AUSTRALIAN OS9 NEWSLETTER

sides 'No. of cylinders' (in decimal) :Interleave value: (in decimal) @FREE Syntax: Free [devname] Usage : Displays number of free sectors on a device @GFX Syntax: RUN GFX(<funct><args>) Usage : Graphics interface package for BASIC09 to do compatible VDG graphics commands @GFX2 Syntax: RUN GFX2([path]<funct><args>) Usage : Graphics interface package for BASIC09 to handle ax: none Usage : graphics/ windows on-line help to topics @IDENT mation from OS = use single lin ecution directory device @INKE tine to input a port to the pro to a memory ents of text files @ es into memory @MAKDIR Syntax: Makdir <pathname> Usage : Creates a new directory file @MDIR Syntax: Mdir [e] Usage : Displays the present memory module directory Opts : e = print extended module directory @MERGE Syntax: Merge <path> ndard output @MFREE Synt RAM memory @MODPATCH module in memory from c arnings -c = compare modul name = link to module C o V = verify module M = ma ntype [opt] Usage : Set mo onitor m = monochrome m ge : Creates and links an OS CS Syntax: Procs [e] Usage Opts : e = display all proc Prints the current data direc the current execution directory path @RENAME Syntax: Rename <filename> <new filename> Usage : Gives the file or directory a new name @RUNB Syntax: Runb <i-code module> Usage : BASIC09 run time package @SETIME Syntax: Setime [yy/mm TPR Syntax: to num @ er @TMODE es the operating parameters of the terminal @TUNEPORT Tuneport </t1 or /p> [value] Adjust the baud value for the serial port @UNLINK Syntax: Unlink <modname> Usage : Unlinks module(s) from memory @WCREATE Syntax:

EDITOR	Gordon Bentzen	(07) 344-3881
SUB-EDITOR	Bob Devries	(07) 372-7816
TREASURER	Don Berrie	(07) 375-1284
LIBRARIAN	Jean-Pierre Jacquet	(07) 372-4675
	Fax Messages	(07) 372-8325
SUPPORT	Brisbane OS9 Users Group	

Addresses for Correspondence

Editorial Material:

Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

Subscriptions & Library Requests:

Jean-Pierre Jacquet
27 Hampton Street
DURACK Qld 4077

Volume 6 August 1992 Number 7

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 6 Number 7

EDITOR : Gordon Bentzen
SUBEDITOR : Bob Devries
TREASURER : Don Berrie
LIBRARIAN : Jean-Pierre Jacquet
SUPPORT : Brisbane OS9 Level 2 Users Group.

Well there is no doubt that we will do it all again! Continue the National OS9 Usergroup for another year, that is.

Subscription renewals for the membership year of September '92 to August '93 has already reached twenty-eight. This of course means that the minimum number of twenty is exceeded.

HARD DRIVES

One of our new members has asked about fitting a Hard Drive to a CoCo3 and has also mentioned the Burke & Burke system with XTC Rom.

The Burke & Burke system comprises an interface board which allows an IBM type controller and hard drive to be used with the CoCo. It really does NEED the use of a MPI (Multi-Pak). In theory the B & B interface could plug straight into the CoCo's ROM port, but this would eliminate the use of a floppy disk controller & drives. So for a practical approach, count on using a MPI.

Don Berrie wrote an article for our newsletter which was included in the April 1989 edition. I have heavily edited this article and included a small part below, ED.

A HARD DISK DRIVE FOR YOUR CoCo

BY Don Berrie - April 1989

For a good general discussion about the available systems for the CoCo, I recommend that you read the article in the March 1989 edition of US Rainbow, pages 44 - 56 entitled "Adding a hard drive to your system", by Marty Goodman.

THE BURKE & BURKE SYSTEM

The hardware requirements. NOT ALL hard drives and controllers will work with the Burke & Burke system, but the main limitation seems to be the controller. Not all PC-compatible hard disk controllers are alike; some will not work with the CoCo XT at all, and some will not fit into the case. The controller case and the interface software are optimised for use with particular controllers.

Burke and Burke have also, when using their XT-ROM system, given us the ability to use an "alternate boot", using an alternate kernel. This alternate kernel is stored on track 129, and looks for a bootfile called altboot.
So there you have it, a hard drive for your system. When you have used this type of system, you can really appreciate the comments of the people who write about OS9 really coming into its own with a hard drive. It's really true. Multiview really shines, and what's more it runs at a speed that IS useable. Profile, a database programme, will amaze you. And of course, the hard drive does not use interrupts, has a 2K buffer (within the controller) and therefore will really support multi user capacity.

Don Berrie

WANTED

Some time ago one of our members, John McGrath, requested that we include a "Wanted" ad for him. John is looking for a RS-232 PAK. If you happen to have one for sale, or know somebody who has, please contact John McGrath, Tamworth NSW 067 618071 (home) or during work hours 018 667 662.

Australian COCOFEST!

I had a call from John Ikin, Melbourne, just after we mailed out our July newsletter. He spoke of a possible CoCoFEST in Melbourne later this year. See also the note by Andrew Donaldson, about page 7. Anyway if you are interested there are a couple of names to contact. No doubt we will hear more as arrangements are finalized.

SUBSCRIPTIONS

I would like to remind you to complete the application/renewal form and mail off your cheque if you have not already done so as the current subscription year ends with this edition. If have been expecting back-issues or we have missed any, please let us know and a correction will be made. To those members who have already responded we thank you for your continued support.

Cheers, Gordon.

A 'C' Tutorial
Chapter 9 - Standard Input/Output

THE STDIO.H HEADER FILE

Load the file SIMPLEIO.C for our first look at a file with standard I/O. Standard I/O refers to the most usual places where data is either read from, the keyboard, or written to, the video monitor. Since they are used so much, they are used as the default I/O devices and do not need to be named in the Input/Output instructions. This will make more sense when we actually start to use them so lets look at the file in front of you. The first thing you will notice is the first line of the file, the #include "stdio.h" line. This is very much like the #define we have already studied, except that instead of a simple substitution, an entire file is read in at this point. The system will find the file named "stdio.h" and read its entire contents in, replacing this statement. Obviously then, the file named "stdio.h" must contain valid C source statements that can be compiled as part of a program. This particular file is composed of several standard #defines to define some of the standard I/O operations. The file is called a header file and you will find several different header files on the source disks that came with your compiler. Each of the header files has a specific purpose and any or all of them can be included in any program. Most C compilers use the double quote marks to indicate that the "include" file will be found in the current directory. A few use the "less than" and "greater than" signs to indicate that the file will be found in a standard header file. Nearly all MSDOS C compilers use the double quotes, and most require the "include" file to be in the default directory. All of the programs in this tutorial have the double quotes in the "include" statements. If your compiler uses the other notation, you will have to change them before compiling.

INPUT/OUTPUT OPERATIONS IN C

Actually the C programming language has no input or output operations defined as part of the language, they must be user defined. Since everybody does not want to reinvent his own input and output operations, the compiler writers have done a lot of this for us and supplied us with several input functions and several output functions to aid in our program development. The functions have become a standard, and you will find the same functions available in nearly every compiler. In fact, the industry standard of the C language definition has become the book written by Kernigan and Ritchie, and they have included these functions in their definition. You will often, when reading literature about C, find a reference to K & R. This refers to the book written by Kernigan and Ritchie. You would be advised to purchase a copy for reference. You should print out the file named "stdio.h" and spend some time studying it. There will be a lot that you will not understand about it, but parts of it will look familiar. The name "stdio.h" is sort of cryptic for "standard input/output header", because that is exactly what it does. It defines the standard input and output functions in the form of #defines and macros. Don't worry too much about the details of this now. You can always return to this topic later for more study if it interests you, but you will really have no need to completely understand the "stdio.h" file. You will have a tremendous need to use it however, so these comments on its use and purpose are necessary.

OTHER INCLUDE FILES

When you begin writing larger programs and splitting them up into separately compiled portions, you will have occasion to use some statements common to each of the portions. It would be to your advantage to make a separate file containing the statements and use the #include to insert it into each of the files. If you want to change any of the common statements, you will only need to change one file and you will be assured of having all of the common statements agree. This is getting a little ahead of ourselves but you now have an idea how the #include directive can be used.

BACK TO THE FILE NAMED "SIMPLEIO.C"

Lets continue our tour of the file in question. The one variable "c" is defined and a message is printed out with the familiar "printf" function. We then find ourselves in a continuous loop as long as "c" is not equal to capital X. If there is any question in your mind about the loop control, you should review chapter 3 before continuing. The two new functions within the loop are of paramount interest in this program since they are the new functions. These are functions to read a character from the keyboard and display it on the monitor one character at a time. The function "getchar()" reads a single character from the standard input device, the keyboard being assumed because that is the standard input device, and assigns it to the variable "c". The next function "putchar(c)", uses the standard output device, the video monitor, and outputs the character contained in the variable "c". The character is output at the current cursor location and the cursor is advanced one space for the next character. The system is therefore taking care of a lot of the overhead for us. The loop continues reading and displaying characters until we type a capital X which terminates the loop. Compile and run this program for a few surprises. When you type on the keyboard, you will notice that what you type is displayed faithfully on the screen, and when you hit the return key, the entire line is repeated. In fact, we only told it to output each

character once but it seems to be saving the characters up and redisplaying them. A short explanation is in order.

DOS IS HELPING US OUT (OR GETTING IN THE WAY)

We need to understand a little bit about how DOS works to understand what is happening here. When data is read from the keyboard, under DOS control, the characters are stored in a buffer until a carriage return is entered at which time the entire string of characters is given to the program. When the characters are being typed, however, the characters are displayed one at a time on the monitor. This is called echo, and happens in many of the applications you run. With the above paragraph in mind, it should be clear that when you are typing a line of data into "SIMPLEIO", the characters are being echoed by DOS, and when you return the carriage, the characters are given to the program. As each character is given to the program, it displays it on the screen resulting in a repeat of the line typed in. To better illustrate this, type a line with a capital X somewhere in the middle of the line. You can type as many characters as you like following the "X" and they will all display because the characters are being read in under DOS, echoed to the monitor, and placed in the DOS input buffer. DOS doesn't think there is anything special about a capital X. When the string is given to the program, however, the characters are accepted by the program one at a time and sent to the monitor one at a time, until a capital X is encountered. After the capital X is displayed, the loop is terminated, and the program is terminated. The characters on the input line following the capital X are not displayed because the capital X signalled program termination. Compile and run "SIMPLEIO.C". After running the program several times and feeling confidant that you understand the above explanation, we will go on to another program. Don't get discouraged by the above seemingly weird behavior of the I/O system. It is strange, but there are other ways to get data into the computer. You will actually find the above method useful for many applications, and you will probably find some of the following useful also.

ANOTHER STRANGE I/O METHOD

Load the file named SINGLEIO.C and display it on your monitor for another method of character I/O. Once again, we start with the standard I/O header file, we define a variable named "c", and we print a welcoming message. Like the last program, we are in a loop that will continue to execute until we type a capital X, but the action is a little different here. The "getch()" is a new function that is a "get character" function. It differs from "getchar()" in that it does not get tied up in DOS. It reads the character in without echo, and puts it directly into the program where it is operated on immediately. This function then reads a character, immediately displays it on the screen, and continues the operation until a capital X is typed. When you compile and run this program, you will find that there is no repeat of the lines when you hit a carriage return, and when you hit the capital X, the program terminates immediately. No carriage return is needed to get it to accept the line with the X in it. We do have another problem here, there is no linefeed with the carriage return.

NOW WE NEED A LINE FEED

It is not apparent to you in most application programs but when you hit the enter key, the program supplies a linefeed to go with the carriage return. You need to return to the left side of the monitor and you also need to drop down a line. The linefeed is not automatic. We need to improve our program to do this also. If you will load and display the program named BETTERIN.C, you will find a change to incorporate this feature. In BETTERIN.C, we have two additional statements at the beginning that will define the character codes for the linefeed (LF), and the carriage return (CR). If you look at any ASCII table you will find that the codes 10 and 13 are exactly as defined here. In the main program, after outputting the character, we compare it to CR, and if it is equal to CR, we also output a linefeed which is the LF. We could have just as well have left out the two #define statements and used "if (c == 13) putchar(10);" but it would not be very descriptive of what we are doing here. The method used in the program represents better programming practice. Compile and run BETTERIN.C to see if it does what we have said it should do. It should display exactly what you type in, including a linefeed with each carriage return, and should stop immediately when you type a capital X. If you are using a nonstandard compiler, it may not find a "CR" because your system returns a "LF" character to indicate end-of-line. It will be up to you to determine what method your compiler uses. The quickest way is to add a "printf" statement that prints the input character in decimal format.

to be continued.....

Are your VEF pictures squashed?
by Bob Devries

One of our Sydney members, Bob Barker, asks "How can I easily tell if a VEF picture is squashed or not?" The first answer that comes to mind is, simple, look at the first byte of the file. On reflection, I realise that

while that is true, it is not really easy to read the first byte of a file, without a lot of rigmarole. So, I wrote two programmes to do it. First, a Basic09 procedure, the small one, and then a C programme, the larger one.

The Basic09 procedure is very simple, it uses the command line argument for a filename, reads the first byte, and prints a line to tell you whether the file is squashed or not. That is all it will do. The C programme is much more sophisticated. It allows a command line switch (or option) to print more information about the file, and will do more than one filename on the command line, or it will read STDIN to get filenames, so you can pipe the filenames to it.

First, here's the code for the Basic09 procedure:

```
PROCEDURE checkVEF
ON ERROR GOTO 10
PARAM file:STRING[80]
DIM path:BYTE
DIM char:STRING[1]
DIM ernum:INTEGER
OPEN #path,file:READ
GET #path,char
IF ASC(char)>127 THEN
PRINT "Squashed VEF picture"
ELSE
PRINT "Unsquashed VEF picture"
ENDIF
CLOSE #path
END
10 ernum=ERR
IF ernum=56 THEN
PRINT "Usage: CheckVEF <file>"
ENDIF
END
```

As you can see, it is simple. The on error goto makes due allowance for the misguided user, who forgets how to use it. If a filename is supplied, the file is opened, a byte is read, and if the first byte is greater than 127, the picture is squashed VEF, else it is not. The file is then closed, and the procedure quits. Simple.

The C programme is much more complex, but then, it also does more. Here's the code:

```
#include <stdio.h>
#define TRUE 1
#define FALSE 0

struct header {     /* structure for packet to read from VEF file */
       char sqsh;   /* squashed = 128 else 0 */
       char sctp;   /* screen type 0, 1, 3 & 4 */
       char pal[16]; /* palette register contents */
};

main(argc,argv)
int argc;
char *argv[];
{
    int verbose = FALSE; /* verbose flag */
    int count = 1;       /* argument counter */
    char file[256];      /* filename string */
    char *fgets(), *gets(); /* prototype function */
    char *r_err;         /* error value */
    int error;           /* error value */
    FILE *ifp,*fopen();  /* file pointer */
    struct header head;  /* declare structure */

    if (argc > 1)
        if (argv[1][0] == '-') /* correct option ? */
            if (toupper(argv[1][1]) != 'V') { /* no... tell him! */
                fprintf(stderr,"Usage: CheckVEF [-v] file file ...\n");
                exit(0); /* and exit */
            } else {
```

```
                verbose = TRUE; /* yes, be more wordy */
            }
    if (verbose == TRUE) count++; /* set counter past switch */
    for(;;) { /* forever loop */
        if (((verbose == TRUE) && (argc > 2))    ((verbose == FALSE) && (argc > 1)))  {
            if (count < argc) { /* command line filenames ? */
                strcpy(file,argv[count++]);
            } else {
                exit(0);
            }
        } else {
            r_err = gets(file); /* or piped ? */
            if (r_err == NULL) exit(errno); /* quit if any error */
        }
        if ((ifp = fopen(file,"r")) == NULL) { /* open the file */
            fprintf(stderr,"Can't open %s\n",file); /* if error, skip */
        } else {
            error = fread(&head, sizeof(struct header), 1, ifp);
            if (error == NULL) {
                fprintf(stderr,"Can't read data from %s\n",file);
            } else {
                printf("%s is %s VEF file.\n",file, (head.sqsh < 0) ? "a squashed" : "an unsquashed");
                if (verbose) { /* if verbose print more info */
                    printf("Picture is ");
                    switch(head.sctp) {
                        case 0:
                            printf("320 x 200, 16 colour\n");
                            break;
                        case 1:
                            printf("640 x 200, 4 colour\n");
                            break;
                        case 3:
                            printf("320 x 200, 4 colour\n");
                            break;
                        case 4:
                            printf("640 x 200, 2 colour\n");
                            break;
                    }
         printf("Palette registers are: %2d, %2d, %2d, %2d\n",head.pal[0],head.pal[1],head.pal[2],head.pal[3]);
         printf("                       %2d, %2d, %2d, %2d\n",head.pal[4],head.pal[5],head.pal[6],head.pal[7]);
         printf("                       %2d, %2d, %2d, %2d\n",head.pal[8],head.pal[9],head.pal[10],head.pal[11]);
         printf("                       %2d, %2d, %2d, %2d\n\n",head.pal[12],head.pal[13],head.pal[14],head.pal[15]);
                }
            }
            fclose(ifp);
        }
    }
}
```

Usage for this programme is as follows:

CheckVEF [-v] file file ...

So you can type 'CheckVEF picture.vef' or 'CheckVEF -v picture.vef'. More than one filename may be given, but the '-v' option must be the first argument. You may also 'pipe' the input to the programme. Say for argument's sake you have a directory full of VEF files. The command then might be 'ls *.vef ! CheckVEF' or 'ls *.vef ! CheckVEF -v'. The output should look something like this:

bilbo.vef is an unsquashed VEF file.

without the '-v' option and with the '-v' option:

```
bilbo.vef is an unsquashed VEF file.
Picture is 320 x 200, 16 colour
Palette registers are: 62,  0, 35, 63
                       49,  4,  7, 56
                       60,  6, 39, 53
                       63, 63, 63, 63
```

As is usual, these programmes will be available from our librarian, Jean-Pierre Jacquet. They will probably end up on disk 12 of the library.

Bob Devries

Australian COCOFEST!

Australians! (and everyone else..) A company called REMCOMS is organising a COCOFEST in Melbourne Australia some time in October! Date is not finalised. Will probably be in Dandenong somewhere.

REMCOMS recently got the rights to sell nearly all the brands of COCO software in Australia. They even are allowed to reproduce some names here. If they have your name, you would have got a catalogue in the mail. (Oh and they run a BBS too).

The COCOFEST is a great idea, my MM/1 will be there ;-) More details as they come to hand. (ask me if you want the phone number.. the hours are complicated)

-Andrew Donaldson.

P.S. I have nuffin to do with REMCOMS.

P.P.S. They are even putting an ad in the Herald/Sun for a week!

Microware's 'pipelines'

Just got the latest issue of Microware's "Pipelines" (Spring 1992). The MM/1 is mentioned TWICE in this issue. First in a new Microware product announcement header. It says, "Microware has recently released two new products for consumer-oriented OS-9 computers systems, such as CD-I and the MM/1." Both are software products, one being ariver for MIDI support, and the other is a graphics library which can be used to display CD+G graphics data.

Me, I am having loads of fun with my MM/1 and a Yamaha PSR-48 synth in MIDI mode by merely using UltiMusEK from KalaSoft and the MIDI Paddle board for the /t1 device port on the motherboard from Kevin Pease, the designer of the original MM/1.

Later, in Pipelines, in the "New Vendor Products" section, the MM/1 is mentioned in a blurb for the relatively new magazine, "The 68xxx Machines", saying, "The magazine covers technical tips and software information for OS-9 based computers like the MM/1."

There is also a couple of VERY interesting articles of how OS-9 is being used in the field, one story of a GESPAC MPU-20 (a 16 Mhz 68020 with 512K ram) diskless system which is the "brain" of an underwater exploration vehicle called "The Sea Squirt", being developed at the Underwater Vehicles Laboratory at MIT (Cambridge).

The other story is how two French organizations are using OS-9 based systems for research and data collection. One, AETA (Applications Electroniques des Techniques Avancees in Fontenay aux Roses) is using a 68030 based system running OS-9 to test astronauts for zero-G orientation abilities. Another French company, ESRF (The European Synchroton Radiation Facility at Grenoble) is using 68030 based systems to monitor and control atomic research.

There is also an announcement of the latest version of OS-9000, Version 1.3, which adds DOS emulation support for OS-9000. This is called "VPC" for "Virtual PC" by Microware.

But for me, the best article, is the centerpiece story, a story written by the well-known Peter Dibble on signal handling with C, a story titled, "Fine Tuning Signal Handling", with several examples of C source code.

Pipelines is free to any OS-9 user (I think, they don't charge me for it!). Just call or write to Microware to get on their mailing list!

--

Zack Sessions
sessions@seq.uncwil.edu
University of North Carolina at Wilmington

"Good health is merely the slowest form of dying."

This message was captured from the comp.os.os9 newsgroup.

```
                    +------------------+
                    !!                !!
                    !!  News Flash...  !!
                    !!                !!
                    +------------------+
```

```
+-------------------------------------------------------------------+
!!                                                                   !!
!! The "OS9 Underground" Magazine now incorporates '68xxx Machines' !!
!!                                                                   !!
!!  Former subscribers of '68xxx Machines' will receive a larger    !!
!!  magazine, with all the articles you were used to... plus you    !!
!!  will get MORE OS9!                                              !!
!!                                                                   !!
!! "OS9 Underground" Subscribers will get almost double the issue   !!
!! now.                                                             !!
!!                                                                   !!
!!  The OS9 Underground is THE Magazine for OS9...                   !!
+-------------------------------------------------------------------+
```

```
          +-----------------------------------------+
          !!                                       !!
          !! Haven't gotten a free trial issue yet??? !!
          !!                                       !!
          !! Reply to this message with your Name and !!
          !! address and I'll send you Issue #1...   !!
          !!                                       !!
          !! Or USMail to:                          !!
          !!               Fat Cat Publications    !!
          !!               4650 Cahuenga Blvd. Ste #7 !!
          !!               Toluca Lake, Ca.  91602  !!
          !!                                       !!
          +-----------------------------------------+
```

```
                  +--------------------------+
                  !! Whadda ya waitin' for? !!
                  +--------------------------+
```

```
 \!/  StG Net International  Zog's Cavern BBS  (818) 761-4135
Z(O)G          In support of OS9 on an OS9 Network!
 /!\  SysOp:Alan Sheltra - Editor of "The OS9 Underground" Magazine
```

Hard disk interface options

>> What interfaces are currently available? What other hardware will I need? What type of hard drive should I be looking for?

The two most popular interfaces are the Burke&Burke CoCo-XT and the Disto 4-in-1 board. I recommend getting the interface first, and then buying an appropriate hard drive and/or controller, since each one has somewhat particular hardware requirements (the CoCo-XT only works a particular type of 8-bit PC/XT controller card, and the Disto drivers require a SCSI drive that supports an uncommon 256-byte sector size).

- Burke&Burke CoCo-XT.

Adapts a PC/XT-style controller card to a CoCo. Requires an appropriate 8-bit MFM or RLL controller card and an appropriate MFM or RLL drive. Pluses: Excellent drivers and support software by Chris Burke; such drives and controllers are often cheaply available new or used; Chris also sells a ROM for booting OS9 from the hard drive. Minus: Requires a Multipak, or some very careful hacking (address conflicts with floppy controller, requires source of +12v). There's also a version with a built-in RTC.

- Disto 4-in-1 card.

This card goes inside a Disto controller, providing an RTC, Parallel printer port, serial port, and a SCSI hard disk interface. Lots of people used to use this with an Adaptec SCSI-MFM controller, but with the prices of embedded SCSI drives coming down out of the stratosphere, that's changing. Pluses: No Multipak required; also gives you parallel and serial port; SCSI drives can be carried to lots of other computers. Minus: requires a Disto Super Controller, Super Controller II, or Mini Expansion Bus and MultiPak; drivers included are not great, much better ones are available free (I don't know if anyone has yet written deblocking drivers to use with 512-byte sectors, so it may still require a drive that can do 256-byte sectors, such as the Seagate N-series. Ken Scales knows about this, I beleive.); may be hard to find, since I think CRC (the company that marketed the Disto products) died a while back, Dave Meyers at CoCoPRO! may have used ones(?).

A number of people have built their own hard disk interfaces, usually SCSI, since the hardware is pretty easy. This does have the drawback of requiring you to write your own drivers, though. <grin>

- Tim Kientzle

Add a second port to your Deluxe RS-232 PAK

In reference to several users asking about a second serial port for CoCo OS9, a reprint of an article I posted to CIS a while ago. It seemed short enough to send up as a list message, Bob Brose. (jriver!os9!boblist@uunet.UU.NET)

IMPORTANT NOTE: DO NOT ATTEMPT THIS MOD UNLESS YOU ARE VERY HANDY WITH A SOLDERING IRON AND ARE EXPERIENCED WITH PIGGYBACKING CHIPS. THERE IS NO WARRANTY EXPESSED OR IMPLIED. YOU ARE RESPONSIBLE FOR YOUR OWN WORK!

Hardware Modifications

Materials needed:
- 6551 chip
- 1489 chip
- 1.8432 Mhz crystal

If you need more than just Transmit data for outgoing lines, a 1488 will also be needed.

Remove the existing 6551 from it's socket.

On the new 6551, bend pins 2,5,6,7,8,9,10,11,12,16 and 17 up so they point directly away from the body of the chip. Place the new 6551 over the old 6551 (line up the pin 1's) and solder the following top pins to the bottom pins 1, 3, 4, 13, 14, 15, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 without getting any solder on the lower parts of the legs of the lower 6551.

Plug the 6551 stack back into it's socket. (if your 6551 wasn't socketed you can still do the mod but it will be harder to solder the 2 chips together on the board, be careful not to short any connections with solder blobs).

Connect a short jumper from the 74ls04 pin 9 to the top 6551 pin 2. This provides the select signal for the new 6551 (FF6C-FF6F).

Assuming a 3 wire port (which is all I needed), connect the following on the top 6551 together:

pins 1, 9, 16, 17

This sets the cts, dcd and dsr lines to low (true for them) which is correct for a three wire line with no hardware handshaking. If you want a full port, DO NOT connect these pins together, you will need to connect 9, 16 and 17 to gates on the soon to be piggybacked 1489 instead.

To pins 6 and 7 on the top 6551, solder a 1.8432 MHZ crystal. This is necessary as the two 6551's cannot share the same crystal because of the way they generate a signal from it. Alternatly, you can make a crystal generator out of spare gates on the 74ls04 with one crystal and feed the signal into both pin 6's on the 6551's (leaving pin 7's unconnected) but this requires much more work and since crystals are about $1 really isn't worth it.

Piggyback the new 1489 on top of the current one making

sure that the pins 1 line up, bending up all pins on the new 1489 except 7 and 14. Solder the 2 pin 7's together and the 2 pin 14's together making sure not to short out any other pins or traces. Connect a wire from pin 3 of the piggybacked 1489 to pin 12 of the top 6551 (this is receive data).

The existing 1488 has 1 free gate which we will use for transmit data. Solder a wire from the top 6551 pin 10 to the 1488 pin 2.

Final assembly:

Get your desired RS232 connector (I used a 25 pin female like the original). Solder a wire from a conveinent ground (I used the pad by the right rear mounting screw to pin 7 on the 25 pin RS232 connector. Solder a wire from pin 3 of the 1488 to pin 2 of the RS232 connector (this is transmit data). Solder a wire from pin 1 of the piggybacked 1489 to pin 3 on the RS232 connector (this is receive data). Note: pins 2 and 3 can be reversed depending on whether you are talking to a modem or terminal.

Test out the RS232 by plugging it into a multipak (protects you from major soldering errors) and powering up your machine. If your machine doesn't act completely normal, turn it off immediatly and recheck all of you connections against the above instructions. If everything is OK, try out a terminal program for the existing RS232, if it works proceed to the software mod section below, otherwise go back and check your work.

End of hardware mods

NOTES: If you want to hook up other input status lines, the piggybacked 1489 can be used to hook up the 3 input status lines, cts dcd and dsr. If you are going to use this port with a CALL IN modem you will need to do this, see the 1489 data sheet for pinouts of the unused gates.

Also, if you need to hook up outgoing status lines like dtr and rts, you will need to piggyback another 1488 on top of the existing one and connect it up. I'm using my second port for a terminal so none of the handshaking lines were necessary.

Software Modifications

I use the port only with OS9 so the changes are minor. You can use the port with rsdos, but you will need to write your own software to do so. Remember in RSDOS if you use both ports as interrupt driven ports, your interrupt routine will have to check both ports to see which one caused the interrupt as they are connected together (PC owners WISH they could actively share interrupts!).
OS9 instructions.

I use T3 for my new descriptor, I did this by taking an existing T2 descriptor and changing the least significant byte of the port address at offset 10h in the descriptor to 6C from 68. Also you need to change the name of the descriptor, I did this by changing the hi bit set "2" which is B2 at offset 38h to B3 (which is a hi bit set "3"). Don't forget to verify the CRC and save out the new descriptor. Create a new boot with the T3 and you are ready to go. The name offset at 38h above will vary from one descriptor to the next because there are so many versions of the acia driver around. I use Bruce Istads with great success and recommend it highly. (SACIA).

I routinely call in on T2 and then connect to another computer via T3 and it works completly perfectly.

Happy Computing!

Robert E. Brose II, N0QBJ
uunet!jriver!os9!bob
compuserve 72067,3021

6309 BenchMarks

I tried posting this on the net a long while back, and I didn't think it made it because of UUCP troubles. Here it is again. Hope some of this info is useful to all of you using or contemplating the use of the 6309.

Boisy Pitre US National OS9 Users Group President

6309 BENCHMARK RESULTS
==========

Using my TIMER utility which benchmarks programs by the system clock, I tested the speed of several popular commands.

These tests were run on a 6309-based 512K Tandy Color Computer 3 running at 1.9625 MHz; Seagate ST-157N-0 Hard Drive with Ken-Ton SCSI Interface. Although the actual speeds of these commands may be biased against the 1.789MHz clock in stock CoCo 3's, the time ratio between the two comparisons are constant. The version of PowerBoost used for these tests was 1.0.

Keep in mind that the issue here isn't the speed of the commands themselves, but rather their speed as it relates to the patched kernel and managers vs. non-patched OS-9.

Note that all commands were either already in memory, or previously loaded to aleviate timelapse in loading from disk. The left column reflects the elapsed time without PowerBooster 1.0 installed, while the right column shows the same command run under the same circumstances.

WITH PowerBooster.

```
WITHOUT POWERBOOSTER                WITH POWERBOOSTER
-------------------------------------------------
Command line:  cobbler /r0  (1)
Start Time:  15:26:34 . . . . . . . . . 15:28:30
Stop  Time:  15:26:50 . . . . . . . . . 15:28:31
Elapsed   :  00:00:16 . . . . . . . . . 00:00:01
Increase: Quite a bit

Command line:  mdir
Start Time:  15:25:48 . . . . . . . . . 15:28:12
Stop  Time:  15:25:52 . . . . . . . . . 15:28:14
Elapsed   :  00:00:04 . . . . . . . . . 00:00:02
Increase: Pretty good

Command line:  mdir e
Start Time:  15:26:01 . . . . . . . . . 15:28:19
Stop  Time:  15:26:09 . . . . . . . . . 15:28:24
Elapsed   :  00:00:08 . . . . . . . . . 00:00:05
Increase: Considerable

Command line:  megaread</dd@ (2)
Start Time:  16:13:37 . . . . . . . . . 16:30:27
Stop  Time:  16:14:50 . . . . . . . . . 16:31:37
Elapsed   :  00:01:13 . . . . . . . . . 00:01:10
Increase: Slight

Command line:  rz</t2       (3)
Start Time:  16:18:08 . . . . . . . . . 16:24:05
Stop  Time:  16:23:14 . . . . . . . . . 16:29:09
Elapsed   :  00:05:06 . . . . . . . . . 00:05:05
Increase: Very very very very little
```

NOTES:

(1) The Cobbler command was benchmarked with a 30,535 byte OS9Boot file written to a RAMDisk.

(2) The MegaRead command is a custom assembly language program which reads 40 blocks of 25,000 bytes each from stdin. This is NOT the same MegaRead utility used by others for benchmarking purposes.

(3) RZ 3.10 was used to download a 69,971 byte text file at 2400 baud (ViVa Modem 24) to a hard drive. No errors or retries occurred.

COMMENTS:

These are estimates based on my particular system and are accurate within 1 second. Some commands (i.e. Cobbler & MDir) show a drastic improvement in speed with the PowerBoost patches applied, while other commands (i.e. RZ & MegaRead) show some to very little improvement.

Indeed, the command which showed the most dramatic increase was the Cobbler utility. This is due to Cobbler's heavy dependence on the F$CpyMem call (used in three different places in the program, and that's not counting looped calls). I was so amazed at how much faster the command ran that I had to try it several times to be sure I wasn't dreaming!

MDir performs a good margin above the unpatched run. This command references the F$CpyMem call twice (again, not counting looped calls) as well as the F$GModDr call. One can definitely tell the difference when running this command, with or without the option.

I threw together my own home-brewed version of MegaRead (which probably doesn't resemble the "standard" version at all). Although it performed slightly better after the PowerBoost, Chris Burke's demonstration with his version at the ChicagoFest showed a vast increase. (If that version of MegaRead is available for copying, I would appreciate a UUENCODED version emailed to me. <hint>)

Although RZ showed hardly any improvement (you could almost say "no improvement"), I DO notice that my modem transfers, especially my UUCP transmissions, are considerably faster with PowerBooster enabled.

Other commands that I noticed significant speed when using PowerBooster: idir, ddir, proc, pmap, paths (these commands are part of Kevin Darling's utility package). I have also noticed speed increases in the MVCanvas graphics program, particularly when moving the canvas vertically using the up/down arrow object in the Tools menu. I would be interested in hearing from others out there with their observations of the PowerBoost's performance.

The ONLY problem that my system has with the PowerBoost software (yes there is ONE problem) is random crashes at various times, especially when running disk intensive C programs like MAKE and REPACK. I assume this has something to do with the exotic patches to my system's modules, but the problem only exists when PowerBoost is installed.

Let's not forget that this is version 1.0 of PowerBoost. Not nearly all of the drivers and system modules that could be patched, were patched. To really squeeze all of the speed out of the 6309, it would take a total rewrite of the kernel, managers, and drivers. Also remember that the programs which were tested don't take advantage of the extra features of the 6309 and were written with the 6809 in mind.

With the implementation of Native Mode in upcoming releases of Burke & Burke's PowerBoost upgrades, another significant increase in speed will be attained. In other words, this is just the "tip of the iceberg."

Lastly, based on just these benchmark figures alone, it is worth anyone's time and money to purchase the PowerBoost upgrade (and no, Chris Burke isn't paying me for this <grin>)

Boisy G. Pitre

NATIONAL OS9 USERGROUP - AUSTRALIA

SUBSCRIPTION RENEWAL / APPLICATION

Subscription Renewal [__] New Subscription [__]

Surname: _____ First Name :_____ Title : ____

Street: _____

Suburb: _____ State :_____ Postcode : _____

Country:_____

Home Phone: _____ Business Phone: _____

Age Group (please tick) Under 18 [__] 18-25 [__] 26-35 [__]
36-45 [__] 46-55 [__] over 55 [__]

Do you run OS9 Level 1[__] OS9 Level 2[__] OSK[__] OS9000[__]

Type of Computer for OS9: _____ RAM _____K.

Diskette 5.25 Number [__] Tracks [___] Sides [__]

Diskette 3.5 Number [__] 720k [__] 1.4meg [__]

Hard Drive Meg [____] Controller Type _____

Printer Type/Model _____

Modem Type _____

Special Interests _____

_____

Can you contribute articles to this Newsletter ? _____

Date : ____/____/____ Signature :_____

Amount Enclosed: $ _____ ($18-00 will cover you for 12 months)
(A$25-00 Overseas)
(CHEQUES PAYABLE TO: **NATIONAL OS9 USERGROUP**)

Please Return Completed Form to :-

NATIONAL OS9 USERGROUP
c/o GORDON BENTZEN
8 ODIN STREET

OR

NATIONAL OS9 USERGROUP
c/o J.P. JACQUET
27 HAMPTON STREET