HARDWARE

DPS8M MULTICS PROCESSOR SIMULATOR MANUAL

DRAFT-7

NOT FOR DISTRIBUTION

SIMULATOR

NOTICE

SIMH COPYRIGHT NOTICE

The following copyright notice applies to the SIMH source, binary, and documentation:

PREFACE

This manual describes the DPS8M Simulator. It is important to note that this manual does **not** provide documentation on Multics, only information about the simulator and some minimal Multics configuration information. For detailed information on Multics please see the following resources:

- http://ringzero.wikidot.com
- http://www.bitsavers.org/pdf/honeywell/multics/

This manual is intended for:

- anyone who wants to understand the various aspects of the DPS8M simulator
- anyone who wants to configure and run the DPS8M simulator
- Multics System / Site Administrators who wish to change the default simulated hardware configuration

The information and specifications in this document are subject to change without notice.

This manual documents version 2.1 of the DPS8M Multics Mainframe Simulator.

Contents

1 GETTING STARTED

This section covers getting started with the simulator. First, getting a copy of the simulator (build or download) will be discussed. Then a review of a quick Multics startup will be shown.

1.1 BUILDING THE SIMULATOR FROM GIT

Building the simulator has been tested on a number of platforms.

DRAFT NOTE: All of the instructions below are for Release 2.0 of the simulator and need to be verified and/or updated for 2.1.

1.1.1 BUILDING DPS8M UNDER UBUNTU

This needs to be tested

Install needed packages:

```
(for 16.04):  sudo apt install git clang
(for 18.04):  sudo apt install git clang libtool m4 automake
```

Build 'libuv'

```
git clone https://github.com/libuv/libuv.git
cd libuv
git checkout v1.23.0
sh autogen.sh
./configure
make
sudo make install
cd ..
```

Build dps8

```
git clone https://gitlab.com/dps8m/dps8m
cd dps8m
git checkout R2.0
make LOCKLESS=1
```

Running the Simulator

```
(Copy 'src/dps8/dps8' to the desired working directory)
./dps8 [boot script]
```

1.1.2 BUILDING DPS8M UNDER MACOS

This needs to be update/tested

Install needed packages:

```
brew install git clang libtool m4 automake
```

Build 'libuv'

```
git clone https://github.com/libuv/libuv.git
cd libuv
git checkout v1.23.0
sh autogen.sh
./configure
make LOCKLESS=1
sudo make install
cd ..
```

Build dps8

```
git clone https://gitlab.com/dps8m/dps8m
cd dps8m
git checkout R2.0
make LOCKLESS=1
```

Running the Simulator

```
(Copy 'src/dps8/dps8' to the desired working directory)
./dps8 [boot script]
```

1.1.3 BUILDING DPS8M UNDER CENTOS

This needs to be tested

Install needed packages:

```
sudo yum install git automake libtool clang
```

Build 'libuv'

```
git clone https://github.com/libuv/libuv.git
cd libuv
git checkout v1.23.0
sh autogen.sh
./configure
make
sudo make install
cd ..
```

NOTE: For some reason libuv is not being found in its default installation location of /usr/local/lib and it was necessary do the following for it to be found:

```
sudo ln -s /usr/local/lib/libuv.so.1.0.0 /usr/lib64/libuv.so.1
```

Build dps8

```
git clone https://gitlab.com/dps8m/dps8m
cd dps8m
git checkout R2.0
make LOCKLESS=1
```

Running the Simulator

```
(Copy 'src/dps8/dps8' to the desired working directory)
./dps8 [boot script]
```

1.1.4 BUILDING DPS8M UNDER FEDORA 30 LINUX FOR WINDOWS

This needs to be tested

Install needed packages:

```
sudo dnf install git mingw64-gcc mingw64-libgnurx libtool make
sudo dnf update perl-Errno
```

Build 'libuv'

```
git clone https://github.com/libuv/libuv.git
cd libuv
git checkout v1.23.0
sh autogen.sh
mingw64-configure
make
sudo make install
cd ..
```

Build dps8

```
git clone https://gitlab.com/dps8m/dps8m
cd dps8m
git checkout R2.0
make CROSS=MINGW64 LOCKLESS=1
```

Running the Simulator

```
Copy these files to your windows machine:
dps8.exe
/usr/x86_64-w64-mingw32/sys-root/mingw/bin/libwinpthread-1.dll
/usr/x86_64-w64-mingw32/sys-root/mingw/bin/libuv-1.dll
```

1.1.5 BUILDING DPS8M UNDER WINDOWS WITH MSYS2

This needs to be tested

Install MSYS2 and packages:

In your browser, go to https://msys2.github.io/

Download and run the msys2 x86_64 installer. Follow the installation instructions on the webpage.

After step 7 ("Now Pacman is fully committed..."), fetch the dps8m code, needed libraries and packages by entering:

```
git clone git://git.code.sf.net/p/dps8m/code dps8m-code
git clone https://github.com/libuv/libuv.git

pacman -S mingw-w64-x86_64-gcc
pacman -S mingw-w64-x86_64-binutils
pacman -S mingw-w64-x86_64-libtool
pacman -S mingw-w64-x86_64-make
pacman -S mingw-w64-x86_64-dlfcn
pacman -S mingw-w64-x86_64-diffutils
pacman -S mingw-w64-x86_64-gettext
pacman -S mingw-w64-x86_64-libgnurx
pacman -S mingw-w64-x86_64-binutils
pacman -S automake
pacman -S autoconf
pacman -S unzip
pacman -S zip
```

Start a MSYS2 MinGW window and build the code:

```
Start -> All programs -> MSYS2 64 bit -> MSYS2 MINGW 64 bit

cd libuv
git checkout v1.23.0
sh autogen.sh
./configure
mingw32-make.exe MAKE=mingw32-make.exe

cd
cd dps8-code
git checkout R2.0
```

```
git fetch
mingw32-make.exe
```

Running the Simulator

Start a Windows command shell:

```
Start -> All programs -> Accessories -> Command Prompt
```

Change directory to where the simulator is to be run from and copy the needed files:

```
cd <wherever>
copy c:\msys64\home\Admin\dps8m-code\rc\dps8\dps8.exe
copy c:\msys64\mingw64\bin\libwinpthread-1.dll
copy c:\msys64\mingw64\bin\libuv-1.dll
```

Run Multics:

```
.\dps8.exe <boot.ini>
```

1.1.6 BUILDING DPS8M UNDER RASPIAN

This needs to be tested

Install needed packages:

```
sudo apt-get update
sudo apt-get install git libtool automake clang
```

Build 'libuv'

```
git clone https://github.com/libuv/libuv.git
cd libuv
git checkout v1.23.0
sh autogen.sh
./configure --prefix=/usr
make
sudo make install
cd ..
```

Build dps8

```
git clone https://gitlab.com/dps8m/dps8m
cd dps8m
git checkout R2.0
make M32=1
```

Running the Simulator

```
(Copy 'src/dps8/dps8' to the desired working directory)
./dps8 [boot script]
```

2 SIMULATED HARDWARE OVERVIEW

2.1 SIMULATED COMPONENTS

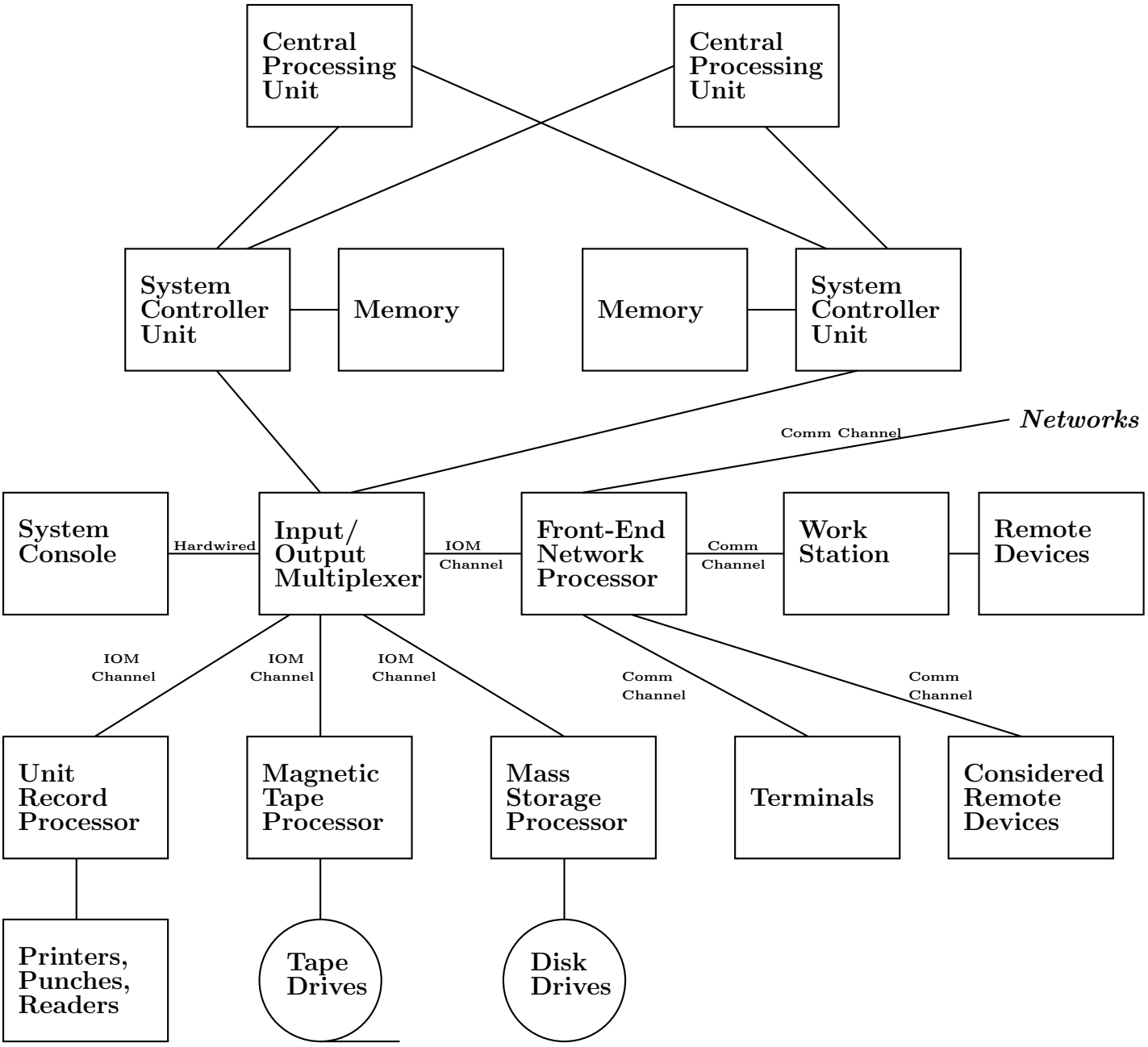
The DPS8M Simulator simulates not only the DPS8M Processor but a complete mainframe computer system with all its peripheral devices. These include:

- Central Processing Unit (CPU)
- Input/Output Multiplexer (IOM)
- System Control Unit (SCU)
- Front End Processor (FNP)
- Tape Drives
- Disk Storage Units
- Printers
- Card Reader
- Card Punch
- Operators Console
- ABSI IMP Interface

With this simulator, in a matter of seconds, you can conjure up a system that would have cost millions of dollars in the 1980s.

2.2 SYSTEM DIAGRAM

Below is a diagram that shows the various simulated hardware components and how they connect together.


Central Processing Unit
Central Processing Unit
System Controller Unit
Memory
Memory
System Controller Unit
Networks
Comm Channel
System Console
Hardwired
Input/ Output Multiplexer
IOM Channel
Front-End Network Processor
Comm Channel
Work Station
Remote Devices
IOM Channel
IOM Channel
IOM Channel
Comm Channel
Comm Channel
Unit Record Processor
Magnetic Tape Processor
Mass Storage Processor
Terminals
Considered Remote Devices
Printers, Punches, Readers
Tape Drives
Disk Drives

3 SIMULATOR CONFIGURATION / OPERATION

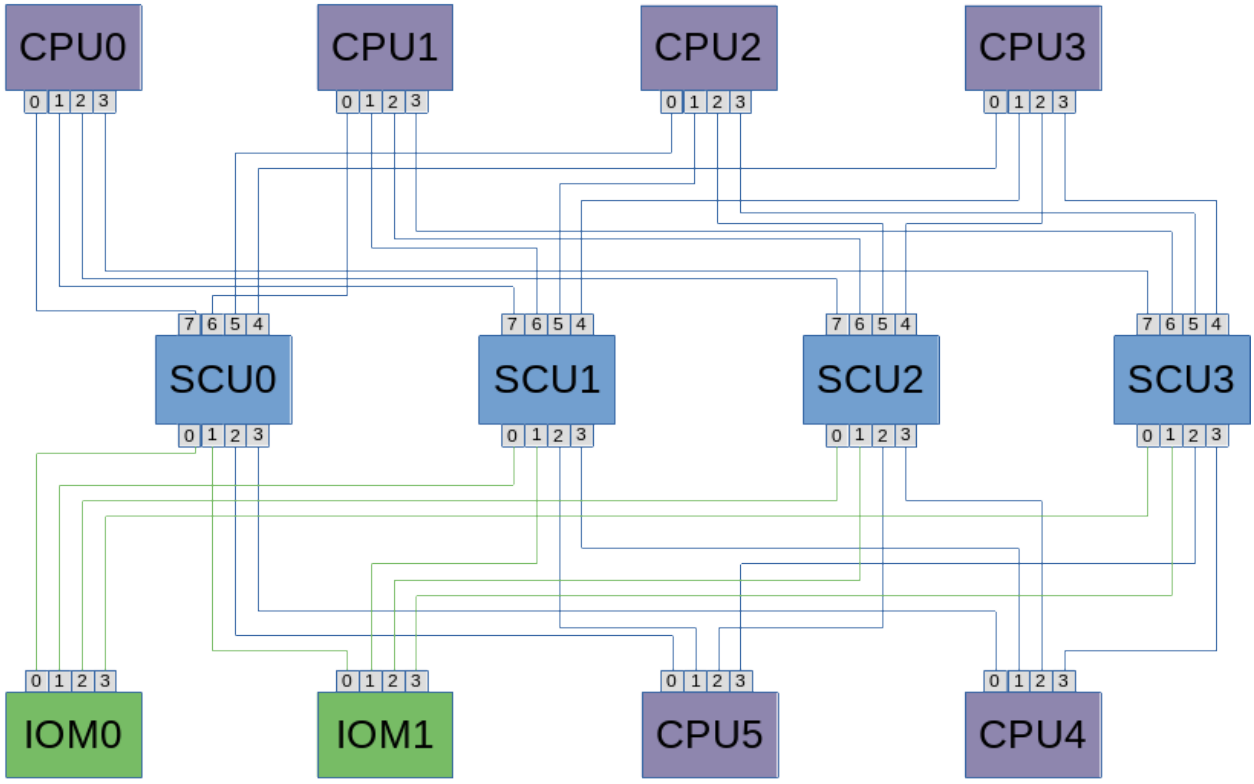
This section covers the configuration of the simulator and how to change it.

Note that the default configuration contains more hardware in it than Multics can handle (see the MULTICS CONFIGURATION section warning). So you want to pick and choose which parts of the configuration meet your needs. The MULTICS CONFIGURATION section shows how to do this.

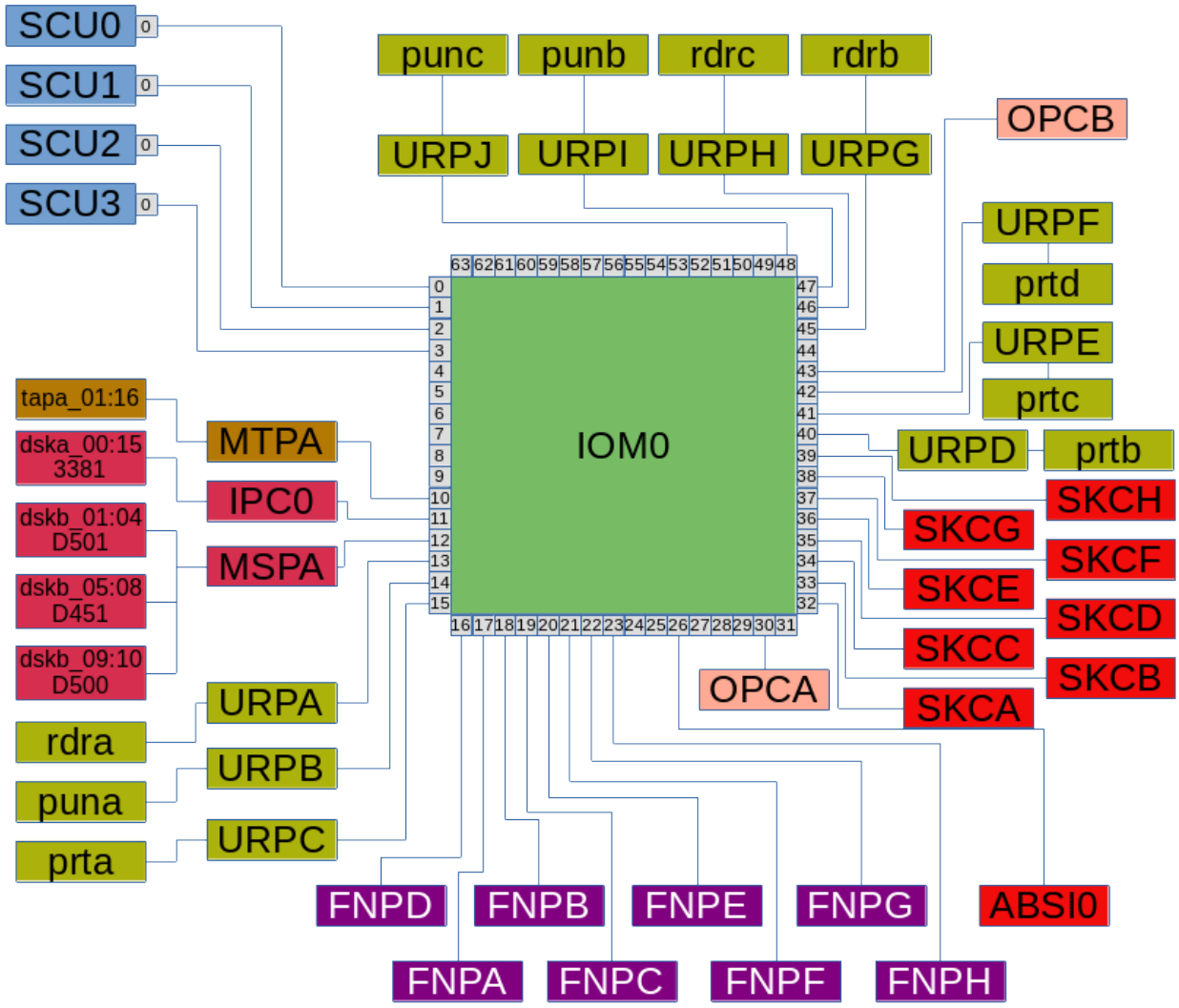
First, the following diagrams show the default configuration of the simulator when it first starts up.

3.1 CPU/SCU DEFAULT CONFIGURATION

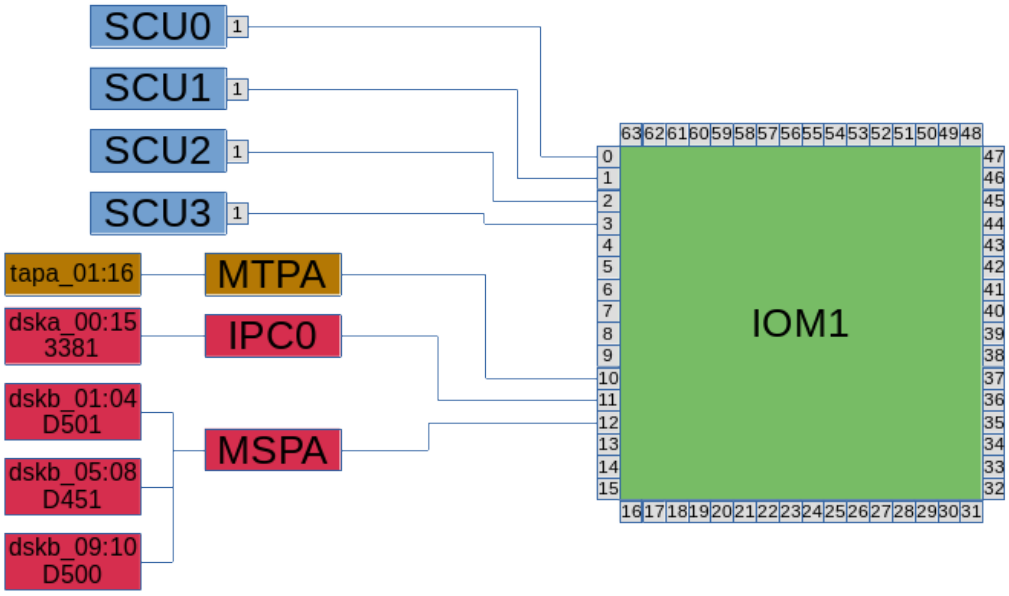
By default there are 6 CPUs, 4 SCUs (with max memory) and 2 IOMs. In most systems, using 2 to 4 of the CPUs will be sufficent. Since each of the 4 SCUs is configured with 16 Mw, the maximum amount of memory is available (64 Mw). That is the maximum phyiscal memory the simulator can provide (and Multics can make use of). Here is how these devices are cabled:


CPU0
CPU1
CPU2
CPU3
0 1 2 3
7 6 5 4
SCU0
SCU1
SCU2
SCU3
IOM0
IOM1
CPU5
CPU4

3.2 IOM0 DEFAULT CONFIGURATION



3.3 IOM1 DEFAULT CONFIGURATION



3.4 CENTRAL PROCESSING UNIT

This section describes the configuration and operation of the simulated DPS8M Central Processing Unit (CPU).

3.4.1 CPU OVERVIEW

The DPS8M CPU has

- 36 bit words
- 18 bit addresses (allowing a segment size of 256 thousand words)
- 15 bit segment numbers (allowing 32 thousand segments of 256 thousand words)
- 1 million instructions per second (1 MIP)
- Direct Memory Access I/O (DMA)
- Two 36 bit accumulators (A and Q)
- Eight 18 bit index registers (X0-X7)
- Eight pointer/address registers (each containing a 15 bit segment number, 18 bit address and 6 bit bit number)
- 36 and 72 bit integer and floating point arithmetic
- 10 digit decimal arithmetic
- String move and compare
- Decimal number formatting (e.g COBOL or PL/1 "PIC")

3.4.2 CPU OPTIONS

These are the currently supported "normal" CPU options:

faultbase

num

data

stopnum

mode

speed

port

assignment

interlace

enable

init_enable

store_size

These options are supported as "hacks" and should not normally be used:

dis_enable

halt_on_unimplemented

disable_wam

report_faults

tro_enable

drl_fatal

useMap

address

disable_cache

3.5 SYSTEM CONTROL UNIT (SCU)

This section describes the configuration and operation of the simulated System Control Unit (SCU).

3.5.1 SCU OPTIONS

mode

maska

maskb

portN (N = 0 - 7)

lwrstoresize

cyclic

nea

onl

int

lwr

These options are supported as "hacks" and should not normally be used:

elapsed_days

steady_clock

bullet_time

y2k

3.6 INPUT/OUTPUT MULTIPLEXER

This section describes the configuration and operation of the simulated Input/Output Multiplexer (IOM).

3.6.1 IOM OPTIONS

model

os

boot

iom_base

multiplex_base

tapechan

cardchan

scuport

port

addr

interlace

enable

initenable

halfsize

store_size

3.7 CARD PUNCH

This section describes the configuration and operation of the simulated Card Punch.

The card punch operates by writing a simulated card deck to a spool file. Each deck will be placed in it's own spool file. The location of these spool files is controlled by the "path" global option (see below).

The file will contain all banner, lace and trailer cards that Multics prints. The lace cards are parsed to extract job information that is used to name the spool file.

Note that there is a utility "punutil" that can be used to extract the various cards from the deck in a more usable format (such as ASCII).

3.7.1 CARD PUNCH FILE FORMAT

Each column in a punched card has twelve rows, designated from top to bottom:

```
& - 0 1 2 3 4 5 6 7 8 9
```

Given the above, an 80 column punched card would then look like this:

&	&	&	&	&	
-	-	-	-	-	
0	0	0	0	0	
1	1	1	1	1	
2	2	2	2	2	
3	3	3	3	3	...
4	4	4	4	4	
5	5	5	5	5	
6	6	6	6	6	
7	7	7	7	7	
8	8	8	8	8	
9	9	9	9	9	

&	&	&	&	&
-	-	-	-	-
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5
6	6	6	6	6
7	7	7	7	7
8	8	8	8	8
9	9	9	9	9

When representing a card column, the most obvious way to do it is to assign a single bit per punch. This means that there are 12 bits for each column. When looking at a hexadecimal dump of a spool file, each column occupies three 4 bit nibbles. The following representation is big-endian:

Column 1												Column 2											
&	-	0	1	2	3	4	5	6	7	8	9	&	-	0	1	2	3	4	5	6	7	8	9
Byte 1								Byte 2								Byte 3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

3.7.2 CARD PUNCH PARAMETERS

This section describes the parameters that can be set for the Card Punches.

The Global Parameters apply to all card punch units while the Unit Parameters apply to a single card punch unit.

Global Parameters

Global Parameters are set using the "`SET PUN XXXX=yyyy`" simulator command (`XXXX` is the parameter being set and `yyyy` is the value to set).

The "`SHOW PUN XXXX`" command will show the current value of a global parameter.

PATH

Sets the path that will be used as the location to write punch spool files. When not set, all punch spool files will be written to the simulator's current directory. When this path is set, the simulator expects there to be a subdirectory for each named punch unit (the simulator will **not** create any directories). For example, if the PATH is set to "/home/tom/punches" and three card punch units are defined (puna, punb and punc), then the following directory structure must exist for punch jobs to be output properly:

```
/home/tom/punches
/home/tom/punches/puna
/home/tom/punches/punb
/home/tom/punches/punc
```

3.7.3 CARD PUNCH OPTIONS

Options are set on a per device basis with the "`SET PUNn CONFIG=XXXX=yyyy`" simulator command (`n` is the unit number with 0 = A, 1 = B, etc, `XXXX` is the name of the option and `yyyy` is the option value).

LOGCARDS

When this option is turned on, a significant amount of card punch diagnostic output will be displayed on the console. This includes an "ASCII Art" form of the punched cards where asterisks (*) are used to represent punched holes. It is recommended to only turn on this option if you are attempting to diagnose a card punch issue.

Valid option values are:

`0`
`off`
`disable`
	Turn off logging of card punch diagnostic data

`1`
`on`
`enable`
	Turn on logging of card punch diagnostic data

3.8 OPERATOR CONSOLE

This section describes the configuration and operation of the simulated Operator Console (OPCON).

3.8.1 OPERATOR CONSOLE OPTIONS

autoaccept

noempty

attn_flush

model

3.8.2 OPERATOR CONSOLE SCRIPTING

3.9 PRINTER

This section describes the configuration and operation of the simulated Printer.

3.9.1 PRINTER OPTIONS

split

3.10 TAPE DRIVES

This section describes the configuration and operation of the simulated Tape Drives (TAP).

3.10.1 TAPE DRIVE PARAMETERS

This section describes the parameters that can be set for the Tape Drives.

The Global Parameters apply to all tape drives while the Unit Parameters apply to a single drive unit.

Global Parameters

Global Parameters are set using the "`SET TAPE XXXX=yyyy`" simulator command (`XXXX` is the parameter being set and `yyyy` is the value to set).

The "`SHOW TAPE XXXX`" command will show the current value of a global parameter.

ADD_PATH

Adds a new tape search path at the end of the current search path list. The parameter format is:

```
prefix=directory
```

For example, to look for tapes with a volume name starting with "BK" in a directory "./tapes/backups":

```
SET TAPE ADD_PATH=BK=./tapes/backups
```

Note that when using relative paths, the starting point is the simulator's current directory. Also note that if a `SET DEFAULT_PATH` is done, all previous **ADD_PATH** entries will be removed.

Adding paths can't be done until the **DEFAULT_PATH** has first been set.

CAPACITY_ALL

Sets the maximum size that can be written to a tape file for all tape drives.

DEFAULT_PATH

Sets the default path to use when searching for tape files. Note that setting this option will clear out any previous **ADD_PATH** parameters.

If **DEFAULT_PATH** is not set, the simulator will default to look for tape files in its current directory.

For example, to look for tapes in a directory "./tapes":

```
SET TAPE DEFAULT_PATH=./tapes
```

Note that when using relative paths, the starting point is the simulator's current directory.

NUNITS

Sets the maximum number of tape drives available.

Unit Parameters

Unit parameters are set on a per Tape Drive basis. Unit parameters are set using the "`SET TAPEn XXXX=yyyy`" simulator command (`n` is the tape drive number, `XXXX` is the parameter being set and `yyyy` is the value to set).

NAME

Set the name of a specific tape drive. For example, to set the name of tape drive 1 to "tapa_01":

```
SET TAPE1 NAME=tapa_01
```

3.10.2 TAPE DRIVE OPTIONS

The tape drives have no configuration options.

3.10.3 CREATING A SEARCH TABLE

When the search table is evaluated during a tape file lookup, it is processed in order with a first match being accepted. For example, if the following commands are given:

```
SET TAPE DEFAULT_PATH=./tapes/general
SET TAPE ADD_PATH=B=./tapes/billing SET TAPE ADD_PATH=BK=./tapes/backups
```

a tape with a volume name of "BK001" will end up in the ./tapes/billing directory and not the ./tapes/backups directory. Since the name starts with a "B", the first entry in the search table will be selected.

To get the intended effect, the commands should be reorder like this:

```
SET TAPE DEFAULT_PATH=./tapes/general
SET TAPE ADD_PATH=BK=./tapes/backups
SET TAPE ADD_PATH=B=./tapes/billing
```

this way the names starting with "BK" will be checked first followed by the more general "B".

Important Note: The simulator "attach" command (most often used in .ini files to mount the boot tape) is not aware of the alternate search directories so, if a directory is needed for an attach command, the path must be specified on the command:

```
attach -r tape0 ./tapes/12.7/12.7MULTICS.tap
```

3.11 CABLING SYSTEM COMPONENTS

This section describes how the simulated components of a system are cabled together.

3.11.1 CABLE COMMAND

This section describes the CABLE command and its options.

This command is responsible for managing the interconnection of major subsystems: CPU, SCU, IOM; contollers: MTP, IPC, MSP, URP;

The unit numbers for CPUs, IOMs, and SCUs (eg IOM3 is IOM unit 3) are SIMH unit numbers; these units can be individually configured to desired Multics unit numbers. However, it is somewhat easier to adopt an one-to-one practice; IOM0 == IOMA, etc.

The following shows all the various options for the CABLE command:

CABLE RIPOUT

Remove all cables from the configuration.

CABLE SHOW

Show the current cabling configuration.

CABLE DUMP

Show the current cabling configuration in great detail.

CABLE SCUi j IOMk l

Connect SCU i port j to IOM k port l. "cable SCU0 0 IOM0 2"

CABLE SCUi j CPUk l

Connect SCU i port j to CPU k port l. "cable SCU0 7 CPU0 7"

CABLE IOMi j MTPk
CABLE IOMi j MTPk l

Connect IOM i channel j to MTP k port l (l defaults to 0).

CABLE IOMi j MSPk
CABLE IOMi j MSPk l

Connect IOM i channel j to MSP k port l (l defaults to 0).

CABLE IOMi j IPCk
CABLE IOMi j IPCk l

Connect IOM i channel j to IPC k port l (l defaults to 0).

CABLE IOMi j OPCk

Connect IOM i channel j to OPC k.

CABLE IOMi j FNPk

Connect IOM i channel j to FNP k.

CABLE IOMi j ABSIk

Connect IOM i channel j to ABSI k.

CABLE IOMi j SKCk

Connect IOM i channel j to SKC k.

CABLE MTPi j TAPEk

Connect MTP i device code j to tape unit k.

CABLE IPCi j DISKk

Connect IPC i device code j to disk unit k.

CABLE MSPi j DISKk

Connect MSP i device code j to disk unit k.

CABLE URPi j RDRk

Connect URP i device code j to card reader unit k.

CABLE URPi j PUNk

Connect URP i device code j to card punch unit k.

CABLE URPi j PRTk

Connect URP i device code j to printer unit k.

3.11.2 DEFAULT CABLE CONFIGURATION

Below are tables showing the default cable configuration. Note that you can always get this information by simply starting the simulator with no ".ini" file and entering the command "cable show" (which is how these tables were generated).

SCU	SCU port	<-->	IOM IOM	IOM port
0	0		0	0
0	1		1	0
1	0		0	1
1	1		1	1
2	0		0	2
2	1		1	2
3	0		0	3
3	1		1	3

IOM	IOM port	<--> -->	SCU SCU	port
0	0		0	0
0	1		1	0
0	2		2	0
0	3		3	0
1	0		0	1
1	1		1	1
1	2		2	1
1	3		3	1

SCU	SCU port	<--> -->	CPU CPU	port
0	2		5	0
0	3		4	0
0	4		3	0
0	5		2	0
0	6		1	0
0	7		0	0
1	2		5	1
1	3		4	1
1	4		3	1
1	5		2	1
1	6		1	1
1	7		0	1
2	2		5	2
2	3		4	2
2	4		3	2
2	5		2	2
2	6		1	2
2	7		0	2
3	2		5	3
3	3		4	3
3	4		3	3
3	5		2	3
3	6		1	3
3	7		0	3

	CPU	<-->	SCU	
CPU	port	-->	SCU	port
0	0		0	7
0	1		1	7
0	2		2	7
0	3		3	7
1	0		0	6
1	1		1	6
1	2		2	6
1	3		3	6
2	0		0	5
2	1		1	5
2	2		2	5
2	3		3	5
3	0		0	4
3	1		1	4
3	2		2	4
3	3		3	4
4	0		0	3
4	1		1	3
4	2		2	3
4	3		3	3
5	0		0	2
5	1		1	2
5	2		2	2
5	3		3	2

				IOM	<-->	controller			
			ctlr		ctlr	chan			
IOM	chan	-->	idx	port	type	type	device	board	command
0	10		0	0	MTP	PSI	0x4e5f98	0x553c50	0x43ed50
0	11		0	0	IPC	PSI	0x4e4208	0x551f20	0x4120e0
0	12		0	0	MSP	PSI	0x4e4368	0x5527a0	0x4120e0
0	13		0	0	URP	PSI	0x4f1b28	0x4f10e0	0x449510
0	14		1	0	URP	PSI	0x4f1b28	0x4f1168	0x449510
0	15		2	0	URP	PSI	0x4f1b28	0x4f11f0	0x449510
0	16		3	0	FNP	Direct	0x4e51f0	0x4e48b8	0x4288c0
0	17		0	0	FNP	Direct	0x4e51f0	0x4e4720	0x4288c0
0	18		1	0	FNP	Direct	0x4e51f0	0x4e47a8	0x4288c0
0	19		2	0	FNP	Direct	0x4e51f0	0x4e4830	0x4288c0
0	20		4	0	FNP	Direct	0x4e51f0	0x4e4940	0x4288c0
0	21		5	0	FNP	Direct	0x4e51f0	0x4e49c8	0x4288c0
0	22		6	0	FNP	Direct	0x4e51f0	0x4e4a50	0x4288c0
0	23		7	0	FNP	Direct	0x4e51f0	0x4e4ad8	0x4288c0
0	26		0	0	ABSI	Direct	0x4de7b8	0x4de5a0	0x403d50
0	30		0	0	OPC	CPI	0x4deea8	0x4de8e0	0x40b5b0
0	32		0	0	SKC	Direct	0x4ef7a8	0x556d60	0x445c20
0	33		1	0	SKC	Direct	0x4ef7a8	0x556de8	0x445c20
0	34		2	0	SKC	Direct	0x4ef7a8	0x556e70	0x445c20
0	35		3	0	SKC	Direct	0x4ef7a8	0x556ef8	0x445c20
0	36		4	0	SKC	Direct	0x4ef7a8	0x556f80	0x445c20
0	37		5	0	SKC	Direct	0x4ef7a8	0x557008	0x445c20
0	38		6	0	SKC	Direct	0x4ef7a8	0x557090	0x445c20
0	39		7	0	SKC	Direct	0x4ef7a8	0x557118	0x445c20
0	40		3	0	URP	PSI	0x4f4c48	0x4f4398	0x44aee0
0	41		4	0	URP	PSI	0x4f4c48	0x4f4420	0x44aee0
0	42		5	0	URP	PSI	0x4f4c48	0x4f44a8	0x44aee0
0	43		1	0	OPC	CPI	0x4e1eb8	0x4e1978	0x40b950
0	45		6	0	URP	PSI	0x4f4c48	0x4f4530	0x44aee0
0	46		7	0	URP	PSI	0x4f4c48	0x4f45b8	0x44aee0
0	47		8	0	URP	PSI	0x4f4c48	0x4f4640	0x44aee0
0	48		9	0	URP	PSI	0x4f4c48	0x4f46c8	0x44aee0
1	10		0	1	MTP	PSI	0x4e5f98	0x553c50	0x43ed50
1	11		0	1	IPC	PSI	0x4e4208	0x551f20	0x4120e0
1	12		0	1	MSP	PSI	0x4e4368	0x5527a0	0x4120e0

	MTP	<-->	IOM	
MTP	port	-->	IOM	port
0	0		0	10
0	1		1	10

	MSP	<-->	IOM	
MSP	port	-->	IOM	port
0	0		0	12
0	1		1	12

	IPC	<-->	IOM	
IPC	port	-->	IOM	port
0	0		0	11
0	1		1	11

	URP	<-->	IOM	
URP	port	-->	IOM	port
0	0		0	13
1	0		0	14
2	0		0	15
3	0		0	40
4	0		0	41
5	0		0	42
6	0		0	45
7	0		0	46
8	0		0	47
9	0		0	48

	ABSI	<-->	IOM	
ABSI	port	-->	IOM	port
0	0		0	26

	SKC	<-->	IOM	
SKC	port	-->	IOM	port
0	0		0	32
1	0		0	33
2	0		0	34
3	0		0	35
4	0		0	36
5	0		0	37
6	0		0	38
7	0		0	39

	OPC	<-->	IOM	
OPC	port	-->	IOM	port
0	0		0	30
1	0		0	43

controller <--> device

MTP	dev_code	-->	TAPE		IPC	dev_code	-->	DISK
0	1		1		0	0		0
0	2		2		0	1		1
0	3		3		0	2		2
0	4		4		0	3		3
0	5		5		0	4		14
0	6		6		0	5		15
0	7		7		0	6		16
0	8		8		0	7		17
0	9		9		0	8		18
0	10		10		0	9		19
0	11		11		0	10		20
0	12		12		0	11		21
0	13		13		0	12		22
0	14		14		0	13		23
0	15		15		0	14		24
0	16		16		0	15		25

MSP	dev_code	-->	DISK		URP	dev_code	-->	URP
0	1		4		0	1		0
0	2		5		1	1		0
0	3		6		2	1		0
0	4		7		3	1		1
0	5		8		4	1		2
0	6		9		5	1		3
0	7		10		6	1		1
0	8		11		7	1		2
0	9		12		8	1		1
0	10		13		9	1		2

device	<-->	controller	
TAPE	-->	MTP	dev_code	type
1		0	1	MTP
2		0	2	MTP
3		0	3	MTP
4		0	4	MTP
5		0	5	MTP
6		0	6	MTP
7		0	7	MTP
8		0	8	MTP
9		0	9	MTP
10		0	10	MTP
11		0	11	MTP
12		0	12	MTP
13		0	13	MTP
14		0	14	MTP
15		0	15	MTP
16		0	16	MTP

```
DISK  -->  CTLR  dev_code  type
 0           0       0      IPC
 1           0       1      IPC
 2           0       2      IPC
 3           0       3      IPC
 4           0       1      MSP
 5           0       2      MSP
 6           0       3      MSP
 7           0       4      MSP
 8           0       5      MSP
 9           0       6      MSP
 10          0       7      MSP
 11          0       8      MSP
 12          0       9      MSP
 13          0       10     MSP
 14          0       4      IPC
 15          0       5      IPC
 16          0       6      IPC
 17          0       7      IPC
 18          0       8      IPC
 19          0       9      IPC
 20          0       10     IPC
 21          0       11     IPC
 22          0       12     IPC
 23          0       13     IPC
 24          0       14     IPC
 25          0       15     IPC

RDR  -->  URP  dev_code  type
 0         0       1      URP
 1         6       1      URP
 2         7       1      URP
PUN  -->  URP  dev_code  type
 0         1       1      URP
 1         8       1      URP
 2         9       1      URP
PRT  -->  URP  dev_code  type
 0         2       1      URP
 1         3       1      URP
 2         4       1      URP
 3         5       1      URP
```

4 MULTICS CONFIGURATION

This section describes how to match the Multics configuration to the simulator.

4.1 TYPICAL SYSTEM CONFIG DECK

The following config deck would be for a typical, moderately sized system. It only uses devices that are cabled by default:

```
clok -delta 8.  -zone pst
iom -tag a -port 0 -model iom -state on
iom -tag b -port 1 -model iom -state on
cpu -tag a -port 7 -state on -type dps8 -model 70.  -cache 8.
cpu -tag b -port 6 -state on -type dps8 -model 70.  -cache 8.
mem -port a -size 4096.  -state on
mem -port b -size 4096.  -state on
mem -port c -size 4096.  -state on
mem -port d -size 4096.  -state on
ipc -type fips -iom a -chn 13 -nchan 1
prph -subsys dska -iom a -chn 13 -nchan 1 -model 3381.  -number 16
prph -device fnpd -iom a -chn 20 -model 6670.  -state on
mpc -ctlr mtpa -iom a -chn 12 -nchan 1 -model 501.
prph -subsys tapa -iom a -chn 12 -nchan 1 -model 500.  -number 16.
prph -device opca -iom a -chn 36 -model 6001.  -ll 256.  -state on
mpc -ctlr urpa -iom a -chn 15 -model 8004.  -nchan 1
prph -device rdra -iom a -chn 15 -model 301.
mpc -ctlr urpb -iom a -chn 16 -model 8004.  -nchan 1
prph -device puna -iom a -chn 16 -model 301.
mpc -ctlr urpc -iom a -chn 17 -model 8004.  -nchan 1
prph -device prta -iom a -chn 17 -model 1600.  -train 600.  -ll 136.
mpc -ctlr urpd -iom a -chn 50 -model 8004.  -nchan 1
prph -device prtb -iom a -chn 50 -model 1600.  -train 600.  -ll 136.
part -part hc -subsys dska -drive 00a
part -part bos -subsys dska -drive 00a
part -part dump -subsys dska -drive 00a
root -subsys dska -drive 00a
sst -4k 3800.  -16k 2100.  -64k 820.  -256k 260.
dbmj 64.  700.  400.  150.  60.  25.
tcd -apt 1000.  -itt 2000.
intk warm 0.  rpvs star
parm dirw
```

4.2 LARGE SYSTEM CONFIG DECK

The following config deck uses most of the default cabling and will allow Multics to boot:

```
clok -delta 8.  -zone pst
iom -tag a -port 0 -model iom -state on
iom -tag b -port 1 -model iom -state on
cpu -tag a -port 7 -state on -type dps8 -model 70.  -cache 8.
cpu -tag b -port 6 -state on -type dps8 -model 70.  -cache 8.
cpu -tag c -port 5 -state on -type dps8 -model 70.  -cache 8.
cpu -tag d -port 4 -state on -type dps8 -model 70.  -cache 8.
cpu -tag e -port 3 -state on -type dps8 -model 70.  -cache 8.
mem -port a -size 4096.  -state on
mem -port b -size 4096.  -state on
mem -port c -size 4096.  -state on
mem -port d -size 4096.  -state on
ipc -type fips -iom a -chn 13 -nchan 1
prph -subsys dska -iom a -chn 13 -nchan 1 -model 3381.  -number 16
mpc -ctlr mspa -model 612.  -iom a -chn 14 -nchan 1
prph dskb a 14 1 501.  4.  451.  4.  500.  2.
prph -device fnpd -iom a -chn 20 -model 6670.  -state on
mpc -ctlr mtpa -iom a -chn 12 -nchan 1 -model 501.
prph -subsys tapa -iom a -chn 12 -nchan 1 -model 500.  -number 16.
prph -device opca -iom a -chn 36 -model 6001.  -ll 256.  -state on
mpc -ctlr urpa -iom a -chn 15 -model 8004.  -nchan 1
prph -device rdra -iom a -chn 15 -model 301.
mpc -ctlr urpb -iom a -chn 16 -model 8004.  -nchan 1
prph -device puna -iom a -chn 16 -model 301.
mpc -ctlr urpc -iom a -chn 17 -model 8004.  -nchan 1
prph -device prta -iom a -chn 17 -model 1600.  -train 600.  -ll 136.
mpc -ctlr urpd -iom a -chn 50 -model 8004.  -nchan 1
prph -device prtb -iom a -chn 50 -model 1600.  -train 600.  -ll 136.
mpc -ctlr urpe -iom a -chn 51 -model 8004.  -nchan 1
prph -device prtc -iom a -chn 51 -model 1600.  -train 600.  -ll 136.
mpc -ctlr urpf -iom a -chn 52 -model 8004.  -nchan 1
prph -device prtd -iom a -chn 52 -model 1600.  -train 600.  -ll 136.
prph -device opcb -iom a -chn 53 -model 6001.  -ll 256.  -state alt
mpc -ctlr urpg -iom a -chn 55 -model 8004.  -nchan 1
prph -device rdrb -iom a -chn 55 -model 301.
mpc -ctlr urph -iom a -chn 56 -model 8004.  -nchan 1
prph -device rdrc -iom a -chn 56 -model 301.
mpc -ctlr urpi -iom a -chn 57 -model 8004.  -nchan 1
prph -device punb -iom a -chn 57 -model 301.
mpc -ctlr urpj -iom a -chn 60 -model 8004.  -nchan 1
prph -device punc -iom a -chn 60 -model 301.
```

```
part -part hc -subsys dska -drive 00a
part -part bos -subsys dska -drive 00a
part -part dump -subsys dska -drive 00a
root -subsys dska -drive 00a
sst -4k 3800.  -16k 2100.  -64k 820.  -256k 260.
dbmj 64.  700.  400.  150.  60.  25.
tcd -apt 1000.  -itt 2000.
intk warm 0.  rpvs star
parm dirw
```

IMPORTANT WARNING:

You may notice if you compare this config deck with the default cabling in the simulator that not all the devices cabled are configured. The reason is known as the "Evil Config Deck Bug" in Multics.

With the ability of the simulator to simulate a system that is so large no one would ever build it in actual hardware, it is possible to trigger bugs in Multics that are caused by running out of space in internal data structures. The above config deck is an example of a configuration that is right on the edge of triggering one of these bugs. If, for example, you add CPU F into the deck above and do not remove any other cards, Multics will crash when attempting to boot.

If you get something like this, you have made a configuration that is too large:

```
bce (boot) 1454.1:  M-> [auto-input] boot -cold
Do you really wish to boot cold and there by destroy the system hierarchy?
M-> [auto-input] y
CONSOLE: ALERT
1454.2 page_fault:  fatal error at loc 2330
1454.2 Multics not in operation; control process:  Initializer.SysDaemon.z.
bce (crash) 1454.2:  M->
```

5 SIMULATOR DEBUGGING

6 UTIILITY PROGRAMS

This section covers several utility programs that are useful when working with the simulator.

6.1 PUNUTIL

6.2 PRT2PDF

7 REFERENCES