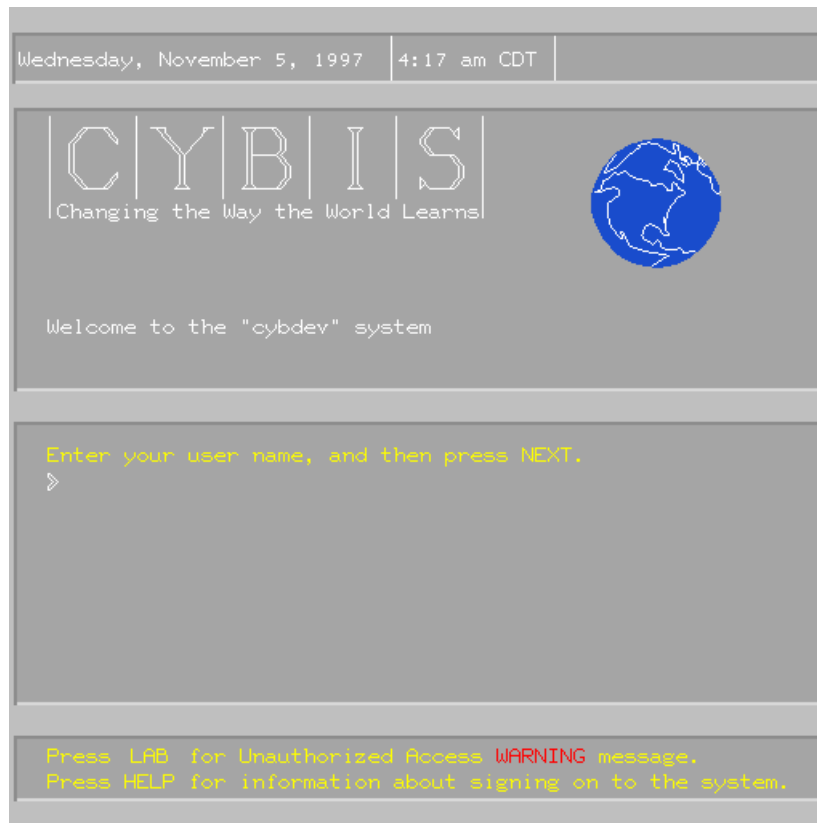


CYBIS Operation and Usage Manual

Quick Reference



Copyright © 2016-2017 Tom Hunter and Paul Koning

January 5, 2017

Table of Contents

Introduction	5
Installing the CYBIS Package	6
Starting NOS and CYBIS.....	7
Starting and Ending an Interactive CYBIS Session.....	13
Starting Pterm	13
Logging into CYBIS.....	14
Logging out of CYBIS	16
Shutting down CYBIS and NOS.....	17
Shutting down CYBIS.....	17
Shutting down NOS.....	20
Introduction to CYBIS.....	23
CYBIS Keyboard	23
Interesting CYBIS Lessons	24
Information about CYBIS.....	24
Games	24
System Administration and Operator Utilities.....	27
Other	27
Writing Tutor code in CYBIS.....	27
Operator Interface	28
Build CYBIS Binaries	30
Interactive Terminal Access to NOS.....	30
Important NOS User Names and Passwords.....	31
Frequently Asked Question and Tips and Tricks	32
Why can't I use the current date when I start up NOS?	32
What is the difference between a system administrator and an author?	32
What Kind of System Help is Available?	32
Why is there a leading "0" (Zero) in front of many lessons?	32
What is a "master file"?	32
What is ECS/UEM used for in CYBIS"?	32
What is the CYBIS CONSOLE program?	32
How do CYBIS and NOS relate to each other?	33
How does CYBIS communicate with Pterm or real terminals like a CDC IST II?	33
What is CSTC?	33

How does CYBIS know which master files are available?	33
How does CYBIS start?	33
Where are the CYBIS sources?	34
What are "binary" type master files?	34
What are "backup" type master files?	34
What protocol is used by the CYBIS terminal connection?	34
What are all these "n" and "o" prefixed versions of CYBIS lesson files?	34
Why does it ask for a "security code" when I try to run "ipedit" or "allocate"?	35
How to create a data file like "acclog1"?	36
How to reload (restart) all CYBIS subsystems (versus a shutdown)?	36
How do I find all required master files?	36
Has Pterm got some trace feature?	36
What is "Development mode"?	36
How to create the first ever Group "s" user (information only)?	36
How to find the list of used master files?	36
What are "general" type versus "master" type master files?	37
How to edit installation parameters when lesson "ipedit" is not working?	37
What is a Site in the CYBIS context?	37
How to disable the default lesson (information only)?	38
What are well known CYBIS groups?	38
What is a CYBIS account?	38
How to list all CYBIS accounts?	38
What is "system backout" in CYBIS?	39
How to edit lesson "notes" settings?	39
How do I print anything?	39
How to switch between "X.CONSOLE." modes?	39
What are these shifted letter shortcuts on the Author mode page?	39
Are a "data set" and a "data file" identical?	39
Why do many lessons have number prefixes which collide with shortcuts?	40
Why do we have so many files starting with "s0" or "a0"?	40
Why do I have to enter DATA (Ctrl-D in Pterm) to start a lesson?	40
How to determine the free space in masterfiles?	40
Where do new files go when no master file is specified?	40
How to edit a lesson which starts when I enter NEXT after the lesson name?	40

How can I learn Tutor to write my own CYBIS lessons?.....	40
What is the “use” statement in Tutor code used for?	40
What are the rules for “codewords” of lessons included with “use”?	41
How to set the “lesson access classes” for an account?.....	41
How to recondense a lesson?	41
Does the condenser produce a binary even if there are errors?.....	41
How to find (and possibly edit) the lesson component that just aborted?.....	41
What is a “micro table”?	42
What is lesson “runnersys”?	42
What type of variables are supported by Tutor?.....	42
What is “common”?	42
What is “storage”?	42
Why when editing non-text blocks do changes appear in multiple blocks?	43
How to export files from CYBIS to NOS?	43
How to edit the Notes Index?	43
How to edit and test lesson “plato”?	43
How to debug lessons?	43
How to creates “pnotes” for a new system group?.....	44
Why doesn’t SHIFT-STOP work on my CDC IST II or III?	44
What is the “echo” output command in the ASCII protocol?	44
What is “GOGO” in the CYBIS context?	44
What are “system lessons”?	45
What are the standard PLATO text entry features?	46
What are the important user types in CYBIS?	46
What are naming restrictions?	47

Introduction

CYBIS (CYber-Based Instructional System) started life as PLATO (Programmed Logic for Automatic Teaching Operations) and was developed at CERL (Computer-based Education Research Laboratory) at the University of Illinois. Later PLATO became a commercial product marketed by CDC. The rights to the PLATO name were sold and CDC rebranded the product CYBIS. Eventually University Online which became VCampus acquired CYBIS from CDC. VCampus is no longer in business and its assets (including CYBIS) now belong to Nat Kannan, its former CEO.

I have been able to secure a copy of CYBIS for Controlfreaks. Paul Koning and I got permission from Nat Kannan to use and share this material under the condition that it won't be resold or used for any commercial purpose whatsoever. This condition applies to the CYBIS software, tools and course content.

With the generous help of Paul Koning I was able to create a "canned" ready-to-run package containing a deadstart tape, disk images, emulator initialization file, Desktop CYBER emulator binaries for various platforms and emulator sources.

This manual describes the installation, operation, system administration and use of this system.

A big **THANK YOU** to Paul Koning and Nat Kannan for making this possible. Thanks also to Gerard van der Grinten for a copy of SES and help with its use to build the CYBIS sources.

Installing the CYBIS Package

- 1) Login to <http://www.controlfreaks.org/> and navigate to “CYBIS Release”. Download the file md5sum.txt which contains an “md5sum” hash for all other files to give some reassurance that the files have not been corrupted during transfer. Check the hash for all subsequently downloaded files.
- 2) Download the CybisRelease1.zip package and unzip it on a drive with at least 8 GB free disk space. The package contains a NOS 2.8.7 deadstart tape image with CYBIS, disk images, an empty directory for persisting memory images and the Desktop CYBER 5.5.0 initialisation file “cyber.ini”.
- 3) There are pre-build binaries of Desktop CYBER 5.5.0 for Windows, Linux or MacOS X. If your system matches either of these download the DtCYBER.binaries.zip package and copy the appropriate binary into the CybisRelease1 directory.
- 4) If none of the supplied binaries are suitable, download the DtCYBER.source.zip package. Extract the sources and build an emulator binary from scratch. Alternatively you can also checkout the source from the Subversion repository using the following command:

```
svn co http://www.controlfreaks.org/DtCyber/trunk
```

Build as appropriate and then copy the binary into the CybisRelease1 directory.

- 5) Install a version of the PLATO/CYBIS terminal emulator. There are two options:
 - a) Download PTERM_V5.0.8.zip from the “CYBIS Release” page and run the appropriate installation program for your platform (Windows Linux or MacOS X). These are the versions I tested with.
 - b) Get the latest version from <https://cyber1.org/pterm.asp>.

Starting NOS and CYBIS

Once the CYBIS disk images, deadstart tape image, "cyber.ini" and the emulator binary have been extracted and/or built and PTERM has been installed the system is ready to run.

Use the following steps to start up the NOS 2.8.7 operating system and then as a second step launch the CYBIS subsystems.

Start the Desktop CYBER emulator. Under Windows the firewall will prompt you if you want to allow the emulator to communicate on the network. Tick the appropriate network types (at least private networks) and then click on "Allow access".

The emulator creates two windows (colours depend on host OS):

- the operator window (examples show white characters on black background);
- the CDC console window (green characters on black background);

The operator window is used to mount and un-mount tapes, show what tapes are mounted, remove printed paper, load and remove punch cards and to shut down the emulation. It allows the user to interface to the emulated peripherals in the same way as an operator of a real system.

The emulated CDC console keyboard behaves like the real thing except for the following differences:

- Left blank key is mapped to "[" and erases any command line which may have been entered.
- Right blank key is mapped to "]" and its use depends on context. Often it is used to cycle through a set of console screens.

The screen shots below are showing examples of both windows:

```
C:\CybisRelease1\DtCyberWin32.exe

Desktop CYBER 5.5.1 - Copyright (C) Tom Hunter
Licensed under the terms of the GNU General Public License version 3

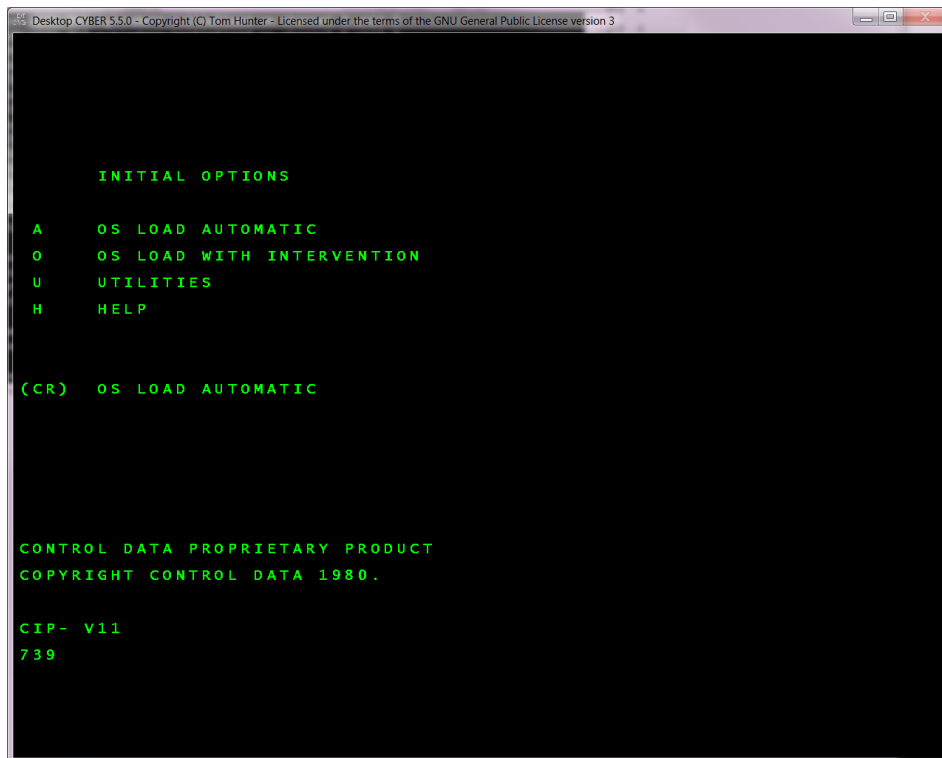
Starting initialisation
CPU model CYBER865 initialised (CM: 4000000, ECS: 0)
PPs initialised (number of PPU's 24)
Channels initialised (number of channels 40)
Using QueryPerformanceCounter() clock at 2.648623 MHz
Status/Control Register initialised on channel 16
Status/Control Register initialised on channel 36
Disk with 843 cylinders initialised on channel 1 unit 0
Disk with 843 cylinders initialised on channel 2 unit 0
Disk with 843 cylinders initialised on channel 3 unit 0
Disk with 843 cylinders initialised on channel 4 unit 0
Disk with 843 cylinders initialised on channel 1 unit 1
Disk with 843 cylinders initialised on channel 2 unit 1
Disk with 843 cylinders initialised on channel 3 unit 1
Disk with 843 cylinders initialised on channel 4 unit 1
Disk with 843 cylinders initialised on channel 1 unit 2
Disk with 843 cylinders initialised on channel 2 unit 2
Disk with 843 cylinders initialised on channel 3 unit 2
Disk with 843 cylinders initialised on channel 4 unit 2
Console initialised on channel 10
Equipment 07, Unit 00 attached to DCC6681 on channel 7
LP3555/512 initialised on channel 7 equipment 7
Equipment 07, Unit 00 attached to DCC6681 on channel 11
CR3447 initialised on channel 11 equipment 7
Equipment 06, Unit 00 attached to DCC6681 on channel 11
CP3446 initialised on channel 11 equipment 6
MT679 initialised on channel 13 equipment 0 unit 0
MT679 initialised on channel 13 equipment 0 unit 1
MT679 initialised on channel 13 equipment 0 unit 2
MT679 initialised on channel 13 equipment 0 unit 3
Two port MUX initialised on channel 15
NPU initialised on channel 5 equipment 7

Desktop CYBER 5.5.1 - Copyright (C) Tom Hunter.
Licensed under the terms of the GNU General Public License version 3.
For details see included text file 'license-gpl-3.0.txt' or visit
'http://www.gnu.org/licenses'.

Operator interface
Please enter 'help' to get a list of commands

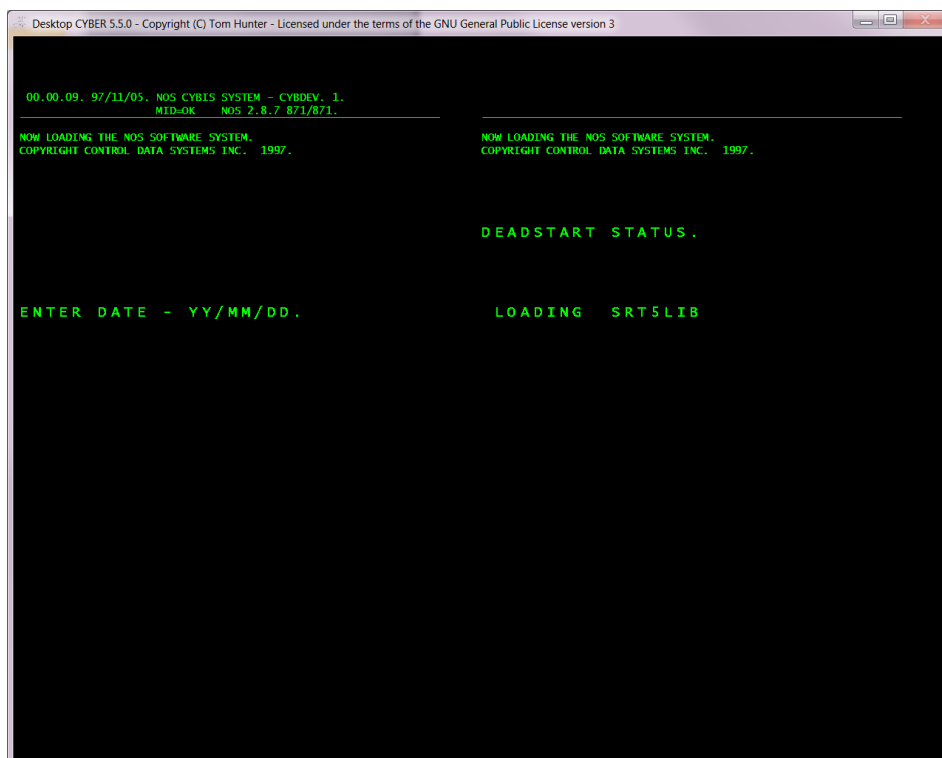
Operator>
```

Desktop CYBER Operator Window



Desktop CYBER Console Window

To load the NOS 2.8.7 operating system, simply press CR (Enter on modern keyboards) in the console windows. The screen will show the following:



The right hand side will update as the various NOS components are loaded until the following shows:

```
Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

00.01.46. 97/11/05. NOS CYBIS SYSTEM - CYBDEV. 1.
MID=OK NOS 2.8.7 871/871.

NOW LOADING THE NOS SOFTWARE SYSTEM.
COPYRIGHT CONTROL DATA SYSTEMS INC. 1997.

DEADSTART STATUS.

ENTER DATE - YY/MM/DD. COMPLETE
```

Here you have the option to enter the current date and then the time. As neither NOS nor CYBIS are fully Y2K compliant and parts of CYBIS make assumptions about valid date ranges I recommend to simply press Enter (effectively 97/11/05). Alternatively dates between 97/11/05 and 99/01/01 are suitable. After a few seconds you should see the following:

```
Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

A. SYSTEM DAYFILE. B,A. SYSTEM STATUS. NAM. REQUEST *K* DISPLAY
00.09.47. 97/11/05. NOS CYBIS SYSTEM - CYBDEV. 1. SEE *A,OPERATOR*
MID=OK NOS 2.8.7 871/871.

00.09.36. AAAEN. RFL(23000)
00.09.36. AAAEN. NS(NIN=099,FDP=YES,RT=YES,MC=500)NLF=NLF
FILE
00.09.36. AAAEN. BUILT 97/11/05. 02.34.23.
00.09.36. NAM X. NV/ 00.09.36.APPLICATION NETTED 0
N - CS
00.09.36. NAM X. NV/ 00.09.36.APPLICATION NETTED 0
N - TVF
00.09.36. AAAEN. NS TRYING NETON.
00.09.37. AAAEN. CS NETON SUCCESSFUL.
00.09.37. NAM X. CS/ 00.09.36. VER 1.8 - 871.
00.09.37. NAM X. CS/ 00.09.36. NCF 97/11/05. 03.24
-27.
00.09.37. NAM X. CS/ 00.09.36. DTCYBER NETWORK
13
00.09.37. AAACN. CPU MS REQD 0.020
00.09.37. AAACN. SETJOB(DC=NO) DISCARD ALL JOB OUTPUT.
00.09.37. AAACN. EXIT. COLP
00.09.37. AAACN. OUT(*OP=E)
00.09.37. AAACN. UNLOAD(*OP=0)
00.09.37. NAM X. NV/ 00.09.37.APPLICATION NETTED 0
N - NS
00.09.38. NAM X. EST 060 - NIP/REGL DN= 1,SN= 2,RL=3.
00.09.38. NAM X. NS/ 00.09.38.VER 1.8 - 871.
00.09.38. NAM X. NS/ 00.09.38.NCF 97/11/05. 03.24.
27.
00.09.38. NAM X. NS/ 00.09.38. DTCYBER NETWORK
00.09.38. NAM X. NS/ 00.09.38.NLF 97/11/07. 10.32.
24.
00.09.38. NAM X. NP/NPUA 00.09.38.NPU NPUA ,AC,002
00.09.38. NAM X. NP/NPUA 00.09.38. SUPERVISION GAINED
00.09.38. NAM X. NP/NPUA 00.09.38. CCP VERSION 301
00.09.38. NAM X. NP/NPUA 00.09.38. PREVIOUS CS NODE 0
00. PREVIOUS NS NODE 000
00.09.44. NAM X. NV/ 00.09.44.APPLICATION NETTED 0
N - IAF
00.09.44. IAF X. NETWORK CONNECTED.
```

Now wait about 5 seconds for the NAM networking to finish initialising (i.e. the last line on the left says “IAF X. NETWORK CONNECTED.”) otherwise CYBIS may not start correctly.

Now type “X.CYBIS.” followed by Enter:

```
Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

A. SYSTEM DAYFILE.                                     B,A. SYSTEM STATUS.      NAM. REQUEST *K* DISPLAY
00.14.32. 97/11/05. NOS CYBIS SYSTEM - CYBDEV. 1.    SEE *A,OPERATOR*
MID-OK NOS 2.8.7 871/871.

00.09.36. AAAEN. RFL(23000)                            CP JSN SC PR FL CPU STATUS
00.09.36. AAAEN. NS(NIN=099,FDP=YES,RT=YES,MC=500)NLF=NLF 1 IAF X 76 603 X NETWORK CONNECTED.
FILE 2 NAM X 77 330 X REQUEST *K* DISPLAY
00.09.36. AAAEN. BUILT 97/11/05. 02.34.23. 3
00.09.36. NAM X. NV/ 00.09.36.APPLICATION NETTED 0 4
N - CS 5
00.09.36. NAM X. NV/ 00.09.36.APPLICATION NETTED 0 6
N - TYT 7
00.09.36. AAAEN. NS TRYING NETON. 10
00.09.37. AAAEN. CS NETON SUCCESSFUL. 11
00.09.37. NAM X. CS/ 00.09.36. VER 1.8 - 871. 12
00.09.37. NAM X. CS/ 00.09.36. NCF 97/11/05. 03.24 13
-27. 14
00.09.37. NAM X. CS/ 00.09.36. DTCYBER NETWORK 15
16
00.09.37. AAACN. CPU MS REQD 0.020 17
00.09.37. AAACN. SETJOB(DC=NO) DISCARD ALL JOB OUTPUT. 20
00.09.37. AAACN. EXITT. COLP 21
00.09.37. AAACN. OUT(*OP=E) 22
00.09.37. AAACN. UNLOAD(*OP=E) 23
00.09.37. NAM X. NV/ 00.09.37.APPLICATION NETTED 0 24
N - NS 25
00.09.38. NAM X. EST 060 - NLP/REGL DN= 1,SN= 2,RL=3. 26
00.09.38. NAM X. NS/ 00.09.38.VER 1.8 - 871. 27
00.09.38. NAM X. NS/ 00.09.38.NCF 97/11/05. 03.24. 28
27. 29
00.09.38. NAM X. NS/ 00.09.38. DTCYBER NETWORK 30
31
00.09.38. NAM X. NS/ 00.09.38.NLF 97/11/07. 10.32. 32
24. 33
00.09.38. NAM X. NP/NPUA 00.09.38.NPU NPUA ,AC,002 34
00.09.38. NAM X. NP/NPUA 00.09.38. SUPERVISION GAINED 35
00.09.38. NAM X. NP/NPUA 00.09.38. CCP VERSION - 301 36
00.09.38. NAM X. NP/NPUA 00.09.38. PREVIOUS CS NODE 0 37
00. PREVIOUS NS NODE 000 38
00.09.44. NAM X. NV/ 00.09.44.APPLICATION NETTED 0 39
N - IAF 40
00.09.44. IAF X. NETWORK CONNECTED. 41

X. CYBIS.
```

After about 30 seconds it will show:

```
Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

A. SYSTEM DAYFILE.                                     B,A. SYSTEM STATUS.      NAM. REQUEST *K* DISPLAY
00.16.21. 97/11/05. NOS CYBIS SYSTEM - CYBDEV. 1.    SEE *A,OPERATOR*
MID-OK NOS 2.8.7 871/871.

00.16.10. AAAMS. CBLTH=21. CONDENSOR SOURCE BUFFER CP JSN SC PR FL CPU STATUS
00.16.10. AAAMS. NETMS=16. NETWORK SYSTEM TABLE SI 1 IAF X 76 603 X NETWORK CONNECTED.
ZE 2 NAM X 77 330 X REQUEST *K* DISPLAY
00.16.10. AAAMS. PWNOT=10. MISSED PASSWORD GENERAT 3 MASI S 76 44 X CYBIS
ES NOTE 4 PLAI S 73 1231 X LOCAL SYSTEM LESSONS INSTALLED
CMP=ON. TURN ON CENTRAL PLATO E 5 FBR1 S 74 130 X FRAMAT
RECUTOR 6 PNII S 74 411 X 0 TERMINALS ACTIVE.
00.16.12. PLAIS. UNIT 0 BINARY1 BINARY D 7 COAI S 72 554 X CONDEN.
00.16.12. PLAIS. UNIT 1 BINARY2 BINARY D 10
00.16.12. PLAIS. UNIT 2 SYSTEM MASTER A 11
00.16.12. PLAIS. UNIT 3 SOFILES GENERAL C 12
00.16.12. PLAIS. UNIT 4 SOSTUFF GENERAL C 13
00.16.12. PLAIS. UNIT 5 SOSUP GENERAL C 14
00.16.12. PLAIS. UNIT 6 SYSTEM1 GENERAL C 15
00.16.12. PLAIS. UNIT 7 SYSTEM2 GENERAL C 16
00.16.12. PLAIS. UNIT 8 SYSTEM3 GENERAL C 17
00.16.12. PLAIS. UNIT 9 SYSTEM4 GENERAL C 20
00.16.12. PLAIS. UNIT 10 SYSTEM5 GENERAL C 21
00.16.12. PLAIS. UNIT 11 DEVELOP MASTER A 22
00.16.12. PLAIS. UNIT 12 PUBA MASTER A 23
00.16.12. PLAIS. UNIT 13 PUBB MASTER A 24
00.16.12. PLAIS. UNIT 14 PUBC MASTER A 25
00.16.12. PLAIS. UNIT 15 PUBD MASTER A 26
00.16.13. PLAIS. UNIT 16 PUBE MASTER A 27
00.16.13. PLAIS. UNIT 17 PUBF MASTER A 28
00.16.13. PLAIS. UNIT 18 PUBG MASTER A 29
00.16.13. PLAIS. UNIT 19 PUBH MASTER A 30
00.16.13. PLAIS. UNIT 20 PUBI MASTER A 31
00.16.13. PLAIS. UNIT 21 PUBJ MASTER A 32
00.16.13. PLAIS. NV/ 00.16.13.APPLICATION NETTED 0 33
N - CYBIS 34
00.16.13. PNIIIS. NETON COMPLETE. 35
00.16.15. PLAIS. RUNNING IN ESM MODE 36
00.16.15. PLAIS. DEVELOPMENT SYSTEM 37
00.16.16. PLAIS. INSTALLING LOCAL SYSLES MOOS 38
00.16.16. PLAIS. LOCAL SYSTEM LESSONS INSTALLED 39
```

Wait until the last message on the left hand side says "LOCAL SYSTEM LESSONS INSTALLED".

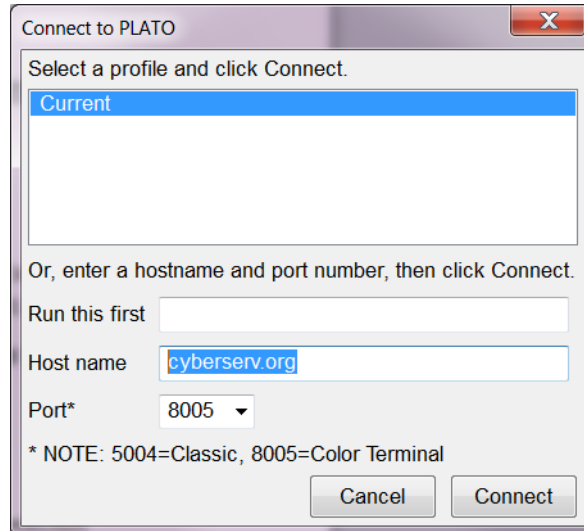
On the right hand side the following CYBIS subsystems should be running: MAS1, PLA1, FOR1, PNI1 and COA1.

If not all subsystems have started, then shutdown CYBIS as described in the chapter "Shutting down CYBIS" and if necessary drop any remaining CYBIS subsystems and possible dump jobs using "DROP,jsn.". After all CYBIS subsystems have shutdown try starting CYBIS again with "X.CYBIS.".

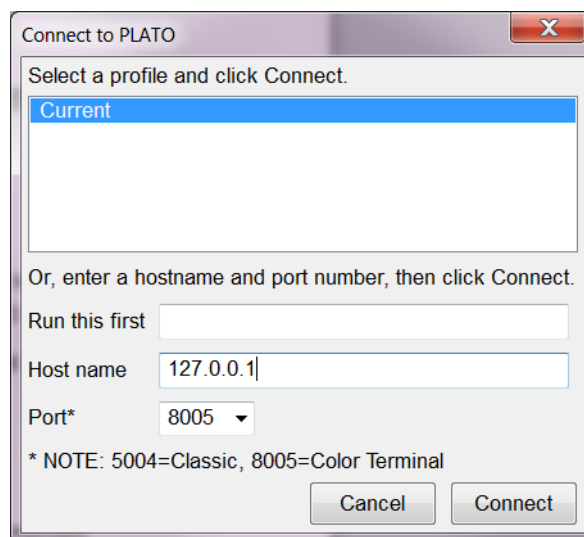
Starting and Ending an Interactive CYBIS Session

Starting Pterm

To start interactive CYBIS sessions launch the CYBIS/PLATO terminal emulation (Pterm). It will prompt you for the connection parameters:



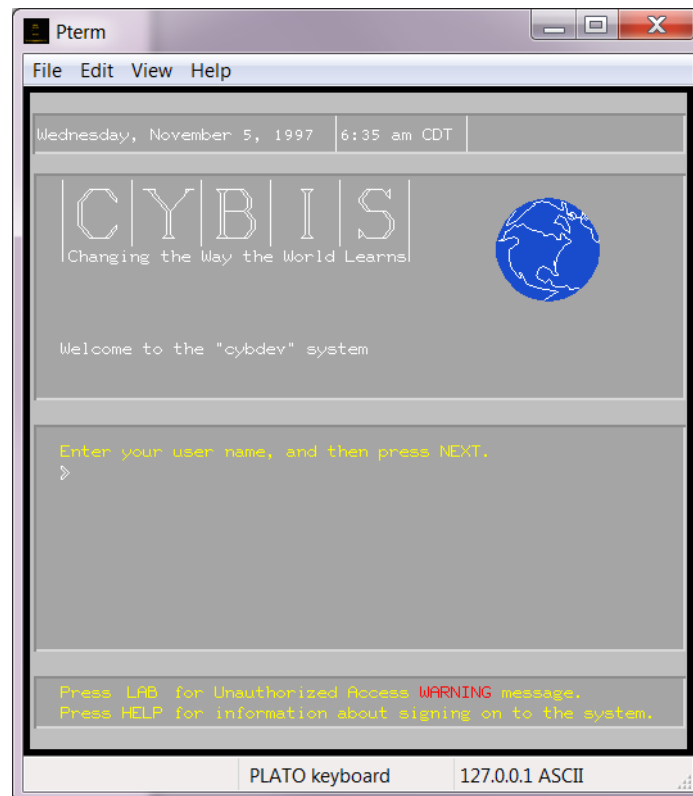
Change the “Host name” parameter from “cyberserv.org” to 127.0.0.1 to connect to your local host. Make sure that the “Port” parameter is 8005 (this is the TCP port number on which Desktop CYBER is expecting connections for CYBIS). PTERM will remember the new IP address for subsequent sessions.



Finally click on “Connect” to establish a Pterm session to CYBIS. After a few seconds the Pterm screen changes to the CYBIS login screen.

Logging into CYBIS

The CYBIS login screen prompts you for a valid user name, group and password:



The following administrator type users are already setup in Group "s" with user names: "admin", "admin1", "admin2" and "admin3".

Additionally there are also the following author type users available in Group "author" with user names: "author", "author1", "author2" and "author3". The author type users are all under account "cybdeva".

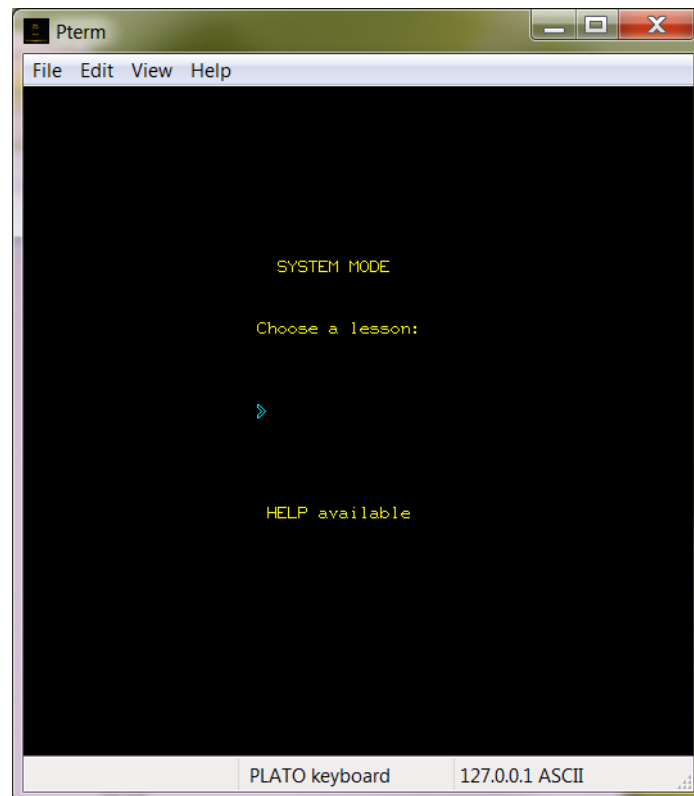
All of these users have the same password "passme".

The ultimate super user is "admin" which owns and has access to everything on this CYBIS system.

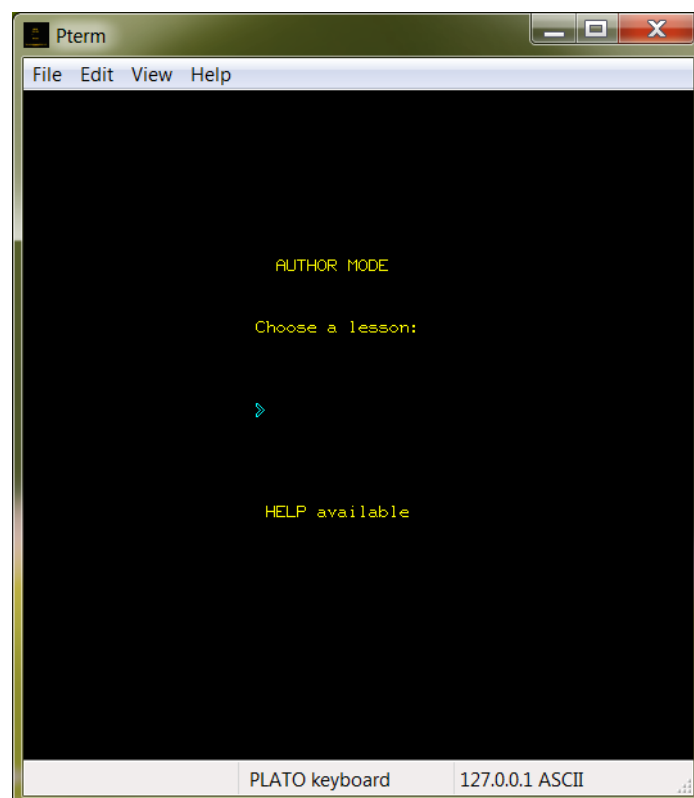
To login enter one of these user names and then follow the prompts to enter the associated Group name and the password.

To see the list of keyboard mappings for all PLATO/CYBIS special keys click on "Help" on the menu bar and then select "Pterm keyboard".

For an administrator type account (Group “s”) you will be presented with the following “System Mode” screen:



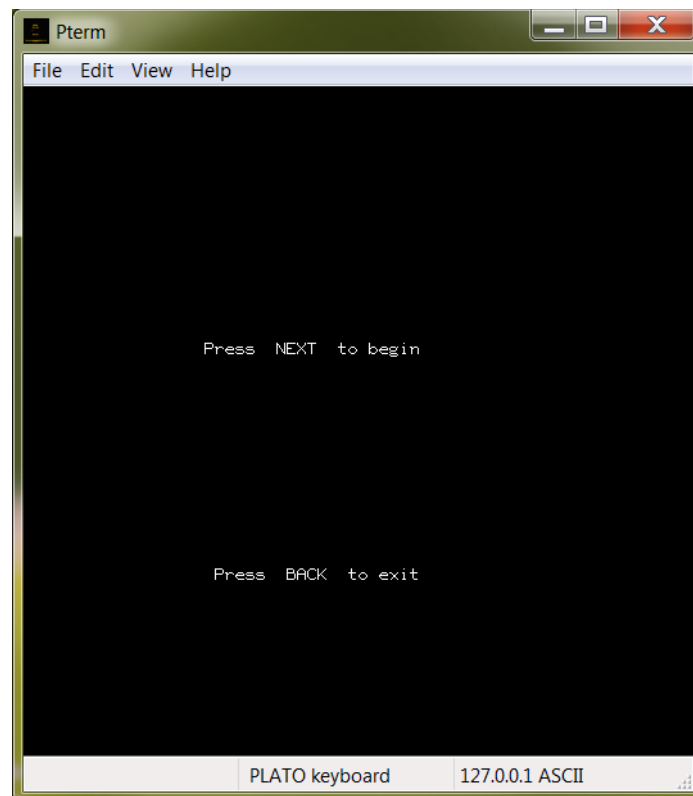
For an author type account (Group “author”) you will be presented with the following “Author Mode” screen:



In either mode you can run most CYBIS lessons. “System Mode” has some additional privileges and is required for some system maintenance and operator tasks (e.g. CYBIS shutdown or creation of new accounts).

Logging out of CYBIS

To end a CYBIS terminal session press SHIFT-STOP to end the currently running CYBIS application if necessary and then press SHIFT-STOP again to logout from the current session:



If you press NEXT on this screen it will take you back to the original CYBIS login screen.

Waiting too long appears to trigger a bug in Pterm (or CYBIS) causing it to no longer respond to the NEXT key so that the CYBIS login screen won't reappear. In this case simply close the Pterm window and start a new instance of it.

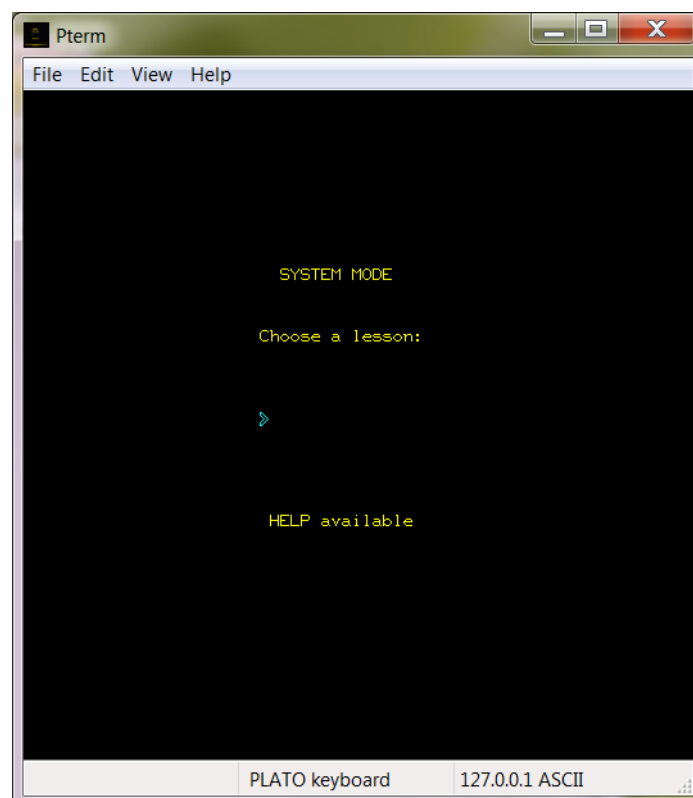
Shutting down CYBIS and NOS

Before you can stop Desktop CYBER you have to cleanly shut down first CYBIS and then the NOS operating system. If the system is not cleanly shut down as described below, NOS disks and CYBIS master files may be left in an inconsistent state and subsequent runs may fail.

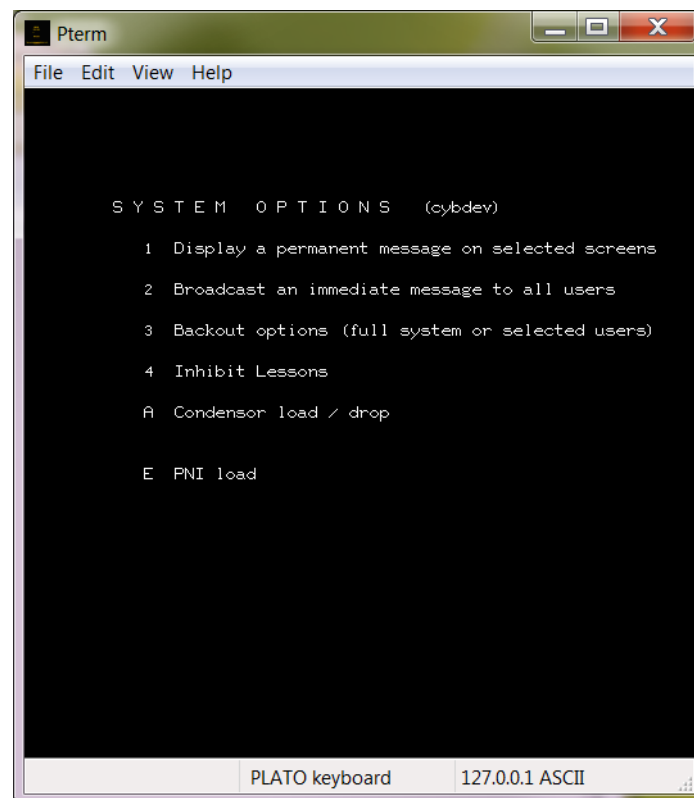
Shutting down CYBIS

The CYBIS shutdown procedure is documented in the operations manual. If not performed correctly the file system is so simple that it is not likely to be affected. What may be affected are lessons that use "common" - shared ECS for data or communication - which is checkpointed back to disk at times such as on shutdown.

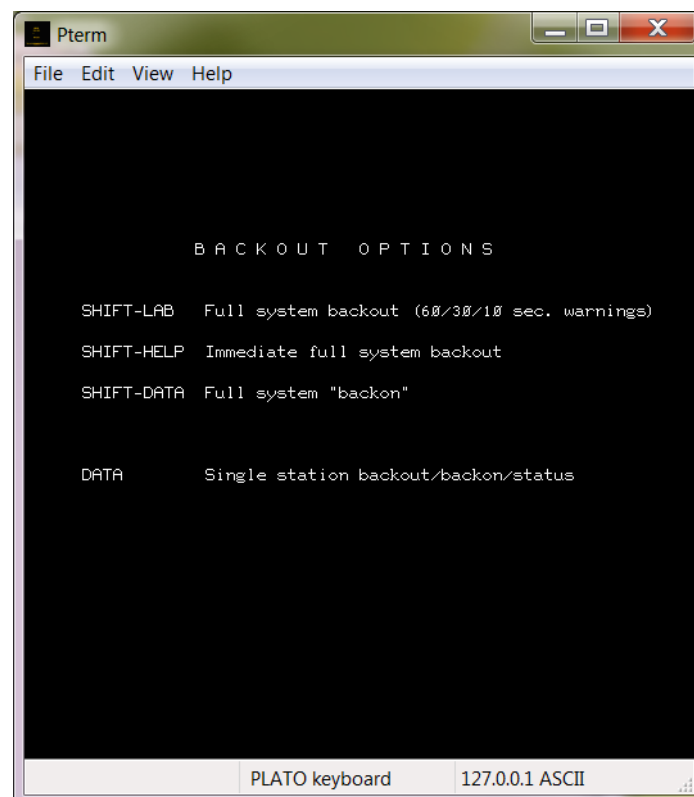
To shut down CYBIS first you have to sign in as one of the administrators under Group "s" so that you are at the "System Mode" screen:



Now press "1" to bring up the System Options screen:



Select "3 Backout options" and the screen changes to:



Press SHIFT-HELP for immediate full system backout. After one or two seconds the screen changes back to System Options. Press SHIFT-STOP twice to exit and log off.

On the console screen the display shows “Full System Backout Completed”:

```

Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

A.  SYSTEM DAYFILE.                                B,A.  SYSTEM STATUS.      NAM. REQUEST *K* DISPLAY
05.48.05. 97/11/05. NOS CYBIS SYSTEM - CYBDEV. 1.  SEE *A,OPERATOR*
MID-OK   NOS 2.8.7 871/871.

05.46.56. PLAIS. UNIT 2  SYSTEM MASTER A          CP JSN SC PR FL CPU STATUS
05.46.56. PLAIS. UNIT 3  SFTFILES GENERAL C        1 IAF X 76 350 X NETWORK CONNECTED.
05.46.56. PLAIS. UNIT 4  SOSTUFF GENERAL C          2 NAM X 77 332 X REQUEST *K* DISPLAY
05.46.57. PLAIS. UNIT 5  S0SUP GENERAL C            3 MAS1 S 76 44 X CYBIS
05.46.57. PLAIS. UNIT 6  SYSTEM1 GENERAL C          4 PLA1 S 73 1231 X FULL SYSTEM BACKOUT COMPLETED.
05.46.57. PLAIS. UNIT 7  SYSTEM2 GENERAL C          5 FOR1 S 74 130 X FRAMAT
05.46.57. PLAIS. UNIT 8  SYSTEM3 GENERAL C          6 PNI1 S 74 411 X
05.46.57. PLAIS. UNIT 9  SYSTEM4 GENERAL C          7 COA1 S 72 11 X
05.46.57. PLAIS. UNIT 10 SYSTEM5 GENERAL C         10
05.46.57. PLAIS. UNIT 11 DEVELOP MASTER A         11
05.46.57. PLAIS. UNIT 12 PUBA MASTER A             12
05.46.57. PLAIS. UNIT 13 PUBB MASTER A             13
05.46.57. PLAIS. UNIT 14 PUBC MASTER A             14
05.46.57. PLAIS. UNIT 15 PUBD MASTER A             15
05.46.57. PLAIS. UNIT 16 PUBE MASTER A             16
05.46.57. PLAIS. UNIT 17 PUBF MASTER A             17
05.46.57. NAM X. NV/ 05.46.57.APPLICATION NETTED 0 20
N - CYBIS                                         21
05.46.58. PLAIS. UNIT 18 PUBG MASTER A             22
05.46.58. PLAIS. UNIT 19 PUBH MASTER A             23
05.46.58. PNI15. NETON COMPLETE.                 24
05.46.58. PLAIS. UNIT 20 PUBI MASTER A             BIO X 70 2 IDLE.
05.46.58. PLAIS. UNIT 21 PUBJ MASTER A             MAG X 76 35 X MAGNET.
05.47.00. AABDS. EDITING COMPLETE.                SYS S 100 0 SYSTEM DATE 97/11/05.
05.47.00. AABDS. UNLOAD(SYSTEM)
05.47.00. AABDS. ENDIE(CONDEN)
05.47.00. AABDS. REVERT.
05.47.00. AABDS. MODE(1)
05.47.00. AABDS. RFL(40000)
05.47.00. AABDS. *DIS.
05.47.00. AABDS. IFE,( NE. ),COND.
05.47.00. AABDS. ELSE(COND)
05.47.00. AABDS. CONDEN.
05.47.00. PLAIS. RUNNING IN ESM MODE
05.47.00. PLAIS. DEVELOPMENT SYSTEM
05.47.00. PLAIS. INSTALLING LOCAL SYSLES MOOS
05.47.00. PLAIS. LOCAL SYSTEM LESSONS INSTALLED
05.47.44. PLAIS. FULL SYSTEM BACKOUT COMPLETED.

```

Enter “K,MAS1.” to bring up the K-Display for Mastor and then enter “K.STOP” to shutdown CYBIS:

```

Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

K.      MAS1                                B,A.  SYSTEM STATUS.      NAM. REQUEST *K* DISPLAY
05.52.20. 97/11/05. NOS CYBIS SYSTEM - CYBDEV. 1.  SEE *A,OPERATOR*
MID-OK   NOS 2.8.7 871/871.

MASTOR K-DISPLAY

SUBMIT FILE = CYBIS

SECUR      = OFF

K. STOP

CP JSN SC PR FL CPU STATUS
1 IAF X 76 350 X NETWORK CONNECTED.
2 NAM X 77 332 X REQUEST *K* DISPLAY
3 MAS1 S 76 44 X CYBIS
4 PLA1 S 73 1231 X FULL SYSTEM BACKOUT COMPLETED.
5 FOR1 S 74 130 X FRAMAT
6 PNI1 S 74 411 X
7 COA1 S 72 11 X
10
11
12
13
14
15
16
17
20
21
22 BIO X 70 2 IDLE.
23 MAG X 76 35 X MAGNET.
24 SYS S 100 0 SYSTEM DATE 97/11/05.

```

After a few seconds all CYBIS subsystem control points have terminated and the console display shows:

```
Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

K.          MASI          B,A.  SYSTEM STATUS.      NAM.  REQUEST *K* DISPLAY
05.34.26.  97/11/05.  NOS CYBIS SYSTEM - CYBDEV. 1.  SEE *A,OPERATOR*
          MID=OK      NOS 2.8.7 871/871.

JSN NOT FOUND

CP  JSN  SC  PR  FL  CPU      STATUS
1  IAF  X   76  350  X  NETWORK CONNECTED.
2  NAM  X   77  332  X  REQUEST *K* DISPLAY
3
4
5
6
7
10
11
12
13
14
15
16
17
20
21
22  BIO  X   70   2   IDLE.
23  MAG  X   76  35  X  MAGNET.
24  SYS  S  100   0   SYSTEM DATE  97/11/05.

K .
```

Shutting down NOS

Now enter left-blank (i.e. “]”) and “AB.” to get rid of the stale K display and then shut down NAM with the command “IDLE,NAM.”. After a few seconds of activity NAM has shut down and the NAM Dayfile has printed on the BIO control point. The console screen shows:

```
Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

A.  SYSTEM DAYFILE.      B,A.  SYSTEM STATUS.
05.38.19.  97/11/05.  NOS CYBIS SYSTEM - CYBDEV. 1.  SEE *A,OPERATOR*
          MID=OK      NOS 2.8.7 871/871.

05.38.12.  AAPN.  EXIT.  NS
05.38.12.  AAAON.  IF (.NOT. FILE (TRCLEV3,AS)) DEFINE (TRCLEV3=
          NVT0100)
05.38.12.  AAAON.  DEFINE (TRCLEV3=NVT0100)
05.38.12.  AAAON.  SKIPEI (TRCLEV3)
05.38.12.  AAAON.  COPYEI (TRCLEV2,TRCLEV3)
05.38.12.  AAAON.  FILE NOT FOUND - TRCLEV2.
05.38.12.  AAAON.  PURGE (ZZNV100/NA)
05.38.12.  AAAON.  IF (FILE (ZZZZZDN,AS),NTRCLV1)
05.38.12.  AAAON.  RENAME (TRCLEV1=ZZZZZDN)
05.38.12.  AAAON.  REWIND (TRCLEV1)
05.38.12.  AAAON.  IF (500.NE.0) SKIPR (TRCLEV1)
05.38.12.  AAAON.  SKIPR (TRCLEV1)
05.38.12.  AAAON.  COPYBF (TRCLEV1,TRCLEV3)
05.38.12.  AAAON.  EOI ENCOUNTERED.
05.38.12.  AAAON.  EOI. 1 FILE 2 RECORDS 784 WORDS.
05.38.12.  AAAON.  BKSP (TRCLEV3)
05.38.12.  AAAON.  SKIPR (TRCLEV3)
05.38.12.  AAAON.  IF (.NOT. FILE (TRCLEV3,EOF)) WRITEF (TRCLE
          V3)
05.38.12.  AAAON.  ENDOF (NTRCLV1)
05.38.12.  AAAON.  RETURN (TRCLEV1,TRCLEV2,TRCLEV3)
05.38.12.  AAAON.  ENDIF (NOTRACE)
05.38.12.  AAAON.  ATTACH (NVFLST=NVL0100/NA,M=W)
05.38.12.  AAAON.  IF (.NOT. FILE (NVFLST,AS)) DEFINE (NVFLST=NV
          L0100)
05.38.12.  AAAON.  DEFINE (NVFLST=NVL0100)
05.38.12.  AAAON.  SKIPEI (NVFLST)
05.38.12.  AAAON.  NOTE (DFL,NR)/NVDA100
05.38.12.  AAAON.  DAYFILE (DFL)
05.38.12.  AAAON.  USER DAYFILE PROCESSED.
05.38.12.  AAAON.  PACK (DFL)
05.38.12.  AAAON.  PACK COMPLETE.
05.38.12.  AAAON.  COPYEI (DFL,NVFLST)
05.38.12.  AAAON.  EOI ENCOUNTERED.
05.38.12.  AAAON.  EOI. 0 FILES 1 RECORD 435 WORDS.
05.38.12.  AAAON.  SETJOB (DC=ND)
05.38.12.  AAAON.  EXIT.  NVF
05.38.19.  IAF X.  WAITING FOR NETWORK.

CP  JSN  SC  PR  FL  CPU      STATUS
1  IAF  X   76  350  X  WAITING FOR NETWORK.
2
3
4
5
6
7
10
11
12
13
14
15
16
17
20
21
22  BIO  X   70   2   IDLE.
23  MAG  X   76  35  X  MAGNET.
24  SYS  S  100   0   SYSTEM DATE  97/11/05.
```

Now enter “UNLOCK.” followed by “CHECK POINT SYSTEM.” (the command auto-completes after a few characters):

```
Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

A. SYSTEM DAYFILE.                                UNLOCK
06.02.30. 97/11/05. NOS CYBIS SYSTEM - CYBDEV. 1.
MID=OK NOS 2.8.7 871/871.

05.58.12. AAAON. DEFINE(TRCLEV3=NVTO100)
05.58.12. AAAON. SKIPEI(TRCLEV3)
05.58.12. AAAON. COPYEI(TRCLEV2,TRCLEV3)
05.58.12. AAAON. FILE NOT FOUND - TRCLEV2.
05.58.12. AAAON. PURGE(ZZNV100/NA)
05.58.12. AAAON. IF(FILE(ZZZZZDH,AS),NTRCLV1)
05.58.12. AAAON. RENAME(TRCLEV1=ZZZZDH)
05.58.12. AAAON. REWIND(TRCLEV1)
05.58.12. AAAON. IF(500.NE.0)SKIPR(TRCLEV1)
05.58.12. AAAON. SKIPR(TRCLEV1)
05.58.12. AAAON. COPYOF(TRCLEV1,TRCLEV3)
05.58.12. AAAON. EOI ENCOUNTERED.
05.58.12. AAAON. EOI. 1 FILE 2 RECORDS 784 WORDS.
05.58.12. AAAON. BKSP(TRCLEV3)
05.58.12. AAAON. SKIPR(TRCLEV3)
05.58.12. AAAON. IF(.NOT.FILE(TRCLEV3,EOF)) WRITEF(TRCLE
V3)
05.58.12. AAAON. ENDIF(NTRCLV1)
05.58.12. AAAON. RETURN(TRCLEV1,TRCLEV2,TRCLEV3)
05.58.12. AAAON. ENDIF(NOTRACE)
05.58.12. AAAON. ATTACH(NVFLST=NVLO100/NA,M=M)
05.58.12. AAAON. IF(.NOT.FILE(NVFLST,AS))DEFINE(NVFLST=NV
LO100)
05.58.12. AAAON. DEFINE(NVFLST=NVLO100)
05.58.12. AAAON. SKIPEI(NVFLST)
05.58.12. AAAON. NOTE(DFL,NR)/NVDA100
05.58.12. AAAON. DAYFILE(DFL)
05.58.12. AAAON. USER DAYFILE PROCESSED.
05.58.12. AAAON. PACK(DFL)
05.58.12. AAAON. PACK COMPLETE.
05.58.12. AAAON. COPYEI(DFL,NVFLST)
05.58.12. AAAON. EOI ENCOUNTERED.
05.58.12. AAAON. EOI. 0 FILES 1 RECORD 435 WORDS.
05.58.12. AAAON. SETJOB(DC=ND)
05.58.12. AAAON. EXIT. NVF
05.58.19. IAF X. WAITING FOR NETWORK.
06.00.00. SYS S. SYSTEM DATE 97/11/05.
06.01.19. SYS S. DS, UNLOCK.

B,A. SYSTEM STATUS.
SEE *A,OPERATOR*

CP JSN SC PR FL CPU STATUS
1 IAF X 76 350 X WAITING FOR NETWORK.
2
3
4
5
6
7
10
11
12
13
14
15
16
17
20
21
22 BIO X 70 2 IDLE.
23 MAG X 76 35 X MAGNET.
24 SYS S 100 0 SYSTEM DATE 97/11/05.

CHECK POINT SYSTEM.
```

After a few seconds the checkpoint has completed:

```
Desktop CYBER 5.5.0 - Copyright (C) Tom Hunter - Licensed under the terms of the GNU General Public License version 3

A. SYSTEM DAYFILE.                                UNLOCK
06.03.18. 97/11/05. NOS CYBIS SYSTEM - CYBDEV. 1.
MID=OK NOS 2.8.7 871/871.

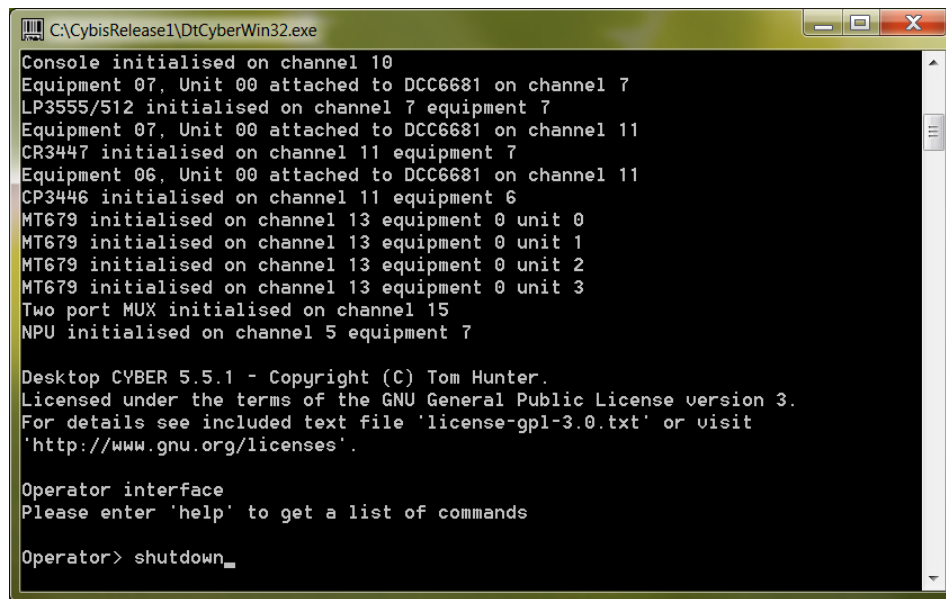
05.58.12. AAAON. DEFINE(NVFLST=NVLO100)
05.58.12. AAAON. SKIPEI(NVFLST)
05.58.12. AAAON. NOTE(DFL,NR)/NVDA100
05.58.12. AAAON. DAYFILE(DFL)
05.58.12. AAAON. USER DAYFILE PROCESSED.
05.58.12. AAAON. PACK(DFL)
05.58.12. AAAON. PACK COMPLETE.
05.58.12. AAAON. COPYEI(DFL,NVFLST)
05.58.12. AAAON. EOI ENCOUNTERED.
05.58.12. AAAON. EOI. 0 FILES 1 RECORD 435 WORDS.
05.58.12. AAAON. SETJOB(DC=ND)
05.58.12. AAAON. EXIT. NVF
05.58.19. IAF X. WAITING FOR NETWORK.
06.00.00. SYS S. SYSTEM DATE 97/11/05.
06.01.19. SYS S. DS, UNLOCK.
06.02.47. SYS S. DS, CHECK POINT SYSTEM.
06.02.47. IAF X. SYSTEM CHECKPOINT ABORT.
06.02.47. IAF X. IAF REPRIVED.
06.02.47. IAF X. OPERATOR DROP.
06.02.47. IAF X. IAFEX2.
06.02.47. IAF X. TERMINATION IN PROGRESS.
06.02.47. IAF X. IAFX 0.001 KILO-FL DECREASES.
06.02.47. IAF X. IAFX 0.104 PERCENT CPU USAGE.
06.02.47. IAF X. IAF TERMINATED.
06.02.47. IAF X. IFE,FILE(OUTPUT,AS),DUMP.
06.02.48. IAF X. ENDIF(DUMP)
06.02.48. IAF X. IF,EF,EQ,0,SET,EFG=1.
06.02.48. IAF X. ENDM(LOOP)
06.02.48. IAF X. WHILE((EFG.NE.0).AND.(EF.LT.SPE).AND.(
SW2.EQ.0),LOOP)
06.02.48. IAF X. ENDM(LOOP)
06.02.48. IAF X. IFE,FILE(ZZZZOUT,AS),ZOUT.
06.02.48. IAF X. ENDIF(ZOUT)
06.02.48. IAF X. REVERT. IAF END.
06.02.48. BIO X. SUBSYSTEM ABORTED.
06.02.48. BIO X. EXIT.
06.02.49. SYS S. CHECKPOINT COMPLETE.

B,A. SYSTEM STATUS.
SEE *A,OPERATOR*

CP JSN SC PR FL CPU STATUS
1
2
3
4
5
6
7
10
11
12
13
14
15
16
17
20
21
22
23
24 SYS S 100 0 CHECKPOINT COMPLETE.
```

The final console command is “STEP.” to stop all PP activity and any further disk accesses. **THIS IS IMPORTANT – DON’T FORGET IT.**

The NOS operating system is now fully shut down. To terminate the emulation go to the operator window and type “shutdown” and then press the Enter key:



```
C:\CybisRelease1\DtCyberWin32.exe
Console initialised on channel 10
Equipment 07, Unit 00 attached to DCC6681 on channel 7
LP3555/512 initialised on channel 7 equipment 7
Equipment 07, Unit 00 attached to DCC6681 on channel 11
CR3447 initialised on channel 11 equipment 7
Equipment 06, Unit 00 attached to DCC6681 on channel 11
CP3446 initialised on channel 11 equipment 6
MT679 initialised on channel 13 equipment 0 unit 0
MT679 initialised on channel 13 equipment 0 unit 1
MT679 initialised on channel 13 equipment 0 unit 2
MT679 initialised on channel 13 equipment 0 unit 3
Two port MUX initialised on channel 15
NPU initialised on channel 5 equipment 7

Desktop CYBER 5.5.1 - Copyright (C) Tom Hunter.
Licensed under the terms of the GNU General Public License version 3.
For details see included text file 'license-gpl-3.0.txt' or visit
'http://www.gnu.org/licenses'.

Operator interface
Please enter 'help' to get a list of commands

Operator> shutdown_
```

The operator window closes a few seconds after “shutdown” has been entered. All buffers are flushed to disk and CM and PPU memories and tape controller conversion tables are saved.

Introduction to CYBIS

This Quick Reference is not a complete guide to CYBIS but barely skims the surface and reflects the author's superficial knowledge of CYBIS. It aims to get the user of this CYBIS release started. It only covers "System Mode" and "Author Mode" and ignores Student and Instructor Modes.

Other material available from Bitsavers and/or Controlfreaks websites will have much more depth.

Included with this release are two text files "Plato Installation Manual.txt" and "Plato Operator Guide.txt". Both provide detailed background information for administrators and operators.

CYBIS Keyboard

The CYBIS keyboard has a set of special keys not found on modern PC keyboards. For Pterm key mappings on modern keyboards click on Help and then select "Pterm keyboard". As a mnemonic, most (but not all) of these function keys are accessible as control-x where x is the first letter of the function key name (for example, control-d for the DATA key). Note also that all the function keys except for TERM come in shifted variants. For example, control-shift-S will produce the SHIFT-STOP key.

Sometimes in documentation or help text you will see a "1" after the key name to indicate the shifted version of the key. For example STOP1 represents SHIFT-STOP.

Here is an incomplete list of some of the most important keys:

CYBIS Key	Description
NEXT	Pressing NEXT is similar to Enter on modern keyboards and is typically used to indicate to CYBIS that you have finished typing an answer to a prompt.
SHIFT	The SHIFT key is always used in combination with another key by holding down the SHIFT key and while still holding it down pressing another key. It allows you to enter upper case characters, but also allows numeric, punctuation and function keys to have to values. Of special interest are some of the "shifted" function keys like SHIFT-NEXT or SHIFT-STOP or SHIFT-DATA.
SHIFT-STOP	This key combination is used typically used to exit out of a CYBIS lesson or when in the System Mode or Author Mode display to log out of CYBIS.
HELP	Pressing HELP provides online help in most (but not all) contexts. If the help extends of multiple pages, then NEXT will page forward and BACK will page backward. Normally after the last page of help the CYBIS lesson redisplay the original page HELP was pressed in.
DATA	When you enter a CYBIS lesson name on the System Mode or Author Mode screen the DATA key executes the lesson. In this context the NEXT key invokes the editor for the lesson (but still gives you the option to execute the lesson by pressing DATA). This takes some time to get used to if you are new to System or Author Mode in CYBIS. I still often press NEXT instead of DATA when trying to run a lesson.
BACK	Pressing BACK typically takes you back one step, but is also often used to browse backwards (e.g. in Notes files).

Interesting CYBIS Lessons

There is a range of CYBIS lessons for everyone. Invoke a lesson by typing the lesson name in the System or Author Mode screen followed by DATA.

You will see a warning from many lessons which have not yet been “condensed” saying in red: “NO BINARY – Lesson must be condensed”. It then suggests to press DATA to condense or NEXT to choose a new lesson. This is a known issue with this version. Simply press DATA and the lesson will condense and run normally.

Information about CYBIS

These lessons provide general help about CYBIS, how to write Tutor lessons, the NOS operating system, how to operator CYBIS etc.:

“Ointrotop”	Introduction to keyboard and function keys
“Ointrotut”	Introduction lesson to the TUTOR language
“Onotesintr”	Introduction to using notes
“Okeyboard”	Introduction to using the PLATO keyboard
“aids”	Main documentation repository
“sysaids”	Documents system commands available only to administrators
“s0ascers”	Documents the ASCII protocol between CYBIS and Pterm

Games

CYBIS has some iconic games:

“Oempire”	Star Trek like game.
“Oemphelp”	Help for “Oempire”
“ 2avat”	Avatar (note the leading space)
“Oairfight”	3D simulation of a dogfight among jet fighters
“Oadgame”	Advertising game
“Oaerogames”	Aerospace engineering games
“Oants”	Aumbers game
“Oareneg”	Chemistry game
“Obackgam”	Backgammon
“Obagels”	Numbers guess game
“Obattleshi”	Sea Battle – attempt to destroy Plato’s fleet
“Obees”	Bee hive
“Obingo”	Game of bingo
“Obiocytes”	Biorythm
“Occttt”	Tic Tac Toe
“Ocheckers”	Checkers

"0cocos"	The coconut story
"0concentra"	Concentration games
"0contract"	Contract bridge
"0crball"	Baseball betting
"0crosswdn"	Crossword puzzle
"0darts"	Darts
"0deutsch"	The trucking game
" 2dice"	Dice game
"0dogfight"	Dog fight game
"0drib"	Fractions basketball
"0edl"	Horse race
"0fishwar"	Air war simulation
"0freecell"	Game of cards
"0fun"	Maths game
"0hangman"	Ordeal of a hangman
"0hangspy"	Hang a spy
"0hifive"	A game of chance and skill
"0hunt"	Deer hunt
"labyrinth"	Labyrinth
"2mahjongg"	Mahjongg
"mahjongg"	Mahjongg
"0mate"	Checkmate
"0mazewar"	Maze war
"0mlcnim"	NIM game
"0moonbattl"	Game of arithmetic knowledge and speed
"0moonwar"	Multiplayer shooting game
"0moria"	Moria
"0mreact"	Maths drill game
"0musgame2"	Key spinner game
"0musgame8"	Music concentration game
"2nova"	Star destroyer
"0obs"	Obstacle course
"0pind"	Decimal pinball
"0pinw"	Pinball

"0playgo"	Game of GO
"0pogo"	Game of splash
"0port"	Space port
"0pzk"	Tank war
"0p106con1"	Vectors and kinematics contest
"0racetrack"	Racetrack
"0racing"	Racing games
"0react"	Reaction game using touch panel
"0sea"	Torpedo numbers game
"0solitaire"	Card game with touch panel
"0swat"	
"0syng"	Synthesis race
"0syng2"	Synthesis race (another one)
"0tictac"	Tic Tac Toe
" 2tkm"	The Kings Mission game
"0tokens"	Token Solitaire
"0tricks"	Dr. Lobo's psychic experiments
"0ttt"	Tic Tac Toe
"0tubs"	Pick a tub (maths game)
"0vegas"	Learn to play Keno
"0wallstree"	Invest in securities
"wilderness"	Fantasy simulation
"0wmg04"	Probability game
"0ychess1"	Chess program

System Administration and Operator Utilities

"operator"	system account/file options
"accounts"	main lesson for editing CYBIS accounts
"s"	manage Group "s" (signons are created/changed here)
"author"	manage Group "author" (signons are created/changed here)
"ipedit"	system operations parameters
"u"	system utilities menu
"user"	list of users active on the system

Other

"catalogs"	shows a list of CYBIS courseware
"prints"	prints a CYBIS file on a NOS printer – wait for the print to finish (BIO is idle and the NOS DSD I-display shows LQ030 is idle) and then remove the listing via the Operator Interface command "rp 7,7".
"search"	search files for strings.

Writing Tutor code in CYBIS

To help new authors to get started I have created 2 minimal lessons ("mytut1" and "mytut2") under account "cybdeva". To edit and use these lessons sign in with one of the "author" type accounts ("author", "author1", "author2" or "author3") using group "author" and password "passme".

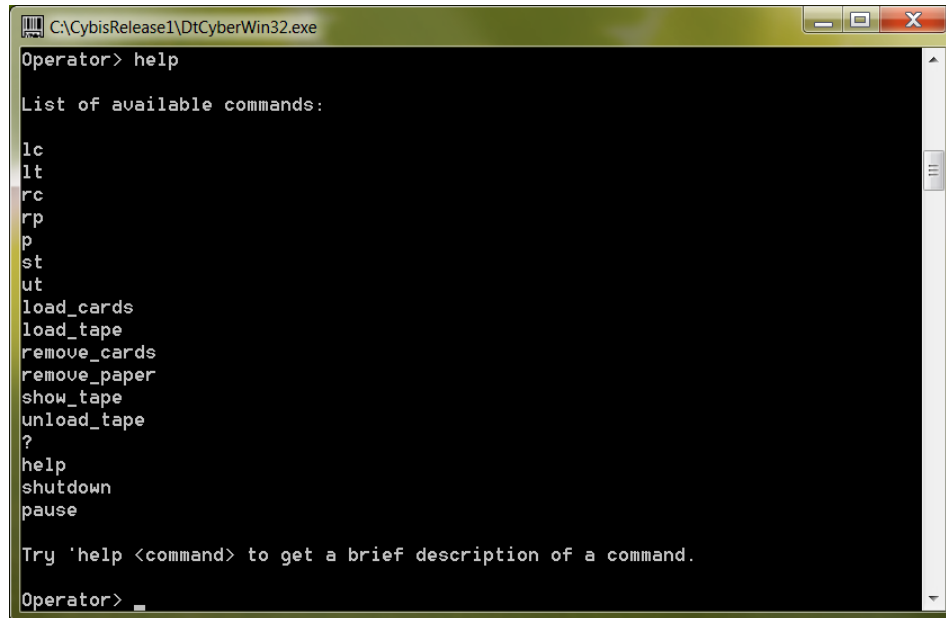
Once signed in at the author prompt type "mytut1" (or "mytut2") and NEXT to edit the lesson. Press HELP to learn about editing. To condense and run the lesson from within the edit environment press SHIFT-STOP. To run the lesson from the author mode prompt type the lesson name and then press DATA.

To get started with Tutor programming I recommend working through "Introduction to Tutor" and for more in-depth coverage "The Tutor Language 1978".

To create more lessons, sign on as user "admin", group "s" and password "passme" and at the System Mode prompt type the account name "cybdeva" followed by NEXT. Then NEXT for "file management options" and then "1" for "Create a file". Select option "a" to create a lesson, type the file name and NEXT and then again NEXT when prompted for "Enter masterfile" (don't care) and "5" when prompted for "Enter number of parts". This will give you a reasonable size lesson file to experiment with.

Operator Interface

The operator interface is command line based and prompts for commands in its text window. Note that case matters. All commands have to be followed by the Enter key to execute the command. You can type in “help” or “?” to find out what commands are available:



```
C:\CybisRelease1\DtCyberWin32.exe
Operator> help

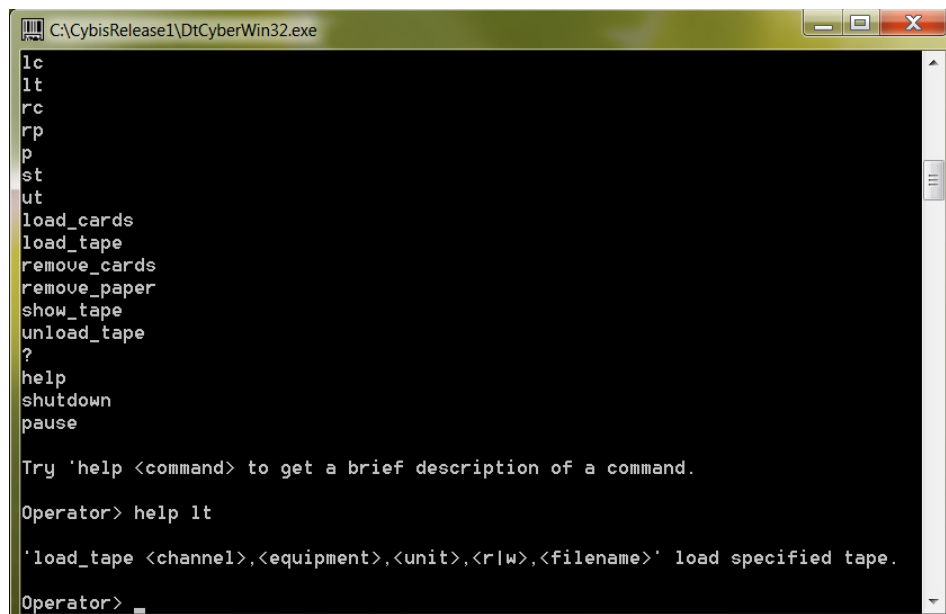
List of available commands:

lc
lt
rc
rp
p
st
ut
load_cards
load_tape
remove_cards
remove_paper
show_tape
unload_tape
?
help
shutdown
pause

Try 'help <command> to get a brief description of a command.
Operator> _
```

All commands have an abbreviated version of only one or two letters. For example “lt” is the same as “load_tape”, or “st” is the same as “show_tape”.

To get help for the parameters of the individual commands type help and the command name. For example:



```
C:\CybisRelease1\DtCyberWin32.exe
lc
lt
rc
rp
p
st
ut
load_cards
load_tape
remove_cards
remove_paper
show_tape
unload_tape
?
help
shutdown
pause

Try 'help <command> to get a brief description of a command.
Operator> help lt

'load_tape <channel>,<equipment>,<unit>,<rlw>,<filename>' load specified tape.
Operator> _
```

Most commands require parameters which identify the peripheral you want to interact with using the following types of information:

- channel number
- equipment number
- unit number (only for tape drives)

Some peripherals require additional parameters:

- file name (card reader and tape drives)
- read/write mode flag does the same as the write enable ring for a real tape drive ('w' is equivalent to ring in; 'r' is equivalent to ring out)

Some commands (e.g load_tape or load_card) required a file name specifying what file to use. The path name of the file **must not** contain spaces.

To shut down the emulation type the following in the operator window:

shutdown

The emulation will terminate cleanly and flush all pending buffers to disk. Do not kill the emulation by any other means.

Build CYBIS Binaries

It is not possible to build CYBIS in an interactive terminal session. Instead to perform a build of CYBIS modules you have to use the NOS console (i.e. DIS). The full build of all CYBIS modules takes about 15 minutes with Desktop CYBER running on an Intel Core I5-6400 with a clock speed of 2.7 GHz.

Enter the following console commands:

```
X.DIS.  
USER,PLATO,PASSME.  
GET,XSES.  
XSES.  
.      (the full stop or period enters AUTO mode and starts processing)  
GET,BALL871.  
BALL871.  
.      (enters AUTO mode and start processing)
```

Be patient or even better go out for lunch.

Once the build has finished the binaries are available as the following direct access permanent files under user "plato":

```
CBIN  
FBIN  
MBIN  
PBIN  
PNIBIN  
KRMDBIN (debug version of PNI)  
PPUBIN  
UBIN
```

For testing these binaries can be inserted into the running system using SYSEDIT (shut down CYBIS first). Use LIBEDIT to create a new deadstart tape.

The build PROC for PNI is BPNI871.

Interactive Terminal Access to NOS

This system provides full interactive terminal access to NOS via IAF and NAM by connecting to TCP port 6610.

For Windows users I recommend to download the modified version of TeraTerm from the Controlfreaks website (under topic "Desktop CYBER Emulator"). This version of Teraterm supports full screen editing via FSE. To use FSE with TeraTerm enter the NOS command: "SCREEN,TTERM." before running FSE.

Important NOS User Names and Passwords

The following user names and passwords are important:

User name	Password	Contents of user catalog
sys	passme	CYBIS subsystem startup (only one indirect access file "CYBIS").
cybismf	cybismf	This user is UI 377773 and only System Origin jobs have access to it (e.g. via DIS). No terminal access is possible. The IST load files are kept under this user on the family disks. Also all master files are under this user on various disk packs accessed via PACKNAM (see PROC MFNX on the deadstart tape).
plato	passme	CYBIS sources and build PROCs. "BALL871" does a full system build.
ses	passme	SES build system with the CYBIL related utilities.
netadmn	netadmn	Network Definition Language (NDL) and related material. "NDL18" is the currently used version. Build the NDL via PROC NDLJOB.
install	install	Some of the installation and source OPL files (e.g. OPL871).
thunter	passme	Terminal Definition File (TDF) source for the version of TeraTerm on the Controlfreaks website. The source is "STTERM". The compiled and active version is "TTERM". Compile with the NOS TDU command. To use FSE with TeraTerm enter the following NOS command: "SCREEN,TTERM."

Frequently Asked Question and Tips and Tricks

The following are questions and answers to CYBIS installation and administration issues. Many of them were raised in emails by the author with replies from Paul Koning. These are in chronological order starting with the earliest questions first.

Why can't I use the current date when I start up NOS?

Both NOS and CYBIS were developed well before the year 2000 so their chosen date representations do not support years beyond 1999. There are official patches from CDC to "fix" the problem in NOS but in reality the "fix" just means that the system could be used for a bit longer with real dates and in a few years would break again. CYBIS too could be fixed but it would take some effort.

For a historic computer system it is reasonable to run with dates in the past, so I recommend using dates between 97/11/05 and 99/12/31 (or just pressing Enter when prompted for the date during deadstart).

What is the difference between a system administrator and an author?

A system administrator is an author with special privileges. The differences come from the group name ("s" rather than a regular group). The "SYSTEM MODE" screen (as opposed to "AUTHOR MODE" screen) is a recent change to remind the administrator of the privileges. At CERL everyone got "AUTHOR MODE" whether in group "s" or not.

What Kind of System Help is Available?

Start lesson "aids" and when it shows the "INFO" screen press NEXT and select option "b" for "Author Resources". Another one is "sysaids".

Why is there a leading "0" (Zero) in front of many lessons?

The leading "0" meant that it was a published lesson. On CDC systems, this involved accounting for usage and royalty payments to the lesson authors. That is why the names were different on CDC systems for such lessons.

What is a "master file"?

It is a NOS permanent file used as a container for a CYBIS or PLATO file system. So instead of reading the disk directly as CERL PLATO would do, CYBIS has to read the NOS permanent file which adds some small overhead.

What is ECS/UEM used for in CYBIS?

The use of ECS (directly via RE/WE instructions) is pervasive throughout CYBIS. It's not like NOS where ECS is handled as a solid state disk and accessed only through disk I/O primitives that could be tweaked. All the CYBIS CPU components have ECS field length assigned to them, and among other things use it as a medium for high bandwidth message passing both among CPU programs and PPU programs. This version is actually configured to use UEM.

What is the CYBIS CONSOLE program?

The CONSOLE program is a primitive CYBIS terminal program that runs on the CYBER console. It is invoked via the DSD command "X.CONSOLE.". A text only terminal is displayed on the left screen, and a help page showing the keyboard layout on the right. Right blank toggles shift state, left blank toggles between text and function keys. The right screen changes accordingly. Usually it prompts

"Lesson desired" - BACK (left blank then 'b') then drops you into "Author mode". At this point you don't have a user name or group name, but you have "s" privileges.

The program has limitations, but it is necessary to do some of the CYBIS setup. For example, it is the only way to get onto a system without having a valid login with a known password, or it allows you to create group "s" if it doesn't exist yet.

If for some reason you want to use the console terminal (perhaps because others don't work yet) and you do want to appear as a signed in user, at the "Lesson desired" page, enter "plato". That will give you the login display, and you can go through username, group name, and password as usual.

To exit the CONSOLE program press the following keys in order: left blank, right blank and 'x' (that means '[', ']', and 'x' on Desktop CYBER).

How do CYBIS and NOS relate to each other?

CYBIS runs as a set of subsystems under NOS but bypasses NOS in many places. Not quite so much in the CDC / VCampus version as in the CERL edition. NOS just does startup, scheduling, provides disk containers in form of direct access permanent files (master files) and terminal communications (via NAM).

How does CYBIS communicate with Pterm or real terminals like a CDC IST II?

CYBIS uses a small subsystem called PNI written in CYBIL to provide the terminal interface. PNI interfaces via NAM/PIP to the NPU emulation of Desktop CYBER. The network settings are configured via an NDL file (NDL18) under NOS user "netadm" with password "netadm". Desktop CYBER's network configuration is defined in "cyber.ini" under the section specified in "npuConnections" which points to "npu.cybis". The current configuration allows for 32 normal terminal connections on TCP port 6610, 30 Pterm connections on TCP port 8005 and 2 special RS232 terminal server connections for real terminals like the CDC IST II. Loading of the IST II or III firmware from CYBIS/PNI is not yet reliable.

What is CSTC?

CSTC is the assembler needed to assemble CYBIS or PLATO code. PLATO uses a modified Compass, with an additional directive CST, which is like SST but it saves more stuff, specifically common symbols as required by the PLATO sources. You can actually use it everywhere - CERL did exactly that. But CDC practice was to call it CSTC and use it only for PLATO. Paul Koning has what appear to be the actual mods that turn COMPASS into CSTC. The mods were done by Don Lee of CERL. No attempt has been made to build CSTC using these mods.

How does CYBIS know which master files are available?

The PROC MFNX on the deadstart tape attaches all the master files spread over multiple disk packs accessed via the NOS PACKNAM directive. There is no directory of master files, instead at start-up, CYBIS asks MASTOR what all the master files are, and loads their directory. All master files that are of type "general" or "master" then have their directories essentially merged into a single namespace which is the namespace of files/lessons you can see as an author or student. MASTOR simply expects every local file that's a direct access file to be a master file.

How does CYBIS start?

The starting point is PROC CYBIS on the deadstart tape. That loads the "prime time" CYBIS, i.e., the one that lives on the deadstart tape. On development systems, there's also a notion of a "non-prime

time" or "development" CYBIS, which is found in a set of direct access files and loaded by PROC CYBDEV which is unsupported and not working on this system. My standard procedure is simply to make a new deadstart tape when I want to test stuff. Our current tapes are fast enough for this.

CYBIS is invoked by "X.CYBIS." from the console (DSD). It calls MFNX, gets the CONFIG file, and starts MASTOR. The CONFIG file is record TEXT/CONFIG on the deadstart tape.

When MASTOR starts, it looks for a permanent file "CYBIS" under user "SYS". The file contains calls to load the CYBIS subsystems to NOS control points (PLATX, CONDX, FRAMX and PNIX).

Where are the CYBIS sources?

All CYBIS sources are under user "plato". Back at CERL, the official sources were in PLATO files ("code" type lessons). I believe CDC made the OPL since that's the form they are comfortable with. And in fact, the CYBIS system has those code files, but it's clear they are produced from the OPL, not the other way around. CYBIS files of type "e" indicate they contain NOS program sources

What are "binary" type master files?

CYBIS wants a binary pack to put lesson binaries into when they are "condensed". Ideally you want enough space to hold all the binaries that are in use, but if you don't have enough old entries are discarded.

What are "backup" type master files?

The "backup" type masterfiles are typically used for doing file transfers: you put things on a "backup" type masterfile, then ship the masterfile, then load that on the destination system. The point is that PLATO's file name search only looks at "master" and "general" packs by default. To access a file on a "backup" or "binary" master files, you have to select it explicitly, by master files name.

To change a "backup" type master file to "general" type:

- shutdown CYBIS
- attach the "backup" type mastefile (xyz in the example below)
- then: "mfalter(mf=xyz,pt=general)"
- restart CYBIS

What protocol is used by the CYBIS terminal connection?

CYBIS uses 7 bits with parity. Pterm conforms to the CBIS/PLATO custom encoding. The protocol is documented in CYBIS "document" type file "s0ascers".

What are all these "n" and "o" prefixed versions of CYBIS lesson files?

There are lot of "n" versions of files – standard system lessons with "n" prefixed to the name. This is a CYBIS system convention that development is done in those, for example "nedit" or "nplato", then once it looks good it is copied into the primary lesson (and the old primary one goes to the "o" prefixed lesson). The "n" lessons are restricted to system staff. And when in an "n" lesson, by convention a '*' is plotted in the upper right corner so you're aware that you're not in the official version. Also by convention, "n" files go into account "s0nver", and "o" files into "s0over".

The "n" versions aren't hard-coded, but they are a development convention supported by a couple of tools. They are defined as system lessons just like their non-n counterparts (apart from not being set up to trap shift-stop). That's about all that is needed for things to work.

As an additional convention, when you're in the "n" version of one of those lessons, the jumps to related lessons also use the "n" version. The idea is to make the set of components look like a single larger system. For example, if you just run "nedit" which tells me I am in the n version by showing a "*" in the top right corner. As you wander around the various bits of a lesson, you may be in nedit1, or nedit2, etc., but you stay in the "n" series.

Various bits of code that jump around different lessons know rules like "if I'm 'edit' then jumpout to editmicro, else jumpout to neditmicro", so the "n" lessons act as a set. Note that this is just a coding convention.

As for copying over, there's a specific option in lesson "operator" for that, option "g" for "copyover file contents", then "n" to copy <file> to o<file>, n<file> to <file>. So before you do that, make sure there exists an o<file> -- for example "oplato". These are content copies, not file creators. For that operation to work, all three files have to exist, and be the same size. There's a bit of magic: the codewords are not copied over so whatever the edit protection was on the original main lesson remains in effect. If after doing this you discover a problem, you can reverse it ("o" to copy current to new, old to current). Of course this is only one level of "Undo".

Typically, this "copyover file contents" also deletes the binary so the lesson will recondense when next requested. In a few cases that doesn't happen; I think "plato" is one of those. You can edit or inspect the lesson and hit shift-stop to force a condense. That may tell you it is "in use" although it isn't always - sometimes the reference count is forced to 1 to keep a lesson resident even when it's not actively in use. You can then enter SHIFT-HELP to condense it anyway. It used to be that it was a bad idea to do this for lesson "syslib". I don't believe that is true anymore. In any case, reloading PLATO will also do the job.

Why does it ask for a "security code" when I try to run "ipedit" or "allocate"?

At the "Author" mode (or "System" mode) page, if you enter a lesson name and NEXT, that is a request to edit the lesson. For that, you need write access to it. System lessons by convention have their "change code" set to "no access permitted", i.e., you can't write them. Well, not except from the console running "X.CONSOLE." and then entering BACK to go into "Author" mode.

As mentioned earlier, the editing rule for system lessons is that you edit and test the "n" prefixed version, whose change code is set to "group s" meaning that anyone logged in into that group has write access. Then once debugging is done, you use the "copyover" operation (lesson "operator", option "g") to for example copy lesson "ipedit" to "oipedit", "nipedit" to "ipedit" (or whatever the lesson name is).

If you want to execute a lesson, enter the lesson name and DATA (Control-D in Pterm). Also, if you want to inspect rather than edit, enter the name and LAB (Control-L in Pterm). Inspecting requires a match either on the change or inspect code. For system lessons such as "ipedit", the inspect code by convention is "account system", i.e., anyone logged into any group that is part of account name system (such as groups s, p, or pso) is allowed in.

How to create a data file like “acclog1”?

File “acclog1” is a “data file”, not to be confused with a “data set”. A data file is basically a log file. You can create it; make it an 18 part data file.

While manipulating files through the lesson “accounts” machinery is the normal practice, there is another way that doesn't depend on the account machinery to be fully operational - lesson “operator”. Option “f” lets you do file operations, such as creating, deleting, or renaming them. It lets you specify what account to operate on.

How to reload (restart) all CYBIS subsystems (versus a shutdown)?

To shut down: lesson “sysopts”, option 3. If you aren't dealing with other users you can do an immediate backout. That leaves the terminal you're using still logged in. Log it out and then go to the NOS K display for “MAS1”. “K.STOP” tells it to exit, killing all the CYBIS subsystem. “K.RELOAD” tells it to reload the CYBIS subsystems.

How do I find all required master files?

Lesson ipedit, option “g”, has the list of “required master files”. Those are the ones expected to be there. A mismatch isn't fatal, but a bunch of things complain if they don't match. You can either correct a mismatch by changing what masterfiles are attached, or by editing the list through that ipedit option, or some of each. Then a restart should show the corrections.

Has Pterm got some trace feature?

Control-“j” toggles trace mode on and off; the status bar at the bottom shows “Trace” if it's on. The trace file is “ptermxxxx.trc” in the install directory, where xxxx is the process ID number. Pterm keeps adding to that file during the entire run, so you can toggle trace on and off repeatedly and still get just the one file.

What is “Development mode”?

“Development mode” is a flag that controls a variety of things. This system is in development mode. It's a match of the system ID (SID in the config file) against a table in deck msubs (at ndevsys). If you want to pick a new system ID you may want to add the chosen name there. The currently used “CYBDEV” is already in the table and is a perfectly fine generic “CYBIS development” name.

How to create the first ever Group “s” user (information only)?

This is not needed in this working system, but provides some useful background information.

In “X.CONSOLE.” at “Lesson desired” enter BACK. That gives you the “Author” mode page. Enter “s” to edit the group. Option 2, c, 3, to add the user. NEXT, edit the new user (DATA from the add page, or from the “c” menu in the group editor). Option 4 to “set allowable options”. Press DATA to set all. That creates a valid Group “s” user you can then sign on with in Pterm. Exit CONSOLE by pressing the following keys in order ‘[’, ‘j’ and ‘x’.

Further users can be added via Pterm after logging in using that first username.

How to find the list of used master files?

In lesson “operator”, option “c” shows the actually used master files. Lesson “ipedit” is where you see the required list - they are expected to be the same. More precisely, the set of “master” and

"general" packs is expected to match. But if they don't, you can still do things, and in particular file operations. The reason for the warning is that you might find yourself recreating a file which is actually on a missing pack, so if you fix the missing pack issue you now have two files.

What are “general” type versus “master” type master files?

The difference is that "general" type master files are available for file allocation through accounts. The “master” type master files are not - you can only put files there through "operator" by explicitly asking for a specific master file to be used. Both types contribute to the default file namespace (while binary and backup packs do not).

How to edit installation parameters when lesson “ipedit” is not working?

This is not needed in this working system, but provides some useful background information.

If you can't get ipedit to run, another option would be to manually edit the parameters common in file "sysfile". From “X.CONSOLE” in author mode, enter "sysfile" then NEXT. Keep entering ‘+’ until you see the block name "iparams". Enter the letter to the left of that name, then NEXT to edit and then ‘1’ to change/inspect common. For example to change the list of master files required, press ‘+’ until you see master file names show up. I see those starting at word 45. Enter "a" to set data entry mode to alphanumeric; "o" for octal. Once you see an entry to change, enter "e" for edit mode; it asks you for the word number and the new data, then advances to the next line so you can keep adding values. I'm not sure if the entries have to be together (no blank lines in the middle). If you see a master file name you don't want, just blank it out (octal entry mode, new value 0). If you need to add a name, alpha mode is easiest (add it to the end or into an empty slot in the middle of the list). To find a block by name type “x”.

What is a Site in the CYBIS context?

"Site" refers to either "physical site" or "logical site" - usually the latter. A physical site is simply the 32 stations sharing a common site number (all but the lower 5 bits of the station number). That term comes from "site controller", a device used in the old (pre-ASCII) PLATO communication system to drive 32 terminals from the TDM signal broadcast from the central system over a TV signal.

A "logical site" is a collection of terminals belonging to a given organization. It has a name, and an ECS allotment. Originally, that's about all there was to it. The "E" option from Author Mode shows ECS usage by the currently logged on terminals in the site. Lesson "site" is the site manager tool, it allows you to see more things, and also perform operations like send a message to a terminal on the site, or force a user off (this is called "backout").

Logical sites are defined by a system option. In lesson site, SHIFT-DATA gets you there. Option "a" lets you see the logical site assignments for the terminals in a given physical site (i.e., 32 terminals at a time). Option "c" gives the logical site numbers (which is how option a shows assignments), site names, and current ECS allocation.

The "logical site access controller" (LSAC) feature was developed in response to the need of site managers to police the site users better. In particular, at CERL there tended to be a lot of game players, and their usage might interfere with the intended use (educational programs etc.). The LSAC is a program that can be used to verify that someone is allowed to log on right now, or later on, once they are logged on it can be used to check that they aren't doing stuff they aren't allowed to do right now.

How to disable the default lesson (information only)?

CYBIS as received originally was configured to automatically drop the user into a "default lesson" or more correctly into a "logical site access controller". The following describes how I disabled this:

Start lesson "site", SHIFT-DATA for system options, option "e" for "logical site lesson table", enter your site name, then clear out the entries shown there.

What are well known CYBIS groups?

Group "s" is one of the privileged groups. Others are "p", "pso", "o" and "m". Groups "s" and "p" are intended for development system staff, "o" is for system operators, "pso" is for the user support team (the ones who answer questions from regular users). Group "m" was used only at CERL, I believe, by hardware people.

The specific privileges vary. Users of group "s" you can think of as "superusers", and group "p" has close to the same privileges. Group "o" enables only privileges related to system operations – things like startup and shutdown. Group "pso" enables privileges needed for user support. On this CYBIS system, only groups "s" and "author" exist, but that should be sufficient.

What is a CYBIS account?

In PLATO, "account" is a file space management concept. PLATO files (lessons and other files such as groups or datasets) are created explicitly and then their content is manipulated. You don't create or delete files as part of normal program operations, unlike most other systems. When you create a file, you chose its size, expressed in "parts". A "part" is 7 320-word blocks (35 NOS disk sectors). Files can have 1 to 18 parts, except datasets and namesets which can be up to 63 parts. (A "dataset" is basically a binary data file made up of fixed length data blocks - the record size is selected at creation time. A "nameset" is an indexed file - records are arranged in record chains identified by a key, the "name".)

An "account" is used to delegate space management. The account owner is given some number of parts (by the system staff), which are allocated to some named account which lists that owner. The owner, or others who have been given access to the account, can then create or otherwise manage files within the space limits given to the account. Lesson "operator" is a system tool which lets system staff manipulate files for any account.

An account is a file space management and file ownership concept. It isn't a namespace. (That was considered as a further evolution but was not implemented) So groups, like any other file, are system-wide in the sense that all you need is the name to access it. (This applies to general and master packs, not backup packs.) But every file, groups included, belongs to some account, and its file space (number of parts) is charged against the limit of that account, the file can be controlled (access changed, renamed, or deleted) by the account manager of that account, and so on.

How to list all CYBIS accounts?

Lessons "accounts", SHIFT-DATA for "System options", option 'b' for "account summaries/logs/file information", '1' for "List of accounts".

What is "system backout" in CYBIS?

A "system backout" means forcibly logging off all terminals. That process is done roughly by pretending shift-stop is being pressed on all the logged-on terminals. This allows the currently running lessons to do their clean-up, including stuff like writing ECS to disk.

Normal system backout starts with a notification message to all logged-in terminals, giving them 60, 30, 10 second warning. Immediate backout skips the warning phase.

The correct shutdown procedure is to run system backout (immediate is fine if you don't have others logged in), then on the console under DSD "K,MAS1." followed by "K.STOP."

How to edit lesson "notes" settings?

In lesson "notes", shift-data for system options (this isn't mentioned on that display), then option 'h'.

How do I print anything?

"R" at author mode (or directly lesson "prints"). For regular users, that gives you the printout request page. For system users, it shows the pending requests. SHIFT-BACK goes to the printout request page, but there's an extra option "shift-data to print a single file". Option "f" is "print log maintenance". There is also various setup stuff, "c" for the printer table, and "e" for the available print types and the corresponding submit decks.

Printing a file submits a batch job that runs the appropriate printing program, for example "tprint" to print a lesson. The submit decks are in file "prtsub", so you can look there and find the source blocks containing the decks that will be submitted for the given print type.

There are some system settings on user records in the group editor which relate to NOS job submission. Check "special options", batch jobs should be "enabled" and a NOS user name set. Mine is set to "plato".

How to switch between "X.CONSOLE." modes?

With "X.CONSOLE." there is a distinct author/normal mode. You get from "normal" to "author" via BACK. The distinction author/normal doesn't exist in Pterm sessions.

The "lesson desired" page is specific to the console. It is generated in the CONSOLE COMPASS code. It doesn't exist for other terminals.

How do you get from "author" to "normal"? On the author (system) mode page, shift-stop logs you off, that's true for CONSOLE as well as any other terminal. That produces "press NEXT to begin". On the console (but no other terminal), BACK from that page gets you to "lesson desired".

What are these shifted letter shortcuts on the Author mode page?

SHIFT-DATA from author mode shows all of them. For example 'A' is Aids ("INFO"), 'Z' is alarm, 'E' is ECS allocation. And 'F' is the lesson catalog.

Are a "data set" and a "data file" identical?

No, a "data set" and a "data file" are entirely different. A "data file" is a log; a "data set is" a binary block data container. Each has its own editor, which allows you to view it in a manner meaningful to that kind of file.

Why do many lessons have number prefixes which collide with shortcuts?

Why do many lessons have number prefixes which collide with shortcuts? For example I start the Avatar game with "2avat", but I have to prefix it with a space otherwise it executes the "2" shortcut. Why not just call it "avatar"?

The convention is that lessons starting with 0 are "published lessons", and then some got names starting with 1, 2, or 3 for special purposes. I'm not sure why the Avatar ones start with 2, but we got them that way. The "2" shortcut only applies to system staff, so normal users don't notice this annoyance.

Why do we have so many files starting with "s0" or "a0"?

The reasoning is that non-system staff is subject to naming rules when creating files ("accounts" enforces these). One is a length limit (no 10 character names, I think). The other is 2 characters minimum, or maybe 3, and the first two must be letters. So file names that have a digit in the first or second position are essentially reserved names. Originally a whole bunch of system files were created with ordinary names, blocking those from general use. At some point the s0 convention was put into place to allow new system files to be created that could not conflict with user-owned file names.

Why do I have to enter DATA (Ctrl-D in Pterm) to start a lesson?

Having to type DATA (Ctrl-D) to invoke a lesson is counter-intuitive. NEXT drops you into the editor. It would make more sense the other way around.

Yes but remember that DATA on the PLATO keyset is a dedicated key, just like NEXT. So it's no harder to type one or the other.

How to determine the free space in masterfiles?

Use lesson "operator" option "c" (master files on disk and space left).

Where do new files go when no master file is specified?

When no master files is specified at creation they are allocated to the master file of type "General" with the most free space.

How to edit a lesson which starts when I enter NEXT after the lesson name?

Edit or inspect it by typing "edit", then the lesson name and NEXT to edit or LAB to inspect.

How can I learn Tutor to write my own CYBIS lessons?

Lesson "aids" (info) is the documentation to use to learn Tutor. There's also Bruce Sherwood's text book, you might find that useful. Once you get past the basics, when reading system lessons like "accounts" or "plato", you'll encounter commands not documented there. Those are system commands (commands accepted only in lessons listed in the "system lesson" table). The documentation for system commands lives in lesson "sysaids".

What is the "use" statement in Tutor code used for?

It is similar to the "#include" in C. It is used to include lesson components from other files.

Normally a "use" command names blocks, and the lesson referenced is one of the "associated files". In system lessons, a different form of "use" is allowed that explicitly names the file, rather

than relying on the associated file name. That case applies if the use "lesson name" is set to "multi-lesson".

In system lessons the "use" command refers to another lesson and a source block (by name) in that lesson file. It includes that text into the referring lesson at that point, like "#include".

The "use" command can have one argument (a block name) or two (lesson, block). For the single argument form, the lesson name is the one given on the "associated files" page of the directory information

Unlike C practice, "use" often brings in executable code (think of it as library code) because Tutor doesn't have object libraries the way Unix does.

What are the rules for "codewords" of lessons included with "use"?

Various Tutor operations ("common", "use", "jumpout" and "attach") that refer to another file require that the two files have matching codewords for the code related to the operation.

To inspect or change the codewords edit the lesson, open block "a" and the select option "c" for Codewords.

A bit of notation: a word in hyphens (like -use-) means the Tutor command of that name.

How to set the "lesson access classes" for an account?

You can see them if you edit the account (e.g. "system") and press DATA twice. But to set those flags, you run lesson "accounts", SHIFT-DATA for System Options, "a" for Edit/Inspect, "1" for Edit, now enter the account name (e.g. "system"), LAB for Lesson Access and "1-50" NEXT to enable all. Press BACK repeatedly to exit.

How to recondense a lesson?

The normal rule is that SHIFT-STOP while editing a lesson forces it to be recondensed. If someone is using (running) the lesson, you will be told and have to elect to kick them out, or cancel the condense. By special rule, SHIFT-STOP while INSPECTING recondenses the lesson also only if you're signed in using a system staff group (s and probably a few others). That's useful because a lot of system lessons are set up to prohibit editing, and this way you can force a recondense even without edit access. That applies also to other non-editable lessons; the published lessons are typically set up that way also.

Does the condenser produce a binary even if there are errors?

The way the condenser works, even if there are errors, it produces a binary. That may be good enough, or not. There is a "reserved word" - essentially a predefined constant - named "zcondok" that is true (-1) if the condenser was happy, false (0) if not. Incidentally, most reserved words have names starting with "z" as a way to avoid name clashes, though some (older) ones don't.

How to find (and possibly edit) the lesson component that just aborted?

Often you can hit SHIFT-STOP to exit into author mode and then NEXT (by itself) to edit the lesson you were just in. Alternatively, you can use another terminal and lesson "site" to see what lesson your terminal is in.

What is a “micro table”?

A “micro table” is related to text entry. A micro table is a table of strings, which are mapped to "micro" keystrokes: control/m then another key. Think of it as somewhat like alt or control codes, except that "micro" is a prefix. Micro tables may for example be used to supply a Hebrew keyboard, or for convenient editing shortcuts. Typically, they are used when running a lesson (via the "micro" command). But you can also set up a micro table to be loaded whenever you edit a lesson. That is configured via one of the lesson information pages, accessed by DATA from the block directory display. ("Associated files").

A "font" is another thing that can be loaded - those are the terminal programmable characters. You might see that used as an associated file in lessons that use other character sets, for example Hebrew or Russian language lessons. In either case, if the load fails, you get an error message. It doesn't interfere with editing, and has no effect on running the lesson. It just means that whatever convenience the author intended with that associated file isn't available.

What is lesson “runnersys”?

It performs a similar role to Unix CRON. For details see the “Plato Operator Guide”.

What type of variables are supported by Tutor?

TUTOR defines three sets of "variables" - CM resident data the lesson can manipulate. One set is "student variables", 150 words. Another is "common", 1500 words. The third is "router variables", 50 words. They have builtin names n1-n150, nc1-nc1500, and nr1-nr50 respectively if you want to interpret them as integers, or v1-v150, vc1-vc1500, and vr1-vr50 to interpret them as float. "Router variables" are only available in "router lessons". Student and router variables are preserved from session to session in the student record on disk (in the group) if the user type is "student", but not otherwise.

In general, lessons use symbolic names for variables, which is done with the "define" command. You can see those at the start of a lesson, or in blocks referenced by “use” commands near the start. There are also "segment" variables - those are arrays defined as bitfields (think of them as somewhat like a VFD statement). There's a lot of detail in Tutor command “define” - study AIDS for more information.

What is “common”?

A common is a chunk of disk (the blocks of the given name in a lesson). When in use, it's copied into ECS. Changes are "checkpointed" (written back) to the disk copy at times. And when a lesson wants to manipulate common data, it normally brings it into CM (into nc variables with -comload-). There are some exceptions: the -transfr- command allows writing directly to the ECS copy of common or storage, which is useful for some special purposes. One example is moving large blocks of data, larger than can conveniently be manipulated as nc variables.

What is “storage”?

A similar idea to “common” is "storage". That's also a chunk of ECS, swapped in/out using -stoload- (just like -comload-). The difference is that "storage" is private, not shared, and anonymous (not from disk). It's used when you need a lot of data, more than fits in the 150 student variables. It can also be used for I/O buffers, especially in system lessons. Remember that CYBIS disk I/O is to/from ECS, so storage acts as the ECS buffer for disk I/O.

Why when editing non-text blocks do changes appear in multiple blocks?

Text type blocks (source, text, listing) are treated as individual blocks of 320 words. All other block types are referenced by name, when created you are asked for the desired length (if applicable) and the system then creates that many consecutive blocks with the given name. So, for example, a 900 word common takes 3 blocks. Editing any of the 3 has the same effect - it simply means that you're editing that common. Which block you named when you started the common edit makes no difference. The same applies to other block types that can be multi-block constructs, such as charsets.

How to export files from CYBIS to NOS?

Use the "pf" program. It is documented in lesson "nosaid".

How to edit the Notes Index?

In lesson "notes" press the undocumented SHIFT-DATA key to get to "Notes System Options" and then select "h" for "Edit the Notes Index". Make any changes as required (Help is available) and then use LAB to install the new version.

How to edit and test lesson "plato"?

Lesson "plato" is the CBIS login screen. To make changes you MUST use the "new lesson" mechanism. So if it doesn't already exist, copy "plato" to "nplato" using lesson "operator" (option "f" for "File options" and then 5 for "Make a copy of a file"). Try "nplato" to see that it works. You just execute it as a lesson, and run through the normal sequence. There's one difference: in standard PLATO (though not, I think, in the CYBIS version) during login you have to press SHIFT-STOP after the group name. This foils fake login programs, because they end up being aborted by that keystroke before they can prompt for the password. This works because "plato" is one of a small set of system lessons that are configured to treat shift-stop as a normal keystroke that they can act on. However, "nplato" is not such a lesson. So the convention is that, while testing, you press SHIFT-LAB instead. When you run nplato, you'll note a * in the upper right corner. In general system lessons are set up to do this when you run the "n" version.

So you log in normally to your group "s" signon, then run "nplato" to test your changes. If it succeeds, you'll be logged in but it will drop you into the author mode page of "nedit" (the new version of "edit"). SHIFT-STOP at that point will sign you off. If while running "nplato" something goes wrong, you can hit SHIFT-STOP and it will abort. Depending on where you were, this may drop you back to the author mode page, or it may leave you signed out (if so, sign back in). See below under "How to debug lessons" for more information.

How to debug lessons?

For debugging, you might find "term-step" helpful. It's a bit clunky but still valuable. AIDS has some info, and it has (of course) help built in. The way it works: at any prompt, you can press the TERM key and enter "step". Alternatively, at any point in the code where you want to start step mode, use the command -step on-. This will erase the bottom two lines of the screen and give you the current unit, command, and source line. Pressing NEXT will step one command. You can also examine variables. BACK (or SHIFT-BACK?) exits step mode, as does the command -step off-. This doesn't have nice stuff like breakpoints, or the ability to step over subroutines (-do- commands) but it's better than nothing.

How to create “pnotes” for a new system group?

To create it, edit account “system”, “file options”, create a file, option d (“Personal notes file”), then enter the group name, masterfile (NEXT to have the system pick one), then it asks for space parameters. Those can be changed if needed, so for now some placeholders would be 20 users, 1000 notes total.

Why doesn’t SHIFT-STOP work on my CDC IST II or III?

There are two key maps. The original one is used initially; the other one is enabled if the terminal supports XON/XOFF flow control and answers to that effect when the host sends an “enable flow control” echo code (echo 0x52). The host knows this was done because the reply is 0x53 rather than the default 0x52. The two key maps are largely the same; the differences are in the codes for Access (the “square” key), SHIFT-SUB, TAB, SHIFT-HELP, SHIFT-STOP, SUPER, and apostrophe. So if the wrong key table is being used, or if the two sides are not in agreement, SHIFT-STOP would indeed be affected but stop would work. XON/XOFF must be disabled in your terminal server program. Note that you also need two stop bits otherwise the IST II or III firmware download fails (or is unreliable).

What is the “echo” output command in the ASCII protocol?

PLATO always had an “echo” output command, which tells the terminal to send back an “echo key” with the specified low order bits in it. As a way to get status or ask for stuff in an upward compatible way, various specific echo data values were given specific meanings, and the various interesting reply values are all encoded as echo key responses with a value different from the request value. The result is that a terminal which doesn’t understand the request simply treats it as a plain echo and replies with the data that was sent.

So, for example, enable flow control is echo code 0x52. Reply 0x52 means not supported (the terminal simply echoed the value). Reply code 0x53 means “flow control supported and I just enabled it”.

Similarly, there are terminal type queries (0x70) - reply 0x70 means the terminal doesn’t know that query, which makes it a PLATO IV (original Magnavox) terminal, while other values refer to one of the many different microprocessor based terminals.

What is “GOGO” in the CYBIS context?

At CERL, the system was both a production system and a development system. The way this was handled is through “prime time” and “non-prime time”. During prime time the system ran the tested “good for production” version, and was supposed to be up at all times without interruptions. Non-prime time was for running new test versions; ideally that would be pretty stable as well but there weren’t guarantees. Normally the current non-prime version would migrate to prime status weekly.

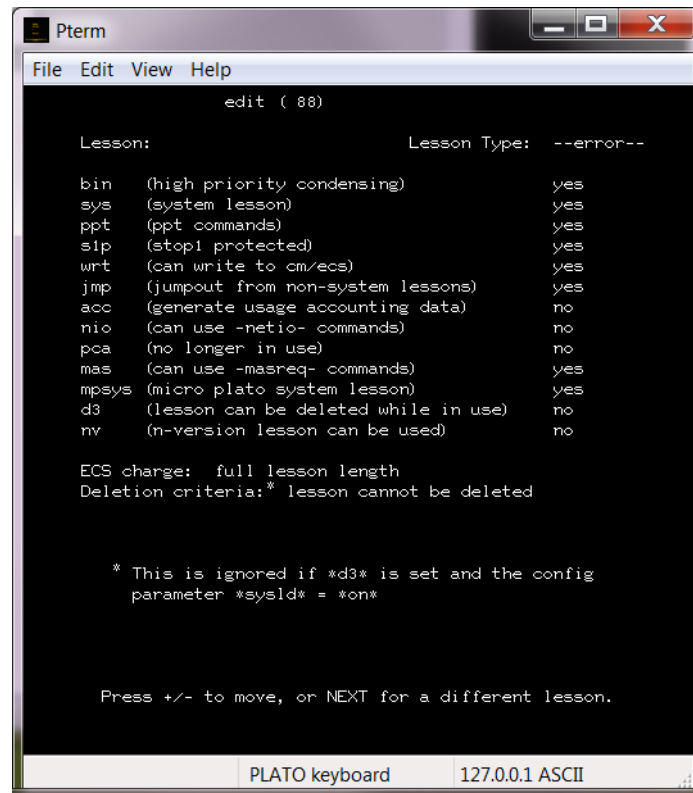
The prime time version of PLATO was on the deadstart tape. The non-prime time version was loaded from permanent files, so developers could reassemble just one component (say, FRAMAT), run a backout, and reload the non-prime version. For reasons I never learned, the non-prime time version was called “GOGO”.

Given that deadstart is so fast on DtCYBER, getting the GOGO version to work was not even attempted. At CERL, with slow tape drives, it certainly was.

What are "system lessons"?

CYBIS has a list of "system lessons". Each entry has a set of flags associated with it to say what specifically that named lesson can do beyond the normal things.

The system lesson table is manipulated by lesson "s0syslst". Option 1 gives the list of system lessons by name (the standard list), while option 4 lets you see the local (installation specific) additional system lessons. The "n" lessons are not explicitly listed as their own entry; instead they inherit the attributes of the regular entry, if that is enabled. Consider the settings for "edit":



```
Pterm
File Edit View Help
edit ( 88)

Lesson:                                Lesson Type: --error--

bin  (high priority condensing)         yes
sys  (system lesson)                   yes
ppt  (ppt commands)                    yes
s1p  (stop1 protected)                 yes
wrt  (can write to cm/ecs)              yes
jmp  (jumpout from non-system lessons) yes
acc  (generate usage accounting data)   no
nio  (can use -netio- commands)         no
pca  (no longer in use)                 no
mas  (can use -masreq- commands)        yes
mpsys (micro plato system lesson)       yes
d3   (lesson can be deleted while in use) no
nv   (n-version lesson can be used)     no

ECS charge: full lesson length
Deletion criteria: * lesson cannot be deleted

* This is ignored if *d3* is set and the config
parameter *sysld* = *on*

Press +/- to move, or NEXT for a different lesson.

PLATO keyboard  127.0.0.1 ASCII
```

The main one is "sys" which says the system commands are enabled. The other flags add some additional capabilities that are not given to every system lesson. For example "s1p" means the -stop1- command is allowed, which lets the lesson treat SHIFT-STOP as a normal key it handles rather than as the "abort" key. The "wrt" flag enables -writecm- and -writecs- commands. "mas" enables -masreq- commands. For example, "edit" and "plato" have "s1p" set but not "mas"; lesson "console" has "mas" set so it can ask for console display information, but not "s1p" because it doesn't want to handle shift-stop keys.

Also, the "sysnv" flag means: if you're on a "development system" (which cybdev is because its name appears in the development systems table in the source code) then "nedit" is also a system lesson; otherwise it isn't (for some reason we don't have the "sysnv" flag).

You can create your own system lesson for experimentation. Typically you'd set "sys" but not most of the others; you can add the more magical flags if needed. When you make changes to the local system lesson table, you have to use SHIFT-HELP from the main menu to put those changes into effect (by reading them into ECS where the condensor finds them); if you don't, you won't see the changes until the next restart.

Note that you can't change the flags for the standard entries like "edit", only the local ones. The standard table is in the sources, in deck "sysless" but it's probably not a good idea to mess with it.

For testing changes to edit, the normal practice is to modify nedit. If you want to make a private copy so you aren't touching nedit - perhaps worthwhile if you're doing drastic stuff - make a local system lesson entry for that copy, with the same flags as the original with the possible exception of s1p. (The reason for not setting s1p is that errors in such lessons may leave you with no reasonable way to abort, while having s1p off means shift-stop works normally.)

What are the standard PLATO text entry features?

The text entry features are invoked by the CYBIS function keys EDIT (control-E), "square" (control-Q) and COPY (control-C).

In general, when you're entering text (at an "arrow" prompt, the character that looks like a hollow greater than sign) EDIT and square are available. COPY often is but not always. They work like described below.

If a "copy buffer" has been set, COPY fetches and displays the next word from it, SHIFT-COPY the entire remaining copy buffer, and Square the next character. Typical usage of the copy buffer is for the previous content of the line when you're doing a replace command, the contents of the previously inserted line when you're inserting multiple lines, etc. The specifics vary, but in many cases, if there's an obvious interpretation of COPY for that particular context, it will be there and have that meaning. For example, when renaming files in accounts or operator, COPY would give you the old name. When creating files, if you're doing several in a row, COPY will be the name of the previously created file. And so on.

EDIT is standard (except when explicitly disabled, which is rare). It's a mode toggle. Initially, if you've entered some text and press EDIT, the text is copied to the edit buffer (internal to PLATO) and erased. Subsequent presses of EDIT then retrieve the next word, SHIFT-EDIT the rest of the edit buffer, and Square the next character. So this is the tool if you're entering a line of text and notice a typo earlier in the line. It's not quite the same as cursor editing of the partial line the way modern systems do, but it's a pretty effective substitute.

This is fully documented in the lesson editor help.

What are the important user types in CYBIS?

An important type of user is "author". That looks just like administrators except that you don't get the system options. When you log in you get the "AUTHOR MODE" page (same as "SYSTEM MODE" except for the label). And just like admins, authors can execute lessons (name then DATA) as well as inspect (name LAB) or edit (name NEXT). For the latter, they need access privilege to the lesson in question. System admins have access to a lot of stuff; regular authors are more limited. Some standard files are open inspect, lesson "library" is an example because its purpose is to show sample code.

Some authors may have additional privileges. An admin might create an account, and make a given author the account owner. That author is thereby delegated the ability to create files of various types, including groups, lessons, and other stuff. Other authors who want a lesson to work on would request space from an account owner; typically one of the system admins. (At CERL it would be the account owner for the account related to your academic department.)

Regular authors are author type records in groups whose names are not among the special ones. The special names are s, p, o, pso, coserv -- you can see the list in the documentation for the -check- command in sysaids. Other groups are normal. So to create a normal user record, start by creating a normal group (say "users"), then create an author record in that group.

Record types other than author are probably not that interesting. They are created in a similar way, though, in non-system groups. (System groups don't have non-author records in them as normal practice, except for special cases that I can't remember.) A student is either locked into a given lesson whose name is set in the student record, or in a "router" which is a lesson variant whose task it is to "route" the student through a sequence of lessons. The student record stores lesson state; when a student signs out and back in, at least part of the last activity is preserved and you continue where you left off (more or less, subject to what the lesson wants to do).

There is a standard router provided by the system that is driven by "instructor files" as a way to set a lesson curriculum. Those files are normally edited by people with instructor records. For details check lesson AIDS (INFO).

Finally, a "multiple" is like a student record, except that unlike all other record types, it can be concurrently in use at multiple terminals. Because of that, a multiple doesn't preserve the execution state as a student record does; it always starts at the predefined lesson set in the record. Multiples are often used for demos of various kinds. An example is user "diag" in group "m". After signon this user directly goes to lesson "diag" which is a set of terminal tests including various test displays (note that user "diag" and group "m" are not implemented on this version of CYBIS but the "diag" lesson exists and is functional).

What are naming restrictions?

Account names are limited to 7 chars max, groups to 8 chars max. All file names must be alphanumeric, starting with a letter or digits 0-4. (But digits are used for specific purposes. I would avoid them entirely for groups and accounts).