digital

XXDP
DEC X/11

PROGRAMMING CARD

METHOD	RKDP	TCDP	TMDP	THDP
BM792	LA 773100 SR 77406 START	LA 773100 SR 777344 START		
MR11-DB	LA 773110 START	LA 773120 START	LA 773136 START	
BM873-YA	LA 7773010 START	LA 773030 START	LA 773050 START	
BM873-YB	LA 773030 START	LA 773070 START	LA 773110 START	LA 773150 START
TOGGLE	LA 010000 DEP 012737 000005 177404 000001 LA 010000 START wait 1 sec LA 000000 START	LA 777342 DEP 004003 DT rewinds examine DEP 000001 LA 000216 DEP 012737 000005 177342 000777 LA 000216 START	LA 010000 DEP 005137 172524 012737 060011 172522 000777 012737 060003 172522 000777 LA 010000 START wait 1 sec HALT LA 010014 START wait 1 sec HALT LA 000000 START	LA 010000 DEP 012737 001300 172472 012737 177777 172446 012737 000031 172440 105737 172452 100375 012737 177400 172442 005037 172444 042737 000007 172452 012737 000071 172440 105737 172440 000100 000375 000137 000000 LA 010000 START

TADP	RXDP	RPDP	RBDP	RSDP
LA 773300 START				
		LA 773154 START		
LA 773230 START		LA 773100 START		
LA 773524 START		LA 773350 START	LA 773320 START	LA 773000 START
LA 001000 DEP 012700 177500 005010 010701 062701 000052 012702 000375 112103 112110 100413 130310 001776 105202 100772 116012 000002 120337 000000 001767 000000 000755 005710 100774 005007 017640 002415 112024 LA 001000 START	LA 001000 DEP 005000 012701 177170 105711 001776 012711 000003 005711 001776 100405 105711 100004 116120 000002 000770 000000 005000 000110 000000 000000 000000 LA 001000 START	LA 001000 DEP 012705 176716 012715 177400 012745 000005 105715 100376 005007 LA 001000 START	LA 10000 DEP 012700 176700 012710 000023 005060 000034 005060 000006 012760 177400 000002 012710 000071 105710 100316 005007 LA 10000 START	LA 001000 DEP 012705 172044 012745 177400 012745 000071 032715 100200 001775 100762 005007 LA 001000 START

XXDP RESIDENT MONITOR COMMANDS

F ⟨CR⟩	Set console fill count
D ⟨CR⟩	Directory on the TTY console
D/F ⟨CF⟩	Short directory on the TTY console
D/L ⟨CR⟩	Directory on the line printer
D/L/F ⟨CR⟩	Short directory on the line printer
R COPY ⟨CR⟩	Starts the copy program
R FILENAME ⟨CR⟩	Starts the indicated program
L FILENAME ⟨CR⟩	Loads the indicated program
S FILENAME ⟨CR⟩	Starts the desired program that was loaded under "L" command
S ADRS ⟨CR⟩	Starts program at specified address
C FILENAME ⟨CR⟩	Runs desired chain table
C FILENAME/QV ⟨CR⟩	Runs desired chain in quick verify

XXDP RESIDENT MONITOR ERRORS

INVCMD/SW	Invalid command and/or switch, check command just given
DEVERR	Device error on input device
EOM	End of medium, occurs during input operations when the program attempts to input and the file is at an end. Serious problem, file in storage is probably wiped out.
INVADR	Invalid address, must be even
CKSMER	Checksum error during "LOAD" command.
POFLO	Program too large to load within available core space.
INVNAM	Invalid character used for file name
NEXFIL	Non-existent file, file does not exist on medium.

UPD1 PROGRAM COMMANDS

FILL ⟨CR⟩	Sets up terminal for correct print after CRLF
CLR ⟨CR⟩	Clears core below update program
XFR ⟨CR⟩	Permits making program self-starting or non-self-starting
DUMP DEV:FILNAM.EXT	Writes memory contents in ABS format
LOAD DEV:FILNAM.EXT	Loads ABS format program(.BIN, .BIC)
PIP DEV1:FILNAM.EXT= DEV2:FILNAM.EXT	Copy file from device 2 to device 1
SAVE DEV:FILNAM.EXT	Writes memory contents onto contiguous blocks
MOD ADR	Modifies core contents
CORE	Types protection limits
LOCORE ADR	Enters low protection limit
HICORE ADR	Enters high protection limit
DIR DEV:	Types dev directory on TTY
ZERO DEV:	Zero device directory
BOOT DEV:	Loads block 0 of dev starting at LOC 000000
SAVM DEV:	Writes 4K onto dev starting at block 30
START	Starts program at its transfer address
START ADR	Starts program at ADR
^C(Control)	Return to command mode(open output file is CLOSED)
DEL DEV:FILNAM.EXT	Deletes file from device directory
REN DEV:NEWNAM.EXT = DEV:OLDNAM.EXT	Renames old file

UPD2 PROGRAM COMMANDS - Same as UPD1 plus

DIRLP DEV:	Types dev directory on line printer
ACT	UPD2 "ACT MODE"
NOTACT	UPD2 Out of "ACT MODE"
FILE DEV: DEV:	Copies file(s) from one device to another,deleting file of same name before doing the transfer
FILEF DEV:FILNAM.EXT	Same as file except that with cassette or magtape fast transfers are performed
FILET DEV:FILNAM.EXT	Reads file and checks (no dir for device checking) (Errors (file "TEST")
FILEL DEV:FILNAM.EXT	Loads files (Assumes ABS format) Checking for device and checksum errors
FILEG DEV:FILNAM.EXT	Loads files (Assumes contiguous format) Checking for device and file size errors

FILED DEV:FILNAM.EXT	Deletes named files
TEXT DEV:FILNAM.EXT	Creates Test file for printing or for command execution
FILCMP DEV: DEV:=DEV: FILNAM.EXT	Compare two files on the XXDP Mediums,(UPD2R Only)
PATCH DEV:FILNAM.EXT	Enables the user to patch an ABS format program on any XXDP random access device
PRINT DEV:FILNAM.EXT	Outputs a file to the line printer
TYPE DEV:FILNAM.EXT	Outputs a file to the console terminal
DO DEV:FILNAM.EXT	Executes a command file
ASG PHYSICAL = LOGICAL	Assigns a physical device to a logical device name
EOT	Writes end of tape mark (File) on magtape or cassette after tape has been positioned
^Z(CONTROL Z)	End input to a text file
*	Used for file naming to mean "ANY" (Any file name or any file extention)
?	Used for file naming to indicate a wild character (Any character will match it)
#OR ;	Used in a file of executable commands to start a comment line which is to be typed during execution
$	Same as # but causes a halt after the comment is printed
/LP	Line printer output
/N	Aborts type outs

ERRORS

INVCMD	Invalid command
INVDEV	Invalid device
INVADR	Invalid address
INVNAM	Invalid file name
NEXFIL	Non-existent file
DELOLD	Delete old file before giving command
DEVERR	Device error on either input or output device
NOTRDY	Paper tape device is not ready
CKSMER	Checksum error
EOM	End of medium
DEVFUL	Device full
INVCOR	High core limit lower than lower core limit
DIRERR	Invalid name in device directory
DELERR	Bit map error during delete operation on dectape or disk
POFLOW	Program too large to load within existing core space
INVSW	Invalid switch specified in command string
DUMP ERROR	Act mode only data dumped on output device does not match

PERIPHERALS

PR:	PC11 High speed paper tape reader (UPD1, UPD2)
PP:	PC11 High speed paper tape punch (UPD1, UPD2)
KB:	TTY Keyboard, or low speed reader (UPD1, UPD2)
PT:	TTY Printer and punch(UPD1, UPD2)
DTN:	TC11 Dectape (UPD1.N=0 or 1), (UPD2, N=03)
DKN:	PD11, N=0 or 1) (UPD2, N=0-3)
MTN:	TM11/TU10 Magtape 7/9 Track (UPD2, N=0-3)
CTN:	TA11 Cassette (UPD1, N=0 or 1), (UPD2, N=0 or 1)
DXN:	RX11/RX01 Floppy disk (UPD1, N=0 or 1) (UPD2 N=0 or 1)
MMN:	TM02/TU16 Magtape (UPD2 Only, N=0-3)
DPN:	RC11/RP02/RP03 (UPD2 only, N=0 or 1)
DBN:	RP04 Disk (UPD2 Only, N=0 or 1)
DSN:	RS03 RS04 Disk (UPD2 Only N=0 or 1)

PROGRAM NAMING CONVENTIONS (Ex. DCFPKA#)

D =	Diagnostic program, and is not used in naming a problem
C =	A = 11/05, 15, 20 Processors B = 11/40 Processor C = 11/45 Processor E = 11/70 System Z = All Processors X = DECX/11 Exerciser Software
FP =	Option Designation
K =	A thru Z = Program Designation 0 thru 9 = Overlay Designation
A =	A thru Z = Revision Designation
# =	# = MCN Level 0 = No MCN Issued

EXTENTIONS

SAV	Non-chainable core image file
SAC	Chainable core image file
BIN	Non-chainable, ABS Format
BIC	Chainable, ABS Format
OBJ	Object module. Linking required
LIB	Libary file, used with other programs
TXT	ASCII File, Used for text files

HOW TO USE THE COPY PROGRAM

The copy and verify commands must be used with like mediums

EXAMPLES:

Copy DK1: = DK0: ⟨CR⟩	Copies DECPACK Drive 0 (Master) Unto Drive 1 (Scratch) and verifies
Copy DT2: = DT1: ⟨CR⟩	Copies DECPACK Drive 1 (Master) Unto Drive 2 (Scratch) and verifies
Verify DK1: = DK0: ⟨CR⟩	Verifies that DECPACK Drive 0 is Identical to Drive 1

COMMANDS:

FILL	UPD1/UPD2 Equivalent
BOOT	UPD1/UPD2 Equivalent
DIR	UPD1/UPD2 Equivalent
DIRLP	UPD2 Equivalent
COPY	Copies and verifies like mediums
VERIFY	Verifies like mediums

FORMATTING

TC11/TU56

Load YPTCB0⟨CR⟩
Tape on drive zero
(Remote and write enabled)
Write ALL and write T&M enabled
Start 600 ⟨CR⟩
1st pass tape halts
Disable T&M
Press continue
3rd pass tape halts
Formatting complete
Disable write ALL

RK11/RK05	Load ZRKIB0 ⟨CR⟩
	Disk on drive zero
	Write enabled
	Start 200 ⟨CR⟩
	Program types heading
	Type 3 ⟨CR⟩
	Set SR = 000000
	Press continue
	Program types drive 000000
	Program types pack good
	Formatting complete
RP11/RP03	Load ZRPDB0 ⟨CR⟩
	Disk pack on drive zero
	Start 200 ⟨CR⟩
	Program types "Unit:"
	Type 0 ⟨CR⟩
Program Types	Set the format enable switch
	Set the RP11 write enable switch
	Set the selected unit write enable switch
	Strike any teletype key when ready
Program Types	Reset the format enable switch
	Strike any teletype key when ready
	TEST COMPLETE
RH11/RP04	Load ZRPLB0 ⟨CR⟩
	Disk pack on drive zero
	Disable write protect
	Start 200 ⟨CR⟩
Program Types:	You Type:
Drive	0 ⟨CR⟩
22 Sector	D
Program mode	D
Starting CYL, TRK	D
Ending CYL, TRK	D
Select Pattern	D
Program Types:	FORMAT COMPLETE

CREATING A XXDP RP02/RP03 DISK PACK

ZERO DP1:
LOAD DP0:RPDP.BIN
SAVM DP1:RPDP.SAV
LOAD DP0:UPD1.BIN
DUMP DP1:UPD1.BIN
LOAD DP0:UPD2.BIN
DUMP DP1:UPD2.BIN

CREATING A XXDP RP04 DISK PACK

```
ZERO DB1:
LOAD DB0: RBDP.BIN
SAVM DB1: RBDP.SAV
LOAD DB0: UPD1.BIN
DUMP DB1: UPD1.BIN
LOAD DB0: UPD2.BIN
DUMP DB1: UPD2.BIN
```

CREATING A XXDP RS03/RS04

```
ZERO DS1:
LOAD DS0: RSDP.BIN
SAVM DS1: RSDP.SAV
LOAD DS0: UPD1.BIN
DUMP DS1: UPD1.BIN
LOAD DS0: UPD2.BIN
DUMP DS1: UPD2.BIN
```

CREATING A NEW XXDP DECTAPE

```
ZERO DT1:
LOAD DT0: TCDP.BIN
SAVM DT1:
DUMP DT1: TCDP.BIN
LOAD DT0: UPD1.BIN
DUMP DT1: UPD1.BIN
LOAD DT0: UPD2.BIN
DUMP DT1: UPD2.BIN
```

CREATING A NEW XXDP DECPACK

```
ZERO DK1:
LOAD DK0: RKDP.BIN
SAVM DK1:
DUMP DK1: RKDP.BIN
LOAD DK0: UPD1.BIN
DUMP DK1: UPD1.BIN
LOAD DK0: UPD2.BIN
DUMP DK1: UPD2.BIN
```

CREATING A NEW XXDP MAGTAPE

```
ZERO MT1:
LOAD MT0: THDP.BIN
SAVE MT1: THDP. SAV
LOAD MT0: TMDP.BIN
SAVE MT1: TMDP.SAV
LOAD MT0: THDP.BIN
DUMP MT1: THDP.BIN
LOAD MT0: TMDP.BIN
DUMP MT1: TMDP.BIN
LOAD MT0: UPD1.BIN
DUMP MT1: UPD1.BIN
```

```
LOAD MT0: UPD2.BIN
DUMP MT1: UPD2.BIN

****NOTE****WHEN USING TM02/TU16 USE THE SAME
        PROCEDURE BUT SUBSTITUTE MM FOR MT
        NOMENCLATURE.
```

CREATING NEW XXDP CASSETTE

```
ZERO CT1:
LOAD CT0: TALDRB.BIN
SAVE CT1: TALDRB.SYS
LOAD CT0: GTP.BIN
DUMP CT1: GTP.BIN
```

CREATING A NEW XXDP FLOPPY DISKETTE

```
ZERP DX1:
LOAD DX0: RXDP.BIN
SAVM DX1: RXDP.SAV
LOAD DX0: UPD1.BIN
DUMP DX1:UPD1.BIN
LOAD DX0: UPD2.BIN
DUMP DX1: UPD2.BIN
```

DEC X/11

GETTING STARTED

NEEDED TCDP containing Dec X/11 Modules
TADP containing Dec X/11 Modules
RKDP containing Dec X/11 Modules
TMDP containing Dec X/11 Modules
PAPER TAPE Dec X/11 Modules

NOTE: On all XXDP packages Modules will be files.

DEC X/11 CONSISTS OF

Configuration PROG.BIN	DXQA*
Monitor's Module.LIB	DXQL*
General Products #1.LIB	DXQL*
General Products #2.LIB	DXQL*
Communications options #1.LIB	DXQL*
Communications options #2.LIB	DXQL*
Lab or Industrial options #1.LIB	DXQL*
Lab or Industrial options #2.LIB	DXQL*
New Options.LIB	DXQL*

* Letter will be version

NOTE: Advised that Dec X/11 be used in connection with XXDP for ease of use. Much preferred to paper tapes. Dec X/11 should reside on media which contains XXDP Monitor for ease of Booting.

LOADING UNDER XXDP

Load XXDP Monitor as shown under loading XXDP Monitor

Monitor identifies itself

DXQA* (self-starts)
(Program responds with date)

NOTE: Must use DXQA* first in all cases to build run-time exerciser, and then first library called must be Monitor Library - DXQL*.

FACILITIES UNDER CONFIGURATOR

PROG ASKS FOR:	Core Size	User Response
Sys, Size	4K	20,000
	8K	40,000
	12K	60,000
	16K	100,000
	20K	120,000
	24K	140,000
	28K	160,000
	or greater	

NOTE: All entries and commands terminated with ⟨CR⟩ unless otherwise specified.

Date — DD-MM-YY

Define Output — Device run-time exerciser will be when completed (i.e., Papertape, DT, DK, etc.)

Define Input — Device and name of library to be read in from

Module Wanted (three responses possible)

NO — Do not include particular Module in run-time system.

Y — Want Module included in run-time exerciser using default parameters.

YES — Want Module included in run-time exerciser but wish to change default parameters (i.e., DVA - device address; VCT - device vector; BRI & BRZ - interrupt levels of device; DVC - specifies number of drives; SRI - modify execution of module due to particular feature in device.

SWITCHES /MLP — core map printed on line printer.
/MP — core map printed on console.
⟨CR⟩ — no core map desired.
/FM — BREAK OUTDEV:<
IN DEV:NAME.EXT/FM

NOTE: /FM is used with MT or TA for a fast mode directive.

COMMANDS

CNF	Enter configure mode
MDL mod nam	Enter module name in configuration table
MDL	Type current module entry contents
DVA (addr)	Enter device address
VCT (addr)	Enter device vector address
BR1 (n)	Enter BR1 level
BR2 (n)	Enter BR2 level
DVC $(n)_{10}$	Enter device count
SR1 (n)	Enter SR1 value
CL	Clear configuration table
EX	Exit configure mode

POINT MODNAM
Find module name and type entry contents

NXT — Point to next entry and type contents

KI — Kill current entry

FILL — To change fill parameters for console device

GETC DEV:FILN.EXT
Get configuration table from device specified

SAVC DEV:FILN.EXT
Store configuration on device specified (Extension must be .CNF)
Boot and start Monitor from device specified must be device 0.

LINK DEV:FILN.BIN<DEV:
Link and output exerciser module onto directory device specified from device specified

LINK DEV:<DEV:
Link and output exerciser module onto non-directory device specified from device specified

MAKE — Initiates exerciser generation via make command

NOTE: If user wishes to end make mode and not process any more Libraries, he may type CTRL Z (↑Z) in response to "do you want module xxxx?" following NO, Y, or YES answer. (core map output) If no more modules are needed from current Library, user may end Y, YES or NO with CTRL 'Y' (↑Y) and prog. will ask for next input.

CHECK DEV:NAME.EXT
Checks file for checksums and correct object format

MERGE DEV:NAME.EXT<DEV:
Builds library from input according to data in configuration table (extension must be .LIB)

BREAK DEV:<DEV:NAME.EXT
Splits library into its component modules
*see /FM

PIPM DEV:NAME.EXT<DEV:NAME.EXT
Transfers object module or library from one device to another

ZERO DEV: Zeros directory

TYPEC — List configuration table on console device

PRINTC — List configuration on line printer

ERRORS UNDER CONFIGURATOR

INVCMD Invalid command

INVDEV Invalid device

INVADR/DATA
Invalid address or data address must be even dev count max 16_{10}
vector specifications 774_8 max

NEXFIL File specified does not exist on medium

DELOLD Another file exists with the same name. Delete old or give new name to file

INVNAM Invalid name given, give correct name, invalid characters in name

CKSMER Checksum error when reading a formatted finary block, fatal error

EOM End of medium

DEVERR Device error

DEVFUL Output device full

INPUT BUFFER OFLO
Block size of input file too large for program's input buffer

?INVCMD NOT IN CNF MODE
Command given must be given while in configure mode

?CNF TABLE FULL
Configuration table full only 60 entries allowed.

?COR EXCD
Core exceeded, the run-time exerciser exceeds the core size of system

?NOT FOUND
Occurs during 'point' command while making or editing a configuration table, pointed to non-existent name

ERR01 Symbol table error, prog error

ERR02 Global search failure in RLD

ERR03 PC Mod Command not in RLD

ERR04 OBJ Module does not start with GSD, Bad Module

ERR05 First entry in GSD not OBJ Module name, Bad Module

ERR06 Cannot find section name specified in RLD

ERR09 Jump table index out of range

ERR12 Load Module output error

FACILITIES UNDER MONITOR

COMMANDS

MAP — Types Map of Modules in exerciser, indicates Module starting address and Module status

SEL — Selects all modules for execution

DES — Deselects all modules

SEL (Module Name)
Selects specified module

DES (Module Name)
Deselects specified module.

MOD (address)
Used to examine and/or modify the contents of storage
MOD Address — Close with ⟨CR⟩ or change and close with ⟨CR⟩ or open next using ⟨LF⟩ after change or make no change and open next with ⟨LF⟩

MOD (Module Name Addr)
Same operations as above apply.
MOD DCAA0 20 — opens 10th octal word of Module DCAA0 (relative loc 20)

RUN (Address optional)
Starts run-time exerciser. Only those modules selected start. The optional address will cause the program to relocate to the nearest 1K boundary and start. Relocat is normal from this point on.

RUNL (Address Optional)
Starts exerciser and stays locked. Does not relocate. Address specified must be 1K boundaries and program will lock.

KTOFF — Disable memory management

KTON — Reenables memory management

CTRLC (↑C) — Ends execution of exerciser. Types run summary, returns to command mode.

SWITCH REGISTER OPTIONS

SR15=1	Drops Module after error. Module dropped - printout accompanies.
SR14=1	Inhibit dropping Module after 20 errors
SR13-1	Inhibit error and Module message printout
SR12-1	Enable end of pass printouts
SR10-1	Report all data errors. Set to 0 only first 3 errors of block transfer reported.

NOTE: SR=060000 — Inhibits all printouts except Monitor messages and module halts. Good for scoping operations.

KEYBOARD COMMANDS AND ERRORS

A through Z
0 through 9
SPACE
⟨CR⟩
⟨LF⟩
RUBOUT
(↑C)
All other characters invalid

⟨CR⟩	Normal Terminator
RUBOUT	For corrections on current line
↑C	Start command string over

KBUF OFLO
Too many characters typed

INVALID COMMAND
Command not recognized by Monitor

INVALID/NEX NAME
Specified non-existent Module or invalid characters

INVALID ADDR/DATA
Odd address or non-octal numbers specified

NOTE: Module names always 5 characters.

STATUS WORD OF MODULES

Bit 15=1	Module is an IOMOD
Bit 15=0	Module is BKMOD or NBKMOD
Bit 14=1	Module selected
Bit 14=0	Module deselected
Bit 13=1	Module dropped during run
Bit 13=0	Module not dropped during run
Bit 12=1	Module IOMODX
Bit 12=0	Module not IOMODX

Bit 11=1 Module IOMODR
Bit 11=0 Module not IOMODR
Bit 10=1 Module IOMOD?
Bit 10=0 Module not IOMODP.
NOTE: Low byte — processor status assumed when running module
0 = IOMOD and NBMODS
20 = BKMODS — bit 4 is T bit.

PRINTED MESSAGES
SYSTEM ERROR

YYYYYY PC pushed on stack at time of failure
ZZZZZZ Processor status at time of failure
SSSSSS Contents of stack pointer (RG) immediately after the trap (virtual ADDR)
0000XX 4 if bus error
10 if reserved instruction trap
NOTE: In command mode, remains in command mode.
Run mode — restarts, pass and error counts not cleared.
Chain mode — exits to XXDP Monitor.

ERROR — To report other than data errors

NNNNN Failing Module Name
PCXXXXXX
18 bit physical address of ERROR $ CALL
APC YYYYYY
Assembled PC of ERROR $ CALL
PASS#NNNNN
Pass number at which error occurred (decimal)
ERR#NNNNN
Error count in current run (decimal)
ACSR AAAAAA
CSR ADDR of failing device, 0 if not applicable
CSRC CCCCCC
Contents of failing device CSR, 0 if not applicable
STATC SSSSSS
Contents of failing device status reg, if applicable.

EXTENDED ERROR

NOTE: Same as above plus up to eight additional values. Must refer to erring module listing to obtain meaning.

DATA ERROR — To report errors data errors except IOMODX

NNNNN Failing Module Name

PCXXXXXX
18 bit physical address of DATER $ CALL

APC YYYYYY
Assembled PC of DATER $ CALL

PASS#NNNNN
Pass number at which error occurred (decimal)

ERR#NNNNN
Error count for current run (decimal)

CSRA AAAAAA
CSR addr of failing device

S/B BBBBBB
Expected data

WAS WWWWWW
Obtained data

WRADR DDDDDD
Address of expected data

RDADR EEEEEE
Address of bad data

MONITOR DATA ERROR

Reports data errors for IOMODX using CDATA $ CALL to have Monitor check. IOMODX write buffers not mapped. Error discovers error during check, reports via DERRX $ CALL.

NOTE: Printout of error same as data error above plus one word argument.

WORD#ZZZZZ
Position of word being checked in buffer

NOTE: All errors in a given transfer counted as one error as far as error count concerned. Not incremented until all errors reported. Total number of words in transfer reported and total number of errors encountered also reported. (Two other possible printouts)

MEMORY MANAGEMENT ABORT HALT

Status register bit 0 cleared (KT11 enable bit)
Status register bit 2 will contain the virtual address of instruction that failed.
Halts

ENDPAS

Used by module to indicate pass completed. Inhibit for better throughput. SR12=1 printout enabled.

NNNNN Module name

PCXXXXXX

See other printouts: same

APC YYYYYY

See other printouts: same

ENDPAS NNNNN

Pass number completed (decimal)

NOTE: Operation of module continues unless:

1. Module BKMOD — In which case Monitor starts next BKMOD.
2. Chain mode — Allowed one pass only.
3. Program relocation allowed, after WBUF rotation. After relocation they are restarted.

DROPPED

Following printout module dropped and not allowed to execute for remainder of current run.

NOTE: Occurs

1. After error if SR15=1.
2. After 20th error if SR14=0.
3. Due to abnormal condition (No drives available DT, DK, etc.)

RUN SUMMARY

Printed at end of exerciser run. Terminated. Cleared by one of the following

1. ↑C — 1st will type run summary.
   — 2nd inhibit further typing of run summary.
2. All selected modules dropped due to errors detected.

NNNNN Module name

AT XXXXXX
ADDR

STAT SSSSSS
Status

PAGCNT CCCCC
Pass count (decimal)

ERRCNT EEEEE
Error count (decimal)

NOTE: Run summary helps detect hung modules since not all modules have watchdog timer.

ROTATION ENABLED

Write buffer rotation enabled.

Range: From XXXXXX To YYYYYY

NOTE: Occurs if sufficient free core exists above last module. No free core = no message. Write buffer assigned within address range of Monitor.
APC 100 of IOMODX may be modified to change write buffer size requested from Monitor.

PWR FAILURE

PWR Failure
Printed upon restart from power failure. Run mode active at time of power fail reactivated. If not goes back to command mode under Dec X/11.

ASCII MESSAGES

Allows modules to report:

1. Error condition
2. Status condition
3. End of pass statistics

EX.

LPAA0 PC XXXXXX APC YYYYYY PASS#NNNNN
LP is off line
RKAA0 PC XXXXXX APC YYYYYY PASS#NNNNN
Data transfers XXXXXX
Soft errors YYYYYY
Hard errors ZZZZZZ

KT11 ENABLED

KT11 Enabled

Standard Monitor determines KT11 present and allowed.

RELOCATED TO XX0000

Will be printed if KT11 present and 8K of free core exists and not inhibited by RUNL command.

DEC X/11 MODULES

I/O MODULES

IOMOD	Once started by Monitor is driven strictly by interrupts and runs continuously. Depends on expected interrupts. If they do not occur may become hung.
IOMODX	Tests NPR devices. Utilizes Monitor's rotating buffer.
IOMODR	Cannot run when program relocated (offset no 0)
IOMODP	Can only be run when exerciser is relocated to even 32K boundaries.
BKMOD	Trace mode. A trace trap occurs after every module instruction. To permit servicing of I/O modules, run one at a time.
NBKMOD	Non-trace, run first one after the other.
EX.	Module to check parity core memory. ON awakes after first run if parity error occurs.

MODULE – TWO SECTIONS

1. Front end
2. Module code itself

FRONT END

MODNAM	5 bytes. Name ASCII
XFLAG 1	1 byte. Force IOMODX to write from each bank of memory.
ADDR	1 word. Addr of first register of device to be tested.
VECTOR	1 word. Device's vector
BR1	1 byte. 1st BR level
BR2	1 byte. 2nd BR level if any
DVID 1	1 word, device count. Each bit indicating a device. One bit set for each one.
SR 1	1 word, internal switch register for module.
STAT	Status word as given before.
INIT	1 word, module start addr.
SPOINT	1 word, addr to load in stack pointer when first starting module.
PASCNT	1 word
ERRCNT	1 word
SVR0–SVR6	6 words. Locations to save modules' registers codes and stack pointer when control given back to Monitor.
CSRA	1 word. Addr of failing device CSR.
SBADR/ACSR	1 word. Data error addr of good data/error call contents of failing device CSR.
WASADR/ASTAT	1 word. Data error contains ADDR of bad data/error call contains contents of status register of failing device.
ASB	1 word. Contains expected good data.
AWAS	1 word. Contains actual data – bad data.
RSTRT	1 word. Module's restart ADDR.

NOTE: Additional words in IOMODX only.

RBUFVA	Module's read buffer virtual ADDR.
RBUFPA	Module's read buffer buffer physical ADDR - Low 16 bits.
RBUFEA	Module's read buffer extended ADDR bits. Shifted to positions 4 and 5
RBUFSZ	Module's write buffer physical address, low 16 bits, assigned by Monitor when servicing GWBUF $ CALL from module.
WBUFEA	Assigned write buffer extended ADDR bits, shifted to bit positions 4 and 5
WBUFRQ	Write buffer size requested by module (in words)
WBUFSZ	Write buffer size allocated to module by monitor (in words)

NOTE: End of IOMODX Special Area.

FREE 1 word, reserved for future monitor use.

STACK AREA
32 words, module's stack module runs and operates on own stack.

For Additional Information on the Module Codes refer to:

1. DEC X/11 User's Documentation and Reference Guide DXQBA-*-D
2. DEC X/11 Programmer's Guide
3. Training Documents

Prepared by Educational Services
Digital Equipment Corporation

PRINTED IN USA EJ 1014 76 10A/03 15-04