
RLOGIN.FTN	CONTROL USER PROCESS ON TERMINAL SIDE OF REMOTE LOGIN (MOS IDLE).
RNGRCV.PLP	Level II protocol receive for Ring Network.
RNGSND.PLP	Level II protocol transmit logic for Ring Network.
SLAVE.PLP	GIVEN A REQUEST MESSAGE IN BUF, SLAVE CALLS THE TARGET SUB SENDS A RESPONSE.
SLAVER.PLP	THE ROOT TO ALL SLAVE INVOKATIONS...ACCEPTS CALL & DEFINES FIRST MSG BUFFER.
SLAVE_CK.PLP	It is called by DF UNIT to check the usr type, if U$NPX ge SLAVE ON UNIT.
SLCNET.PMA	SMLC INTERRUPT SATUS HANDLER FOR X.25 LEVEL 2.
SLCNSB.FTN	SUBROUTINES FOR PRHDLC TO SLCNET INTERFACE.
SLCPLR.PLP	Call PRHDLC when a queue has something to process.
STOPME. FTN	PRINTS ERROR AND STOPS NPX PHANTOM.
STPNC.PLP	ROUTINE TO GATHER PNC STATISTICS DATA.
STRBL.PLP	ROUTINE TO MOVE THE RING BREAK INFORMATION TO A RING 3 BUFFER
TICKL2.PLP	Tick off level 2 clocks.
TRNRCV.PLP	TRANSMITS AND RECEIVES MESSAGES TO AND FROM SLAVES IN ONE OPERATION UNDER QUITPROTECTION.

UPUS1.PLP	Subroutine to update user status words.
UPUS2.PLP	Subroutine to update user status words.
UPUS3.PLP	Subroutine to update user status words.
User_Valid.Plp	Validate the network initializing user
VCINFO.PLP	Subroutine to check the network status.
X$ADCL.PLP	Routine to add declaration to DCL list.
X$ADR.FTN	Modules to decode addresses from incoming calls.
X$ASGN.PLP	Assign primitive for general users.
X$BID.PLP	Routine to build a restart ID packet (rev 20+).
X$BRLG.FTN	Associate an incoming remote login call request with a user
X$BVCB.PLP	Bind an incoming call request to a declaration block and a
X$CACP.FTN	ROUTINE TO ACCEPT A CALL.
X$CLOK.FTN	BACKGROUND CLOCK FOR LEVEL 3 X.25 - SHOULD RUN EVERY 10 SECONDS.
X$CLRA.FTN	ROUTINE THAT CAN BE USED TO CLEAR ALL CONNECTIONS A USER OWNS
X$CNFG.PLP	Return pointers to various configuration related objects.
X$COPY.FTN	ROUTINE TO COPY PACKET INTO AN UNWIRED BUFFER.
X$CREQ.FTN	PROCESS AN INCOMING CALL REQUEST.
X$DDCL.PLP	Remove declaration from list.
X$DIAG.PLP	Handling of level 3 diagnostic packets.
X$FCNF.FTN	Fortran callable version of configuration data lookup routine.
x$fcty.plp	Facilities parsing for call request/incoming call packets.
X$FLDS.FTN	X$FLDS - Get all of the fields in a CREQ, ACCEPT, or CLEAR p
X$GBCD.FTN	X$GBCD - ROUTINE TO COPY BCD DIGIT STRING TO ASCII STRINGS
X$GETU.FTN	ROUTINE TO HANDLE OUTPUT PACKETIZING
X$GIVU.FTN	X$GIVU - ROUTINE TO TRY TO GIVE DATA PACKETS TO USER LEVEL
X$GVVC.FTN	PASS CONTROL OF A VIRTUAL CIRCUIT TO ANOTHER USER
X$HDOW.FTN	ROUTINE TO SHUTDOWN X.25 LEVEL 3 FOR A GIVEN HOST
X$HDWN.PLP	Clears all virtual circuits to the specified host
X$IDNT.FTN	Routine to build a restart ID packet (rev 17.3+)
X$IPKT.FTN	TAKE INCOMING PACKETS FROM LEVEL II PROTOCOLS
X$LINK.FTN	Links network table entries together for HDX on-the-fly configuration
X$LINK.PLP	Line network table entries together for HDX on-the-fly configuration
X$LOOP.FTN	ROUTINE TO PROCESS PKTS THAT START AND END IN THE SAME MACHINE.
X$MAP.PMA	POINTERS TO IMPORTANT NETWORK STRUCTURES.
X$NORM.FTN	DECODE CMND BYTE AND DO ROUTINE WINDOW UPDATES/CHECKS
X$NTFY.FTN	WAIT ON AND KICK USER'S NETWAIT SEMAPHORE
X$PRIM.FTN	NETWORK PRIMITIVES
X$RLG.FTN	HANDLE USER SIDE OF REMOTE LOGIN
X$RLT.FTN	LO-THRU MODULES - TERMINAL SIDE OF REMOTE LOGIN
X$RSET.FTN	ALLOW A USER TO CAUSE A RESET ON ONE OF HIS VIRTUAL CIRCUITS.
X$RT.PLP	Ring 0 support for route through configuration information
X$RTL.PLP	Set up this process to run as the route-through server
X$SRNR.FTN	X$SRNR - ROUTINE TO SEND RNR ON A VIRTUAL CIRCUIT

X$STAT.FTN	ROUTINE TO RETURN STATUS INFORMATION TO USER SPACE
X$UASN.PLP	Unassign primitive for general users
X$ULNK.PLP	Unlink the network table entries and put a site 'offline'
X$USRQ.FTN	ROUTINE TO PUT VCB IN A USER'S QUEUE OF VCBS
X$UTIL.FTN	ALL OF THE NETWORK SOFTWARE UTILITY ROUTINES
X$VID.PLP	Routine to verify a restart ID packet (rev 20+)
X$VLNK.PLP	Verifies that network table entries are linked together as expected
X25DEF.PMA	X.25 NETWORK COMMON DEFINITIONS (UNWIRED)
XLASGN.PLP	Extended declaration of interest in incoming calls
XLGC$.FTN	XLGC$ - GET ALL OF THE FIELDS IN A CONNECT REQUEST PACKET
XLUASN.PLP	Unassign an extended declaration
XMTRCV.PLP	Transmits and receives message to and from slaves in one operation under quit protection.

2.6 RJES-ROUTINES

DATCPY.PLP	This routine copies data to the Trace Buffer !!
GETCP.PLP	PH/WRK - return pointer to area used to pass PH config
GRTS.PLP	Protocol specific handler for the GRTS protocol.
GRTSCK.PLP	GRTS Protocol Specific Check module
HASP.PLP	HASP protocol specific RJPROC code
HASPCK.PLP	HASP Protocol Specific Check module
PHDBG.PLP	PH - returns addresses of common area for protocol handler
readqt.plp	Routine reads entry off primos queue
RJ$ATT.PLP	RJ1 interface routine - allows process to attach for lin
RJ$I.PLP	RJI routines return information/data to the user from the protocol handler
RJ$MSG.PLP	RJPROC message returning routine
RJ$O.PLP	RJI routines will output blocks, control messages, detach (disable) line
RJALQU.PLP	Create a queue and a queue control block given a chunk o memory
RJCDF.PMA	COMMON DECLERATIONS FOR RJE EMULATORS
RJCKPC.PLP	To valid the request protocol and character code
RJCMTR.PLP	Configure MTR sub-process for protocol handler
RJCPY.PLP	RJI-PH-routine copies xmit blocks into wired xmit buff
RJDBG.PLP	Debug gate returns pointer to RJI common blocks for work
RJDLIN.PLP	Deconfigure line
RJES_ VERSION.PMA	List of RJES version numbers !!
RJEVNT.PLP	Event handler for the Rjproc system
RJGBDQ.PLP	RJI-PH routine - get a data block off a devide queue
RJINI.PLP	Cold start code for RJE emulators
RJLINE.PLP	Low level routines for Rjproc
RJMNIT.PLP	Ring o code required to run the Monit facility
RJPCDF.PMA	Protocol handler common declerations for rje emulators
RJPHFS.PLP	rje emulators - routine manages the dim free store area
RJPHLC.LPL	rje emulators - routine assigns a line control block
RJPHS.PLP	RJI Ph routine - modify protocol handler state in the wo RJI database
RJPLO.PLP	Logout code for protocol handlers
RJPMSG.PLP	RJPROC message printing routine
RJPROC.PLP	Main driver for RJE emulator process
RJQ.PLP	RJI queueing routines using ROCB
RJRBRQ.PLP	Protocol handler - copy contents of receive block and queue the worker
RJRECV.PLP	Receive routines for RJPROC
RJRQST.PLP	Worker request processor for RJPROC
RJRTRY.PLP	Routines supporting RJPROC retry mechanism
RJSCH.FTN	This program sets up DMC channels for a logical SMLC line
RJSLCFG.PLP	Configure HSSMLC, MDLC and LYNX for RJE use
RJTIM.PLP	Timer routines for the Rjproc system
RJTWKR.PLP	Send Messages to Ring3 Workers via RJI
RJUNDO.PLP	Logout code for RJE emulators.
RJWLO.PLP	Logout code for RJI workers.
RJWRFS.PLP	RJE emulators - routine manages RJI system free store

RJWRLC.PLP	RJE emulators - routines assign and unassign control block line
rjxmit.plp	Transmit routines for RJPROC
X80.PLP	X80protocol handler
X80CK.PLP	X80 Protocol Specific Check module
XBM.PLP	XBM line events and timeouts
XBMCK.PLP	Determine type of message from MTR (XBM Link level) processing
XBMCOM.PMA	ALLOCATE SPACE FOR XBM CAT QUEUES

2.7 SNAS-ROUTINES

PRIMOSCOMO_-E

SNA$CF.FTN	Create the Free Storage classes for SNA Free Storage
SNA$CX.FTN	Create the Free Storage classes for PRIME/SNA RJE
SNA$IADM.PLP	Administration Control Request Gate
SNA$IAN.PLP	Create and send a START 3270 LECB to the LU Manager
SNA$ICLS.PLP	Close established Mate-Manager connection
SNA$IGD.PLP	Build and send a GET DEVICE LECB to the LU Manager
SNA$IGE.PLP	Retrieve a message for a LU Mate from the LU Manager
SNA$IOPN.PLP	Open connection between mate and manager
SNA$IRD.PLP	Build and send a RETURN DEVICE LECB to the LU Manager
SNA$IRS.PLP	Build and send a RECOVER SESSION LECB to the LU Manager
SNA$ISS.PLP	Build and send a SUSPEND SESSION LECB to the LU Manager
SNA$IST.PLP	Build and send a CHECK STATUS LECB to the LU Manager
SNA$ISTA.PLP	Administration Status Request Gate
SNA$ISTP.PLP	Adminstration Stop Request Gate
SNA$IWR.PLP	Build and send a WRITE DATA LECB to the LU Manager
SNA$PH.PLP	Create an SNA Service for an SNA Adminstrator
SNASEC.PLP	Security Check for SNA gates
SNA_GATES.PM	Gated interludes to standard routines for Free Storage and IPQNM
SNA_ICHK.PLP	Do connection checks for Gate routines
SNA_IGET.PLP	Obtain a buffer from Mate free pool
SNA_ICCK.PLP	Set the specified Interlock
SNA_IRO.PMA	PRIME/SNA Interactive Ring O Database.
SNA_IRCV.PLP	Obtain a request queued from the LU Manager
SNA_IRLS.PLP	Return a buffer to free pool
SNA_ISND.PLP	Queue a request to the LU Manager

3 SYSTEM CONFIGURATION INFORMATION

1. PROCESS EXCHANGE MECHANISM - PXM

- process control blocks (PCB's)
- ready list
- wait list

MEMORY MANAGEMENT

- descriptor table address registers (DTAR's)
- segment descriptor words (SDW's)
- page maps (HMAP's, LMAP's, MMAP's)
- ptuseg
- paging disk map (PDMAP)

FILE SYSTEM

- locate buffers
- unit tables (UT's)
- unit table entries (UTE's)

2. STACKS

- interrupt stack (INTSK)
- page fault stack (PGFSTK)
- unwired ring0 stack (SUPSTK)

3. OTHER AREAS

- user profile common (UPCOM)
- per user data common (PUDCOM)
- locks
- disk queue blocks (DQB's)
- supervisor common (SUPCOM)
- user type array (UTYPE)
- configuration common (FIGCOM)
- register save area (RSAV)

4 PROCESS CONTROL BLOCKS - PCB's

1. THE PXM IS MADE UP OF 3 MAIN ELEMENTS

   - PROCESS CONTROLL BLOCKS
   - READY LIST
   - WAIT LIST

1. Used by both the software and the microcode.
2. Contains all the essential information of all the processes on the system
3. The PCB's are located in segment 4, location '600 - '640.

5 READY LIST

1. The ready list is used by the micrcode to indicate priorities and dispatch processes.

2. A series of PCB's actually make up the ready list as well as two 32 bit registers called PPA and PPB located in the microcode scratch area (PPA-pointer to process A and PPB-pointer to process B)

3. The dispatcher (user -22) always runs the highest priority and can preempt any process.

6 WAIT LIST

1. The wait list specifies a group of processes that are waiting for an event to occur. The wait list is made up of 2 major elements:

   - a semaphore
   - a data base made up of PCB's

2. Each process or linked lists of PCB's in the wait list, wait on a semaphore for the event to occur.

7 MEMORY MANAGEMENT

- descriptor table address register (DTAR)
- segment descriptor words (SDW)
- page maps (HMAPS, LMAPS, MMAPS)
- ptuseg
- paging disk map (PDMAP)

8 DESCRIPTOR TABLE ADDRESS REGISTER - DTAR

1. There are 2 DTAR's in each PCB 4 DTAR's in each register set.

2. contains the physical address of segment descriptor tables.

3. Together with the SDT's and the hardware page maps the DTAR is used to help translate virtual to physical addresses in the STLB logic.

9 SEGMENT DESCRIPTOR WORDS - SDW

1. SDW's contains the physical address of the start of the page table for a given segment.

2. Must be wired.

3. The tables for DTAR 2 and DTAR 3 are located in each user's page fault stack (PGFSTK).

10 PAGE MAPS (HMAPS, LMAPS, MMAP)

1. The HMAPS and LMAPS contains the physical page number.

2. Beginning with rev 19.2, PRIMOS supports different versions in order to support 16MB of physical memory for various type processors.

3. The HMAPS and LMAPS are setup by the software and must be wired.

4. The HMAPS and LMAPS are used by the software and the microcode.

5. The MMAPS is a software only database which must be wired.

6. The MMAPS contains one entry for each physical page whether they are in use or not.

11 PAGE TO USER SEGMENT (PTUSEG)

1. One entry (2 words) for every segment in the system.

2. Does not contain entries for "WINDOWED" segments.

3. The initial VMFA segments are allocated at the end.

4. PTUSEG can be found in segment 14. Each entry (2 words) will contain the user # and the segment #. Each time a user logs out, that particular segment for that user is available for use.

12 PAGING DISK MAP (PDMAP)

1. Used to allocate records on the paging surface.

2. Each bit represents 8 records on the paging disk.

3. PDMAP can be found in segment 14.

13 FILE SYSTEM

- LOCATE BUFFERS
- UNIT TABLES (UT's)
- UNIT TABLE ENTRIES (UTE's)

14 LOCATE BUFFERS

1. Serves as a cache for disk access.

2. Active, one per user is mapped to a buffer

3. Each buffer has an associated buffer control block (BCB) 1BCB/locate buffer. each BCB is wired.

4. Locate buferrs are only wired when in transistion. a process can only own one locate buffer at a time.

5. locate buffers can be found in FS>LOCATE.PMA .

15 UNIT TABLES (UTs)

1. A UT is a list of pointers to UNIT TABLE ENTRIES (UTE's).

2. 1 UT per user. A maximum of 32768 units per user.

3. Contains attach points and file units.

4. Per user UT's are allocated and deallocated dynamically

16 UNIT TABLE ENTRIES (UTEs)

1. A UTE desribes a file system object that is currently in use via the file system.

2. One UTE exist per open file or attach point and contains all necessary file information.

3. A type for each of the following:

   - attach points
   - local files
   - remote files

17 STACKS

- INTERRUPT STACK (INTSK)
- PAGE FAULT STACK (PGFSTK)
- UNWIRED RING0 STACK (SUPSTK)

18 INTERRUPT STACK (INTSK)

1. The INTSK is located in segment 4. This is the only stack used for all interrupt processes.

2. Contains the phantom interrupt code for all the controllers as well as the RSAV area for machine checks.

3. For more information on the INTSK, refer to SEG4.PMA and PRIMOS INTERNALS.

19 PAGE FAULT STACK (PGFSTK)

1. This area is wired when a user logs in and is limited in size.
2. The page fault handler is locted in segment 6.
3. The following fault handlers exist in segment 6:
   - process fault
   - page fault
   - UII
   - access violation
   - semaphores (overflow or underflow)
   - segment fault
   - pointer fault
4. any other faults taken in ring0 will take a halt instruction.

20 UNWIRED RINGO STACK (SUPSTK)

1. The SUPSTK is only allocated 8 pages.

2. The SUPSTK is located in segment 6003.

3. The SUPSTK is the most used stack by PRIMOS.

21 OTHER SYSTEM INFORMATION

1. USER PROFILE COMMON (UPCOM)

   - UPCOM is located in segment 6000/16000

2. PER USER DATA COMMON (PUDCOM)

   - PUDCOM is located in segment 6000/000000.

3. SUPERVISOR COMMON (SUPCOM)

   - SUPCOM is located in segment 6/1400

4. CONFIGURATION COMMON (FIGCOM)

   - FIGCOM is located in segment 14/700

5. REGISTER SAVE AREA (RSAV)

   - The RSAV area is located in segment 14/2000 - 14/3777.

22 LOCKS

1. N1LOCKS

- multiple readers or 1 writer locks
- Set of hierachical system locks
- Prevents deadlocks (deadly embrace) and race conditions
- Allows access to critical databases to only one process at a time (if writing)

2. MUTUAL EXCLUSION LOCKS

- Strewn over the entire system.

23 DISK QUEUE BLOCKS

- Database used to communicate information about disk requests between a user process and the disk interrupt process.

- Each entry has multiplexed data.

- The amount of disk queue blocks available depends on the revision of primos.

24 VARIOUS CONFIGURATION INFORMATION

1. MAXPAGE is located in segment 14.

2. The ECCCNT count is located in segment 4. This is where the memory ECCC errors are recorded.

3. PAGEDEV is located in segment 14.

4. ALTPAGEDEV is located in segment 14.

5. VPDEV is located in segment 11. This is where the data partitions are kept from the addisk command.

6. NUSR is located in segment 6.

7. DISKIO is located in segment 6. This is where the disk queue request block begins.

25 CRASH DUMP DEBUGGING APPROACH

NOTE

ONCE THE TAPE DUMP HAS BEEN READ IN, WE MUST DETERMINE WHAT WENT WRONG. THE FOLLOWING PROCEDURE IS SOME BASIC STEPS THAT SHOULD BE TAKEN WHEN ANALYZING ALL TYPES OF CRASH DUMPS.

WITH THE AID OF THIS STUDENT GUIDE AND OTHER REFERENCE MATERIALS YOU WILL BE ABLE TO EXAMINE THE CRASH DUMPS AND ATTEMPT TO RESOLVE THE PROBLEM(S) IN A LOGICAL MANNER. THE OBJECTIVE IS TO ELIMINATE AREAS OF THE SYSTEM THAT MAY NOT BE A FACTOR IN A PARTICULAR DUMP, SUCH AS I/O, MEMORY, CPU, OR PRIMOS.

ACTIONS	COMMANDS
1. DETERMINE THE SYSTEM MODEL TYPE, PRIMOS REVISION, MICRO-CODE REVISION, AND HALT TIME/DATE.THIS INFORMATION IS VERY USEFUL. REFER TO COMMAND 1.	1. USE THE **DATE** COMMAND. THE HALT TIME/DATE WILL BE DISPLAYED ON THE TOP LINE, FAR RIGHT.
2. IF THE SYSTEM TYPE IS AN 850, REFER TO COMMAND 2 TO DETERMINE WHICH CPU (ISU) HAS HALTED.IF NOT AN 850, CONTINUE WITH ACTION 3.	2. USE THE **RD AP** COMMAND (REGISTER DUMP FOR ASSOCIATED PROC) AT LOCATION XXXX, IF CONTENTS=041004, ISU#1 HALTED. IF CONTENTS=102010, ISU#2 HALTED.
3. DETERMINE WHAT USER WAS EXECUTING WHEN THE SYSTEM CRASHED (LIVE USER). REFER TO COMMAND 3. IF SYSTEM TYPE IS AN 850, THERE IS A LIVE AND LAST USER FOR BOTH PROCESSORS. REFER TO COMMAND 3a.	3. USE THE **RP -LIVE** COMMAND TO DISPLAY THE LIVE USER AND REGISTER PRINT. 3a. THE LIVE AND LAST USER COMMANDS ARE AS FOLLOWS. FOR ISU#1, **RP -LIVE** AND **RP -LAST**. FOR ISU#2, **RP -LIVE -SLAVE** AND **RP -LAST -SLAVE**.

ACTIONS	COMMANDS
4. ACQUIRE MORE DETAIL OF THE LIVE USER SUCH AS, (1)USER NAME,(2)PROCESS ID,(3)LAST WAIT LOCATION, (4)USER PRIORITY,(5)LOCKS OWNED. REFER TO COMMAND 4.	4. USE THE STATUS <user#> COMMAND FOR MORE DETAIL OF THE LIVE USER OR ANY USER.
5. FIND THE MODULE IN WHICH THE LIVE USER WAS EXECUTING. REFER TO COMMAND 5. NOTE: *this is normally considered the halt location.*	5. USE THE LOSEARCH XX/XXXXXX COMMAND FOR THE EXECUTING MODULE NAME. XX/XXXXXX BEING THE PB REGISTER ADDRESS TAKEN FROM THE STATUS INFO. OR THE RP LIVE INFO.
6. EXAMINE IN MORE DETAIL THE STEPS THE LIVE PROCESS TOOK, UP UNTIL THE SYSTEM HALTED. THIS IS DONE BY RETRACING THE LIVE USERS PROCESS STACK. REFER TO COMMAND 6.	6. THE TRACE <user> COMMAND WILL ALLOW YOU TO TRACE A USERS STACK FRAMES. SUBCOMMANDS ALLOW YOU TO EXAMINE INDIVIDUAL STACK FRAMES IN MORE DETAIL SUCH AS THE DMSTK COMMAND.

NOTE

UP TO THIS POINT, WE HAVE DETERMINED WHO THE CURRENT AND LAST RUNNING USER WAS AS WELL AS, WHERE THE USER HALTED AND THE STEPS IT EXECUTED PRIOR TO, AND UP UNTIL THE SYSTEM HALTED OR HUNG. WE'VE ALSO DETERMINED WHICH ISU HALTED(850 SYSTEM ONLY).

THE FOLLOWING STEPS WILL CONTINUE TO BREAK DOWN THE CRASH DUMP EVEN FURTHER BY REFERRING TO THE TROUBLESHOOTING FLOW CHARTS.

ACTIONS	COMMANDS
7. DETERMINE IF THE SYSTEM EXPERIENCED ANY CHECKS i.e. MCHK,MMOD,ECCC,ECCU, REFER TO COMMAND 7.	7. USE THE CHECK COMMAND. IF SOME TYPE OF CHECK EXISTS, REFER TO FLOW CHART CH100.
8. DETERMINE THE HALT LOCATION FROM STEP 5. REFER TO COMMAND 8.	8. IF THE HALT IS A MMOD_ REFER TO FLOW MM300. IF THE HALT IS A BOOT0 REFER TO FLOW BT500. IF THE HALT IS A PAGES_ REFER TO FLOW PA600.IF THE HALT IS A IPAGF_ REFER TO FLOW PF700.IF THE HALT IS A PGMPA_ REFER TO FLOW PG800.
9. IF THE SYSTEM APPEARS BE HUNG REFER TO COMMAND 8	9. FOR A SYSTEM HANG PROBLEM REFER TO FLOW HG400.
10. IF NONE OF THE ABOVE TYPE HALTS EXIST, REFER TO COMMAND 10.	10. IF NO CONCLUSIVE DATA IS ISOLATING THE PROBLEM FROM THE PREVIOUS STEPS USE THE DOC UTILITY IN AN ATTEMPT TO ISOLATE THE PROBLEM.

TROUBLESHOOTING FLOW CHART

- MACHINE CHECK HALTS

- MISSING MEMORY MODULE HALTS - MMOD__

- SYSTEM HANGS

- LABELED HALTS - BOOT0/PAGES__/IPAGF__/PGMPA__

MACHINE CHECK HALTS - MCHK_

CH100

(A)

1. Determine the type of machine check by decoding the DSW registers for the appropriate CPU type.
2. If registers decode to an ECCC or ECCU, refer to CH200-A.
3. If registers decode to a machine check with an internal CPU parity error, refer to CH-A100-A.
4. If registers decode to a DMX/IO parity error, continue with CH100-B.
5. If registers decode to a MMOD_ refer to MM300-A.

If a machine check has occurred, a parity error exists in the system, normally due to bad hardware. In a 9955 system, a single bit parity error results in a recoverable machine check. A double bit parity error would halt the machine.

(B)

1. Use the **LOSEARCH** command on the machine check location.
2. Restore the machine check location segment and access the offset. Backward trace the instructions being executed prior to the halt. refer to CH-A101.

This procedure will display the routine being executed at the time the system halted as well as, the instructions executed prior to the halt.

(C)

1. Determine the last vectored interrupt that occurred. This address is saved for 9000 model cpu's only. Refer to CH-A101.

The last vectored interrupt can be very useful when attempting to isolate a bad controller that could have issued bad parity.

MACHINE CHECK HALTS - MCHK_

CH101

(A)

1. Check for any disk activity.
2. Check for any discrepancies in the disk subsystem such as, DMA overruns, controller hangs, or disk I/O timing problems. refer to CH-A102-A.

If any disk activity exists in the system, there should be used disk queue blocks. The amount of disk queue blocks available will depend on the revision of PRIMOS.

(B)

1. Use the TT -U1M command to Check the user 1 message buffer for any disk error messages.

Any disk errors that occur in the system will be displayed on the system console. The user 1 message buffer will contain the most recent error messages.

(C)

1. Check each used disk queue block for more detail of each disk process. Refer to CH-A102-C.

Examining each used disk queue block will allow you to acquire more data of each disk process such as, (1)the user #, (2)the partition #, (3)disk statuses, (4)op code, and other data.

If the disk subsystem appears to be fine, continue with flow CH102.

MACHINE CHECK HALTS - MCHK_

CH102

(A)

1. Check for any AMLC controllers configured in the system. .
2. Determine which AMLC controller had recently interrupted. Refer to CH-A103-A.

The AMLC phantom interrupt code located in segment 4 will contain the device address of any AMLC controller configured. This area will also contain the last AMLC controller that interrupted.

1. If the AMLC controllers do not appear to be a problem, refer to CH103.

MACHINE CHECK HALTS - MCHK_

CH103

(A)

1. Determine the last user to have assigned the tape drive.
2. Verify that the user still owned the tape drive at the time of the system halt. Refer to CH-A104-A.

The objective here is to determine if any tape activity existed at the time of the halt and weather or not the tape subsystem may of had any effect to the system halting.

(B)

1. Determine whether the user had completed its tape process.
2. Check for any faults or aborts by tracing the user's stack and checking the user's PCB. Refer to CH-A104-C.

Tracing the user's stack and looking at the user's PCB should help reveal any problems the user may have encountered during its tape session. Any problems could be hardware or software induced.

(C)

1. If the tape subsystem does not appear to be a problem, refer to CH104-A.

MACHINE CHECK HALTS - MCHK_

CH104

(A)

1. Check the live and last user's PCB for any useful data that may help find the origin of the problem.
2. Trace the last user's stack for any fault or aborts that could have effected the live process when the system halted. Refer to CH-A105-A.

Here are some final checks in an attempt to find any problems that may have been overlooked. The live and last user PCB frame and the live and the last users stack frame may contain some useful data.

(B)

1. Check the file system for any inconsistencies. Refer to CH-A105-C.

This option checks the unit tables for any suspecting hash thread problems. Segment 14 will contain the register save area for any register information.

(C)

1. Check the page maps for consistency. Refer to CH-A105-E.

This option checks the HMAP, LMAP, and MMAP table for consistencies. If any errors have occurred, it will be displayed.

(D)

1. Check the status of all the PRIMOS N1LOCKS.
2. Check RSAV are for any register information that may be useful. Refer to CH-A105-G.

If the problem to the system halt is not found up to this point, a suggestion would be to run DOC as well a referring the tape dump to an analyst well experienced in autopsy. Do not replace any unnecessary hardware.

END MCHK_ FLOW

MACHINE CHECK HALTS - ECCC/ECCCU

CH200

(A)

1. Check and decode DSWRMA.
2. Determine the amount of eccc errors that may have occurred. Refer to CH-A200-A.

Determining the amount of eccc errors could help verify the reason why the system may of halted. Primos will record a maximum of 2000 eccc errors prior to going to silent mode. The DSWRMA register will help break down the errors to the ppn and memory array board.

(B)

1. Use the page check command to verify the integrity of the hmap and lmap tables. If any discrepancies are displayed, refer to CH-A200-C.

Because the memory eccc or eccu may not be the entire reason for the system halting,be sure to verify what other processes the CPU may have been executing by referring back to CH100.

END ECCC/ECCU FLOW

MISSING MEMORY MODULE - MMOD_

MM300

(A)

1. Check for any eccc/eccu memory parity errors.
2. Check the memory page map tables for any discrepancies.
3. Decode the DSWRMA register. Refer to MM-A300-A.

A MMOD_ halt is a memory location that has been addressed that either doesn't exist or is unavailable. This can be a difficult halt to troubleshoot. The problem is normally hardware induced and can be located in memory, I/O, or CPU.

(B)

1. Check the DSW registers for any I/O operation being executed from a controller that could have issued a bad address.
2. Determine what controllers may have issued a DMX request or has just completed a DMX operation. Refer to MM-A300-C.

If the MMOD_ halt occurred during a DMX request or DMX transfer, the halt could have been caused by a bad address issued from a controller. Possible incorrect values in the DMA registers,dmc cells, dmt channel programs,or dmq headers.

END MMOD_ FLOW

SYSTEM HANGS

HG400

(A)

1. Use the **STAT US** command to check the amount of active users.
2. Check the ready/wait list.
3. Determine if the majority or all of the active users are waiting on the same semaphore. Refer to HG-A400-A.

System hangs can be very difficult to troubleshoot because the amount of data available is very limited. The problem can be hardware,software, or micro code induced. The most effective means of problem isolation is to begin troubleshooting the pxm area.

(B)

1. Check the PCB of the live and last user as well as other users that are suspected of hanging the system.
2. Determine what N1LOCKS may be owned by any suspected user(s). Refer to HG-A400-C.

Because the backstop process does not wait on a semaphore and runs when the pxm is less active and the clock process preempts any process at certain time intervals, steps 1 and 2 are not necessary if the live process is the clock or backstop.

(C)

1. Check for any I/O activity that may have caused the system to hang.
2. Check the system micro code revision for any reported problems. Refer to HG-A400-D.

It can be very common for an I/O controller to cause a system hang, especially communication controllers. Determining what I/O process may have been running when the system hung, can prove to be good practice in isolating the cause of the hang.

END SYSTEM HANG FLOW

LABELED HALTS - BOOT0

BT500

(A)

1. Check the user 1 message buffer.
2. Trace the live user's stack. Refer to BT-A500-A.

High level languages normally call the boot routine to halt, which normally halts the machine at BOOT0. This halt is normally software induced but, can be hardware related.

(B)

1. Check for any significant changes or modifications in the system configuration. This would include the hardware and software. Refer to BT-A500-C.

This halt can tend to appear after PRIMOS or some other high level language has been modified or revised.

(C)

1. Check for any unusual activity in the disk subsystem and the communications area. This may help when problem isolation is difficult for this particular halt. Refer to BT-A500-E.

If PRIMOS modifications may perform differently from machine to machine due to differences in the system configuration. Checking for unusual events could help isolate these type problems.

END BOOT0 FLOW

LABELED HALTS - PAGES_

PA600

(A)

1. Check the fault address of the live user.
2. Check the live user's stack to reveal any repetitive faults that may have occurred that could have corrupted or trashed the unwired ring0 stack. Refer to PA-A600-A

A PAGES_ halt indicates that a page fault has occurred in the unwired ring0 stack in segment 6003. This halt is handled by the software in the ring0 fault table. This problem can be either hardware or software induced.

(B)

1. check for excessive disk and/or network activity. Refer to PA-A600-C.

Excessive paging or disk errors during paging process could have an impact on the unwired ring0 stack becoming overflowed or trashed. Excessive network errors due to bad hardware could also cause the unwired ring0 stack getting trashed.

(C)

1. Check memory for any discrepancies or inconsistencies. Refer to PA-A600-E.

Unreliable hardware in the CPU and memory will have an impact on this halt when referencing areas of memory due to paging.

END PAGES_ FLOW

LABELED HALTS - IPAGF_

PF700

(A)

1. Check the live users stack for any faults.
2. Check memory for any discrepancies or inconsistencies. Refer to PF-A700-A.

This halt normally occurs when a page fault has occurred while hardware interrupts are inhibited. This halt is handled by the software in the interrupt fault table but, can be hardware or software induced.

(B)

1. Check for any I/O activity. Refer to PF-A700-C.

A bad device interrupt issued from a controller could cause an IPAGF_ halt.

(C)

1. Check for the interrupt stack (segement 4) being overflowed. Refer to PF-A700-E.

An excessive amount of paging activity and interrupt activity could have an impact in the interrupt stack being overflowed. this is normally due to a problem with PRIMOS.

END IPAGF_ FLOW

LABELED HALTS - PGMPA_

PG800

(A)

1. Check memory for any discrepancies or inconsistencies.
2. Trace the live user's stack for any process faults. Refer to PG-A800-A.

This halt normally indicates that, the address of a page map entry was just calculated but is not in a segment that contains page maps. This halt is normally hardware induced, usually the CPU or memory.

(B)

1. Check the live user's PCB frame.
2. Check the live user's page map tables for an invalid page map entry. Refer to PG-A800-C.

Checking the live user or suspected user's PCB frame will contain any fault addresses and any abort flags that may have occurred.

END PGMPA_ FLOW

MACHINE CHECK HALTS - MCHK_

CH-A100

(A)

1. If registers decode to an rcm parity error, replace cs board or the CPU hardware that contains the rcc logic. If not, continue.

The rcm logic is a register located on the cs board that is used as a pipeline register to hold the contents of the current microword being executed. This logic is all microcode controlled and the microcode must be reloaded if a parity error occurs in this logic.

(B)

1. If registers decode to a parity error detected along the BPA or bma, replace the associated CPU logic(i.e. cs board, j board, a board), providing an I/O controller did not pass along bad parity to the CPU.

The I/O controllers interface to the CPU along the BPA and BPD buses. any bad parity detected along this bus can be initiated by either the CPU or the I/O.

(C)

1. If the I/O is suspected as the problem, refer back to CH100 and continue debugging the I/O.

MACHINE CHECK HALTS

CH-A101

(A)

1. If an EIO instruction is found prior to the machine check location, then decode EIO address(refer to sys. arch guide). The EIO address will also display the register name which contains the device address located in the RP -LIVE display.
2. The device address that is displayed in the address most likely caused the halt providing the cpus i/o interface (CS BD.) did not generate the parity error. This may be confirmed by the PIO instruction type.

The EIO instruction is performed by program to implement I/O. This instruction will send control information to a peripheral device as well as move data between a device and the cpu. The EIO instruction is not used for DMX transfers.

(B)

1. If no conclusion at this point, refer to CH100-C.

(C)

1. Dump segment 14 locations 2400 - 2477. Location 2432 will contain the vectored interrupt address, i.e. DUMP 14 2400 2477.
2. Use the **LOSEARCH** command on the vectored interrupt address in segment 4 to display the interrupt process, i.e. LOSEARCH 4/<vec add>.
3. The controller associated with interrupt process could be suspected bad.

The cpu monitors the I/O backplane for any interrupt requests. There are two types of interrupt modes standard and vectored. The controller supplies it's vector address along the BPA in vectored mode, whereas the software routine supplies the pointer address to the interrupting controller in standard mode.

(D)

1. Continue to isolate the I/O, refer to CH101.

MACHINE CHECK HALTS - MCHK_

CH-A102

(A)

1. Use the DD -USED command to display any disk queue blocks currently in use.
2. Use the DD -METER command to verify if any problems with disk I/O has occurred. Note: all values in the disk metering are accumulative.
3. If the disk meter displayed any dma overruns or registered any controller hangs, a problem exists in the disk subsystem. Most probable cause could be a disk drive, disk controller, or a cable. Continue with flow to further isolate.

If there are no disk queue blocks in use, then most likely there was no disk activity. If the disk meters show any significant errors, then the disk susbsystem could be suspected as the source to the problem.

(B)

1. If disk activity exists, refer to CH101-B. If no activity exists, refer to flow CH102-A.

Any disk errors that may be displayed in the console message buffer should help in guiding you to the suspect drive or controller. Further isolation can be made with the following steps shown below.

(C)

1. Use the DD <dqb #> command to display the disk queue block in detail. Together with the steps shown above, this will help isolate the problem to the partition level as well as the user who initiated the disk process.
2. Trace the stack of the user displayed in each active disk queue block and check for any faults or aborts that may have occurred. The problem may have occurred due to unreliable disk media or the disk drive hardware.
3. If a problem is suspected in the disk subsystem, it should be isolated at this point.

The disk queue blocks will display numerous amount of data which is helpful in further isolating the problem to a disk drive, controller, or partition.

(D)

If the disk susbsystem appears to be fine, continue with flow CH102.

MACHINE CHECK HALTS - MCHK_

CH-A103

(A)

1. Check the AMLC phantom interrupt code (PIC) located in segment 4 locations 105 - 124. If no AMLC controllers are configured, refer to CH103-A.
2. Check the AMLC common interrupt handler located in segment 4 location 436. Use the restore command in order to use symbolic mode. At location 436, there will be a DAC instruction pointing back to the location of the last interrupting AMLC device address.

Checking the AMLC PIC code and the AMLC common interrupt handler display the AMLC controller addresses, if any are present as well as, the last AMLC controller that interrupted.

(B)

1. If the LIVE or LAST user was an AMLC process, then the interrupting controller address found from the previous step can be suspected bad.
2. If the last vectored interrupt was an AMLC process, then the interrupting controller address found from the previous step can be suspected bad.

(C)

If the AMLC controllers do not appear to be a problem, refer to CH103-A.

MACHINE CHECK HALTS - MCHK_

CH-A104

(A)

1. Dump segment 4 locations 207-212 (MTPTRS). These locations will contain the user pcb number who used the tape subsystem last.
2. Dump segment 6 locations 23177-23203 (MT1LCK) and segment 6 locations 23204-23207 (MT2LCK). These locations will contain the user number who has the tape drive assigned to it.

These locations contain the mag tape pointers(MTPTRS),mag tape 1 lock and mag tape 2 lock(MT1LCK/MT2LCK) is useful in isolating the tape subsystem. If any tape activity was present, these areas will contain the USER ID currently using the tape subsystem.

(B)

1. If no tape activity exists, refer to CH104-A. Otherwise, continue with CH103-B

(C)

1. Use the TTY <user #> command to check the user's tty buffer who last owned the tape drive to determine if it ever completed it's tape utility. Continue to isolate tape subsystem with the following steps.
2. Use the TRACE <user #> and the PCB <user #> command to check the user's stack and pcb frame for any faults or aborts that may have occurred due to bad hardware or possibly a software problem. Verify the hardware before suspecting the software.

The user's tty buffer should display if the user ever completed it's tape session with the message "magrst or magsav completed".

CONTINUE

MACHINE CHECK HALTS - MCHK_

(D)

1. If the LIVE or LAST user was a tape process, then the tape subsystem can be suspected bad if any problems were found from the previous steps.
2. If the last vectored interrupt was a tape process, then the tape susbsystem can be suspected bad if any problems were found from the previous steps.

(E)

1. Use the TT -U1M message command to verify if any tape activity was performed from the system console. There are potential problems when using the system console for MAGSAVS and MAGRSTS.

MAGSAVS and MAGRSTS should not be executed from the system console due to it's priority. The system console receives messages constantly and if the console is busy with MAGSAVS or MAGRSTS, the messages must wait thus, potentially causing a hang or a halt.

(F)

If the tape susbsystem does not appear to be a problem, refer to CH104-A.

MACHINE CHECK HALTS - MCHK_

CH-A105

(A)

1. Use the PCB <user #> command to display the user's pcb format(refer to sys. arch. guide). Check for any abort flag bits enabled that may have generated a user process fault.
2. If any faults are found from the previous step, it may have an impact on the system halt. At this point use the trace <user #> command to check the user's stack for the subroutine being executed when the fault or abort may have occurred.

The pcb frame as well as the user's stack may reveal some useful data that may have been overlooked. The SYSTEM'S ARCHITECTURE GUIDE has the pcb format breakdown.

(B)

1. If no conclusion, refer to CH104-B.

(C)

1. Check the file system with the FS command. If any inconsistencies are encountered, the unit tables involved will be displayed.
2. If any problems are found from the previous step, use the UTE <user #> <ute #> command to further isolate the particular problem. The problem could be related to a disk drive problem or the disk media.

The file system check, as well as checking the file unit tables, could help determine any problems that may exist with the hash tables possibly due to a hardware problem in the disk subsystem.

(D)

1. If no conclusion, refer to CH104-C.

CONTINUE

E

1. Use the PA command to check the page map tables. Any errors will display the segment and page number in error as well as the user number.
2. Use the PM <segment #> <user #> command to further isolate the error. The problem could be related to a memory eccc error or memory could be configured incorrectly.

Checking the page map tables could help find a problem with memory that may have not been displayed by the machine check option such as memory eccc's or possibly a hole in the memory configuration. Note: for model 850 system only, segment 4 page 77 will be displayed as not available. Disregard this message. This portion of memory is wired aside for the SSU board.

F

1. If no conclusion, refer to CH104-D

G

1. Use the LOCKS command to display the N1LOCKS status. Check for any abnormalities.
2. Check the RSAV area in segment 14 locations 2000-3777.

END MCHK_ FLOW

MACHINE CHECK HALTS - MCHK_

(A)

1. To verify if any memory eccc errors have occurred, you must first use the **SYMBOL** command on ECCCNT (SY ECCCNT). This is the area in primos where memory eccc errors are recorded.
2. Use the **RESTORE** command to access the location from the previous step (in octal). The amount of eccc's the system encountered (if any) will be displayed. Replace the appropriate memory chip(s) or board(s).

If a memory parity error occurred during a DMX transfer could halt the machine with a MACHINE CHECK therefore it's important to check for any memory eccc's when troubleshooting a MACHINE CHECK dump.

(B)

1. If no conclusion, refer to CH200-B.

(C)

1. If any discrepancies are displayed from the **PAGECHECK** command, the user and segment number will be displayed. Note: for model P850 system only, segment 4 page 77 will be displayed as not available. Disregard this message. This portion of memory is wired aside for the SSU board.
2. Use the PM <user #> command to acquire more detail of the particular portion of memory that primos has flagged as a problem. The problem could be related to the eccc/eccu error(s) due to bad hardware in the memory. Note: be sure to check logrec for any memory errors.

The page map table check command will display a 64 word HMAP and a 64 word LMAP table, respectively.

END ECCC/ECCU FLOW

MISSING MEMORY MODULE HALT - MMOD_

MM-A300

(A)

1. To determine if any memory eccc or eccu errors exist, use the procedure outlined in flow CH-A200-A. If any exist, replace the appropriate memory hardware.
2. To determine if any memory discrepancies exist in the page map tables, use the procedure outlined in flow CH-A200-B.
3. If the DSWRMA register is valid, it will contain the last memory address referenced. This could become useful if a DMX operation was in progress.

The objective in this flow is to isolate any memory problems that may have surfaced from a memory address being referenced by stack pointers, indirect pointers, or a DMX transfer during an I/O operation.

(B)

1. If no conclusion, refer to MM300-B

(C)

1. If The DSW registers indicate an I/O or DMX operation in progress, use the procedure outlined in flows CH-A101 thru CH-A104 to isolate a bad I/O controller or bad CPU hardware that may have caused the problem.

(D)

1. Once the I/O and the MEMORY have been ruled out as the problem, isolate any logic of the CPU that may be suspected bad in the operation that has just been executed.
2. If I/O,MEMORY, and the CPU cannot be isolated to any suspected bad hardware, run the DOC utility (refer to DOC USER'S GUIDE).

If the I/O does not appear to be the problem, the CPU logic could be suspected bad. Checking the I/O path for the operation that has been executed could help isolate the bad CPU hardware.

END MMOD_ FLOW

SYSTEM HANGS

(A)

1. The **STAT US** command will display the user's logged in as well as the ready/wait status, locks owned, and their base registers.
2. Use the **RESTORE** command to look at the ready list located in segment 4 beginning at location 600(refer to PRIMOS internals guide). If there is an excessive amount of users in the ready list on level 626, this could indicate a problem with the scheduling of users or executing the ready user's processes, possibly due to a software or hardware problem. trace the ready users stack to determine what routines they may have been executing. If all the ready users were attempting to execute the same routines, this could be an indication of a software problem with that particular routine.
3. Use the **LOSEARCH** command on the user's "waiting at" locations acquired from step 1. If the majority of the users appear to be waiting on the same semaphore, this could also indicate a problem with the scheduling of users most likely, the majority of the processes calling on the same routines. This also can be software induced.

The objective of looking at the PXM, will help in determining what processes may have been executing at the time of the system hang, as well as processes waiting on semephores that may be significant to the system hanging.

(B)

1. If no conclusion, refer to HG400-B

CONTINUE

SYSTEM HANGS

(C)

1. Use the **PCB <USER #>** command to check for any user's that may have acquired any process fault or process aborts. This information may be useful in determining why there is a heavy load in the PXM. If any faults or aborts did occur, trace the user's stack to determine what routine may have caused the problem.
2. Use the **LOCKS** command to display what locks are owned. If a lock problem exists, accessing of critical data bases could possibly cause The heavy process exchange overhead therefore, a hang could occur. This normally indicates a software problem. If the PXM appears to be fine, refer to HG400-C.

The data acquired from steps 1 and 2 can become helpful in isolating any faulting routines or accessing of critical data bases due to a lock problem.

(D)

1. To determine if any i/o activity may have caused the system hang, refer to procedure outlined in in flows CH101-CH103.
2. Check any CSBs or FCOs that may have any information related to known system hangs due to microcode.

If a problem exists in the I/O, it is frequently d related to a comms controller. Occasionally, the system could appear it's hung but is actually running very slow due to excessive communications or possibly an applications problem. Be sure to check what I/O process may have been active when the system hung.

END SYSTEM HANG FLOW

LABELED HALTS - BOOT0

BT-A500

(A)

1. Use the TT -U1M -CRLF command to view the user 1 message buffer. An error message of some kind should be displayed on console prior to the system halting at BOOT0. This message should help determine what routine could have failed.
2. Use the TRACE <USER #> command to retrace the steps the live user executed prior to calling BOOT. The displayed subroutines should reveal any problems. The problem is usually software related.

Most routines that fail and call BOOT will print an error message at the system console prior to halting the system with BOOT0. The BOOT routine waits for the system console output buffer to be flushed prior to halting at BOOT0. The console message is a key factor in determining what routine may have failed.

(B)

1. If no conclusion, refer to BT500-B.

(C)

1. Some general problem areas to check for would be:
   - Has PRIMOS been changed recently ?
   - Is PRIMOS modified ?
   - Any new programs running ?
   - Was any new microcode installed recently ?
2. If any of the above changes exist, be sure to verify the integrity of the product under an operating system load.

BOOT0 halts tend to appear after some recent change has been made in the software or possibly after PRIMOS modifications.

(D)

1. If no conclusion, refer to BT500C.

LABELED HALTS

(E)

1. Use the FS command to check the file system for any problems that may exist with the file system.
2. Use the TRACE command to determine if live user's stack trace indicated any conditions signaling a "PAGING DEVICE FULL" or perhaps a "LOGIN DISCONNECT"..

END BOOT0 FLOW

Certain conditions can tend to cause BOOT) halts such as, PAGING DEVICE FULL, file units not closed on UTDALC, LOGINS or LOGOUTS, LOGABORTS, and LOGIN DISCONNECTS. Check for these type of conditions in the live user's stack.

LABELED HALTS - PAGES_

PA-A600

(A)

1. Use the RP -LIVE command , the register print will display the fault address. If the fault address contains 6003/20000, this indicates that the unwired Ring0 stack has been overflowed.
2. Trace the live user's stack and check for any recursive faults or conditions such as "SIGNAL$". These occurrences could corrupt the unwired Ring0 stack. This is can occur from hardware problems with the disk subsystem or the network.
3. A trashed unwired Ring0 stack can occur from frequent user procedure calls such as "R$CALL". This is normally due to bad CPU hardware or a peripheral device.

The unwired Ring0 stack is only allocated 8 pages, located in segment 6003. A process runs out of unwired Ring0 stack at address 20000. The page fault handler checks the fault address prior to halting the machine.

(B)

1. If no conclusion, refer to PA600-B

(C)

1. Use the procedure outlined in flow CH101 to check for any disk activity.
2. Use the procedure outline in flow CH102 for any network activity present. Any network activity with recursive login aborts, logins, or dcd drops could overflow the unwired Ring0 stack.

Certain conditions that can normally trash or overflow the unwired Ring0 stack is, paging device full,disk errors, excessive attach points, logins, log aborts, and output buffers full.

(D)

1. If no conclusion, refer to PA600-C

CONTINUE

LABELED HALTS - PAGES_

PA-A600
CONT.

(E)

1. Use the **PAGE** command to check memory for any inconsistencies. If any exist, use the proceedure outlined in flow CH200.

END PAGES_ FLOW

LABELED HALTS - IPAGF_

PF-A700

A

1. Trace the live user's stack to reveal any faults and fault types. An excessive amount of faults in the user's stack could indicate a hardware problem related to the particular process.
2. Use the Page command to verify if any problems may exist in the page map tables which could have possibly induced the problem.

Separate tables exist for interrupt process vs. user process. Some faults are legal for a user process but not for an interrupt process. The user stack will reveal any faults that may have occurred.

B

1. If no conclusion, refer to PF700-B

C

1. Use the procedure outlined in flows CH101-CH103 to isolate any I/O related problem.
2. Check for the last vectored interrupt outlined in flow CH-A101-B.

Systems running many comms controllers could be a factor in an IPAGF_ halt due to the fact that they all use the same interrupt stack. This could contribute to the interrupt stack becoming overflowed or a bad interrupt address caused by a bad controller.

1. If no conclusion, refer to PF700C.

D

1. Use the Symbol command to find the location of SEG4SZ(SEGMENT 4 SIZE).
2. Use the **RESTORE** command to look at location 0 and 1 of segment 4.
3. The value acquired from step 2 should not be greater that the value acquired from step 1. If the value is greater, then the interrupt stack has been overflowed. This problem is software related.

All interrupt processes use the same stack therefore, the interrupt stack could become overflowed due to excessive page faults. Page faults cannot be handled by the interrupt process, all memory for them must be wired.

END IPAGF_ FLOW

LABELED HALTS - PGMPA_

PG-A800

(A)

1. Use the PAGE command to verify the integrity of the page map tables. If any errors exist, use procedure outlined in flow CH-A200-B to further isolate.
2. Trace the user's stack to reveal any recursive page faults. If any have occurred, this may be related to an invalid PMT entry on the page just referenced.

The software stores the segment number in the stack before doing calculations. After restoring from memory into the L register, the L register becomes trashed. Replace the CPU board containing the register files (E1 for 9955 and 9950).

(B)

1. If no conclusion, refer to PG800-B.

(C)

1. Use the PCB <USER #> command to reveal the user's fault vector and concealed stack frame (refer to Systems Arch. Guide). These areas will contain specific data about the page fault(s).
2. Use the procedure outlined in flow CH-A200-B to display the HMAP/LMAP entry for the specified user and segment number acquired from PG-A800 A1.

All page faults cause a branch in execution through the user's page fault vector to the page fault table code. A CALF(Call Fault Handler) is then executed. The state of the system at the time of the fault is saved in the user's PCB concealed stack frame.

END PGMPA_ FLOW

User's Guide for DOC Automated Analysis

Barry I. Needalman

15 August 1985

Table of Contents

1 Overview

DOC is an automated crash dump analysis tool. It provides both an analysis which attempts to pinpoint the cause of failure and a variety of reports. The reports present various aspects of the failed system's state in a format which enables a knowledgable engineer to analyze the crashes which DOC cannot handle automatically. The current version of DOC has been oriented toward diagnosing hardware failures to the Field Replaceable Unit. This document describes how to run DOC and how to interpret its output.

DOC is intended to be used as a screening tool to determine the most likely problem when a customer calls Central Dispatch. Therefore, the program has the facilities to perform remote diagnosis. A separate document describes the configurations and procedures which must be in place at the customer site to enable remote DOC analysis.

2 Restrictions on DOC's Use

The current version of DOC will diagnose failures on P750, P9750, P9950, and P9955 systems running PRIMOS Revision 19.1 or later.

DOC operates primarily by understanding the implications of the values of many state variables in the PRIMOS operating system. Although the DOC can configure itself according to CPU type and PRIMOS Revision, it is unaware of any modifications from the master disk versions. Changes in modules which DOC does not use in its analysis are accomodated automatically. In particular, the program assumes that all of the DIMs (Device Interface Modules) contain the same code as the master disk version of PRIMOS. DOC ignores any non-standard devices. The file LOC_VAR_nnn.PL (where nnn is the revision number) gives the names of modules which must not have been changed from the mater disk version.

The current version of DOC ignores any ICS1 controllers in the configuration. DOC's analysis will still be valid with the exception that the effects of the ICS1 will be ignored.

3 Performing a DOC Automated Analysis

3.1 Machine Configuration

Since DOC performs "Remote Diagnosis", the machine on which the DOC application executes must have some communications facilities. Both a connection to a X.25 public packet switching network (eg. TELENET) and an assignable asynchronous line should be in the configuration. The asynchronous line should be attached to an auto-dial modem. Particular support is planned for the Racal-Vadic Auto-Dial VA212.

If both the crash dump file (and other required files) and the DOC application are on the same machine, DOC must still LOGIN to another user process on the machine. Primenet

Remote Login (X.25) should be used in "loopback" mode. Thus, the Primenet software must be present and correctly configured on the machine.

DOC must have a UFD for its use. The user process MUST be attached to the UFD while the application is executing. This is required because DOC's self-configuring ability loads in code modules at run time. The user process running the DOC application requires ALL access rights (except the right to change protection) to the UFD because a report of the diagnosis is written into the directory.

Salford Lisp/Prolog must be installed on the machine running DOC. This is a non-Prime product for which a per CPU license is required.

4 Learning to Use DOC

DOC has a menu oriented user interface and most operations are "self evident." This document will only summarize how to use DOC. It is essential to try the program while reading this document to gain full knowledge of how to use the DOC application.

DOC has two modes of operation, DSW decode and crash dump analysis. The crash dump analysis has a variety of useful functions which are described below.

5 DSW Register Decode Using DOC

DOC has the ability to decode the Diagnostic Status Words of the machines of which it has knowledge. In this mode, the information is manually input from the keyboard and no connection to a "remote" machine is needed.

6 Crash Dump Analysis Using DOC

6.1 Logging in to the "Remote" System

The first step in a DOC crash dump analysis is to login the system on which the crash dump and related information is located. The program supports both X.25 and assignable asynchronous line. X.25 is the generic name for Primenet Remote Login and public packet switching data networks such as TELENET. When you use X.25, you will be prompted for a system name. Answer a name which appears in the STAT NET output on the machine running DOC or the numeric network address of the remote machine.

After physical connection to the remote system has been accomplished, DOC will ask you to login. At this point, your terminal will behave as if you are running the NETLINK program. That is, your terminal will behave as if it was attached to the remote system. At this point you must type all of the commands and passwords necessary to LOGIN to the remote machine and attach to the special directory used on the remote machine for DOC analysis. Commands

which use control characters to manipulate the terminal (eg cursor movement) may not work correctly. DOC outputs important instructions which should be observed. Commands such as ATTACH, LISTF, and LD may be given.

After LOGIN has been accomplished and the process on the remote machine has been attached to the proper directory, the command doc_comm must be entered as if it resided in CMDNC0 on the remote machine. doc_comm is NOT a command, but a distinguished string which DOC uses to know that the LOGIN function has been accomplished. If you input the doc_comm string as "type-ahead" (ie before the appropriate Primos command prompt has appeared), then all following output will not be output on your terminal. The typed-ahead commands WILL be processed, but the output will not be displayed. Therefore, it is recommended that the doc_comm command NOT be typed-ahead.

After the doc_comm pseudo-command has been given, DOC will use the remote connection to send commands to the remote system. Your terminal will be used to give commands to DOC.

7 Interpreting Results of DOC's Automated Analysis

Normally, Automated Diagnosis is normally run first. Based on the results of the diagnosis, other reports may be run. The following information will help you to interpret the output of the diagnosis.

If the system halted at a halt instruction which is coded into Primos, DOC will tell you something about the cause of the halt. Since these halt instructions have been coded into Primos by the operating system developers, there is (should be) a specific problem with the system detected. Usually, what is detected is an inconsistency in the data structures used by Primos. For most "coded" halts, DOC is unable to determine what caused the inconsistency. DOC always prints out a message giving some kind of explanation of what class of errors caused the halt.

7.1 Machine Checks

Machine check errors are handled fairly completely. In many instances DOC is able to determine which board(s) are the most likely site of the problem. When the peripheral controllers are the cause, the Controller Status Report should be consulted to gain more information about which controller(s) should be suspected.

7.2 Missing Memory Module

Missing Memory Modules have fair coverage. DOC will print out the virtual address which caused the missing memory check to be taken. This is the address in DSWRMA. The backed-up value of DSWPB is the instruction being executed when mmod check occurred. If

the check occurred when DMx was NOT in progress, then the mmod is associated with some memory address referenced during the execution of the instruction (eg address of instruction, effective address of instruction, indirect pointers, stack pointer, etc). If the check occurred DURING DMx, then the check was caused by an address issued by a controller. Consult the Controller Status Report to isolate suspects. The contents of DSWRMA may contain some address associated with the transfer. If more than one controller was requesting DMx, at the time of the check, then DSWRMA may not contain relevant information. Missing memory checks are often associated with corrupted virtual memory databases within Primos. Another possibility is incorrect values in dma registers, dmc cells, dmt channel programs, and dmq queue headers.

DOC checks for overrun DMC cells. If the overrun DMC cell, is associated with an AMLC; then a likely cause is an unterminated line. An unterminated line has the effect of inputting null characters faster than PRIMOS can empty the tumble table.

7.3 Uncorrectable Memory Parity Errors

Ordinarily, the offending virtual address is in DSWRMA. However, in some cases DSWRMA may not be valid. Use the techniques discussed under Missing Memory Module to isolate likely addresses. If the check occurred during DMx, then consult the Controller Status Report to determine likely transfer addresses.

7.4 Ring 0 Stack Overflow

Although this condition is often caused by software bugs, there are at least two hardware problems which can be responsible. If there is an unterminated AMLC line connected to the system which is picking up noise, it is possible that process aborts (PABORT) can be issued faster than they can be handled. If there is an uncorrectable disk read error on the unwired portion of the Ring 0 Stack, then a stack overflow will result. An disk error message will appear in User 1's message buffer and a page fault on the stack will appear in stack trace of the live user.

7.5 Fortran STOP

This form of coded halt is used by some network related routines. Trace the stack of the live_user to find out which subroutine called F$HT (the Fortran STOP routine). Consult the source code for that routine. The value of the A-register of the live_user should correspond to the number following the STOP statement which caused the halt.

7.6 BOOT0_ AND BOOT_

Boot is another form of emergency halt. Usually, there is an explanatory message in User 1's terminal output buffers. DOC will print out these buffers for BOOT halts.

7.7 Controller Status Report

The Controller Status Report will print an entry for every controller which is present in the configuration and has been used at some time since cold start. For example, if the hardware configuration contains a magtape controller but no magtape was ever ASSIGNed; then DOC will conclude that the magtape controller is not present and will print no information about it.

The information about each controller includes the name of the controller, its device address (address used in PIO instructions), if an outstanding DMx request is pending, and the direction of the pending transfer. DOC cannot determine if a DMx cycle for a given device was actually in progress at the time of the halt; only that the controller has been issued a request which will result in DMx and that the operation has not yet completed.

Information is also printed about the DMx registers and cells used by each controller. The type of DMx (DMA, DMC, or DMQ), where the cell is located, the contents of the cell, the number of words left to transfer, the virtual address of the NEXT transfer, the Segment 0 address which that virtual address has been mapped to, the physical address, and the Primos symbolic name for the virtual address. All numbers preceded by a colon are OCTAL.

DOC cannot determine if a DMT cycle may have been in progress. The disk controllers use DMT to access channel programs. Thus, the disk controllers must be included in a list of controllers responsible for an error occurring during DMx.

7.8 Ready Wait Report

The Ready Wait Report prints a formatted list of the ready list and of semaphores with waiting processes. Each semaphore is associated with an event which a process may wait on. Each process is identified by its user number. Processes with negative user numbers are interrupt processes. The process numbers are in octal. All numbers preceded by a colon are OCTAL.

The Ready Wait Report can be used to determine if processes are waiting for unusual events or if an unusual number of processes are waiting for a particular event. The report is particularly useful when debugging a hang.

The live_user will be the highest priority ready process. The ready list is printed first by priority level and then by the order in which the processes are queued on that level. That is, the order of the process numbers corresponds to the order in which the processes will be

run.

Processes waiting for terminal input are reported as a group even though each process has a separate semaphore for this event. Processes waiting for disk requests and time slice are also grouped together. These categories are common events and waiting there is indicative of normal system behavior.

7.9 LOGBUF Contents Report

The LOGBUF contains a list of "recent" system errors or exceptions which would have been posted to the LOGBUF file if the system had not halted. Not all of the entries are processed. System cold start and disk mount entries are ignored. The information is similar to the information which is output by the LOGPRT program.

7.10 User 1 Output Buffers

This report prints the content of both the user 1 terminal output buffer and the "message buffer." PRIMOS often writes specific error messages which describe the cause before executing the halt. These messages are not always actually printed before the system stops.

7.11 Autopsy Command

This option feeds a single command to the AUTOPSY program which is running on the remote machine and prints the results. At the current time, commands which result in a great deal of output (eg. STATUS ALL) will not work correctly. AUTOPSY commands which enter a different command mode (eg. TRACE, RESTORE, VSPD) will leave AUTOPSY in that mode. Thus, other DOC functions will NOT work until another AUTOPSY command has been issued to exit the mode.

7.12 Virtual - Physical Address Translation

Given either a virtual or physical address, this function will supply the complement. Since hardware deals in physical addresses and PRIMOS in virtual addresses, this command can be used to relate addresses suspected in a particular problem (eg. missing memory module).

7.13 Diagnostic Status Decode

This function decodes the DSW registers as if a machine check had been responsible for the halt. Note that the contents of the registers is only guaranteed to be valid after an actual check. Thus, the output of this command may be invalid if the DSW registers do not contain values from an actual recent check.

Site Administrator's Guide for DOC Automated Analysis

Barry I. Needalman, George Deak

Table of Contents

Version 20.1

Site Administrator's Guide for DOC Automated Analysis

1 Summary

This document describes the machine configuration and procedures which are needed at a customer site to enable remote failure diagnosis by the DOC intelligent crash dump analyzer. DOC can help diagnose halts or hangs which have occurred on systems running PRIMOS Revision 19.1 or later on the following CPUs: 750, 9955, 9750, 9955_mod2, 9755, 2450, 2550, 2655, 9650, and 9655.

When a hang or halt occurs on a Prime system, the customer's computer room operator must take a "crash dump." This operation transfers the contents of the system's user visible registers and all of physical memory onto a magnetic tape. Once the tape dump operation has been completed the failed system may be warm or cold started to resume service to users. The operator then transfers the information on the magtape into a disk file in a previously established UFD. Prime Customer Service is then requested to perform a remote analysis.

The customer site system administrator must establish a LOGIN-able ufd in which is kept the programs used by DOC, crash files to be analyzed, and the PRIMOS ring maps for the operating system on the failed machine. The customer site must have some facilities to allow LOGIN from a remote terminal, either an X.25 public data network connection (eg. TELENET) or an asynchronous line connected to a modem and telephone line.

2 Establishing a LOGIN Account for DOC

The DOC program operates by logging into to the customer's machine as an ordinary user and running a Prime supplied user level application program. This program accesses the crash dump file from the failed machine and the associated ring maps and sends the requested information over the remote link.

2.1 Choosing a Machine

If your site has more than one Prime system you should choose one or more of the machines to run the DOC remote diagnosis application. The CPU type of the machine is unrestricted - any 50-series system will run the customer site program. Even the system which experienced the failure may be used, provided it will run Primos for approximately 30 minutes. The system must be running PRIMOS Revision 19.0.0 or later.

The machine chosen must have some form of remote login capability. If the site has a machine which is directly connected to an X.25 public network, such as TELENET, then that machine would be a good choice. An alternative is a modem connected to an asynchronous line used for login. The line should NOT be connected to the system console remote port. A

1200 baud Bell 212 compatible modem is preferable, but the system will operate at 300 baud using a Bell 103 compatible modem. A normal telephone line whose number can be directly dialed (ie does NOT go through an operator) should be connected to the modem. The line should have no other extensions and should have no "options" which can cause noise on the line. (eg. The line should NOT have "call waiting" since this feature causes a tone to be sounded when a second incoming call is made to an already busy line.) Communication via X.25 public data network is preferred because it offers lower error rates and high speed.

3 Contents of UFD

The UFD used by DOC for remote analysis must contain the following files:

```
DOC_COMM.RUN

Primos RING maps with names changed appropriately (See below).

Crash Dump Files
```

DOC_COMM.RUN is the application program which is run during DOC analysis. It is identical to AUTOPSY with the name changed.

The crash dump should be transferred from tape to disk using DOC_COMM.RUN.

```
OK, ASSIGN MT0
OK, AT dumps_ufd
OK, RUN DOC_COMM
> read <name you wish to give to dump on the tape> 0
```

The RING maps are the "SEG load maps" of the version of PRIMOS the failed machine was running at the time of the failure. There are two files. At a site running unmodified PRIMOS, the RING files are the files RING0.MAP and RING3.MAP from the master disk UFD PRIRUN. The names of the files must be renamed to append the PRIMOS Revision number to the end of the name. For example, if PRIMOS Revision 19.1.1 is being run, then the master disk file PRIRUN>RING0.MAP should be copied into the UFD as RING0.MAP.19.1.1 If the site is running more than one operating system revision (on different machines or at different times), then the UFD used by DOC should contain a pair of appropriately named RING maps for each version. If the site has modified and reloaded PRIMOS, then it is important that the maps from the modified system be used. The RING maps MUST match the operating system version for DOC to give accurate diagnosis. The revision number that a system is currently running can be determined by giving the PRIMOS command 'STAT SYSTEM'. The command prints the revision number on the user terminal. Alternatively, if the failed system is unavailable, run DOC_COMM.RUN on the dump being analyzed. That is: type r doc_comm <dump_name> in PRIMOS. DOC_COMM will tell you the exact name of the ring0 map it is looking for (if it does not find it in the same ufd as the DOC_COMM program program itself). Rename the failed machine's ring maps to match the names sought by

DOC_COMM.RUN.

The LOGIN account used by DOC needs only READ access to the files in its UFD. DOC accesses ONLY the files in its UFD. (Note: The Customer Service Engineer running DOC may use LISTF or LD to verify the contents of the UFD during the process of starting the analysis but this capability is not essential.) The site administrator may restrict access of DOC's LOGIN account appropriately. The LOGIN account used by the computer room operator to transfer crash tapes into the UFD will, of course, need all access rights except the right to change protection.

The size of each crash dump file is determined by the amount of physical memory on the failed system. The dump file has one disk record for each 2KB of physical memory. Dump files may be deleted after analysis but the original crash tape should be retained until it is certain that the problem has been resolved.

4 Computer Room Procedures

The site should establish a procedure to ensure that a crash dump tape will be taken EVERY time there is a system hang or a halt. The procedure for taking a crash dump tape is given in the Prime System Operator's Guide (DOC5038-19x) in Chapter 13 as action D in Table 13-2 Recovery Procedures. For machines not equipped with the VCP (ie has knobs and switches type control panel), see appendix E, Table E-2. A supply of blank or "scratch" tapes should be located physically near the machines. The tape dump operation should take well under five minutes even on a large memory configuration machine, including the time to mount the tape.

The procedure for taking a crash dump tape on a system equipped with a Virtual Control Panel (VCP) is as follows:

1a. If the machine is hung (ie the red stop light is off and no "halted at" message has been printed on the system console, then the system must be manually stopped. Do NOT use the SYSCLR button on the system cabinet. Instead, use the VCP command STOP. If the console is not in "VCP mode", then type <ESC> <ESC>. (ie. Press the ESC key twice.) The console prompts 'CP> ' when in VCP mode.

1b. If the machine fails to respond to the STOP command, then press the SYSCLR button located on the front face of the system cabinet near the top. This method of stopping the system will not save the live registers but DOC will still be able to perform an analysis. DOC assumes in that step 1a above has been tried and has failed to halt the machine as fact that it uses in its reasoning process.

2. Mount a tape, with the write ring in place, on Drive 0.

3. Enter the SYSCLR command. CP> SYSCLR

4. Enter RUN 775. CP> RUN 775
The crash dump will be written on the tape and then the tape will rewind. If the tape does NOT rewind, start at step 2 with a different tape.

5. Dismount the tape and remove the write ring.
Label the tape with the date and time and the machine name (if more than one machine at site).

6. Warm or Cold Start the machine according to instructions in the System Operator's Guide and established site procedure.

7. Record the dump action with identifying tape number in the system logbook together with any "special circumstances" relevant to the problem.

5 Requesting a DOC Analysis

When the tape contents have been transferred to a file in the UFD set up for DOC's use, a call should be made to request analysis. You will need to supply the following information.

1. The complete X.25 public data network address of the machine or the telephone number of the dial-up LOGIN asynchronous line.

2. Instructions on how to LOGIN to the system including the user id and password of the account set-up for DOC diagnosis. Make certain to give ALL information needed to LOGIN to the machine including additional passwords and billing accounts numbers if they are used at the site.

3. The name of the UFD and crash file. It should always be named CRASH.ticket_nr. If you are using the obsolete password protection, the NON-OWNER password to the UFD (if not blank) must be supplied.

4. The name and "serial number" of the failed system so that the Customer Service Engineer can identify the machine.

5. Any information relevant to the problem which has been noted the system logbook or is otherwise known.