Repository NitrOS9 3.3.0 to NitrOS9 EOU Version 1.0.0 Transition Manual
Released April 19, 2023

NOTE: If you are brand new to **NITROS9/EOU**, this is not the document for you. You should instead read the beginner's documentation that comes with the NITRS9/EOU download, called:
"**NitrOS9 Ease of Use - Beginners documentation.rtf**"

NOTE: If you are upgrading from a previous version of **NitrOS9/EOU**, this is also not the document for you. You should check the version release notes to see what has changed from the previous version. (On the other hand, if you have skipped multiple versions, then this document will be helpful in figuring out all the updates at once. If you have made a custom boot for an earlier version of **EOU,** you may want to back up your custom **env.file** and **startup** so that you can copy them back onto this new release). Also please note that some applications are still Works In Progress and may crash; if there is no icon it is likely because of that (except some command line utilities).

This document is specifically for those familiar with **NitrOS9 3.3.0** from the
NitrOS9 repository who want to know what has been changed, upgraded, bug fixed,
etc. from that release.

1ST UP, A THANK YOU TO VARIOUS CONTRIBUTORS TO **NITROS9/EOU**:

Many thanks to **R. Allen Murphy** for gathering all the Alpha, Beta and Final release notes in order to create this document, and in getting the final version made with a table of contents, etc.

Nick Marentes & Rob Inman for making new icons for various programs. Thanks to **Rob** as well for giving us a pre-set up **RSB** for us to install and test with.

Todd Wallace for his new **IBM CGA font** (with all CHR$(32-255) supported), as well as his programs **CCTPlayer, COCOIRC, CocoWX,** and **DOSDir.**

Jay Searle for helping get the major project of redoing/updating the entire set of OS-9 Level II manuals off the ground. The **Technical Reference** and the **Windowing System** are complete already, and can be downloaded at:
http://www.lcurtisboyle.com/nitros9/nitros9_docs.html
Work on the next parts of the manual have started as well, although these next sections are the largest and/or have the most changes.

Jeff Teunissen - For giving us permission to included his new, more modernized C compiler '**DCC**' (and related parts). He also supplied the new **HELP** command system that includes enhancements like sub-topics.

Fred Provoncha - For rewriting the **CONTROL** panel program completely from scratch, now up to version 3.0.

All the **original authors** of various programs that we now have set up to run from **GShell**, of whom there are too many to mention.

Also thanks to our beta testers throughout the development of **NitrOS9/EOU**: **Aaron Doughty, Michael Furman, Rob Inman, Ron Klein, David Ladd, Grant Leighty, Nick Marentes, Nick Marotta, Mark Overholser, Floyd Resler, John Shawler, Terry Steege, Steve Strowbridge, Tim Lindner, Ed Snider, David Lightman, Bill Pierce, Ken Waters** and **Todd Wallace.**

Please report any problems, bugs found, questions, etc. in the **Discord #NitrOS9EOU** group, or via email to me at **curtisboyle@sasktel.net**. Please include details of the problem, which CPU version you were running, and which emulator (or real hardware) you ran into the problem with.

Table of Contents

Changes/additions since NitrOS-9 V3.3.0 (2014)

This will be divided into sections based on your role as a **NitrOS9/EOU** user. We have divided this into 4 categories (note that some may overlap somewhat, depending on your use case):

1) People primarily running applications. This will be a list of the pre-installed programs that are ready to run from **GShell** (and **GShell** itself), and also some more common ones that you would run directly from the **SHELL** command line. Please note that you will occasionally see some programs that don't have **GShell** icons; this *usually* means that they are a work in progress that isn't quite finished yet. (We develop **NitrOS9/EOU** from within itself). This section will let you know in which folders to find the programs to launch from, and a brief description of what the programs do. Some of these will have help that you can access from **GShell**. Others may have their own manuals in **/dd/docs** or even online**.**

These will be shown in a folder hierarchy (note: not all folders have apps; ones that do not will not be shown) so that you can figure out where to find them.

2) People doing application programming. This will list any updates to the various languages, compilers, emulators, etc. that a programmer making regular applications would be interested in. So things like **BASIC09, DCC, CP/M emulator, RSB, ASM, RMA,** etc. and any related libraries, how to set your programs up to launch properly from **GShell**, etc. You may also need information on the new **System Calls** and **Drivers;** see the 3rd category below for more details.

3) People doing or directly accessing system programming. This will explain what has changed/been added in **File Managers, Device Drivers**, the **OS kernel**, etc. For those that fall under the 2nd category above, some of this may apply to you as well so it may be worth giving a quick look through. The updated **Technical Reference** manual will have full descriptions of updated **System Calls** and newer **Drivers** and can be downloaded here:

http://lcurtisboyle.com/nitros9/EOUDOCS/NitrOS-9_EOU_Level_2_Technical_Reference_Manual.pdf

4) System administrators. This will be for people who plan on using **NitrOS9/EOU** in a multiuser environment, and will cover things like login options, file attributes, etc.

1) Pre-installed User applications & User settings

GSHELL (1.27)

GShell as of **NitrOS9/EOU Version 1.0.0** is now up to version 1.27 and has many enhancements from version 1.26 that was part of **NitrOS-9 version 3.3.0**. To run it (and you can run it multiple times in different windows, even with different screen resolutions), just type **GSHELL** in any Coco 3 screen window (not from a Coco 1/2 32 column VDG window). Note that any sizable windows you create will be in the same screen and color resolution as the **GShell** that they are launched from; running multiple **GShell**'s at different resolutions will let you launch these windows at the resolution(s) that you want. Updates and changes since version 1.26 are listed below:

1) The **<?>** icon in the upper right (the **HELP** function) has been changed to use **Jeff Teunissen**'s enhanced **HELP** command, allowing for things like Sub-topics. It also should accept mouse clicks as well as key presses to continue listing help after it auto-pauses.

2) GShell can now launch programs into a **VDG** screen directly without needing any "helper apps" as in the past. This takes less RAM and launches such programs faster (this does require changes to the **AIF** file - see the **Programmer**'s section for details). This includes both OS-9 Level 1 programs and OS-9 Level 2 programs that use direct screen writes. (One thing of note - Level 1 programs - especially those that use medium resolution graphics - will take a lot of **System RAM**. It is recommended that you only run one of these at a time. Some larger ones that double buffer may not run with a full EOU boot at all - you may need to create a stripped down Boot set and ditch **MultiVue/Gshell** for these.)
NOTE: With a Drivewire enabled boot, some of these may not run (a full set of Drivewire drivers takes a lot more **System RAM).**
Quick note about Level 1 programs: Since these were written originally for a Coco 1 or 2 (which did not have real lowercase until the T1 VDG came out in 1985), some of those programs require uppercase key responses to work properly. Once these programs are launched and running, hit **<CTRL-0>** to switch to uppercase only on that screen (**Biosphere** is an example).

3) Since a lot of people run **NitrOS9/EOU** from a real Coco with a **CocoSDC**, we took out the 3rd floppy drive (**/d2**) from the default drive list, freeing up a spot for things like RAM drives (**/r0**) and Drivewire virtual drives (like **/X0, /X1**). If you are one of the rare people who actually have 3 floppy drives hooked up at once, you can add it back in from the **Control** program (see it's own section below). This was done because **GShell** currently gets (up to) the first 5 devices from the **/dd/sys/env.file** to display on screen as it's default drives and locks those in. If you are using all 5, it won't let you add or change any of them. If you have 4 listed, it will lock those 4 in, but it will let you add/change the 5th entry whenever you want.

4) GShell make use of a new System call in **IOMAN** to change device descriptors - this way it requires less RAM and can handle larger directories.

5) GShell now supports alternate icon name labels (rather than just the name of the program to launch). This helps with with things like some packed **BASIC09** programs (which sometimes showed up as just "**RUNB"**) and the Sierra games (which all showed up as "**SIERRA")**, etc. You can also just rename it something totally different to your liking and it will even accept spaces. How to do this is explained in section 2 (**Application Programming**) under **AIF File format changes**. Please note that a maximum of 10 characters will show up on a 40 column **GShell** window, and a maximum of 11 characters on an 80 column window; you should take that into consideration when doing alternate names (anything past that will just be cut off).

6) To aid with backwards compatibility with the original **Tandy/Microware** version, **GShell** will now consider both 24 and 25 line high windows as "full screen" and not create a sizable window for programs that were specified as 24 line screens. It will just create the appropriate size full screen window automatically instead.

7) The **File** menu has an added option - **DUMP**. This does a hexadecimal and ASCII dump of the the file or folder that you have selected in **GShell**, pausing every so many lines. You can hit a key to continue onto the next screen's worth of data (**NOTE: DUMP** is set up to look best on an 80 column window). Hit **<CTRL-E>** or **<CTRL-C>** to abort.

8) The **File** menu options **DUMP** and **STAT** have been enhanced to use an 8x8 font (rather than 6x8) which is both easier to read and also prints to the screen much faster. Both of these commands will fit their maximum width output lines properly on the overlay window they run in (if you have **GShell** running in 80 columns).

9) When creating new **SHELL**'s from **GShell** that are going to different screens than **GShell** itself, command line editing is now fully enabled by default in each of them.

10) GShell now auto-senses RAM up to 2 MB by itself, so the **RAM=###** line in the **/dd/sys/env.file** is no longer needed (although you can override your physical RAM with the value in the **env.file** if you need to for testing).

11) GShell now takes 3 less get/put graphics buffers than it did before, saving a little bit of RAM.

12) GShell can now launch a packed **BASIC09** program by double clicking it, even if it doesn't have an **AIF** file set up for it. It will be run in an overlay window on the main **GShell** screen. Please note that some programs were not designed with that in mind and will not display correctly, or worse redefine the window itself (and since **GShell** won't have a clue that happened, act strangely or even crash). However, some programs will run just fine this way (**GPMAP** is an example).

13) Some issues with launching packed **BASIC09** programs that require large data memory reserved have been fixed. Hopefully I got them all.

14) GShell will no longer freeze the mouse when loading a large directory. It originally shut mouse movement off to speed up reading large directories but since on really large ones that can take many seconds, a lot of people assumed that **GShell** crashed (appeared frozen). It now just slows down the mouse update rate drastically (it just updates the mouse cursor position a few times per second) which lets you know that it is still running, but busy.

15) The **Disk** menu, under **Set Devices**, will now warn when the drive list is full (maximum of 5 active devices at a time). You can change any of these that you have previously changed or added on the fly, but not the "locked in" set that came from **/dd/sys/env.file**.

16) While technically a **Programmer** feature, I will mention here that the Keyboard mouse (toggled on/off with **<CTRL-CLEAR>** is now local to each window in EOU, and inherited from parent processes to children. What this means is that if you are using the keyboard mouse in **GShell**, and you launch an app that uses the mouse into another screen/window, then that app will also have the keyboard mouse enabled by default. The text shell windows by default have this off, and any new windows you make from those will inherit the keyboard mouse being off as well (so you command line editing and history keys will all work properly).

17) For those that don't know already, **GShell** supports some keyboard shortcuts, listed below:
SHIFT-UP - scroll up a screen of icons
SHIFT-DOWN - scroll down a screen of icons
= - force refresh of current folder
S - make an overlay window **SHELL** on the **GShell** screen.
$ - make a placeable, sizable window **SHELL** on a different screen
? - run **HELP** for the currently selected icon/program.
Q - **Quit GShell** - will prompt if you are sure - you can either click the mouse in the appropriate box, or hit **Y** or **N**.

Some GShell tips & tricks

Applications that are ran from the main drive contents view in the GUI window use **AIF** files to get the window type and minimum window size (quite a few will always request a full screen). Others will have a minimum size notably smaller than the screen type they are requesting... these ones are **sizable** windows where you position the box on screen to where you want to place the upper left corner of the window (the box it will show while you are moving it around is the minimum size window that program needs to run). One you found your "anchor spot", you click the button to lock that upper left corner in, and then move the mouse to size the window. You click again when you have the size you want, and the program will launch in that window. If you launch another sizable application, and the **AIF** specifies the same screen type as the sizable window you made earlier, it will let you anchor and size your new window on that same screen.

A lot of the **Tandy** menu applications (but not all) also allow you to position/size the window that they are to run in, but there is no **AIF** files for these, so how does **GShell** know what screen type to use? The answer is that it will use the same screen type that **GShell** itself is currently running in (which also means if you switch the resolution that **GShell** is running in, then the next time you go to launch a **Tandy** menu application, it will use a new screen with the new window type, rather than the last one you made.

Knowing this, you can set up several screens of sizable windows in different resolutions to match certain application window types, to gather them.

This can help you if you want to run, say, **Rogue** next to a shell on the same screen.

Rogue defaults to 640x200x4; launch it, and make it's size small enough to leave enough room for another app (say about the size of half a screen - vertically or horizontally).

Then, if you set your **GShell** resolution (in the pull down **VIEW** menu) to the same resolution/color (80x25 4 color in this case), any of the resizable Tandy apps (**shell, control panel, calculator, clock**) that can fit it's minimum size on the screen, will let you place it on the same screen as **Rogue**. Some programs can run in very small windows, so you might be able to cram 4 or more windows on a single screen. (**Zone Runner** is another sizable program set up for 640x200x4 - you can with that. Or **Tandy Clock, Calculator**, etc.)

If you want bigger windows, but still mixed together on several screens, you can either change the resolution for each new screen by selecting a different resolution in the **VIEW** menu and then launching a couple programs ... and repeat. OR, you can launch a second **GShell** from a different hardware text screen, and have the second copy launch a few more applications onto the screen resolution of your choice.

Another tip/trick is using the 'S' key from **GShell**, which will launch a **Shell** in an overlay window on the main **GShell** screen. When you do this, you will automatically be put into the directory that you are currently in within **GShell** itself; so you can use **Gshell** to navigate to a directory, hit **S**, and then run utilities or programs on files in that directory directly from a **Shell**. Handy for long pathnames as it can save you typing.

Control (3.0)

CONTROL is one of the programs that is selected from the **Tandy** menu in **GShell** (looks like a mini hourglass on the left side of the menu bar). This has been completely rewritten from scratch by **Fred Provoncha**, and is enhanced in multiple ways (including being able to cover the original **Printer** and **Port** items in the **Tandy** menu as well as the original **Control** all in one program). It also is aware of enhancements and additions to the environment file (**/dd/sys/env.file**) that the original **Control** was not.

Control now uses both a menu bar with some general options (like saving, abandoning changes, quitting with new settings but not saving, etc.) but now also adds a left side tabbed interface to edit each section. These sections include what used to be separate **Tandy** menu items (like **Printer** and **Port** (serial port) mentioned above), as well as additional enhancements that have been added to the environment file in **EOU**. It also adds multiple color theme presets (more on that below). It also has built in **HELP** (under the **Info** menu) that will display help for everything in the current tab you have selected. I will let you use that help to get the full details on what everything does, but I will briefly mention what each tab contains here:

GShell tab - This controls the default resolution that **GShell** will launch in, your default storage device names (floppies, hard drives, RAM drive, Drivewire drives), your default serial port devices (printers, RS-232), the default 4 colors that **GShell** itself uses (which is separate from the System palette that other windows default to), and has Presets for different color schemes that you can use for **GShell.**

Display tab - This lets you select the monitor type and lets you select the systems default 16 colors (note that this tab will show you the system colors, not **GShell**'s own colors, while in this tab, so expect different colors at this point). It also lets you select the **Preset** themes here as well (same as the **GShell** tab).

Mouse tab - This lets you select which port the mouse is in, whether it is low resolution or using the High Res Joystick interface, whether you want the second mouse button to switch windows or just act as button 2, as well as mouse sampling rate and mouse timeout values.

Keyboard tab - This lets you select whether you will boot with the first screen having key click on or off, and set the keyboard delay time before key repeat, as well as the key repeat speed.

Windows tab - This lets you change the settings of /Term (the first window when you boot), including it's X and Y sizes, screen resolution type (40 or 80 column), and foreground, background and border colors.

Printer tab - This lets you select the page width/height and left, top and bottom margins (measured in characters)

System tab - This lets you change some system wide settings when you boot: default data directory, default execution directory, default task priority, whether you want module CRC checking on or off (on checks for corrupted modules, but slows program launches, sometimes by several seconds), **SysGo** text response messages (regular or more verbose), and the **RAM** size override value (as mentioned above under **GShell** above).

Bootup tab - This lets you change how the final stages of booting the computer work, including what program to launch by default for each screen, parameters to send to that program, the name of the **startup** file to launch, the name of a program to auto-execute (normally **AutoEx** for backwards compatibility), parameters to send to the auto-execute program, and any modules that you want pre-loaded on boot.

It should be noted that **Control** does it's changes on the current ACTIVE **env.file**; ie the in in **/dd/sys**. To make these permanent between **SWAPBOOT**'s, you will need to copy them over to **/dd/BOOTS,** over top the env.file for the set that you are currently using. So as an example, if you are booting up using the **SDC** set, you would do the following:
1) Run **Control** and do any changes that would like. Make sure to **SAVE** your changes.
2) At a **SHELL** prompt, do the following:
   A) **cd /dd/sys**
   B) **cp -ov env.file /dd/boots/env.file.sdc** (or which ever set you want your changes to stay permanent).

You can do this for multiple Boot sets if you want them all to have the same **env.file** settings; some users prefer to do things like change the default colors of the main /TERM window between the different boots, so that they can tell at a glance which Boot set they currently have running.

That should be it. Then next time you **SWAPBOOT** to a different boot, and then **SWAPBOOT** back, your new settings will stay intact.

Composite Color set

Both **Grfdrv** and **CoVDG** (the two main "draw something on the screen" drivers) have both had their RGB to Composite color tables changed slightly, based on **Erik Gavriluk**'s (of **Colormax 3 Deluxe** fame) research. All colors in NitrOS9 are based on RGB values, and if you have a composite monitor or RF TV hooked up, they are translated through this table. These should be more accurate than the original Tandy/Microware tables.

Shellplus 2.2a and command line editing/history

While **Shellplus 2.2a** was included with **NitrOS9 3.3.0**, for some reason the window device descriptors did not have some of key character codes defined properly in order for them to work. The device descriptors in **EOU** have all been fixed (including both **CoWin** and **CoVDG** descriptors) so that both the SCF current line editing keys (move left, move right, beginning of line, end of line, insert and delete) as well as the **Shellplus 2.2a** up/down arrow keys to go through command line history should all function properly now. For more details on how these work, please see the **Shellplus** section in the **Beginners** documentation.

Keyboard Mouse functionality

The behavior of the keyboard mouse (explained in the original Tandy/Microware Level II manual) has had it's default changed in **EOU**. It is now a local setting in each window (although if you spawn a window from another one - like launching an application from **GShell** - it will inherit the setting from the window that created it). This helps prevent confusion when one is running a graphic application using the keyboard mouse (an example being **GShell** itself), and then switches to one of the **Shellplus** windows; with the original universal setting, the keyboard mouse is enabled there as well, and then things like backspace and command line editing/history don't work (it's trying to move a non-viewable mouse cursor). This has caused a lot of confusion, so having this setting local to each window (like Capslock already was) makes more sense.
If things like backspace, etc. don't seem to be working on a window, it may be that the keyboard mouse is turned on. A quick way to check is try using both a left arrow and <CTR>-<H> to backspace; if only the latter works, chances are you have the keyboard mouse turned on, and you can toggle it off with <CTRL>-<0> (zero).

File Tree for User applications and files to run from GShell

This section lists the folder tree for programs that you can launch from Gshell by double clicking on their icons. It will also include some folders that will contain data files that can be double clicked to launch their host application automatically. Folder names in this section will be shown in ALL CAPITALS, applications themselves in mixed case. This will only give brief descriptions of each application; full documentation will either be in the application itself or in it's own separate documentation or manual. The ones listed here cover what is installed in **NitrOS9/EOU** as of version 1.0.0; if you are reading this with a newer version of **EOU** there may be more applications than are listed here.

Please note that I am listing the 11 character long folder and application names here; if you are running **GShell** in 40 column mode, you will only see the first 10 characters. A folder in the root directory will have a blank line before the folder name, and be in a larger font:

APPS

GCalendar - a rewrite of the Calendar program that came with MultiVue with a lot of enhancements. Written by Darren J. Kindberg in 1992. Full documentation can be found in **/dd/DOCS/gcal.txt**. If you find that you do not like it, you can re-enable the original version by doing the following from any SHELL window:

cd /dd/cmds
rename gcal gcal.new
rename gcal.org gcal

GalileanSat - Program that shows the relative positions of the 4 Galilean satellites around the planet Jupiter that you would see through binoculars or a telescope, based on a starting date that you give it.

IconGen - An older Icon generator or editor from 1988 by Robert Gault. Has the advantage being able to quickly load/save in the CMDS/ICONS directory, but simpler editing features.

SuperIke - A more powerful icon generator/editor with advanced features like Paint, border, shift, etc. It however has some issues with large directories, so you may want to edit/save in a small folder, and then move the final icon over to **CMDS/ICONS** afterwards once you are finished editing it. It also will show your icon in the current **GShell** palette colors that you have selected. This version is updated to 1.1, which adds **GSHPAL** color support, fixes mouse cursor scaling, gets mouse settings from **env.file**, the Fatbits display is almost twice as fast, the text only icon list uses a narrow font to fit more of the name on the screen, it's more multi-tasking friendly, the 3 machine language subroutine modules have been fixed to NOT have self-modifying code, changes **SHIFT, CLEAR** & **CREATE** functions to use your currently selected color (not hardcoded 0 as the old version did), fixed up overlay window colors.

CocoIRC - Coco IRC client (currently requires having your Coco hooked up to a modern machine via **Drivewire**) by Todd Wallace.

MShell - A powerful text based directory browsing application that lets you copy, etc. files between folders and drives, and includes **Drivewire** support (including updating itself over the internet). It will also support OS9, DECB and PC drives for copying files from/to. If you want to use a GUI for file manipulation beyond very simple things like rename and delete, use this instead of **GShell**. Be aware that this program can take a lot of RAM with large directories; this should not be an issue on 1, 2 or 8 MB RAM Coco, but could cause issues if you are running several other applications on a 512K RAM Coco.

Ultimuse3 - A power music editor meant to use with MIDI synthesizers (both through the bitbanger port, or with a more powerful MIDI card like the MIDI Maestro from Retro Renovations (**http://www.go4retro.com/products/midi-maestro/**). Even if you don't have a MIDI synthesizer, you can use it as a high res, up to 16 voice musical notation editor.

CocoWX - A Coco weather client that you can get live weather from, using the internet (currently requires **Drivewire**). You can also run it from the command line for some extra options. NOTE: This requires at least version 0.6 of **pyDrivewire**. It currently defaults to display in full graphics mode; if you want to change it to text mode, change the AIF file to have option **'-t'.** Full documentation is in **/dd/DOCS/cocowx_readme.txt**.

CHECKBOOK

Checkbook+ - A checkbook register program by Joel Hegberg. This uses ALT key combos to get to the various menu options.

DYNACALC

Dynacalc

Please note that this may not run properly from GShell at the moment; Dynacalc uses the same default extension as GCalendar, and GShell gets confused. It should run fine from the command line though.

A VisiCalc type spreadsheet.

PHANTOMGRAPH

Phantomgraph - A graphing/charting program that allows both raw data entry as well as importing from things like **Dynacalc**. We have it defaulting to 640x192x4, but you can edit the **AIF** file and change the type to a different graphics mode if you wish.

PLANET_ENGINE

Planet - Another tool for astronomers, this one shows the positions of constellations, planets, the milky way band ad the sun and moon based on the vantage point and time that you enter the latitude and longitude for. You can also view planet orbits and where in their orbit's each planet is, as well as what phase each planet (and the moon) is at from Earth for the specific location and time you specify.

DEMOS

Bounce

BASIC09 "Mona Lisa" Bouncing ball demo.

Bounceit

Alan Dekok demo of many bouncing small sprites plus scrolling bars. BREAK or <CTRL>-<E> to exit.

Demo

Full screen version of the **MultiVue** demo, where you can select drawing circles, lines or bars. Bars in particular are much faster now.

GUIBDemo

Demo program written in **BASIC09** to use Sean Driscoll's **GUIB BASIC09** library (included with **EOU)**.

SeMaze

Allen Huffman's 3-D wireframe maze walkthrough demo.

BOUNCE96

Bounce96

Alan Dekok's Bouncing Ball demo that shows his library of calls that allow a programmer to completely take over the computer, and return control to NitrOS9 after the program is done. As an example, he is controlling the border colors per scanline during this demo. All multi-tasking is stopped (until the demo times out at a minute & a half, or if one hits the CLEAR key), but it shows that some of the fancier techniques used in DECB games _can_ be used under NitrOS9, if you use these routines to preserve the state.

RAYBOUNCE

Raybounce

Demo of 9 pre-rendered frames of ray-trace bouncing balls (taking 9 screens). This will likely require a 1 MB machine or higher, unless you kill some windows, etc. first. Hold down the **ENTER** key to exit.

SIERRA_XMAS

Sierra XMas

The Sierra Christmas demo, where you can customize some of the messages.

DOCS

This folder contains full documentation for a variety of programs, usually the ones included by the original authors when the software was released. They have a variety of extensions, but most of them can be viewed either with ***List*** from the ***Files*** menu in **GShell**, or with the **VU** utility from a Shell. There is one exception: As an experiment, I tried making a duplicate copy of an earlier version of the ***"NitrOS9-Beginners_docs"*** with an .OS9 extension, that is meant to be listed from a Shell command line (it restarts it's own window, uses custom fonts, etc.). You can try it and see if you like it. I am not sure if I am going to pursue this further, or design a new way of doing it; but be forewarned that listing it from **GShell** itself will completely screw up **GShell**, since it is changing the window and fonts without **GShell** knowing that this has happened.
These documentation files will usually have far more detail than what is found with the **HELP** command.

EMULATORS

CPM

HITCHHIKERS

CPM-Hitch

Run Infocom's **Hitchhiker's Guide to the Galaxy** text adventure game, emulating a Z-80 CP/M system to do so.

SUPERCALC

CPM-SprCalc

Run **SuperCalc** Version 1.12 spreadsheet, emulating a Z-80 CP/M system to do so.

TURPASCAL

CPM-TurPasc

Run **Turbo Pascal** Version 3.01A, emulating a Z-80 CP/M system to do so.

WORDSTAR

CPM-WordSta

Run **Wordstar** Release 3.30, emulating a Z-80 CP/M system to do so.

ZORK1-3

CPM-Zork1 - Run **Zork I**, emulating a Z-80 CP/M system to do so.
CPM-Zork2 - Run **Zork II**, emulating a Z-80 CP/M system to do so.
CPM-Zork3 - Run **Zork III**, emulating a Z-80 CP/M system to do so.

RSB

RSB VDG

Run Chris Burke's **RSB** (Radio Shack/Microsoft BASIC) in a 32 column VDG screen (for Coco 1/2 compatibility).

RSB Wndw

Run Chris Burke's **RSB** (Radio Shack/Microsoft BASIC)in a 40 column Coco 3 screen (for Coco 3 compatibility).

GAMES

LEVEL1

ADVENTURE

INTERBANK - Work in progress. Not functioning yet.
THE_CITY - Work in progress. Not functioning yet.

ARCADE

Sentinel

Low res arcade shooter by Jim & Charlie Gerrie. 3 Skill levels. You will likely want to tap the keys individually for each move (unless you slow down the system key repeat speed from the default setting; key repeat can get way ahead of your moves).

ZeroGravity

256x192 arcade game where you have to collect items within a time limit while flying and bouncing around at high speed. There is no gravity, so you don't slow down unless you thrust in an opposing direction.

CARD_GAMES

Blackjack

Low resolution Las Vegas Blackjack.

Color 8's

256x192 4 color Color 8's game (similar to Crazy 8's)

CNPoker

Low resolution Poker game

CASINO

Slots

Text based slot machine game

SIMULATORS

BIOSPHERE

Biosphere

Graphic simulation of trying to keep a planet's **Biosphere** alive by balancing the number of animals and plants to make a balanced ecosystem. You can even genetically mix animals (or plants) together to create new hybrid species. NOTE: if lettered commands don't appear to be working, you may need to turn on CAPSLOCK (with CTRL-zero). CTRL-C to quit.
Also this version is patched to run in color on both RGB and composite monitors.

STRATEGY

Bombaway

256x192 2 player Artillery style game, with variable terrain and wind conditions.

Cavehunt1

Text based game where you wander through a cave, finding and swapping weapons to efficiently attack the creatures in the cave. Similar to **Hunt the Wumpus**.

Ratmaze

3-D 256x192 maze game where you try to find the cheese. Your only help is to get how far from the cheese you are at any time.

StarTrekL1

NitrOS9 Level 1 text based **Star Trek** game. 10 skill levels.

Sweeper

Mine Sweeper type game in 128x192 4 color graphics. 3 skill levels.

LEVEL2

ADVENTURE
TEXT
Adventure
The original Colossal Cave Adventure ported to OS-9.
Cavehunt2
Text based game where you wander through a cave, finding and swapping weapons to efficiently attack the creatures in the cave. Similar to **Hunt the Wumpus**.
Gargoyle
Gargoyle Castle text adventure game in 80 columns.
Star Wars
Text adventure game in 40 columns.
YorkDeed
The Deed of the York text adventure game in 80 columns.
ROGUE
Rogue
Plays an enhanced version of **Rogue**, that allows you to play **Rogue** in it's full graphics font in a sizable window. It also has the "cheat / test" mode enabled by default; please see:
http://www.lcurtisboyle.com/nitros9/rogue-cheatmodeinstructions.txt
for full instructions on how to use this mode)
STRANDED
Stranded
Plays **Stranded... And Out of Gas!** text adventure game, that is presented in a multi-paned 3D look graphical window.
GRAPHICAL
AGIQUEST1
AGI Quest1
"AGI Quest 1" is a homebrew game using the Sierra AGI engine.
BLACKCAULDRON
Cauldron
Plays **Black Cauldron**, one of the many Sierra On-Line games backported to the Coco 3.
CAITLYNS
Caitlyn's
Plays **Caitlyn's Destiny**, a homebrew game using the Sierra AGI engine.
ENCLOSURE
Enclosure
Plays **Enclosure**, a homebrew game using the Sierra AGI engine.
GOLDRUSH
Gold Rush
Plays **Gold Rush**, one of the many Sierra On-Line games backported to the Coco 3.
KQ1
KingsQuest1
Plays **Kings Quest 1** (the original **Kings Quest**), one of the many Sierra On-Line games backported to the Coco 3.
KQ2
KingsQuest2
Plays **Kings Quest 2**, one of the many Sierra On-Line games backported to the Coco 3.
KQ3
KingsQuest3
Plays **Kings Quest 3**, one of the two official Sierra On-Line games for the Coco 3.
KQ4
KingsQuest4
Plays **Kings Quest 4**, one of the many Sierra On-Line games backported to the Coco 3.
LSL
LeisureSuit
Plays **Leisure Suite Larry**, one of the two official Sierra On-Line games for the Coco 3.
MANHUNTER1
Manhunter1
Plays **Manhunter: New York**, one of the many Sierra On-Line games backported to the Coco 3.
MANHUNTER2
Manhunter2
Plays **Manhunter: San Francisco**, one of the many Sierra On-Line games backported to the Coco 3.
MOTHERGOOSE
MotherGoose
Plays **Mother Goose**, one of the many Sierra On-Line games backported to the Coco 3.
OPERATION_R
Opr:Recon
Plays **Operation:Recon**, a homebrew game using the Sierra AGI engine.

POLICEQUEST
PoliceQuest
Plays **Police Quest**, one of the many Sierra On-Line games backported to the Coco 3.
SPC_QUEST0
SpaceQuest0
Plays **Space Quest Chapter 0: Replicated** (a homebrew prequel to the original Sierra **Space Quest**)
SPC_QUEST1
SpaceQuest1
Plays **Space Quest Chapter I: The Sarien Encounter**, one of the many Sierra On-Line games backported to the Coco 3.
SPC_QUEST2
SpaceQuest2
Plays **Space Quest Chapter II: Vohaul's Revenge**, one of the many Sierra On-Line games backported to the Coco 3.
SPC_QUESTX
SpaceQuest
Plays **Space Quest: The Lost Chapter**, a homebrew game using the Sierra AGI engine and meant to take place after **Space Quest Chapter II**.
STARCOMNDR
SpaceCom
Plays **Star Commander I**, a homebrew game using the Sierra AGI engine.
TIMEQUEST
Time Quest
Plays **Time Quest**, a homebrew game using the Sierra AGI engine. This is Demo version 2 so I don't think it is complete.
V
V
Plays **V**, a homebrew game using the Sierra AGI engine, based on the hit TV series.
VOODOOGIRL
Voodoo Girl
Plays **Voodoo Girl Queen of the Darned**, a homebrew game using the Sierra AGI engine.

ARCADE
Invaders
Plays **Invaders-09** (Space Invaders style game)- keyboard or joystick controls, multiple levels & # of shots, written by Allen Huffman. Full documentation is in /dd/DOCS.
Lander
Plays **Lander** (a Lunar Lander style game) - keyboard controls
PacOS9
Plays **PACOS9** Version 2. Joystick controls, high score saves and 8 mazes to go through.
QBert09
Plays a **Q*Bert** style game written in **BASIC09**. Uses left joystick. This version has been patched to fix where it was painting in the squares; due to Robert Gault's changes to the line drawing routines in **NitrOS9** (to make them more accurate), this program would sometimes paint outside where it was expecting to and mess up the screen.
Smash
Plays **Smash**, a **Breakout** style game for joystick or mouse. Multiple levels and multi-ball, and lets you design your own levels as well with a text editor.
Tetris
Plays 40 column text color **Tetris** with various levels. Keyboard controls.
Thexder
Plays the NitrOS9 version of the original Coco 3 **Thexder** cartridge. Keyboard controls, 5 levels. *NOTE*: If run from the command line in this folder, there are extra options to pick from. This version has also been fixed to run properly on machines with 2 MB of RAM.
BONK
Bonk
Bustout style game written in **BASIC09**, with multiple space backgrounds. Requires joystick or mouse.
GEMQUEST
GemQuest
Gem Quest game written in **BASIC09** - arcade/adventure game. Requires joystick (2 button recommended).
KORONIS
KoronisRift
Koronis Rift by **Epyx**. Requires joystick. This has been patched to cleanly exit back to **GShell** on exit. Also some very minor optimizations on both CPU's.
KYUMGAI
Kyum-Gai
OS9 port of the **Sundog Systems** arcade classic. 2 Button joystick required. NOTE: if run from the command line you can choose the starting level.

RESCUE
Rescue
Rescue on Fractalus by **Epyx**. Requires joystick. The versions included with EOU are optimized compared to the original **Epyx** versions: 10-12% faster on the 6809 version and 30-40% faster on the 6309 version. They are also patched to cleanly exit back to **GShell.**
SNAKEBYTE
snakebyte
Play **Snakebyte** game. Multiscreen "**Snakes**" type game, with 14 levels included.
snakescreen
Snakebyte level editor. You can edit existing levels (including the originals) and add your own.
SPACEZAP
Space Zap
Clone of the arcade game of the same name written in **BASIC09**. Joystick required.
BOARD_GAMES
ChessMV
Text based **Chess** game.
CyrusChess
Port of **Cyrus Chess** ROM Pak to OS9 and upgraded to 640x192x4 graphics, ported by Chris Burke. Joystick optional. This has been patched to not override the system mouse settings.
Mixup
High res graphical pairs matching game. Joystick/mouse required.
Cocothello
Othello game. Joystick/mouse required.
Peg Leap
Peg jumping game in 80 column text.
SeaBattle
Graphical version of **Battleship**. Joystick/mouse required.
Yahtzee
Yahtzee high res game for 1-4 players. Joystick/mouse required.
BATTLESHIP
Battleship
Battleship 1998. 1 player, 40 column text version of **Battleship** written in **BASIC09**. Speech support, and some digitized sound effects.
KNIGHTS
Knights
Knight's Tour game - try to get a Knight (using it's legal moves) to hit every spot on the chessboard once. Joystick/mouse required.
MAGICSTONES
MagicStones
2-6 player board game, with each player attempting to knock out opponents stones.
MASTERMIND
MasterMind
Classic figuring out color pattern **MasterMind** board game.
SHANGHAI
Shanghai
Port of classic **Shanghai** cartridge to NitrOS9.
ShanghaiCrd
Port of classic **Shanghai** cartridge to NitrOS9 with custom playing card based tile set.
ShanghaiTrf
Port of classic **Shanghai** cartridge to NitrOS9 with custom traffic sign based tile set.
SHANG_B09
Shanghai
BASIC09 based version of **Shanghai**, that includes uncovering a hidden picture under all the tiles.
ShangB09ED
Tile editor for **Shanghai/BASIC09**
CARD_GAMES
Klondike2
CocoPro Solitaire in Klondike play mode. Joystick/mouse required.
Solitaire
CocoPro Solitaire in Regular play mode. Joystick/mouse required.
Klondike
Klondike Solitaire by ColorSystems. Joystick/mouse required.
PyramidSol
Pyramid Solitaire by ColorSystems. Joystick/mouse required.
SPOKER
Strip Poker
Port of **Strip Poker** from the IBM PC. ***WARNING! ADULT CONTENT.***
EDUCATION
Eliza
The famous very early "AI" program, where the computer plays being your psychiatrist.

CARMEN
Carmen
Where in the World is Carmen San Diego. NOTE: the game says to hit any key to exit the list of detectives or the Hall of Fame screens; this is incorrect. You have to hit <CTRL>-<BREAK>.

SIMULATORS
Golf
80 column text **Golf** simulation, which uses color to draw the course.

GameOfLife
Conways Game of Life simulator. Start an initial colony, and then let the rules in the simulation dictate whether the colony dies out, stabilizes or expands. High res joystick interface highly recommended (or keyboard mouse). This program has been patched to leave the system mouse settings alone. Full documentation (including special keys) are in **/dd/DOCS.**
Football
80 column text **Football** simulator, that uses color to draw the football field.
Trek
80 column color text **Star Trek** game.
Zone
Zone Runner - runs in a sizable window. Space trading game.
FLIGHTSIM2
FlightSim2
Plays **Flight Simulator II**. If you are running the 6309 version of EOU, it is more multi-tasking friendly than before.
MICROSCOPIC
Mic.Mission
Plays **Laser Surgeon: The Microscopic Mission** by **Activision**. Simulation where you control a small ship inserted into the bloodstream of patients and you have navigate to fix various health issues, without getting destroyed or having the patient die.
SUBBATTLE
Sub Battle
Plays **Sub Battle Simulator** by **Activision**. This version has been patched to exit cleanly to **GShell** when you hit **<CTRL>-<Q>** to quit.

STRATEGY
Bombaway2
Coco 3 high res version of **Bombaway2**, a two player **Artillery** style game.
Cubist
Plays **Master the Cube** - an original puzzle game where you have match the the large playfield squares to the small one on the right. You click center points between 4 squares to rotate all 4 squares clockwise. Requires joystick/mouse.
Minefield
Plays strategy game where the player picks how many mines they have to find (10-40), and then has to navigate from their start point (happy face) to the end (a crown) without hitting a mine. Each move gives a hint by saying how many mines are surrounding the player. Requires joystick/mouse.
STrek
80 column black and white text **Star Trek** game (Coco 3 enhanced version of the Level 1 version listed above).
Subhunt
Plays an 80 column text game similar to **Battleship**, except 1) you are firing depth charges from a surface ship, and 2) you fire into a 3D plan (3 depths of 6x4 areas to shoot at).
Sweeper2
Plays a Coco 3 graphics enhanced version of **Mine Sweeper**.
Towers of A
Towers of Atlantis - an 80 column color text version of the classic Tower of Hanoi game, where you have to move your disks one at a time so that they stack on one of the two outside towers. You can not place a larger disk overtop a smaller one.
COCODLE
CocoDLE
Plays a port of Rick Adams DECB version of **CocoDLE** (a Coco port of the popular **Wordle** game) to NitrOS9, written in **BASIC09**. Supports keyboard editing keys with your previous guess in the keyboard buffer.
SOKOBAN
Sokoban
Plays a 80 column text version of the popular **SocoBan** strategy game, where you push gold into special spots without trapping yourself or the gold. 50 levels included, and saves high scores. NOTE: There was a special font for this around at one time (I had it in the 1990's), but I have not been able to find it. If you have it, please email me at: ***curtisboyle@sasktel.net***

GFX_APPS

MVCanvas
MVCanvas graphics editor (ala Coco Max). Supports 4 different screen/color resolutions, and both uncompressed and compressed **VEF** graphic file formats. Mouse/joystick required, high resolution joystick interface (or keyboard mouse) recommended. The full manual is available on the **Color Computer Archive.**
Artist
Color Computer Artist - an early graphics editor that his it's own custom file format. Mouse/joystick required, high resolution joystick interface (or keyboard mouse) recommended.
FontDemo
Program to let you see each (or all) of the fonts installed in **EOU** (49 as of **EOU Version 1.0.0**). Shows both 6x8 and 8x8 fonts.
FontEd
An early first generation font editor, joystick required. Only does up to 128 character fonts.
fontman
A font editor, mouse required. Only does up to 128 character fonts.
HOME_PUBLIS
Home Pub.
Home Publisher by **Spectral Associates/Tandy**. Requires joystick/mouse. The 6309 version has been patched to properly handle it's internal system call overrides.

MUSIC

NOTE: The MUSIC folder & subfolders are a work in progress.

CCT

cctplay
Chiptunes player for 3rd party sounds cards by Todd Wallace, used to play music in the .CCT format. Defaults to the Zippsterzone **MegaMiniMPI's OPL3** sound chip, but you can edit the AIF file to default to the Zippsterzone **PSG (Programmable Sound Generator)** card's soundchip (**YM1249**) or the **GMC (Game Master Cartridge)** card's soundchip (**SN76489**), depending on which of these 3 you have available.

LYRA

This has a gathering of Lyra files (original for DECB), but there are 2 possible programs to run these with: **LYRA** itself (which has a native OS9 version where you can edit music), or **LYRAPLAY** which simply plays the music (and also has it's own MIDI descriptor & driver merged in; I am not sure if they are compatible with other MIDI player/editors like **Ultimuse**) so there are no icons for these yet, until testing (and deciding whether the editor or the player should be the default) is done.

MIDI

This is a gathering of raw MIDI files. There is a **MIDIPLAY** program (that requires a MIDI synth) that can run through either the serial I/O port on the back of the Coco with a MIDI cable, or it can use **MIDI Pak** hardware (I am guessing the same one as the current **MIDI Maestro** from **Retro Innovations**, but I don't have either to test with).

MUSICA

This is a gathering of (up to) 4 voice music files created by the DECB program **MUSICA**. Unlink a lot of the other music file formats, this does not require any external hardware (sound cards, MIDI, etc.) and works with a stock Coco 3. It also supports for the **Orch-90** and **Speech Systems** stereo 8 bit DAC cards, and you can edit the AIF to make one of those the default output device if you wish (it currently uses **-d** to specify the Coco 3 internal 6 bit DAC).
Musica II
This is the player program... don't double click on this one by itself; this icon is used to associate the extension **.mus** with the player. You can play any of the songs in this directory by double-clicking them. Hit any key during playback to abort the song.
ORCH_90
These are **Orchestra-90** based music files - but I am not sure which player program under **NitrOS9** will play them. Once I figure that out we will enable these (I know there is one; I just can't remember the program name).

ULTIMUSE
These are **Ultimuse .ume** music files, meant to be edited or played via **Ultimuse III+** itself, or through some separate players. Not sure what would be best to default here (it will also require a MIDI

synthesizer & either MIDI serial cable or MIDI hardware cartridge as well).

VGM

This contains the icon for **CCTPLAY** to use the **MegaMini's OPL3** chip to play **.vgm** music files (VGM stands for "Video Game Music"). This requires a **MegaMiniMPI** to play.

PICTURES

The icons in this directory are simply to notify **GShell** of which formats it supports with the **VIEW** program (written by Tim Kienztle). None of them are meant to be ran on their own; the folders contain the actual pictures to view in a variety of formats:

CM3

CocoMax III format pictures, including palette animation.

GIF

cross-platform **GIF** pictures. Most of these hav the extension **.GIF** and will display in color (with dithering) as best as the **VIEW** program can do on the Coco 3. If you rename a file to be **.GBW** instead, that will tell VIEW to run in a grey scale (4 color) mode instead, but with higher resolution (640x192 vs. 320x192) which sometimes produces better results.

MCP

Original **MacPaint** (512x384 black and white) format pictures go here. **VIEW** will scale it to the Coco screen (if you use **VIEW** from the command line instead, you can get it to display differently, including splitting it onto multiple screens vertically).
Note: previously these were .**MAC** files. But that caused a conflict with Mac sound files, so the images were changed to us the **.mcp** extension.

MGE

Color Max 3 & **Color Max 3 Deluxe** format pictures.

VEF

MVCanvas, etc. format pictures (the **Coco 3/OS9** standard graphics format. Aside from **GIF**, this is the only format that can display different resolutions (320 and 640 wide, 2,4 or 16 colors depending on the mode).

SOUND

This is a folder of sound (as opposed to music) files (although some are musical in nature). These are digitized sound samples in a variety of formats that are played by the PLAY command. Do NOT double click the ones just called "**PLAY**" - those are to define the icons for **GShell** for the various sound file types. It supports raw data (either signed or unsigned 8 bit), original **Mac** style sound files, Amiga **8SVX** files (**.amg** extension), **.AU** files from **SUN** computers, and **.WAV** files from **Windows** (this latter will handle 8 or 16 bit, mono or stereo, as long as they are uncompressed. The pitch will get more inaccurate at higher sample rates and/or bit depths further away from the Coco's native 6 bit DAC sound capabilities). **PLAY** can also play stereo to the **Orch-90** if you call it from the command line with a **-o** option.

UTILITIES

Control

Run the **Control** program (also available form the Tandy menu in **GShell**) in it's own full screen.

Towel

The **TOWEL** utility by Allen Huffman, which is a file/directory handling utility that runs in a hardware text screen, but allows both keyboard or joystick/mouse input. More powerful than **GShell**'s own file handling capabilities, but not as powerful as the full blown application (listed above) **MShell**. Uses way less memory than **MShell**, though.

Useful command line programs & utilities for non-programmers

BUILDDIR (utility program)

This program lets you build special floppy disks ONLY to be "schizo"; that is, they will show files under both **NitrOS9** and **DECB**. Type **builddir -?** for help.

BOOT_WIN (prank program)

Ok... this isn't *useful*, but it is funny. It's an old demo I wrote in the late 1980's and I used to have it as the beginning of the startup file (doesn't work as well with the new **EOU** start screens, so it's not enabled by default). But fun.

CORNERCLOCK (utility)

This can be run from any type of window (hardware text, VDG or a graphics window) and should be run as a background task (ie '**cornerclock&'),** This will run a clock in the upper right corner that updates every second, but sleeps most of the time so it takes almost no CPU time away from your other programs.

Please note that if you start typing a command in the SHELL, it will stop updating the clock until you finish typing in your command and hit ENTER, and then it will start updating again. You can run this multiple times on multiple windows too, if you wish. It also has 3 options that you can set:

- **-** (a minus sign) : display the clock inversed
- **h[xx]** : xx is a 1 or 2 digit number telling how many characters from the left the clock should start at.
- **v[yy]** : yy is a 1 or 2 digit number telling how many characters from the top the clock should start at.

Do not actually enter the **[** or **]** characters for the X,Y start positions. If you want the clock at 20,5, you would do: **cornerclock h20 v5&**

Please note that these coordinates are 1 based, not 0 based. Settings X,Y start coordinates would normally be used if a program wants to keep a clock at a specific point on the screen; if you put it in the middle of a larger window with a Shell, then every time the window scrolls a trail of old clock times would scroll up the screen. On the horizontal, please allow at least 9 characters to the right of your X coordinate start.

NOTE: CORNERCLOCK IS NOT FULLY FUNCTIONING ON THE 6809 VERSIONS OF NITROS9 (it starts fine, then sometimes disappears from the screen, even though it is still running).

DECB (folder)

These are programs that run under Disk Extended Color BASIC and can be launched from **NitrOS9** (they have to be a SINGLE file, self contained), (***WARNING***: This will shut down **NitrOS9** in the process - so think of it as "*Exit NitrOS9 and launch this program*").
The program used to launch these programs is called **RUSTY** and has it's own built in help (either type **RUSTY** by itself, or **RUSTY -?** to see this help). It can launch .BIN files (default) or tokenized .BAS files (**-b**); it also can launch with options: RGB or Composite colors, regular or double speed and a several others. Feel free to add your own favorites here using the **RSDOS** utility (see below)!

GPMAP (graphics buffer utility)

This program will list all active **GET/PUT** buffers in the system - including special groups for fonts, patterns, mouse cursors, etc. It has been updated to be faster, and it is now aware of more graphics groups than before, adding:

- **Stdwnd** (group $CE): **GShell / MultiVue** extended graphics
- **StdCd16** (group $CF): standard 16 color playing card graphic set (55 playing card buffers)

The new source code is in **/dd/SOURCECODE/BASIC09/gpmap.b09**

HELP (help utility)

The new **Help** command system allows for subtopics which has much more complete help vs. the older version. It has page pausing built in (even if not running **MultiVue**), and allows clicking the mouse button or hitting the spacebar to continue. It has a limited wildcard like function- if you type a few letters and hit **ENTER,** it will list all topics that start with those letters. While a lot of HELP files have been added in **EOU**, there is a long way to go still. Especially with things like subtopics - try getting help on **BASIC09** and see how deep you can go.

KEYCLICK (Key click utility)

This new utility will let you turn Key clicking on or off in whichever window (or VDG screen) that you run it from. By default, **EOU** is set up with keyclick turned on on the **/Term** window, and off on all others. You can change the default **/Term** setting by changing an entry in the **env.file**.

Usage is **KEYCLICK ON** to turn it on and **KEYCLICK OFF** to turn it off.

NOTE: If you want to run it from a batch file (like STARTUP, for example) you will need to used **KEYCLICK ON <>>>/1** so that it will keep the setting after the batch file is done running.

MEEP (standalone sound effect program)

This is a fun little audio digitized sample, baked into a standalone program.

PCDOS (disk utility)

This program will let you transfer files to/from real **MS-DOS** disks. It is slow, but it does the job. Built in help by typing:

PCDOS -help

It has special options for transferring text files (converting DOS's carriage return/line feed to OS9's carriage return, and vice versa). I believe it requires a real floppy controller (I haven't tried it on an **MSDOS** image of any sort). The standard floppy controllers can handle all **MSDOS** formats up to 80 track double sided (720K), if your physical drive can handle it. Full documentation in **/dd/docs.**

RSDISK (disk utility)

This program lets you transfer files to/from real **DECB** (**RSDOS**) disks, but with some extra options compared to the older **RSDOS** utility (see it's own entry below). It will let you specify different numbers of tracks (for ADOS, etc. style disks), which side of a double sided disk, and copy ALL files from a **DECB** disk to an OS9 directory/folder. Type **RSDISK -?** for help. Unlike **RSDOS,** it doesn't appear to work with the CocoSDC.

RSDOS (disk utility)

This program will let you transfer files to/from real **DECB** (**RSDOS**) disks. Built in help by typing:
RSDOS -help
Getting files is fairly simple; if you are putting a file onto a **DECB** disk, you will need to use modifiers to specify what type of file (binary, data file, etc.). Full documentation in **/dd/docs.**

RUSTY (DECB program launch utility)

As described in the **DECB** entry above, this program lets you load a DECB program (BIN or tokenized BAS), shut NitrOS9 down and launch the program. Multi-file programs can not be launched from here (without some trickery), but it will work with Coco 1/2 or Coco 3 programs, and has options (type **RUSTY** by itself for help) for monitor type, single/double CPU speed on launch, etc.

*NOTE: ON A **COCOSDC**, IF YOU HAVE SDC EXPLORER SET TO AUTOMATICALLY COME UP ON BOOT, YOU **MUST** HOLD DOWN THE SHIFT KEY WHEN YOU HIT 'Y' <ENTER> UNTIL THE PROGRAM BOOTS.*

Smartwatch Utilities

For those of you with a **Smartwatch** real time clock installed, 3 Smartwatch utilities (**GETCLK**, **SWREAD** and **SWSET**) have been installed. You should be able to replace the **SETIME** line in your **STARTUP** file(s) with **GETCLK**. This should initialize the OS9 software clock to your **Smartwatche's** time, meaning that you shouldn't have to type in the time in at all any more. (We still need to get the native drivers fixed, but this stopgap should help those of you with

a Smartwatch).

SWAPBOOT (NitrOS9 Boot file & environment swap utility)

This program will let you swap to a different boot environment, each with their own **OS9Boot** (the OS itself), **startup** file (batch file to run when booting) and **env.file** (system parameters used by **MultiVue** (**GShell**) and other programs). Several are included with each EOU release (how many and which ones will depend on which distribution of **EOU** you use - 6809, 6309, with or w/o **GIME-X**, etc.).

All sets are stored in **/DD/BOOTS**. A set will always have the same extension (an example is ".dw", for the bitbanger port version of **Drivewire**. **SWAPBOOT** will ONLY consider files that have a complete set (**OS9Boot**, **startup** and **env.file**, all with the same extension) for allowing you to switch which boot will be active the next time you reboot. You can add your own with your own extensions (maximum extension is 20 characters). There is special handling for boot files for emulators (**VCC**, **OVCC**, **MAME** and special **XRoar** releases); they *MUST* have "**emu**" as part of the extension name (this tells **SWAPBOOT** that the **OS9Boot** file will be written to a boot floppy rather than a boot hard drive). We have tried to make the set names self-explanatory as to what they support: "**dw**" means **Drivewire**, "**softclock**" means software clock, "**hwareclock**" means hardware clock, "**SDC**" means for the **CocoSDC**, etc. You can choose to follow that convention or name them something that makes sense to you (noting the special **emu** case noted above). One advantage of keeping reboots as a set is that you can do things like change the default monitor type and default screen resolution for composite vs. RGB, or make a bootfile with different default window colors so that you know which boot set you currently have installed just by seeing the windows come up, etc.

One thing of note: If you decide to create your own set, the **startup** and **env.file**'s you can make in **/dd/BOOTS** with a text editor (probably easiest to copy an existing one and then edit them to customize them) fairly easily. For the **OS9Boot** (until the next release of **KwikGen** is ready), the easiest way is probably to **kwikgen** the ACTIVE **OS9boot** (on /dd, or /d0 if using an emulator), and then copy/rename it to **/dd/BOOTS** with the proper boot set extension name, and then run **SWAPBOOT**. This way you will keep your new settings in a set. If you edit **startup**, **env.file** and **OS9Boot** for the current boot (so **/dd/OS9Boot** (or **/d0/OS9Boot**), **/dd/startup**, and **/dd/SYS/env.file**), those settings will stay until the next time you run **SWAPBOOT**, and then those changes will be lost (since those files will get overwritten by the new set).

Default sets included are:

- Standard SDC (**.sdc**)
- SDC with DriveWire (**.dw**)
- SDC with DriveWire via RS232 Pak (**.dw_rs232pak**)
- Emulator with software clock (**.emusoftclock**)
- Emulator with hardware clock (**.emuhwareclock**)

GIME-X versions currently only have the first 2 options listed above.

UNZIP (zip file extraction utility)

This works on **ZIP** files created on a multitude of systems, allowing you to **unzip** the files on your Coco. Good way to get a group of files to **EOU** from a real floppy (see **PCDOS**) or onto VHD images from your modern computer's OS. Type **UNZIP** by itself for help. There are two of the many options that are usually the most useful; **-a** will convert CRLF line endings for **MSDOS** to just CR for **NitrOS9**; and **-d** to create any subdirectories that are in the original **ZIP** file.

VED (powerful text editor)

This text editor is written in machine language, so it is quite fast and compact (the program itself takes less than 7.5K) with a lot of powerful features (including importing/exporting text, text rules, global search and search/replace, and a ton of others) and leaves you a 53K buffer to work with. It is my personal favorite, although there are many more editors included in EOU if you prefer them (**SCRED**, **ED**, **EDIT**, **MintEd**, **Sled**, **uemacs**, **vim** (which is **VI**), etc.). If you want to learn **VED**, it's original author (Bob van der Poel) has let us upload the entire manual for version 2.9 in PDF form, which you can get here:

http://lcurtisboyle.com/nitros9/EOUDOCS/VED2.9_manual.pdf

VIEW (Multi-format graphics file viewer)

This multi-format graphics viewer was mentioned earlier for viewing common Coco 3 graphics formats like Coco Max III, Color Max 3, RAT, VEF, GIF, etc - but it does many more specialty formats, lets you export a graphic from any of types it can read to VEF format (compressed or uncompressed), time how long an image is displayed, and more. Type **VIEW -help** for full options, and **VIEW -formats** to get a full list of the formats supported. This includes Coco 1/2 formats, clipart from MVCanvas & Home Publisher, DS-69 digitizer pictures, Compuserve RLE, Atari ST DEGAS, and more.

*NOTE: On the **6309 version of EOU** is a special experimental version that has both extra features and speed optimizations. It is NOT the default one, as it has bugs with some Coco Max III files. But it has multiple additional features that you may want try. You can load this version on EOU/6309 thusly:*
LOAD VIEW-45a-6309
Aside from speedups on some formats (up to ~50% on some GIF's, 20-30% on MGE, etc.) and using 200 vertical lines to draw rather than just 192, it adds the following features:
***-bufgif** : Makes buffered get/put buffers to do fast animated GIF's, and (mostly respects the Netscape extensions for times between frames, etc.) This takes more RAM, but can run very fast. Supports up to a maximum of 225 animation frames, and/or free RAM on your machine.*
***-sml :** On GIF's, will rescale a GIF to the largest size that can completely fit on a screen and keep the pixel ratio the same.*
***-big** : On GIF's will scale a GIF to the width of the screen (may cut off the top or bottom if it is too tall).*
***-huge** : On GIF's will draw a GIF at 1:1 pixel ratio, possibly cutting off parts to big to fit from all 4 sides.*
***-ss :** Similar to the original **-s** (draw on current screen that **VIEW** is running on), except it will NOT clear the window first. You can use this to draw on a current graphics screen without destroying any parts of the screen that it doesn't need to draw on. Because **VIEW** normally "best guesses" what graphics mode to use depending on the picture specs, you usually want to use this with the original **-t#** option to force the screen type (5,6,7 or 8).*
***-sx### :** Specify the X coordinate that the left side of the GIF will start displaying on the screen (to position a smaller GIF horizontally).*
***-sy### :** Specify the Y coordinate that the top of the GIF will start displaying on the screen (to position a GIF vertically).*
***-signal# #** - This takes two numeric parameters, with a space between them. The first parameter is the process ID that you want to send a signal too, and the 2nd is the signal # you want to send. **VIEW** will send the signal was a graphic is finished rendering, to the let a program know that it is complete.*

NOTE: If you have no use in viewing Coco Max III files at the moment, you can make View 4.5a the default on the 6309 version of EOU by doing:
CD /dd/cmds;CP -ov view-45a-6309 view

VU (powerful full screen text viewer)

This only runs (currently) in an 80x24 hardware text screen. It allows viewing text files of any size, lets you jump to a specific percentage of the way into a file, lets you scroll left/right to read files that have lines wider than the screen, has built in help (hit '**?**'), lets you go up or down to the beginning/end of the file /screen/line, global search, and writing out to a file (or printing) the current screen out. Really fast too, since it writes directly to screen memory rather than print through the operating system itself.

CGA full 224 character font

Thanks to Todd Wallace, we have a duplicated of characters 32-255 from the original IBM PC CGA font, including fancy line drawing characters, etc. This has been added as buffer number $42 in the $C8 font group. If you want try it on a 40 column (4 color) screen type the following in full screen **Shell** window:
DISPLAY 1B 24 1B 20 6 0 0 28 19 0 2 2 1B 3A C8 42 1B 21

And to see it on an 80 column (2 color) screen:
DISPLAY 1B 24 1B 20 5 0 0 50 19 0 1 1 1B 3A C8 42 1B 21

ENV.FILE DEFAULTS

EOU Version 1.0.0 has some specific defaults set when you download it that are different depending on whether you are running from real Coco 3 hardware, a GIME-X test system, or an emulator. These are based on the minimum capabilities of each, but you can change them (what and how to change is found in the separate "**env_file_documentation.rtf**" file that comes with an EOU download):

A) Real Coco 3 (6809 or 6309)
- defaults to Composite video mode
- defaults to 40 column /**TERM** window and 40 column **GShell.**

These settings makes it readable on all systems, but if you have RGB or VGA output (or good eyes!) you can change this (See separate documentation file "**env_file_documentation.rtf**" for details).

B) Real Coco 3 with GIME-X (6809 or 6309)
- defaults to RGB video mode
- defaults to 80 column /**TERM** and **GShell**.

C) MAME, VCC or OVCC Emulators (6809 or 6309)
- defaults to RGB video mode
- defaults to 80 column /**TERM** and **GShell**, and software clock.

(The software clock setting is the default because a lot of people who have downloaded the various emulators for the first time do not know about (or how) to set up the additional virtual hardware needed to enable a hardware clock. There are notes in the **Beginners** documentation included in **EOU** on how to turn the hardware clock on for some of the emulators).

It should be mentioned that there is a new version of the **SysGo** module (this is what actually boots up the user experience, including starting up your initial windows). This module now reports some of your hardware set up (RAM, Drivewire, CocoSDC, etc.) and also processes part of the **/dd/SYS/env.file** to set up some system & window defaults. See the separate **env.file** documentation for a full list, but be aware that you can change a lot of settings by editing the **env.file** text file. (The older **SysGo** did most of it's initialization based on the **Init** module; but this is beyond the capability of a casual user). There is also a special option for owners of a **Coco3FPGA** (thanks to Michael Furman), which bypasses checks for a **GIME-X** and **CocoSDC** (and caused a crash on the **Coco3FPGA**. If you have questions/comments on this functionality (or using **EOU** with a **Coco3FPGA** in general, please contact Michael directly. (He is **mikeyn6il** on the Coco Discord, and is on the Coco Facebook group).

The **DriveWire** detection can generate several responses:
- **Not Detected** : No **DWIO** driver in the **OS9Boot** file.
- **Installed** : There is a **DWIO** driver in the **OS9Boot** file
- **(Inactive) - Checking** : This should only show up briefly if Drivewire is installed and it is checking to see if it can access at least one of drives **/X0**, **/X1** or **/X2Active** : The driver is installed AND it has detected a **DriveWire** server hooked up and running. It will also report the type or version of **DriveWire** installed.

2) Updates for Applications programmers

This section will cover updates that programmers of regular applications will most likely be interested in. These will be command line applications in almost all cases, running from **/dd/CMDS**. It should also be mentioned that fair bit of source code is included in **/dd/sourcecode/*** which covers assembly, BASIC09 and C source code for a variety for programs.

The currently most complete documentation for the installed **Shellplus 2.2a** is in **/dd/DOCS.** It is in 2 parts; the original Shellplus 2.1 (which covers almost all the new features in **Shellplus**), and Shellplus 2.2a (which covers all the command line history additions & some bug fixes). All of this documentation will be included in the upcoming updated **OS-9 Commands** manual, once it is completed.

AWK

Walter Zambotti's port of the Unix scripting language for processing text. **HELP AWK** to get a list of options. You can also see a small sample AWK program with a couple of data files in **/dd/AWK_SAMPLE.**

BASIC09/RUNB

Both the 6809 and 6309 versions have some speed enhancements now. **RUNB** for both CPU's have both been shrunk as well. The 6309 versions get a bigger speed increase than the 6809 optimizations. Here is a list of some of the optimizations (6809 is usually 1-5% depending on the operation; 6309 can be 10-16% faster). The actual speedup will depend on the values being used as well as what functions
First up, for both CPU's: Some special circumstances that **BASIC09/RUNB** has special optimizations for that are not documented in the BASIC09 manual:

1. **INTEGER DIVIDE:** If the dividend (the number you are dividing) fits as an 8 bit number, it is optimized to run through the loop up to half as many times as a 16 bit number would take. Which means it can take up to almost 50% less time.
2. **INTEGER DIVIDE** or **MULTIPLY** by 2 are specially optimized. In the case of **INTEGER DIVIDE**, dividing by 2 twice in a row is actually a little faster than a single divide by 4. Zero divided by anything also is specially optimized.
3. For any trig functions that you use, using **Radians** instead of degrees saves around 1000 CPU cycles per function (Radians is the "native" method used; it has to convert from degrees to Radians which slows down using Degrees instead).

Expected speedups by CPU:
6809:

- **INTEGER** based: Multiplication (1-3% faster), Division (3-5% faster), MOD (3-5% faster), FOR/NEXT loop (2-2.5% faster if STEP value <>1).
- **REAL** based: Addition/subtraction (~2% faster), multiplication (2.5-3% faster), division/MOD (2.5-3% faster), FOR/NEXT (.5-1% faster).
- **LIST**ing a source code line (either in EDIT or DEBUG mode) is slightly faster.
- Passing parameters to another procedure is marginally faster.
- Very slight speed increase copying a string literal.

6309:

- **INTEGER/REAL based:** Some of these were already optimized in **BASIC09** even in NitrOS9 3.3.0, and now **RUNB** has been caught up. Some functions like **Integer Divide** and **Integer MOD** can be up to 50% faster. Some others are 10-16%, and a few are lower. Optimizations for 6309 are: REAL addition, subtraction, multiplication, division/MOD (up to 10% faster), FOR/NEXT loops; INTEGER addition, subtraction, multiplication, division, MOD, FOR/NEXT loops.
- **LIST**ing a source code line (either in EDIT or DEBUG mode) is slightly faster.
- Passing parameters to another procedure is marginally faster.
- Plus all of the previously done (3.3.0) **BASIC09/6309** optimizations have been backported to **RUNB/6309** as well.
- Very slight speed increase copying a string literal.

RUNB for both CPU's comes with **INKEY**, **SYSCALL**, **GFX** & **GFX2** all pre-merged (this includes new versions of **GFX** and **GFX2**; see below).

BASIC09 for both CPU's comes with **BNKEY, BYSCALL** and **BFX** pre-merged (for consistencies sake, **BFX2** is also in the **/dd/CMDS** directory, although it is not pre-merged). The B versions are identical to the G versions merged with **RUNB**; the only difference is the first letter of the module names. Why, you may ask? One problem with developing large programs for **BASIC09** is that **BASIC09** takes 24K for its own code and at least 8K for your program, variables. etc. If you have **INKEY**, **SYSCALL**, **GFX** & **GFX2** pre-merged into **RUNB** (which all combined takes just under 15K),

any time your try to call one of those 4, it has to map in the entire set fo files merged with them - which includes **RUNB.** So calling INKEY (which is <100 bytes) actually will take a whopping 16K out of your **BASIC09** memory map, which leaves you a maximum of 24K to used for your own program and variable storage (and over a 1K of that is for **BASIC09**'s internal use). By making a merged **BASIC09** with the special named versions of 3 of those modules, then if you call any of **BNKEY, BYSCALL** and **BFX**, that will take 0K extra from your program. Even if you have to call **BFX2**, it will only take 1 8K block, giving 8K more back to your program (so up to 32K if you use **BF2**, and up to 40K if you only used **BNKEY, BYSCALL** and/or **BFX).** This makes it easier for you to make larger programs and still have full used of the editor and debugger. When you get your program finalized, you can just do a quick global search/replace to rename all calls to these special "**B**" versions back to their original names before **PACK**ing the program (**RUNB** can actually up to 48K for you program in memory at once at this point). You can still **CHAIN** & **RUN** procedures in your 64K space as well, and **KILL** any modules you don't need (ie take them out of your **BASIC09** workspace)

Of course, if your program easily fits into 24K, then you don't need to use any of the special modules. But it's a nice option to have.

Related to **BASIC09** and **RUNB** changes is updates to both **GFX**/**BFX** and **GFX2/BFX2**:

GFX / BFX

There are some very slight speed/size optimizations to this module for using Coco 1/2 graphics commands. The biggest change here is the addition of **FFill** command (Flood Fill) which will paint the current foreground color starting at the current graphics cursor location. There were always display codes to do this; but for some reason it was missed by Tandy/Microware in the **GFX** package.
There is a small test program demonstrating both **FFill** and **GLoc** in:
/dd/SOURCECODE/ASM/BASIC09/TEST_PROGRAMS/gfxtest.bas.
(There is a mistake in the manual, where it says that you can use call **GLoc** with a **BYTE** variable. It needs an **INTEGER**).

GFX2 / BFX2

There are extensive additions here, which are fully covered in the file **'gfx2_edition5 (EOU Version 1.0.0.txt)'**. This covers the additions originally added by Kevin Darling and Kent Meyers back in 1990, plus further additions and changes by us (there are also some general small optimizations as well):

Title - to set window title and sizes
Menu - to set window menus and whether active menu or not
Item - to set menu pulldown items and whether a menu item is active or not
WnSet - set window type (borders, scrollbars, etc.)
Winfo - return window information
GetSel - return menu selection
UMBar - update menu bar
SBar - update scroll bar positions
SetMouse - set mouse parameters (including turning autofollow on/off)
Mouse - return mouse info
OnMouse - sleep until a mouse button is clicked OR a key is pressed
ID - return calling processes process ID
Tone - play a tone
FCircle - draw a filled circle
FEllipse - draw a filled ellipse
Draw - draw based on draw string. No changes to the command itself, but new to this documentation is that you can shorten your **Draw** string by not putting commas between draw directions; commas are only needed for separating X and Y coordinates. **Draw** also supports optional startx,starty coordinates before you specify the draw string. (ie: **RUN GFX2([path],"Draw"[,startx,starty],Drawstring)**. And finally, the parsing of numeric values in the Draw string is slightly faster on both 6809 and 6309.

For those people who watched the seminar that Ken Waters & L. Curtis Boyle did at the **2022 CocoFest** show on using these new commands for doing **MultiVue** menus, the source code used in that presentation can be found in:
/dd/SOURCECODE/ASM/BASIC09/TEST_PROGRAMS/menus_newtest.b09.

The seminar itself can be found on Ken's YouTube page:
https://youtu.be/si1NDycoP6g

It should be noted it that the included version takes advantage of the **OnMouse** change listed above (which was not present during the seminar - allowing for both a mouse button click or a key press).

CALL

Scott Griepentrog's CALL utility. This allows performing a command on multiple files with a single command line. Full documentation in **/dd/docs/call.doc**.

Card Decks *(graphics buffers)*

A nicely done set of 16 color card buffers (55 in all) to draw a complete set of playing cards. They are built to use get/put buffer group 207 ($CF), but are not loaded by default. However, if programs are written right, they can be shared by multiple card games running in the system at the same time.
The buffers themselves are in **/dd/sys/carddeck** and sample BASIC09 code to show how to load and use them is in **/dd/sourcecode/basic09/cardbase.b09.** This has 3 commented subroutines and a small test program to work with the cards. They display fast even on a 6809, possibly allowing card animation.

Enhanced CC3Disk documentation

In **/dd/docs/cc3disk.doc.** The newer CC3Disk driver (for real floppies) can now handle non-OS9 format disks, 256 and 512 byte sector sizes, and drives that start sector numbering at 0 or 1. This .doc file explains how to use these features.

CHOWN

Utility by Jeff Teunissen to change the owner of a file or directory. If you are the superuser, you can change it to any user number.

DCC

This new (and still a work in progress) C compiler for NItrOS-9 Level II being written by **Jeff Teunissen**. This is a rewrite of original C Compiler that Tandy sold. It should handle larger more complex projects, handles unsigned longs, is a little faster, uses a little less memory (and programs it generates will also be a little faster/take less memory). It adds ISO C "stringizing" and token pasting to the #define directive. The optimizer now supports configurable pattern files, rather than hardcoded assembly recognition patterns. This will allow for special optimizations for Level 2 NitrOS-9, 6309 code generation, etc. in the future. There also updated **Library** and **Defs** files for the **C compiler** and **RMA** as well.

Other additions, changes, and bug fixes are:

- Adds the **#error** and **#warning** directives (error will stop a compile when it finds an error, while warning will let the compile continue but still give the programmer warnings).
- The preprocessors also add a few warnings (it should be noted you should use **CC** and **CPP** and not **C.PREP** and **CC1** (if you want to fall back to the older compiler) due to some of the updates. Use **DCC** for all of the newest updates and fixes).
- Fixes switch so that it evaluates an expression properly; ie switch (x+10) now works (it originally ignored the +10).
- **asctime()** has (finally!) had Y2K fixes (so you get "2022" instead of "19122").
- **_prgname()** and **_errmsg()** have been fixed to not put garbage ahead of the program's name.
- Tabs are converted to spaces by the preprocessors, so error messages won't mess up the screen anymore (***why*** did Microware pick the TAB character to move the cursor Up?) (I have wondered that myself! - LCB)
- **DCC** should never leave dead processes running the background anymore; **CC** sometimes did.
- The compiler now treats strings next to one another as a single string, like modern C. These helps with macros and really long strings (like multi-line ones).
- The assembly optimizer can now do Level 2 specific optimizations (using -2 with DCC), and can now be customized using /dd/lib/level1.patterns and /dd/lib/level2.patterns.
- **atof()** is much more accurate.
- **sleep()** no longer hangs under Level 1.
- **mkdir()** has been added.
- **u2otime()** now correctly converts from Unix time to OS-9 time, and also fixes the month when converting C struct tm to OS-9's "getime" format.

You can also grab the manuals that match the current **EOU** release here:
Users Guide: http://www.lcurtisboyle.com/nitros9/EOUDOCS/dccguide.pdf
Library Reference: http://www.lcurtisboyle.com/nitros9/EOUDOCS/dcclibref.pdf

NOTE: Currently **DCC** only outputs 6809 code, not 6309.
NOTE: Jeff is constantly working on the **DCC** suite of programs, so if you want to get the latest releases, you should get it directly from his GitHub:
https://github.com/Deek/CoCoC

The version included with **EOU** 1.0.0 is from mid November of 2022.

Jeff has also made a set of utilities using his new compiler that are included (including **CHOWN** above):

- **GREP / EGREP / FGREP** - Search for a pattern of characters in files. **GREP** is the regular version, **EGREP** is **Extended GREP**, and **FGREP** is **Fixed GREP.**
- **LEX** - generates lexical analyzers. Commonly used with **YACC** (See below).
- **SED** - Stream editor - edits input files to output files using scripts.
- **TR** - Translate characters
- **YACC** - "Yet Another Compiler-Compiler" - A Look Ahead Left-to-Right (LALR) parser generator.

/DD & /H1 descriptor updates

The **/dd** and **/h1** descriptors have been updated to more closely match the equivalent CHS drive geometry, for some older utilities. (The VHD 'drives' actually use LBA mode and ignore the CHS settings for most functions).

Debug

Bug fix for the memory dump option.

DEFS Directory Updates

The **LIB** and **DEFS** directories have been updated to newer versions with new definitions for both assemblers and the C compilers. A new **cstart.r** from Jeff Teunissen as well. There is differences between the 6809 and 6309 versions - most of the register stack offsets are different due to R$W being inserted for the 6309 stack frame). The new ones are called **sys.l_6809** and **sys.l_6309**, and you can use **RLINK** (in your makefile, or manually) to choose which one you are using.

DIR

Updated to edition 9. This shrinks the code slightly (and should be marginally faster) and fixes a longstanding bug when filenames >15 characters are present while running in a window with <48 character width.

DISASM

Updated to edition 6. This fixes a bug that would incorrectly disassemble PULU or PSHU instructions to try and push/pull U on it's own stack. It now pushes and pulls S onto the U stack, as it should. It also adds support for the new IOMAN system call I$ModDsc (see the **System Programmers** section for details. **GShell** is already using this new call).

DISPLAY

The DISPLAY command with 3.3.0 had a bug when processing decimal values - it would produce incorrect results for any '0' digits in the decimal number. This has been fixed.

DOSDIR

Todd Wallace's new **DOSDIR** (an **MS-DOS** style **DIR** command has been added, including sourcecode in **/dd/SOURCECODE/ASM**. **DOSDIR** formats the directory display like **MSDOS**, and reports both file sizes and free space left on the current drive (floppy or hard drive) in decimal instead of hex.

DUMP

Updated to edition 8. Now handles '-' (dash) characters in filenames (instead of exiting with no error message at all).

FILES

Command added to **/dd/CMDS**. This utility accepts * and ? wildcards, and outputs one filename per line (helpful for reading through directories, etc. that you can pipe or save into an output file, rather than have to write your own wildcard parser).

FONTSPLIT

New utility added to split a file with multiple fonts into separate font files with the font # as part of the filename. Source code written in **BASIC09** is in:

/dd/SOURCECODE/BASIC09/FontSplit.b09

NOTE for DECB users: The 8x8 fonts (except the 224 character fonts) are compatible with DECB, if you load them into the right spot (and skip the 11 header bytes that NitrOS-9 uses).

All (almost) 50 of them.

FORMAT

Updated to edition 26. This single **format** command combines the 3.3.0 **format** and the separate utility **format20** (that allows a special format with 20 (instead of 18) sectors per track on real floppy disk drives). Just use the new '**e**' option to enable the enhanced track format. This will increase the storage on your really floppies as follows:

Drive type	Original capacity	New 20 sector capacity
35T, SS	161,280 bytes (630 sectors)	179,200 bytes (700 sectors)
40T, SS	184,320 bytes (720 sectors)	204,800 bytes (800 sectors)
40T, DS	368,640 bytes (1440 sectors)	409,600 bytes (1600 sectors)
80T, DS	737,280 bytes (2880 sectors)	819,200 bytes (3200 sectors)

GShell - AIF file enhancements

There are several additional features that you can use in AIF files (and old AIF files still work - the new features are backwards compatible):

1. **Launch a program in a VDG window: GShell** previously only allowed screen types 1-2 and 5-8 (for 40/80 column text screens, and 4 different graphic screen types respectively). You can now also specify a type 0 (Zero) for VDG. You must also set the X & Y size of the window in the **AIF** file to 32x16. So, line 5 of an **AIF** file is the screen type (set to 0 for VDG) and lines 6 and 7 are the launch window size (so set these to 32 and 16 respectively). (This uses the **I$ModDsc** system call explained in the System Programmers section, so it won't take a large chunk of RAM away from **GShell** that it should be using for directories). **Please note:** to enforce backwards compatibility, color settings in the **AIF** are ignored for VDG windows.
2. **Custom name an application**: You can now have an optional application name to print on **GShell**'s screen under the applications icon, rather than just the name of the application itself. To do this, edit the first line in an AIF (which is the name of the application to launch - the program name found in the **CMDS** directory), and add a pipe 'I' symbol (**CTRL-!** on a real Coco 3 keyboard), followed by the "alternate" name to display. The name before the pipe still has to match the name of the program in the **CMDS** directory, but the 2nd name can be whatever you want. Please note that you can only fit 11 characters on screen for the 80 column/640 pixel mode in **GShell**, and only 10 characters for 40 column/320 pixel mode). Anything past that will be ignored, and I recommend only using up to 11 characters (spaces are ok) to shrink the file sizes (and **GShell**'s internal tables) as small as possible.

IDENT

Updated to edition 9. This fixes a bug where it won't work on files with a dash ('-') character in it (like **DUMP** above), and shrunk slightly in size.

O9GIF

This GIF viewing program has been added, as some older games require it. While it doesn't handle high color GIF's quite as well (or as fast) as **VIEW**, it does take a much smaller memory footprint.

SDC2

Bill Nobel has added a new **SDC2** utility which is an easier to use/remember commands version of Barry Nelson's earlier **SDC** utilities (which are also still present in case you have scripts set up). It has built in help, but will only run if a **CocoSDC** is detected.

TMODE / TMOD2

Because we are missing source for a fair number of older programs, we have restored the original **TMODE** and renamed the new, incompatible syntax version to T**MOD2.** (for both of the above, it was figured that source code is much more available for more recent software and thus it is easier to patch the newer ones to use the *2 versions. We do eventually plan on making a "fat binary" version of both that will accept both syntaxes, and then all programs can just call the one version.)

XMODE / XMOD2

For the same reasons as listed in **TMODE / TMOD2** above, we have restored the original **XMODE** and renamed the new, incompatible syntax version to **XMOD2.** As with **TMODE**, we do plan on making a "fat binary" version that will accept both syntaxes.

General programming notes

1) Get/Put buffer oddities *(that can be solved with programming)* - I have noticed on the **Snakebyte** game (as one example) where some get/put buffers (or even fonts) either appear corrupted or sometimes return "Buffer Size too small" errors if you run **Snakebyte** first, exit it, and then run a different program. From what I am able to determine, this is because **Snakebyte** doesn't clean up it's own **Get/Put** buffers after itself **and** if the next program doesn't pre-clear (kill) it's own buffer group before sizing/reloading them, then you may experience these errors. Most programs do perform **Get/Put** buffer cleanup on exit, but others (especially older ones) do not.

For programmers, probably the safest way to deal with situations like this is to have your own program send a **KilBuf** for your **Get/Put** buffer group for the whole group (usually the group # is your own process # so that you know it is unique to you); so you would send a **1b 2a <process#> 0)** display sequence (or use **KilBuf** from **BASIC09's GFX2** module, or the equivalent in **C**) and then size, load/get the buffers needed for your own program.

This way, you yourself can clean up after any misbehaving program, and you can make sure your **Get/Put** buffers are the size that you actually want.

In the meantime, if you quit the 2nd game (which will then usually clean that group up itself), and then relaunch it, it will be fine.

2) Many program's source code (and smaller sample sourcecode) are included in the **/dd/SOURCECODE**/* directory tree. This includes ASM, C and BASIC09. Not all of it has been tested extensively; they are meant more to be for educational purposes. Some prominent examples:

- Allen Huffman's SNDDRV driver
- sourcecode for Alan Dekok's REBOOT utility
- sourcecode for Alan Dekok's GLIB library routines for doing sprites. This includes the documentation (in **/dd/DOCS**) and a sample demo program. (Of note: this library is used in the **Smash** game).
- sourcecode for NitrOS9/EOU itself
- sourcecode for SWAPBOOT
- Small test programs for some of the newer system calls and features (most of these are in the **BASIC09** sourcecode folder tree)

Bug fixes & new NitrOS9 operating system features

CoVDG

Has the following changes:

- The **LINE** drawing routine for medium resolution graphics had a sign bug in 3.3.0 and could actually crash the machine (or just draw wrong) depending on coordinates sent to it. This has been fixed.
- Reinstated **Select** checks so that the system no longer allows you to **CLEAR** other screens if double buffering was used.
- **Erik Gavriluk**'s more accurate RGB to composite color conversion table added.
- Minor speed and size optimizations.
- 1 known bug remains: occasionally an extra linefeed will on the 32x16 screen, but only once per VDG window (after it is initialized). The bug may not even be in **CoVDG**, but elsewhere.
-

CoWin

Has the following changes:

- Bug that caused the whole system to crash if trying to load a >8K **Grfdrv** fixed (affected 6809 version in particular)
- Minor speed and size optimizations, including to: **DefPal, Arc, Circle, Ellipse, FFill, FCircle, FEllipse**, **Get, Put** and **Point.**
- Change to use different system font for menu bars that only actively uses 7 vertical pixels. This helps to preserved the 3D look of the menu bar by not having text go through the top of the bar.
- **DefColr** in CoWin is faster & shrunk on 6809.
-

Grfdrv

Has the following changes:

6809 version:

- Screen scrolling (especially full width text and graphics screens with no borders) are now about up to 42% faster on a full graphics screen and 37.75% faster on a full 80x25 hardware text screen.
- **GPLoad**'s are faster
- **Get/Put** buffers with large widths are faster
- **OWSet / OWEnd** (overlay window set and end (restore) are faster.
- Dramatically faster clearing of full width graphic screens. Like 340% faster.
- Non full width graphic screen clearing is also noticably faster, as are Clear to End of Line & Clear to End of Screen.
- Horizontal **lines** with no Logic or Pattern settings enabled are faster (particularly on wider lines)
- **Filled Circle**, **Filled Ellipse** and **Bar** are significantly faster (about 1.75 times faster) if no Logic or Pattern sets are enabled.
- Bug fix for **ARC** with scaling - was using 16x16 unsigned multiply - it needs to be signed in certain cases.
- **LSET** with **AND** or **XOR** logic on wider parts of the screen are faster
- Other minor optimizations.
- The **Transparency Character Switch** now does something on hardware text screens. It will let you print text with whatever foreground color, underline and blinking options you have turned on, but it will leave the background color to whatever was already on the screen at that location. You can try a test program to see this in action in:
  /dd/SOURCECODE/BASIC09/Hware_Transparent.b09

6309 version:

- Clear screen, Clear to end of screen and Clear to end of line are a little faster.

Both 6809 & 6309 versions:

- **FFill -** bugfix improvements. There was a bug where painting with a Pattern Set enabled would occasionally get stuck in an infinite loop that locked **Grfdrv** from responding to the rest of the system (if you had a serial terminal hooked up, it would still run fine). The fix uses more stack space though, so it can trigger a stack overflow error more often than previously (it should be noted that the original stock Tandy/Microware version could also trigger a stack overflow error on very complex paint errors). So, when filling (painting), try to make sure that you are doing smaller, enclosed areas to prevent such an error from happening.
- **6 pixel wide & proportional fonts** - these are now a few percent faster.

IOMAN

This has a brand new system call added to allow your programs to modify device descriptors without needing to map

them in (and currently if they are merged with other modules, the system brings the entire package in, which could cause your application to run out memory in it's 64K workspace). Full details are in section **3) Systems programming information** below.

SELECT WINDOW BUGFIX (multiple modules involved)

- There was a bug in the **Select Window** internal call that would "trap" the CLEAR / SHIFT-CLEAR routines (to let you switch windows) so that any application that used more than one screen itself (for double buffering, etc.) would force the windowing system to go back to the application every VSYNC (1/60th of a second), "trapping" the user in that application until it was ended or aborted. This has been fixed. (Some programs that were affected were:
  VIEW, **VIEWGIF,** "Mona Lisa" **Bouncing Ball** demo, **RayBounce** ray traced balls demo).

SIERRA (the program itself that launches all Level 2 Sierra On-Line games)

The currently installed version is the one that is most compatible with modern Coco systems - but not all. It will work with all 512K Coco 3's, and any Coco 3's with the following 2 MB RAM RAM upgrades:

- Cloud9's **Triad+**
- Zippsterzone's **2 or 8 MB RAM upgrades**
- Boysontech's **Boomerang E2** (revision 2 or above)

The current version of **Sierra** will crash on an old style (like **Disto**) 1 or 2 MB RAM upgrades, do to how the MMU registers get read (vs. write). Robert Gault has made a patch for the **Disto** upgrades, but then it crashes on 512K machines. So we have gone for the most common as the default, but if you have one of the older >512K upgrades, you should be able to get Robert's patched version from the respository.

NOTE: These currently contain self-modifying code, so you should only try to run one Sierra game at once. If you want to switch games, **ALT-Z** will exit any of them, and then you can switch folders and try a different one.
I am currently looking into what is needed to fix these up so that the same versions of the program modules will run cleanly on ALL 512K and higher RAM machines (and maybe even fix them so that they don't need self-modifying code to run). But that's a large project.

Also related to **Sierra** games is patches to the **MnLn** program that were made by Guillaume Major which are also included. He has patched this module to allow the **BREAK** key to abort music while it's playing (the original version disabled the keyboard until the complete music piece stopped playing).

3) Systems programming information

New features are generally listed in the new **Technical Reference Manual**, which you can get here:
http://www.lcurtisboyle.com/nitros9/EOUDOCS/NitrOS-9_EOU_Level_2_Technical_Reference_Manual.pdf

EOU *.VHD hard drive images are "SCHIZO DISKS"

The main hard drive images (*.VHD) are now "Schizo" images, thanks to David Ladd. This puts a very small Disk Basic partition onto it, where it has AUTO.BAS preinstalled (this is for running on a **CocoSDC** but has no effect when they are mounted in an emulator).

If you are one who spends the vast majority of your time in NitrOS-9, if you 1) Rename this file to **AUTOEXEC.BAS** and 2) Point your default drive in the SD card's config file to your 6xSDC.VHD, your Coco will autoboot **NitrOS9/EOU** on power up.

DWIO driver updated

A new version of the **DWIO** driver is installed on both the 6809 and 6309 Drivewire boots. This now preserves the DriveWire server version #, making it available for future use.

EMUDISK updated

Much thanks to **Ed Jaquay**, who solved a long standing bug in the **EMUDSK** driver (the virtual hard drive driver used by **MAME** and **VCC/OVCC)** that caused problems when copying from **/h1** to **/dd**. It has also been slightly optimized. I should now also boot properly from hard drive 0 if you reboot after last using hard drive 1.

IOMAN updated

The I$Attach call is slightly faster on the 6809, and allocating a path descriptor is slightly faster. The big change is the addition of the new I$ModDsc (SWI2 $91) system call (documented in the new **Technical Reference** manual found here):
http://lcurtisboyle.com/nitros9/EOUDOCS/NitrOS-9_EOU_Level_2_Technical_Reference_Manual.pdf

For quick reference it is also described here in this document:

I$ModDsc - Modify Descriptor in Memory
Modify **any** bytes in a Device Descriptor module ONLY, *after* the module header.
Unlike **F$Link**ing a module, this call lets you modify the module without having to link the entire file block it is in (like the OSBoot file, which can be 32 or 40K) into your processes' workspace.
Entry Conditions:
X=Ptr to name of module to modify (Can be high bit or CR terminated)
B=# of bytes to change (max 127)
U=Ptr to start of 2 byte change pairs (2 * B)
Byte 0 is the offset into the descriptor to change (>M$DType offsets ONLY)
Byte 1 is the new byte value to write to that offset
Exit Conditions:
CC=Carry clear if no errors; also the descriptor is updated with the header parity and CRC updated as well,
OR
Carry set if error
Error output:
B=Error code (if any). Some possible ones:
$D8 (216) File Not Found error: If the specified module name is not currently loaded in the module directory.
$BB (187) Illegal argument error: Caused by attempting to modify the module header *OR* modifying byte offsets beyond the size of the descriptor.
A=If an Illegal Argument error was returned, then A contains the first byte offset that caused the error.

JOYDRV

Updated - The 3.3.0 JoyDrv would only return Y coordinates up to 191, even if the window was defined as 200. This has been fixed for both normal joystick/mouse reads and reads from the High Resolution Joystick Interface. **JoyDrv**

has also been shrunk a little in size.

KERNEL_UTILITY

New utility added. It's meant for advanced users to allow them to swap new **REL**, **Boot** and **Krn** modules (or any combinations thereof) into an existing boot track file. The resulting file can then moved to a floppy drive track 34 using the utility **KUTIL** (see it's own entry below)**.**

KEYDRV

Now merged back into **VTIO.** This will be (very slightly) faster but it also makes the combined modules more than 50 bytes smaller, and releases some VTIO system RAM as well... which we have plans for.

KRN

Has had some optimizations done in it's 6809 version:

- System calls that go between system and user states are slightly faster.
- The **F$CpyMem** system call has been moved to **Krn** from **KrnP2** (so it now matches the 6309 version). This does a dramatic speedup (several times faster), and also saves some memory in system RAM. You will notice this the most on some utilities like **PROC, PROCS, MMAP, MDIR** and others.
- **F$Chain** is about 750 CPU cycles faster.
- **F$AllPrc** is about 370 CPU cycles faster.
- **F$Debug** is smaller and a little faster on both the 6809 and 6309 versions.
-

KRNP3

Updated to edition #3 (slightly smaller and faster). This module is now installed by default, which gives you full error messages based on the **/dd/SYS/errmsg**.
Example: Trying to open a non-existant file would originally return a simple **ERROR 216.** It will now return:
Error #216 - Path Name Not Found

KUTIL

Utility to GET a kernel track from a floppy into a file, or PUT a kernel track file onto the real kernel track on a floppy disk. Type **KUTIL** by itself to get it's full built in help.

KWIKGEN

This RAM based bootfile editor now defaults to a full size 48K buffer, so it should handle any legitimate size **OS9Boot** file without needed to specify different memory specifiers from the Shell while launching it. Has full built in help as well.

MODBUSTER

A utility used to split a file with merged memory modules into separate files per module (useful on things like **OS9Boot**) has been updated to report errors better, and show the name of each module that is being split out as they are created.

PIPEMAN

Has been very slightly optimized.

RAMMER

Rammer is the RAM drive driver that we have chosen to default with in **EOU**, although if you have an 8 MB RAM upgrade, you may want to also add in Robert Gault's 6 MB RAM driver as well to use the "leftover" 6 MB for a large RAM drive. This one has two descriptors installed by default:

- **/R0** - regular RAM drive
- **/MD** - "memory" drive - useful for debugging - see below

RAMMER has also had some bug fixes and optimizations compared to it's earlier releases, including:

- Allowing the **/R0** RAM drive to be > (1 MB) up to a maximum of 1,608K
- **/MD** can now access all RAM in the system (up to 2 MB)
- 6809 version has been optimized for more speed.

Unlike some other RAM disk drivers, **RAMMER** respects **DMODE** parameters for things like sectors/track, number of

sides and number of tracks. This means that you can use **DMODE** to set up **/r0** to exactly match a real floppy drive, which allows things like **BACKUP** to work copying between a real floppy and the RAM drive in either direction, including special things like kernel tracks. You can also use this as a disk duplicator; match up your **/r0** to your real floppy, **BACKUP** the physical disk to the RAM drive, and the use **BACKUP** to make copies from **/r0** to your floppy drive. This even works on 20 sector per track disks that we mentioned in the new **FORMAT** command. Some things you should know:

1) EOU defaults **/r0** to match a standard Coco 35 track single sided disk in the descriptor. You can change this with **DMODE.**

2) To start up the standard **/r0** RAM disk, you should do the following, in order:

1. (optional) **DMODE /r0** with the # of tracks, sides and sectors per track you want it to have.
2. **iniz /r0** (this will reserve enough RAM for the drive geometry that you specified in the descriptor).
3. **format /r0** to format the RAM drive image just like a normal disk.

That's it... from there on, treat **/r0** just like a normal floppy drive (except faster and doesn't block clock interrupts).

3) If you want the RAM from the RAM drive back for other purposes (need it for extra graphics screens, long **PLAY** commands, or maybe you want to backup some disks that are not the same size/geometry as you currently have it set at), do the following:

1. Make sure you have no programs (including Shell's) running that have paths open on the RAM drive.
2. Type **DDIR** and make note of the value for R0 under the "**Usr Cnt**" (User Count) column (normally should be 1 if you have done a) already).
3. Do a **deiniz /r0** once for every user count you have. When you are done, type **DDIR** again to make sure that **R0** is no longer shown in the active device table. If it is not, then all the RAM it had allocated will be returned to the system. You can now run those memory hogging programs with the newly freed RAM. Or you could change the geometry for **/r0** to something else (example: If you want **/r0** to now be an 80 track double sided 720K disk, you could do a '**dmode /r0 cyl=50 sid=2**' and then go through the same steps in 2) above to create the new size RAM disk.

4) /MD is a special tool for developers. This lets you use a disk editor (like **DEd** or **KwikZap**) to view (and even change!) any of the RAM in your system (up to 2 MB). We actually use this to test system modules and data while the system is running on real Coco 3 hardware. You can do this by doing **DED /MD**. Since this can be dangerous, I would recommend not doing this unless you know how the system works (although viewing it should be fine). If you are setting up a multi-user system (see section 4 near the end of this document) It would be strongly recommended that you remove /**MD** entirely. On the other hand, if you want to explore how the system works, it can be invaluable.

5) Limits in the **/r0** descriptor (using the symbols reported by **DMODE**):

- **sct=** (sectors per track) - This can not exceed 255
- **t0s=** (sectors in track 0) - This can not exceed 255
- The number of cylinders (tracks) **cyl=** multiplied by the number of heads (sides) **sid** can NOT exceed 255.
- The largest RAM drive I would recommend in a 2 MB machine would be 1.6 MB (6400 sectors).
  If you do want that big of a RAM drive, I would suggest using this **DMODE** setting:
  DMODE /r0 cyl=64, sid=2 sct=20 t0s=20
  (This would give you 6400 free sectors.)

REL

You will get a different version of **REL** depending on which version of **EOU** you have installed. This is the very first module that runs when you type **DOS** to boot **NitrOS9**. Amongst other things, it creates the initial boot screen, and we have customized those to help identify which version you are booting:

- **NITROS9 6809** - 6809 based boot (this will run fine on a 6309 as well)
- **NITROS9 6309** - 6309 based boot
- **NITRO68 GIMEX** - 6809 GIME-X boot (runs at 2.86 MHz)
- **NITRO63 GIMEX** - 6309 GIME-X boot (runs at 2.86 MHz)

It should be noted that both GIME-X versions have to slow down to 1.78 MHz when accessing the CocoSDC, but run at full speed for everything else (like updating the screen, running code, multi-tasking). It should also be noted for the GIME-X versions that you *have* to have a static RAM based memory upgrade installed to run at that speed; it will not with older dynamic RAM upgrades.

SCF

Very slight optimization/shrink for built in line editing.

SETMPI

This is a new utility for those with Ed Snider's (Zippsterzone.com) **MegaMiniMPI** written by Jeff Teunissen; it allows

you enable/disable which slots can send IRQ's through immediately. This can help with things like serial ports, which a normal MPI will only let their IRQ's through if you have that slot selected (and will ignore all others).

VTIO (and it's submodules **JoyDrv** & **SndDrv**)

This has had multiple changes:

1. **KeyDrv** has been merged back into ***VTIO*** and is no longer a separate module. Since all 3rd party enhanced keyboards have to emulate the original Coco keyboard anyways (to be compatible with **DECB),** it doesn't make sense to split this off. This ends up being very slightly faster, but also saves some system RAM.
2. **2nd Mouse Button:** The original code to allow the 2nd mouse button to be optionally used to switch windows was hard-coded to only allow the mouse to be on the right port. It now honors the system's current selection (which side you have active for the mouse).
3. **SS.GIP2 SetStat call added:** This call will be expanded on later, so when making calls to **SS.GIP2,** please make sure that any undefined bits in register X should be set to 0, and registers Y and U should also be set to 0. This is to prevent unexpected results in your program in the future when more features are added to this system call.

   SS.GIP2 *(Global Input Parameters 2)* **SWI2 $99**

   Entry: A=current path
   B=$99
   Y=$0000 (reserved for future use)
   U=$0000 (reserved for future use)

   For the following, each flag is handled by 2 bits:
   X=MSB: 0xxxxxxx = Leave 2nd mouse button function unchanged
   10xxxxxx = Disable 2nd mouse button as CLEAR key
   11xxxxxx = Enable 2nd mouse button as CLEAR key
   xx0xxxxx = Leave current key click setting alone
   xx10xxxx = Disable key click on current window
   xx11xxxx = Enable key click on current window
   X=LSB: 00000000 = (reserved for future use)

   Key click is a local window setting (based on current path), and works on both CoWin and CoVDG windows. Note: if you create another window, the key click status is carried over from the parent window. (ie if you have Key click enabled in **/TERM**, and you do a **SHELL i=/w4&** from **/term**, then **/w4** will also have keyclick on).

   2nd mouse button function is system wide, affecting all CoVDG and CoWin windows. It should be noted how the 2nd mouse button as window select works:

   1) Click and release 2nd button by itself - go to next window (same as **CLEAR** key)

   2) Click and release 2nd mouse button while 1st button is held down - It will go to the previous window when you release the 2nd button (same as **SHIFT-CLEAR** keys). If you keep holding button 1 down, you can keep click/releasing the 2nd button to keep going backwards through your windows.

   NOTE: This is different than 3.3.0. In that version, the X position of the mouse dictated which direction through your windows a right mouse button click would go. But on a hardware text window (or a graphics window that doesn't have a mouse cursor turned on) this meant that you had no idea which direction you were going. We feel that this new method (which is 100% predictable on window change direction) improves this function.
4. **Fixed to properly call CoGrf** (not used in EOU, but if you want a stripped down boot file with no **MultiVue** to free up system RAM, this should allow it to run).
5. **SS.ComSt SetStat -** when used with **CoVDG** windows, the true lowercase bit flag (least significant bit of the high byte of the Y register) is fixed to properly turn true lowercase on or off.
6. **SS.GIP SetStat call updated** - This system call has been enhanced. The original version had a nice feature; if register Y was $FFFF on entry, it left your keyboard settings alone, allowing you to set the global mouse settings without forcing an update to the keyboard settings as well. So, we expanded on that and made it so that all 4 individual settings (Mouse resolution, Mouse port, Keyboard repeat start delay, and Keyboard repeat speed delay) will leave the current setting alone for each of those that send a value of $FF:

   SS.GIP *(Global Input Parameters)* **SWI2 $94**

   Entry: A=*path number*
   B=*$94*
   X=*mouse resolution*; in the most significant byte
   0 = low resolution mouse
   1 = optional high resolution mouse adapter
   $FF = leave current system setting alone
   mouse port location; in the least significant byte
   1 = right port
   2 = left port
   $FF = leave current system setting alone
   Y=*key repeat start constant*; in the most significant byte
   $00 = no key repeat
   $01-$FE = clock ticks (1/60th of a second) before key repeat starts

$FF = leave current system setting alone
key repeat delay; in the least significant byte
$00-$FE = clock ticks (1/60th of a second) between key repeats
$FF = leave current system setting alone

4) System Administrator Utilities and Notes:

MTSMON

This utility written by Bill Nobel in now installed with **EOU** by default. This is a multi-terminal version of **TSMON** (which stands for **Time Sharing Monitor**) that came with the **OS-9 Level Two Development System**. This program lets you set up terminal ports (or virtual windows with **Drivewire**) that can be logged into via usernames and passwords, and will set user permissions, etc. for those logins. (See the **Development System** manual on the **Color Computer Archive** to find out how to set up the password file, **LOGIN** program, etc.).

The biggest improvement switching to **MTSMON** from the original **TSMON** becomes apparent when you have a Coco system that will have more than 1 user logging in at a time (from multiple RS-232 ports, for example). The original system required you to set up a separate **TSMON** process on every port, taking more system RAM for process descriptors, path descriptors, etc. Since **MTSMON** will monitor multiple ports from a single program, it takes far less memory for systems with at least 2 active ports.

You make a list of ports (devices) in the file **/dd/SYS/stdports**, each device name separated by a space. This can include **Drivewire** virtual window descriptors and RS-232 ports.

An example: having '**/z1 /z2 /t2**' in **/dd/SYS/stdports** would mean that **MTSMON** will run the **LOGIN** program on virtual windows **/z1** and **/z2** through Drivewire, and on **/t2** on a real RS-232 port.

NOTE: No default **/dd/SYS/stdports** file is premade on **EOU**, since those that will use **MTSMON** will all have unique setups.

Source code is also included in **/dd/SOURCECODE/ASM/MTSMON**.

Alphabetical Index