Desktop Cyber Emulator Version 2.0

Installation and User's Guide

May 29, 2003

Table of Contents

1 Introduction 3
2 Installing the Emulator 4
2.1 Installation under Microsoft Windows 98/NT/XP 4
2.2 Installation under UNIX systems 4
3 Configuring the Emulator 5
4 Running the Emu lator 6
4.1 Running Chippewa OS 6
4.2 Running SMM 7
4.3 Running NOS 1 8
4.3.1 Starting the emulation and NOS 1 8
4.3.2 Enable terminal access 9
4.3.3 Tape and printer handling 9
5 Utilities 11
5.1 9-Track Magnetic Tape to TAP Image Conversion 11
5.1.1 TAP Image Format 11
5.2 Convert ASCII to CDC Display Code and back 11
5.3 Create NOS I-Format TAP Image 12
5.4 Verify NOS I-Format TAP Image 12
5.5 Punching Card Decks 13
Appendix A - Example Inititialisation File 14
Appendix B - Acknowledgments 15

1 Introduction

The Desktop Cyber Emulator is the result of a project which brought back to life the revolutionary design of Control Data Corporation (CDC) Cyber mainframes. The first CDC Cyber's have been designed and built by a small team headed by Seymour Cray at CDCs Chippewa Falls labs in the early 1960s. The Desktop Cyber is emulating CDC Cyber mainframes and peripherals in software.

The software provides a reasonable emulation of a "typical" CDC Cyber 6600, 7x, 17x based system including common peripherals such as console, tape and disk drives, extended core storage, card reader, printer and terminal multiplexer.

The emulator runs the following CDC operating systems: Chippewa OS, SMM, KRONOS 2.1, NOS 1.2, NOS 1.3, NOS 1.4, NOS 2.2 and NOS 2.8.1. The emulator does not support NOS/VE, which requires virtual mode only available in Cyber 180s.

The emulator is implemented in ANSI C and runs under:

- Microsoft Windows 98/NT/XP
- most modern UNIX systems with X11 and POSIX threads - for example: (Red Hat Linux, FreeBSD, NetBSD, Solaris, MacOS X and Cygwin)

I have created this emulator in the hope that it will create renewed interest in CDC Cyber mainframes and that it helps to preserve our quickly vanishing computing history. I also hope that it will have enough momentum to grow by the efforts of other enthusiasts.

This text attempts to give you an overview on how to install, configure and operate the Desktop Cyber Emulator with a few example CDC operating systems.

2 Installing the Emulator

The Desktop Cyber Emulator is distributed in full source as well as a binaries compiled for Microsoft Windows 98/NT/XP and Red Hat Linux 9.0. The distribution is contained in a ZIP archive created using WinZip version 6.3. The contents of the ZIP archive are listed in the file "archive.txt". The following sections detail the installation steps for different host operating systems.

2.1 Installation under Microsoft Windows 98/NT/XP

To install the emulator software under versions of Microsoft Windows, double-click on the downloaded file "dtcyber20b1.zip" or open the file with WinZip. Then extract the archive to the root directory of your local drive (for this description I assume "C:\"). Make sure you preserve the directory structure by ticking "Use Folder Names".

The sources are located in the directory "C:\DtCyber". To build the executable, you will need MS Visual C++ 6.0 or later. The following steps detail the build process:

- In Windows Explorer double click on the file "C:\DtCyber\DtCyber.dsw" to open the Visual C++ workspace.
- From the menu select "Build | Set Active Configuration | DtCyber - Win32 Release" to activate the optimised release build environment.
- Build the executable by selecting from the menu "Build | Rebuild All".
- Copy the executable "C:\DtCyber\Release\DtCyber.exe" into a work directory (in this description "C:\DtCyber\work".
- Copy the files "C:\DtCyber\cyber.ini" and "C:\DtCyber\Os\*.tap" into the work directory.

2.2 Installation under UNIX systems

The Desktop Cyber Emulator sources will build on most UNIX systems which support X11 and POSIX threads. Depending on the version of UNIX you may need to fine tune some compiler and linker command line flag settings in the makefile to successfully build a working executable.

The downloaded file "dtcyber20b1.zip" file has to be unzipped using tools like "unzip" or "gzip". For this description I assume you unzip the file into your home directory "~/". After unzipping the sources are located in "~/DtCyber". The following steps details the build process in the source directory "~/DtCyber":

- Convert all source files and the Makefile to UNIX format using: dos2unix *.c *.h Makefile.*
- Edit the file "Makefile.x11" to point to the appropriate library and include directories.
- Run "make -f Makefile.x11 all" to build the system.
- Copy the executable "~/ DtCyber/dtcyber" into a work directory (in this description "~/DtCyber/work".
- Copy the files "~/DtCyber/cyber.ini" and "~/DtCyber/Os/*.tap" into the work directory.

3 Configuring the Emulator

When the emulator starts, it reads the file "cyber.ini" which configures the emulation environment. The file typically contains multiple sections of the following form:

```
[SectionName]
; comment line
keyword1=value1         ; trailing comment
keyword2=value2
```

A typical "cyber.ini" is shown in Appendix A.

Each section header is followed by a number of lines specifying parameters to the emulator. By default you need to have the section "[cyber]". This can be overridden by a command line parameter when starting the emulator.

The first section is typically named "[cyber]". It defines some basic emulation parameters such as central memory size, what deadstart panel and equipment sections to use. It has the following parameters:

Keyword	**Values**	**Description**
model	6600 or 173	Chippewa OS needs 6600, everything else needs 173
equipment	string	Section name containing the equipment definitions.
deadstart	string	Name of the section containing the deadstart panel settings.
clock	1 - 10	Clock multiplier used to match real time clock to wall clock.
memory	octal number	Central memory size (1000000 is a good value).
ecsbanks	0, 1, 2, 4, 8 or 16	Number of ECS banks.
pps	12	Number of PPUs (must be 12 in this version)
channels	40	Number of I/O channels (must be 40 in this version)
telnetport	decimal number	TCP port number for terminal mux connections.
telnetconns	1 - 16	Maximum number of TCP connections supported.

The section named by the "equipment" keyword in section "[cyber]" defines the attachment of peripheral equipment to channels, their unit number and in some cases file names. The following shows the general syntax of an entry:

```
<est ordinal>=<devtype>,<unit no>,<channel no>[,<optional file name>]
```

Device Type	**Description**
CO6612	Console: unit 0 on channel 10 - no file name allowed
MT607	7-track tape: normally on channel 5 - file name IS allowed
MT669	9-track tape: normally on channel 13 - file name IS allowed
DD6603	Model 6603 disk drive - file name IS allowed
DD844	Model 844 disk drive - file name IS allowed
CR405	Model 405 card reader - no file name is allowed
LP1612	Model 1612 line printer - no file name is allowed
LP501	Model 501 printer - no file name is allowed
LP512	Model 512 printer - no file name is allowed
MUX6676	Model 6676 terminal multiplexer - no file name is allowed

The section named by the "deadstart" keyword in section "[cyber]" defines the Cyber deadstart panel settings used to load PPU 0 with a short bootstrap routine. The deadstart panel settings are up to 16 octal values (one per line) representing the deadstart switch settings.

4 Running the Emulator

Make sure you have sufficient free space (for example NOS 1 you will need at least 100 MB). The emulator will create a large number of files in your work directory. Depending on your emulated disk configuration you may need several GB of disk space on your host system.

For best performance I recommend that your video card and monitor support a resolution of 1200 x 1024. I also suggest a host system with at least 500 MHz CPU and with 128 MB of RAM or more.

Please note that the configuration file "cyber.ini" controls many aspects of the operation of the emulator. Please study this file carefully. It controls the equipment configuation and deadstart panel as well as other system parameters. You can activate the main configuration sections (for example "smm" and "cos") by renaming either it "cyber" or specifying the section name on the command line.

To exit the emulation at any time, hit ALT-O on the console to enter the operator interface and then type "shutdown" in the text window.

The emulator should be executed within its own work directory. When started from a text window it will print a few startup messages and then launch a graphical console window. The graphical console window is used to emulate a Cyber console. The text window from which the emulator was started, will be used to output critical status messages and also to provide an operator interface (by typing ALT-O in the graphical console window).

The console subsystem's keyboard behaves similar to the real thing except for the following differences:

- Right blank key is mapped to "]"
- Left blank key is mapped to "["
- ALT-O enters the operator interface (use it to shut down, mount tapes and remove paper)
- ALT-0 to ALT-9 toggle instruction tracing for PPU 0 to PPU 9
- ALT-C toggles instruction tracing for the CPU
- ALT-E toggles tracing for Exchange Jumps

Please note that instruction tracing is intended for debugging and generates very large files very quickly! I recommend you only trace around interesting sections. When you activate tracing, the execution speed also dramatically drops.

Additional debug support is provided in the form of in-core disassembly of PPU memory, PPU dumps and CPU dumps. These debugging features are enabled by conditional compilation in the source file "main.c".

Please not that a large number of files created in the directory where the emulator executes are for debug support and may contain instruction traces, dumps and disassemblies. These files will normally be empty unless tracing, dumping or disassembly is enabled.

Apart from the very early Chippewa OS all later CDC Operating Systems are copyrighted by CDC's successor Syntegra. For this reason, I have not been able to include any other OS image with the emulator. If you have access to CDC OS tapes and a 9-track SCSI tape drive, you can create a tape image on disk using the included source file "myread.c".

Below is a short description of the how the get started with a few example CDC OS versions.

4.1 Running Chippewa OS

The Cippewa OS was the first operating system written for the CDC 6600. It was entirely coded in octal with a few comments mixed in. It is rumored that Seymour Cray himself wrote most of it. I was lucky to obtain a copy of the original sources. Because in those days nobody bothered with a copyright statement, I am able to freely distribute this OS with my emulator.

The OS reads very simple card decks which you need to convert from ASCII via the included program "punch026". For example:

```
punch026 cards.ascii
```

After executing "punch026.exe" you should find a file "CR405_C12" which is a hollerith card deck. Note that you have to recreate "CR405_C12" each time because the emulation "eats" the cards.

Before running the emulator, verify that "cyber.ini" and the deadstart tape image "cos.tap" exist in your work directory. To launch the emulator, change to the work directory and then enter:

```
dtcyber cos
```

A few seconds after starting the emulation you should see the Chippewa OS console in your graphical console window.

Type the following in the console window to get job processing started:

```
AUTO.
ON10.
3.DIS.
```

This sequence starts the card reader and printer subsystems ("AUTO.") and the card job you created with "punch026" starts running at control point 3.

With "ON10." and "3.DIS." you call the DIS package to control point 3.

With "DCP." you drop the central processor on that control point and you can see an exchange package (register dump).

With "RCP." you request the central processor and the job resumes.

You should be able to bring up the usual displays such as:

AB.	Dayfile and Control Point Status.
CD.	Memory dumps.
C4,2400.	Dump memory from octal 2400.

Note that you can terminate the emulation any time by hitting ALT-O and then typing "shutdown" in the operator interface.

4.2 Running SMM

SMM is a set of diagnostics, which have been used by Customer Engineers to do fault finding and maintenance of CDC Cyber hardware. The diagnostics had their own version of a mini-OS, which could be deadstarted, and allowed controlled execution of the diagnostics.

Before running the emulator, verify that "cyber.ini" and the deadstart tape image "smm.tap" exist in your work directory. To launch the emulator, change to the work directory and then enter:

```
dtcyber smm
```

The emulation will deadstart and bring up a menu which will allow you to start "CPC" (similar to DIS) by entering "T". Once the "CPC" display comes up enter:

"GB".

From within "CPC" you can launch any of the following toys:

WRM, BAT, ADC, DOG, TEN, EYE, BBP, DRW

You will see that the toys will request the display for a particular PP. Simply enter the number of the PP requesting the display followed by a dot and then hit ENTER (for example "1. <CR>").

Or you could launch the following diagnostics (with varying degrees of success):

ATC, BCD, CH2, CMC, D44, DS1, ECD, EJT, ERX, EXC, FMT, LCW, MAP, MCD, MM2, MXJ, PMM, PSP, PSX, SSP,CU1, CT3.

Note that you can terminate the emulation any time by hitting ALT-O and then typing "shutdown" in the operator interface.

4.3 Running NOS 1

NOS 1 traces its origins back to Chippewa OS, MACE and KRONOS. It provides comparatively modern timesharing features. This text uses NOS 1.3 PSR 472 as an example.

4.3.1 Starting the emulation and NOS 1

Before running the emulator, verify that "cyber.ini" and the deadstart tape image "nos472.tap" exist in your work directory. To launch the emulator, change to the work directory and then enter:

```
dtcyber nos472
```

A few seconds after starting the emulation you should see the DEAD START OPTIONS menu in your graphical console window.

Hit "CR" to start the system.

After a short memory and confidence test, you should see the CMRINST screen with instructions. You can page forward with the "+" key. You can use right blank ("]") to switch to CMRDCK1.

The CMRDCK1 entries match the entries in the "[equipment.nos472]" section of "cyber.ini".

The emulated disks need to be formatted before the NOS system file can be written to the disks**. Note that this needs to be done only once**.

To format the disks type the following lines (only the first time):

```
   INITIALIZE,1,AL.
   INITIALIZE,2,AL.
```

To resume deadstart type:

```
   GO.
```

DSD (Dynamic System Display) appears on the console showing the progress of the system load.

Enter a date when requested (for example 78/01/01.).

Enter the time when requested (for example 00.00.00.)

After the copying of the system from the deadstart tape to disk (and CM) is completed and the library directory has been built, the system will start TELEX, INTRCOM, MAGNET and BATCHIO.

When starting NOS 1 the first time, you will need to create a validation file (VALIDUZ) using the following command entered in DSD. **Note you only need to do this once**.

```
X.GENVAL.
```

To shut down NOS 1 you **MUST** use the following sequence in DSD:

```
UNLOCK.
CHECK POINT SYSTEM.
2.GO.
STEP.
```

To finally shutdown the actual emulation type the following in the text console (operator interface) after having entered it using ALT-O:

```
shutdown
```

The emulation will terminate. If you just terminate the emulation without shutting down NOS properly, subsequent deadstarts may report problems. If that occurs you may find that emulated disks will need to be reformated (using the INITALIZE command).

4.3.2 Enable terminal access

You need to create at least one user to be able to do an interactive login. Use the following command sequence in DSD:

```
X.MODVAL.
K,3.
KK.
K.C,newuser.
K.AW=ALL,AP=ALL.
K.PW=passwrd.
K.END.
K.END.
```

Now you should be able to telnet to the local TCP port 6600, but be patient it takes up to 30 seconds for the login prompt to appear. Login using your newly created "newuser" and password "passwrd". When prompted for the system, enter "BATCH".

4.3.3 Tape and printer handling

The emulation's operator interface may be activated at any time by hitting ALT-O. The operator interface is command line based and prompts for commands in the text window used to launch the emulator.

Commands to try it in the text console (operator interface):

```
help
help load_tape
load_tape 13,1,r,<TAP format file path>
help remove_paper
remove_paper 7,7
end
```

The emulation will resume with a new tape mounted on unit 1 of the tape controller on channel 13, and with paper removed from printer unit 7 on channel 7.

Removing paper means that the file used for print output is closed and then renamed with a date and time stamp. The file can then be viewed or sent to a real printer.

To see the newly mounted tape in DSD, enter:

```
E,T.   (should show the newly mounted tape after a few seconds)
```

To actually itemise the tape contents, create a DIS job:

```
X.DIS.
```

And within DIS if the tape image is unlabelled:

```
VSN,51,MYTAPE.
X.DIS.
LABEL,TAPE,VSN=MYTAPE,LB=KU,F=I.
ITEMIZE,TAPE.
```

Or otherwise if the tape image is labelled:

```
X.DIS.
LABEL,TAPE,VSN=<real label>,LB=KL,F=I.
ITEMIZE,TAPE.
```

To be able to later mount another tape in place of the old one, you have to unload the old one first in DSD:

```
UNLOAD,51.
```

With ALT-O you can later invoke the operator interface again to mount another tape.

5 Utilities

A number of utilities have proven useful when working with the Desktop Cyber Emulator. All utilities are provided in source only and may require some trivial modifications when compiled for non-Windows platforms.

5.1 9-Track Magnetic Tape to TAP Image Conversion

You may have 9-track magnetic tapes, which you would like to access from the Desktop Cyber emulator. The emulator does not allow you to directly work with those tapes, but I have written a small utility which will convert 9-track tapes on a SCSI tape drive under either UNIX or VMS systems into the TAP image file format. The UNIX and VMS sources are:

```
~/DtCyber/tools/mag2tap/mag2tap_unix.c
~/DtCyber/tools/mag2tap/mag2tap_vms.c
```

You may need to do some minor tweaking of include files and the hardcoded device-path of the SCSI tape drive to get these to compile and run on your system.

A UNIX-only utility to write 9-track tapes from a TAP image file is at:

```
~/DtCyber/tools/mag2tap/tap2mag_unix.c
```

These TAP utilities use a hardcoded device-path for the SCSI tape drive. When you run these utilities you only have to supply the path name of the TAP image file. For example:

```
./mag2tap_unix nos14.tap
```

will open the tape device driver and read all records up to EOF (two consecutive tape-marks). For each physical record it reads, it will display the record length and write a TAP record to "nos14.tap".

5.1.1 TAP Image Format

These programs support a very simple form of TAP only. Each physical tape record is represented as the following:

4 byte little-endian record length header (least significant byte first)
n bytes of record contents
4 byte little-endian record length trailer (least significant byte first)

This is repeated for every physical tape record. The record length trailer is useful in an emulation to be able to position n-records backwards. Unfortunately tape-marks are an exception to the format described above. A tape-mark is encoded as 4 bytes of zero (without an actual record contents and trailer part).

During the development of the Desktop Cyber emulator it turned out that a few other variants of TAP do exist. There are some, which make you guess what the actual record length is, others encode some additional info in the record length header and trailer. The Desktop Cyber emulation and associated utilities only support the simple definition given above.

5.2 Convert ASCII to CDC Display Code and back

To be able to manipulate NOS text files within the emulator host environment (Windows or UNIX), I needed conversion utilities between standard ASCII text files and CDC Display Code text files. The Windows source (and Visual C++ 6.0 project files) are located at:

```
C:\DtCyber\tools\dtoa
C:\DtCyber\tools\atod
```

To build under Visual C++ 6.00 simply double click on the workspace file (.dsw) and then click on the build button to create an executable.

The CDC Display Code to ASCII converter "dtoa" has the following command line:

```
dtoa <CDC Display Code file>
```

It reads the CDC Display Code file and converts its contents to ASCII and writes the result to standard output. So typically you would redirect the output to a file.

The ASCII to CDC Display Code converter has the following command line:

```
atod <ASCII file> <CDC Display Code file>
```

It reads the ASCII file, does the conversion and then writes the result to the CDC Display Code file.

5.3 Create NOS I-Format TAP Image

To be able to import files into a running NOS system, the files have to be converted into a NOS I-format tape. The source file for a utility, which performs this conversion, is located at:

C:\DtCyber\tools\nosiwrite

To build under Visual C++ 6.00 simply double click on the workspace file (.dsw) and then click on the build button to create an executable.

The utility converts the specified number of files into a NOS I-format TAP image with one logical record per file. Files to be converted have to have a file name with the following format:

ir0001, ir0002, … ir9999

If individual files are missing, then the utility writes a zero-length logical record. Note that this is not a zero length TAP record (or tape-mark), but a 6 byte TAP record containing only the NOS I-format block terminator.

The utility uses the following command line:

```
nosICreate <TAP output files> [optional files count]
```

If no file count is specified, the utility converts only one file.

5.4 Verify NOS I-Format TAP Image

Sometimes it may become necessary to verify that a TAP image is indeed containing a valid NOS I-format tape. The source file for such a utility is located at:

C:\DtCyber\tools\nosiverify

To build under Visual C++ 6.00 simply double click on the workspace file (.dsw) and then click on the build button to create an executable.

The utility takes a TAP image file, which contains a NOS I-format tape, and dumps each physical record to STDOUT. The fields in the dump for every record are:

1. record length
2. block terminator
3. 10 PP words (12 bit each) octal dump
4. Display Code representation of those 10 PP words

If records are invalid or inconsistent, an appropriate message is displayed.

Use the following command line to check an image:

```
nosIVerify <TAP image with NOS I-format records>
```

5.5 Punching Card Decks

Chippewa OS has no other means to input jobs then Hollerith card decks. Walter Spector has written a utility to convert ASCII text files to virtual punch cards, which can be used to feed card jobs into Chippewa OS running under the emulator. The source file for the utility is located at:

```
C:\DtCyber\tools\punch026
```

To build under Visual C++ 6.00 simply double click on the workspace file (.dsw) and then click on the build button to create an executable.

Use the following command line to "punch" a card deck:

```
punch026 <ASCII card deck>
```

The utility reads the ASCII text file (max 80 characters per line) and converts it into a Hollerith card deck with the file name " CR405_C12".

By dragging and dropping this file into the work directory of the Desktop Cyber emulator, you place the card deck into the card reader. Please note that the card reader will delete the card deck, once it has been read.

Appendix A - Example Inititialisation File

```
[cyber]
model=173
deadstart=deadstart.nos13
equipment=equipment.nos13
clock=3
memory=1000000
ecsbanks=4
pps=12
channels=40
telnetport=6600
telnetconns=4
trace=0

[equipment.nos13]
; eq=type,unitNo,channel
01=DD844,0,01,dd884_1_0_nos13x
02=DD844,0,02,dd884_2_0_nos13x
10=CO6612,0,10
05=CR405,0,11
07=LP501,7,07
50=MT669,0,13,nos13.tap
51=MT669,1,13
32=MUX6676,0,05

[deadstart.nos13]
7513    ; DCN 13
3007    ; LDD 07
3415    ; STD 15
1500    ; LCN 00
0000    ; PSN 00
7713    ; FNC 13,0260
0260    ; ...
7413    ; ACN 13
7113    ; IAM 13,6400
6400    ;
0011    ; wxxy w=libdeck, x=cmrdeck, y=dsopts
0000    ; rpss r=recovery, p=-cej,-cpu1,-cpu0, ss=sysondevbits
```

Appendix B - Acknowledgments

I would like to express my thanks to the following people who have encouraged me and helped in various ways:

Wanee Hunter
Clare Johnstone
David Webb
Douglas Quebbeman
Jeff Woolsey
Walter Spector
Tony Epton
Paul Repacholi
John Zabolitzky
Freddy Meerwaldt
James Wiley
John Laird
Kent Olsen
John Gibbins
Steve Peltz
Dave Mausner
Dennis Henriksen
Jeffrey Katcher
Gerard van der Grinten
Mike Arrington
Joe Cychosz
Tim Smart (Syntegra)
Barry Murphy (Syntegra)
Peter Bartsch
Phillip Draughon
Ken Hunter

Please note that this list is probably incomplete and in no particular order. Sorry if I forgot you.

A special thank you goes to the man who was responsible for the design of Control Data Corporation's most successful large-scale computer, the CDC 6600 system.

.

Seymour Cray

Seymour Cray was born in 1925 in Chippewa Falls, Wisconsin; he died in 1996 in Colorado Springs, Colorado, from injuries suffered in a car accident.