OS-9 Newsletter

Volume III No. 5 | Bellingham OS-9 Users Group | May 31, 1992

Modifying 3-1/2in. Drives

By John Schliep

Recently, Donald Zimmerman and Rodger Alexander made some bulk purchases of 3-1/2 inch, 720K Floppy Drives for their respective club members. Some of the drives were manufactured by Epson, some by Teac and some by Mitsubishi. In all cases it was found that the drives could only be "jumpered" or configured as Drive 0 or Drive 1. The following is a step by step procedure for modifying the drives so that they can be configured for Drive 2 or Drive 3.

NOTE: Steps 1 - 3 pertain to drives with 5-1/4inch adapter frames.

1. Remover two front plate screws and remove Front cover of drive.
2. Remove four screws 2 per side from 5 1/4inch adapter that the drive sets in and lift the drive out of the adapter.
3. Unplug the data and power cables from the adapter.
4. Remove two screws holding cover plate on top of drive.
5. Lift up at the front and the cover plate comes off.
6. Find **D0** and **D1** pins and remove jumper located at back of drive on the left side.
7. Note that **D0, D1, D2** and **D3** contact points are labeled on the top and bottom of the drive circuit board. But that **D2** and **D3** do not have jumper pins
8. Turn drive upside down and solder jumper pins at the proper locations for **D2** and **D3**. (1/4 watt resistor leads are just the right diameter for use as pin stock.)
9. Turn the dive over and replace the jumper block across the desired jumper pins to configure you drive for **D2** or **D3**.
10. Reassemble.
11. FINI

Hope this helps the new guys since I'm sure all the old heads have already figured this on out.

--JOHN--

- IN THIS ISSUE -

BasicO9 Part-8

Tutorial

by Scott Honaker & Rodger Alexander

You are going to hate me for this. We need to start this month's installment with a couple of corrections:

CORRECTION 1:

The **TOP** variable must be reset each time you create a new database file, therefore in the **PDS** Procedure, change the following line:

IF Menu="C" OR Menu="c" THEN RUN
Create_DB(DB_Path,Current,DB_Name) \ ENDIF

This line should be changed to:

IF Menu="C" OR Menu="c" THEN RUN
Create_DB(DB_Path,Top,Current,DB_Name) \ ENDIF

Also change the **Create_DB** Procedure from:

PARAM DB_Patch:BYTE; Current:INTEGER;
DB_Name:STRING[32]

to read:

PARAM DB_Patch:BYTE; Top:INTEGER;
Current:INTEGER; DB_Name:STRING[32]

also change:

DIM Option:STRING[1]
Current=0

to read:

DIM Option:STRING[1]
Top=0Current=0

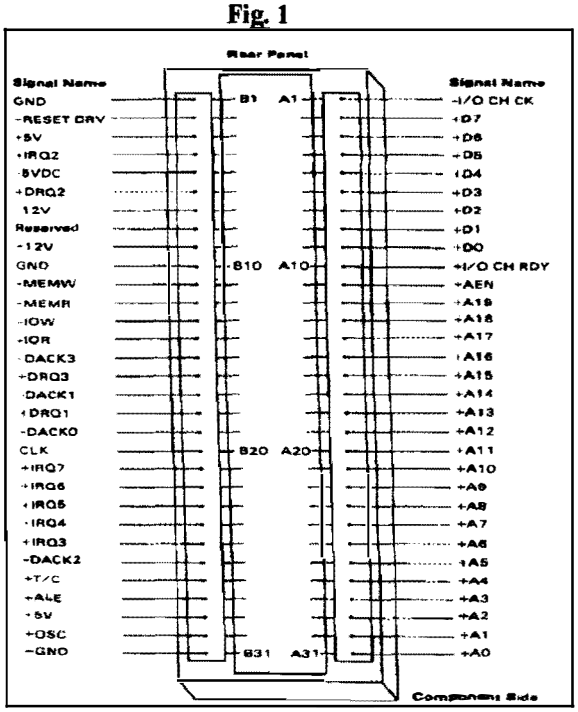
CORRECTION 2:
There was a major mess up in last month's listing of the **Search** Procedure. Find the following lines:
READ offset \ READ length

Replace that line with the following 3 lines:
READ offset
NEXT X
READ length

CoCo to IBM Interface

by Pat Pleuard

Fig. 1



The following diagrams should provide you with the necessary information needed to construct a "CoCo to IBM Bus Interface" so that you can plug an IBM-XT type Hard Drive Controller (MFM or RLL type) to one end of the interface and then plug the interface into a Multipak Interface.

Figure 1 shows an IBM bus socket with the bus pin outs and Signal names that an XT type Hard Drive controller expects to plug into.

Figure 2 is the circuit diagram for the interface card. It's very simple and can be accomplished using point to point wiring with 30 guage wire such as that used for wire wrapping (R.S. Cat.# 278-501). Only two integrated circuits are used: 74LS32 Quad 2-Input OR Gate, and a 74LS04 Hex Inverter. Both chips are very inexpensive and readily available from any electronic supply store. The most difficult part of the construction is soldering the 62 position PC/XT Bus Card-Edge Connector (R.S. Cat.# 276-1453) to the end of the circuit board.

CONSTRUCTION:

To simplify the construction, use a pre-punched plug-in circuit board available from Radio Shack (Cat.# 266-192). This board has 72 position card edge connectors, we only need 40, so carefully cut off the unnecessary connector traces making sure that the remaining 40 traces will line up properly when plugged into the Multi-Pak. Check out the CoCo Bus Pin labels in Figure 2 so that you will properly identify the card edge traces. Mount the two IC in a convenient location on the circuit board, then carefully solder the 30 guage wire to accomplish the connections as shown in the circuit diagram (Figure 2).

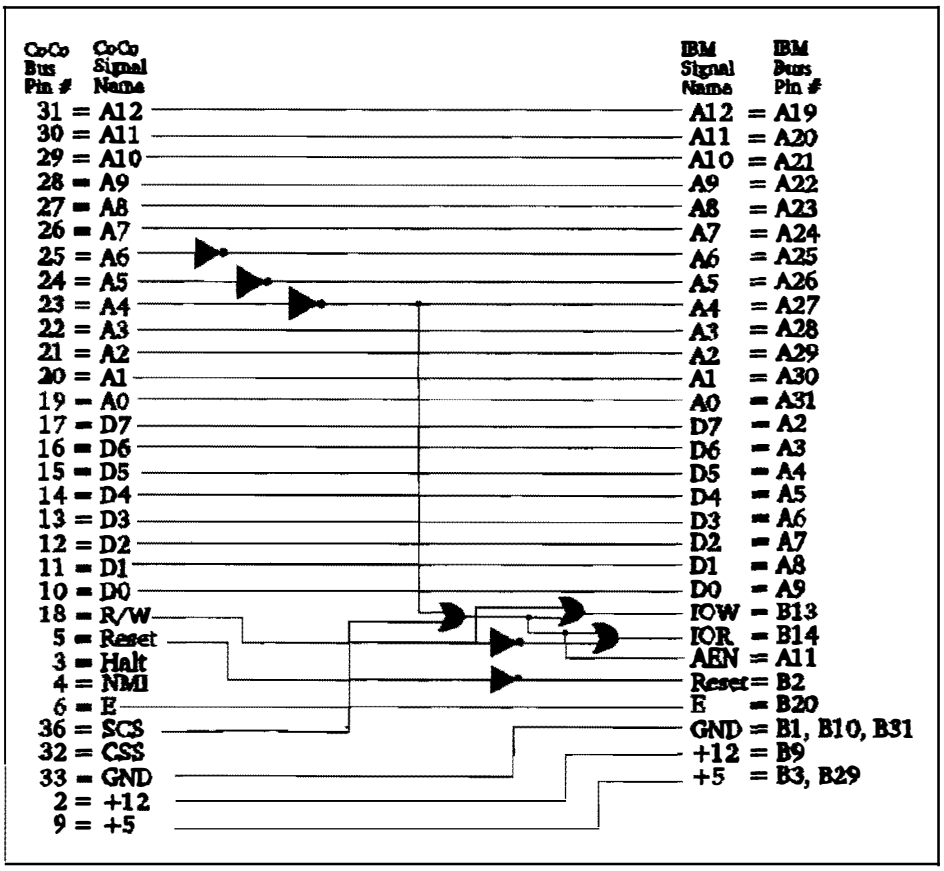
Solder in the integrated circuits or use IC sockets if you wish. Deposit a drop of solder at each desired contact point on the circuit board before you actually start wiring. The insulation on the wire is very thin and heat sensitive. Heat each drop of solder, then bring the wire in contact to the hot solder. The insulation will automatically strip away from the contact points making a very neat and easy connection. Use this method to quickly wire your circuit board point to point. BUT, always check each connection with a continuity testor or ohm meter to insure a good connection.

Except for the tediousness of soldering all those conncection points, this is an extremely easy project.

The only remaining hardware is an XT-MFM or RLL Controller Card which retails for about $50. I've also purchased several cards recently at the AM Computer Swap Meet for $20. Specifically you are looking for a Western Digital **WD1002** series or the **WDXT-GEN** Controller Card. You can also use the **DTC 5150CRH** and **5160CRH** or Adaptec's **2072.** Just make sure that if you are going to use an RLL Hard Drive that you use an RLL type controller. Some MFM Hard Drives can be made to work with an RLL Controller, but you're pushing it! (See Burke & Burke's CoCo XT Manual, pages A16-A20 for a more detailed explanation)

Parts List

Plug in PC Board with 0.1" contact centers
(Radio Shack Cat. # 276-192 at $4.95)
62 Position Card-Edge Connector
(Radio Shcck Cat. # 276-1453 at $2.95)
74LS04 Hex Inverter (approximately $0.49)
74LS32 Quad 2-Input OR Gate (approx. $0.49)



SOFTWARE:

This project is almost too good to be true. For less than $10 in parts you can build your own hard drive interface that looks very similar to the Burke and Burke Interface. and in fact it operates under the Burke and Burke Software. How convenient. But then again.....that's the catch!

Although the circuitry is quite different from Burke and Burke's **COCO-XT** Hard Drive Interface, the results are the same. In order to set up your hard drive descriptor and driver(s) you will need to purchase Burke and Burke's **CoCo XT & CoCo XT-RTC** Software and Documentation Package.

Pat is working on writing software specifically for his interface, but in the meantime, your only solution is to purchase Burke & Burke's CoCo XT Software.

Chris Burke spells it out in very clear terms on the first page of his manual. And I quote.....

In other words, even if you already have a hard drive system using Burke & Burke's Interface, you must still purchase a 2nd software package from Burke and Burke (P.O. Box 733 Maple Valley, WA 98038. 1-800-237-2409 or 206-432-1814). to operate Pat Pleuard's interface.

-- Rodger Alexander --

C-LANGUAGE TUTORIAL

Chapter 3 - Program Control

THE WHILE LOOP

The C programming language has several structures for looping and conditional branching. We will cover them all in this chapter and we will begin with the while loop. The while loop continues to loop while some condition is true. When the condition becomes false, the looping is discontinued. It therefore does just what it says it does, the name of the loop being very descriptive.

Load the program WHILE.C and display it for an example of a while loop. We begin with a comment and the program name, then go on to define an integer variable "count" within the body of the program. The variable is set to zero and we come to the while loop itself. The syntax of a while loop is just as shown here. The keyword "while" is followed by an expression of something in parentheses, followed by a compound statement bracketed by braces. As long as the expression in parenthesis is true, all statements within the braces will be executed. In this case, since the variable count is incremented by one every time the statements are executed, it will eventually reach 6, the statement will not be executed, and the loop will be terminated. The program control will resume at the statement following the statements in braces.

We will cover the compare expression, the one in parentheses, in the next chapter. Until then, simply accept the expressions for what you think they should do and you will probably be correct.

Several things must be pointed out regarding the while loop. First, if the variable count were initially set to any number greater than 5, the statements within the loop would not be executed at all, so it is possible to have a while loop that never is executed. Secondly, if the variable were not incremented in the loop, then in this case, the loop would never terminate, and the program would never complete. Finally, if there is only one statement to be executed within the loop, it does not need braces but can stand alone.

Compile and run this program:

```
/* DOWHILE.C */
main()
{
int i;
  i = 0;
  do {
    printf("The value of i is now %d\n",i);
    i = i + 1;
  } while (i < 5);
}
```

THE DO-WHILE LOOP

A variation of the while loop is illustrated in the program DOWHILE.C, which you should load and display. This program is nearly identical to the last one except that the loop begins with the reserved word "do", followed by a compound statement in braces, then the reserved word "while", and finally an expression in parentheses. The statements in the braces are executed repeatedly as long as the expression in parentheses is true. When the expression in parentheses becomes false, execution is terminated, and control passes to the statements following this statement.

Several things must be pointed out regarding this statement. Since the test is done at the end of the loop, the statements in the braces will always be executed at least once. Secondly, if "i" were not changed within the loop, the loop would never terminate, and hence the program would never terminate. Finally, just like for the while loop, if only one statement will be executed within the loop, no braces are required. Compile and run this program to see if it does what you think it should do.

It should come as no surprise to you that these loops can be nested. That is, one loop can be included within the compound statement of another loop, and the nesting level has no limit.

THE FOR LOOP

```
/* FORLOOP.C */
main()
{
int index;
  for(index = 0;index < 6;index = index + 1)
    printf("The value of the index is %d\n",index);
}
```

The "for" loop is really nothing new, it is simply a new way to describe the "while" loop. Type in and edit the file FORLOOP.C for an example of a program with a "for" loop.

The "for" loop consists of the reserved word "for" followed by a rather large expression in parentheses. This expression is really composed of three fields separated by semi-colons. The first field contains the expression "index = 0" and is an initializing field. Any expressions in this field are executed prior to the first pass through the loop. There is essentially no limit as to what can go here, but good programming practice would require it to be kept simple. Several initializing statements can be placed in this field, separated by commas.

The second field, in this case containing "index < 6", is the test which is done at the beginning of each loop through the program. It can be any expression which will evaluate to a true or false. (More will be said about the actual value of true and false in the next chapter.)

The expression contained in the third field is executed each time the loop is executed but it is not executed until after those statements in the main body of the loop are executed. This field, like the first, can also be composed of several operations separated by commas.

Following the for ( ) expression is any single or compound statement which will be executed as the body of the loop. A compound statement is any group of valid C statements enclosed in braces. In nearly any context in C, a simple statement can be replaced by a compound statement that will be treated as if it were a single statement as far as program control goes. Compile and run this program.

THE IF STATEMENT

This next listing is an example of our first conditional branching statement, the "if".

```
/* This is an example of the if and the if-else statements */
main()
{
int data;
  for(data = 0;data < 10;data = data + 1) {
    if (data == 2)
      printf("Data is now equal to %d\n",data);
    if (data < 5)
      printf("Data is now %d, which is less than
5\n",data);
    else
      printf("Data is now %d, which is greater than
4\n",data);
  } /* end of for loop */
}
```

Notice first, that there is a "for" loop with a compound statement as its executable part containing two "if" statements. This is an example of how statements can be nested. It should be clear to you that each of the "if" statements will be executed 10 times.

Consider the first "if" statement. It starts with the keyword "if" followed by an expression in parentheses. If the expression is evaluated and found to be true, the single statement following the "if" is executed, and if false, the following statement is skipped. Here too, the single statement can be replaced by a compound statement composed of several statements bounded by braces. The expression "data == 2" is simply asking if the value of data is equal to 2, this will be explained in detail in the next chapter. (Simply suffice for now that if "data = 2" were used in this context, it would mean a completely different thing.)

NOW FOR THE IF-ELSE

The second "if" is similar to the first with the addition of a new reserved word, the "else" following the first printf statement. This simply says that if the expression in the parentheses evaluates as true, the first expression is executed, otherwise the expression following the "else" is executed. Thus, one of the two expressions will always be executed, whereas in the first example the single expression was either executed or skipped. Both will find many uses in your C programming efforts. Compile and run this program to see if it does what you expect.

THE BREAK AND CONTINUE

Type in the file named BREAKCON.C for an example of two new statements.

```
/* BREAKON.C */
main()
{
int xx;
  for(xx = 5;xx < 15;xx = xx + 1){
    if (xx == 8)
      break;
    printf("In the break loop, xx is now %d\n",xx);
  }
  for(xx = 5;xx < 15;xx = xx + 1){
    if (xx == 8)
      continue;
    printf("In the continue loop, xx is now %d\n",xx);
  }
}
```

Notice that in the first "for", there is an "if" statement that calls a break if xx equals 8. The break will jump out of the loop you are in and begin executing statements following the loop, effectively terminating the loop. This is a valuable statement when you need to jump out of a loop depending on the value of some results calculated in the loop. In this case, when xx reaches 8, the loop is terminated and the last value printed will be the previous value, namely 7.

The next "for" loop, contains a continue statement which does not cause termination of the loop but jumps out of the present iteration. When the value of xx reaches 8 in this case, the program will jump to the end of the loop and continue executing the loop, effectively eliminating the printf statement during the pass through the loop when xx is eight. Compile and run the program to see if it does what you expect.

THE SWITCH STATEMENT

Type in and display the file SWITCH.C for an example of the biggest construct yet in the C language, the switch. The switch is not difficult, so don't let it intimidate you. It begins with the keyword "switch" followed by a variable in parentheses which is the switching variable, in this case "truck". As many cases as desired are then enclosed within a pair of braces. The reserved word "case" is used to begin each case entered followed by the value of the variable, then a colon, and the statements to be executed.

In this example, if the variable "truck" contains the value 3 during this pass of the switch statement, the printf will cause "The value is three" to be displayed, and the "break" statement will cause us to jump out of the switch.

```
/* SWITCH.C */
main()
{
int truck;
  for (truck = 3;truck < 13;truck = truck + 1) {
    switch (truck) {
      case 3  : printf("The value is three\n");
                break;
      case 4  : printf("The value is four\n");
                break;
      case 5  :
      case 6  :
      case 7  :
      case 8  : printf("The value is between 5 and 8\n");
                break;
      case 11 : printf("The value is eleven\n");
                break;
      default : printf("It is one of the undefined values\n");
                break;
    } /* end of switch */
  } /* end of for loop */
}
```

Once an entry point is found, statements will be executed until a "break" is found or until the program drops through the bottom of the switch braces. If the variable has the value 5, the statements will begin executing where "case 5 :" is found, but the first statements found are where the case 8 statements are. These are executed and the break statement in the "case 8" portion will direct the execution out the bottom of the switch. The various case values can be in any order and if a value is not found, the default portion of the switch will be executed.

It should be clear that any of the above constructs can be nested within each other or placed in succession, depending on the needs of the particular programming project at hand.

Compile and run SWITCH.C to see if it does what you expect it to after this discussion.

Type in and display the file GOTOEX.C for an example of a file with some "goto" statements in it.

```
/* GOTOEX.C */
main()
{
int dog,cat,pig;
  goto real_start;
  some_where:
  printf("This is another line of the mess.\n");
  goto stop_it;
/* the following section is the only section with a useable
goto */
  real_start:
  for(dog = 1;dog < 6;dog = dog + 1) {
    for(cat = 1;cat < 6;cat = cat + 1) {
      for(pig = 1;pig < 4;pig = pig + 1) {
        printf("Dog = %d  Cat = %d  Pig =
%d\n",dog,cat,pig);
        if ((dog + cat + pig) > 8 ) goto enough;
      };
    };
  };
  enough: printf("Those are enough animals for now.\n");
/* this is the end of the section with a useable goto
statement */
  printf("\nThis is the first line out of the spaghetti
code.\n");
  goto there;
  where:
  printf("This is the third line of spaghetti.\n");
  goto some_where;
  there:
  printf("This is the second line of the spaghetti code.\n");
  goto where;
  stop_it:
  printf("This is the last line of this mess.\n");
}
```

To use a "goto" statement, you simply use the reserved word "goto" followed by the symbolic name to which you wish to jump. The name is then placed anywhere in the program followed by a colon. You are not allowed to jump into any loop, but you are allowed to jump out of a loop. Also, you are not allowed to jump out of any function into another. These attempts will be flagged by your compiler as an error if you attempt any of them.

This particular program is really a mess but it is a good example of why software writers are trying to eliminate the use of the "goto" statement as much as possible. The only place in this program where it is reasonable to use the "goto" is the one in line 17 where the program jumps out of the three nested loops in one jump. In this case it would be rather messy to set up a variable and jump successively out of all three loops but one "goto" statement gets you out of all three.

Some persons say the "goto" statement should never be used under any circumstances but this is rather narrow minded thinking. If there is a place where a "goto" will clearly do a neater control flow than some other construct, feel free to use it. It should not be abused however, as it is in the rest of the program on your monitor.

Entire books are written on "gotoless" programming, better known as Structured Programming. These will be left to your study. One point of reference is the Visual Calculater described in Chapter 14 of this tutorial. This program is contained in four separately compiled programs and is a rather large complex program. If you spend some time studying the source code, you will find that there is not a single "goto" statement anywhere in it. Compile and run GOTOEX.C and study its output. It would be a good exercise to rewrite it and see how much more readable it is when the statements are listed in order.

FINALLY, A MEANINGFUL PROGRAM

Type in the file named TEMPCONV.C for an example of a useful, even though somewhat limited program. This is a program that generates a list of centigrade and farenheit temperatures and prints a message out at the freezing point of water and another at the boiling point of water.

```
 /*TEMPCONV.C */
/* This is a temperature conversion program written in  */
/* the C programming language. This program generates /
/* and displays a table of farenheit and centigrade     */
/* Temperatures, and lists the freezing and boiling     */
/*  of water.                                           */
main()
{
int count;       /* a loop control variable             */
int farenheit:   /* the temperature in farenheit degrees  */
int centigrade;  /* the temperature in centigrade degrees */
  printf("Centigrade to Farenheit temperature table\n\n");
  for(count = -2;count <= 12;count = count + 1){
    centigrade = 10 * count;
    farenheit = 32 + (centigrade * 9)/5;
    printf(" C =%4d  F =%4d ",centigrade,farenheit);
    if (centigrade == 0)
      printf(" Freezing point of water");
    if (centigrade == 100)
      printf(" Boiling point of water");
    printf("\n");
  } /* end of for loop */
}
```

Of particular importance is the formatting. The header is simply several lines of comments describing what the program does in a manner that catches the readers attention and is still pleasing to the eye. You will eventually develop your own formatting style, but this is a good way to start. Also if you observe the for loop, you will notice that all of the contents of the compound statement are indented 3 spaces to the right of the "for" reserved word, and the closing brace is lined up under the "f" in "for". This makes debugging a bit easier because the construction becomes very obvious. You will also notice that the "printf" statements that are in the "if" statements within the big "for" loop are

indented three additional spaces because they are part of another construct.

This is the first program in which we used more than one variable. The three variables are simply defined on three different lines and are used in the same manner as a single variable was used in previous programs. By defining them on different lines, we have an opportunity to define each with a comment.

ANOTHER POOR PROGRAMMING EXAMPLE

Recalling UGLYFORM.C from last month, you saw a very poorly formatted program. Now take a look at DUMBCONV.C, an example of poor formatting which is much closer to what you will actually find in practice.

This is the same program as TEMPCONV.C with the comments removed and the variable names changed to remove the descriptive aspect of the names. Although this program does exactly the same as the last one, it is much more difficult to read and understand. You should begin to develop good programming practices now.

Compile and run this program to see that it does exactly what the last one did.

```
/* DUMBCONV.C */
main()
{
int x1,x2,x3;
  printf("Centigrade to Farenheit temperature table\n\n");
  for(x1 = -2;x1 <= 12;x1 = x1 + 1){
    x3 = 10 * x1;
    x2 = 32 + (x3 * 9)/5;
    printf("  C =%4d
    if (x3 == 0)
      printf(" Freezing point of water");
    if (x3 == 100)
      printf(" Boiling point of water");
    printf("\n");
  }
}
```

PROGRAMMING EXERCISES

1. Write a program that writes your name on the monitor ten times. Write this program three times, once with each looping method.
2. Write a program that counts from one to ten, prints the values on a separate line for each, and includes a message of your choice when the count is 3 and a different message when the count is 7.

CoCo Trivia Contest

Submit CoCo Trivia questions to be used at the Northwest CoCo Fest June 26-27. If your question stumps the contestants, you will receive a special NW CoCoFest prize worth $5.

The format of the questions should be:

1. Brief Question
2. Correct Answer
3. 2 different clues
4. Topics are:
   - Hardware
   - Games
   - BASIC
   - Software (other than games)
   - OS-9

If you enclose $1 and a stamped self addressed envelope with you entry, we will send you the official NW CoCoFest Program. Send as many Questions as you like. Those selected will be credited during the event and cited in the program. If your question stumps the contestants, you will receive a special NW CoCoFest prize worth $5. All questions become the property of the NW CoCoFest and the selection of questions by the judges is final.

Send your question(s) to:

NW CoCoFest
c/o Donald Zimmerman
3046 Banner Rd. SE
Port Orchard, WA 98366-8810

All funds generated go toward the NW CoCoFest. In the event there are any remaining funds after all expenses are met, the funds will be held by The Port Orchard Color Computer Club, "Port O'CoCo", as seed money for next year's event or donated to the Computer Bank Charity, a registered non-profit corporation (FED ID 91-1480434, WA ST ID 601-260-547).

Power Boost 6309 cpu from Burke & Burke

FROM THE EDITOR: In the June '92 issue of *The Rainbow*, was a 1/4 page add by Burke & Burke (of Maple Falls, Wa.) introducing his latest addition:

RUN OS9 LEVEL 2 VISIBLY FASTER
GET A $29.95 POWERBOOST FROM
BURKE & BURKE

The add claims that after you install the Power Boost kit, featuring the Hitachi HD63B09E microprocessor, block moves and other functions are up to 4 times faster due to the HD63B09E's added registers and high-speed instruction set. The kit includes software which modifies your OS-9 Level-2 operating system "for faster multi-tasking, graphics, and disk I/O when using the HD63B09E".

A companion **"The 6309 Book by Chris Burke"** is also included in the advertisement for $24.95. A disk is included with an OS-9 Assembler, Disassembler and Debugger patches for the HD63B09E.

After reading the add, I was excited with renewed interest in my CoCo and OS-9. I've got to see this

PowerBoost Kit in action! This will be a must upgrade purchase for those of us using a CoCo-3.

I am assuming that Chris Burke will be at the PNW CoCo Fest to demonstrate the **Powerboost**. I know I'll be there with $55 in my pocket set aside to buy both the kit and the book IF it does everything the add seems to imply

You can be a VIDEO Star

The Port O' CoCo Club in Port Orchard, Washington is requesting submission of video tape presentations for use during its upcoming NW CoCoFest II to be held in Port Orchard June 26-27. The video tapes should be 5-10 minutes long and address a focused, specific area of the CoCo world. These are not to be step by step "How To . . ." tapes, but rather overviews of a philosophic or practical nature. Rather than **"How To Install A Hard Drive"** it should be **"The Value Of A Hard Drive"**. Topics may range from getting started with a CoCo to as deep into OS-9 as you can go! The presentations may promote a product or software, but should be more in an informational mode (like an "info-commercial") rather than a selling mode.

Tape your presentation at your computer or at locations unique to your part of the world. A talking head against a blue background is uninteresting. Consider using your CoCo to generate any graphics that will illustrate your points. You can also feed the output of the CoCo directly into your VCR!!

The NW CoCoFest II is a non-commercial venture and is not funded by big company (in fact, it's not funded by anyone but CoCo users), it is solely supported by the registration fees and staffed by members of the four Color Computer clubs in the Puget Sound area, i.e. Port Orchard, Tacoma, Seattle and Bellingham. Thus we have to be creative in getting the vast pool of knowledge available from around the world. Videos are a viable approach.

There is no charge for submissions. All the video tapes and the information therein becomes the property of the NW CoCoFest II. The presentations may be edited for time. Tapes will not be returned unless postage for return is included with the video. The videos MUST be in VHS standard format (not European PAL or Beta, 8mm or other formats).

All submitters will receive a copy of the NW CoCoFest II program. Selected tapes will be part of the presentation during the NW CoCoFest. CoCo users whose tapes are selected will receive a free NW CoCoFest II T-shirt (in the large size unless otherwise requested). This is a $12 value and a priceless conversation piece among your computer friends for months! **Deadline on tape submission is June 15, 1992.**

Mail tapes to:

NW CoCoFest II
c/o Donald Zimmerman
Port Orchard WA 98366-8810

Four Double Sided Drives on a CoCo Under OS-9

By Bob Devries

This article originally appeared in the Australian OS9 Newsletter of March, 1991.

Here are the details of modifications to my Disto/CRC SCII controller to enable it to use four floppies. The modifications should also work on other controllers, BUT NO GUARANTEES are implied. You're on your own!

There are not four drive select lines, only three and the side select line. Well, that is true, and I had to do a bit of hardware hacking (my main source of enjoyment with the CoCo. After C programming of course!). Well, I'll tell you about how I did the mods. Let me first say that if you decide to try it for yourself, you're on your own! I will not be responsible for any damage.

The disk controller I have is a CRC-Disto Super controller II (with 4-in-1 fitted, but that's by-the-way). Now I would imagine that the mods should be able to be done to any disk controller, but again, NO GUARANTEES.

CABLE:

I used a 23 pin DB connector to replace the 34 pin card edge connector at the end of the cable that you plug into the controller. I wired it along the lines of the Amiga A500 external drive connector. Here is the pinout I used.

DB23 pins	Standard pins
1. Ready (not used on COCO)	(34)
2. Read Data	(30)
3 - 7 Ground	(all odd pins)
8. Motor on	(16)

9. Drive select 2 (14)
10. Disk reset (not used on COCO)
11. Disk change (not used on COCO)
12. +5 Volt supply
13. Side select (32)
14. Write protect (28)
15. Track 00 (26)
16. Write Gate (24)
17. Write Data (22)
18. Step (20)
19. Direction (18)
20. Drive select 3 (not currently connected - read on) (6)
21. Drive select 1 (already used for 5 1/4 drives) (12)
22. Index (8)
23. +12 Volt supply

CONTROLLER:

OK so here's how I modified the controller. Firstly, you'll need a 74LS138 chip, as well as three 10k ohm 1/4 watt or 1/8 watt resistors. Here's the circuit:

74LS138

```
controller drive sel  0   1 |--U--|16   to +5 volt
   "          "    "  1   2 |     |15   to drive sel 3 (pin 6)
   "          "    "  2   3 |     |14   not connected
   connect to ground      4 |     |13   to drive sel 1 (pin 12)
   connect to ground      5 |     |12   not connected
   connect to +5 volt     6 |     |11   to drive sel 2 (pin 14)
   not connected          7 |     |10   not connected
   connect to 0 volt      8 |_____|9    to drive sel 0 (pin 10)
```

The three resistors must be connected one end to +5 volt, and the other ends to pins 1, 2 and 3 respectively.

This little circuit may be connected either in your disk drive case, or in the controller itself, where-ever there is sufficient space. I mounted mine inside the controller, placing the chip piggy-backed on top of another chip of similar size, with all the pins except pins 16, and 8 bent out sideways. I soldered these two pins to the chip underneath, making sure of two things: (1) that the chip underneath was another 74LSXXX chip or 74XX chip (e.g. 7416), and (2), that the piggy-backed chip was the same way 'round. Well, OK, I also checked that I did not blob the solder all over the other chips, but that's normal for any electronics practices.

To make the connection to the circuit, I cut the traces just before the connection to the 34 way edge connector where the drive cable plugs in. Only three need to be cut, pins 10, 12, and 14. There should be no connection to pin 6, and the wire from pin 15 of the 74LS138 chip may be connected here. By the way, I used 28 guage 'Kynar' wire wrap (any fairly thin gauge wire would do).

SOFTWARE:

OK, so now how do you do the modificatin? OS9 needs to have a module patched. The module is CC3Disk. I strongly suggest that you use 'DED' to do the modification. Find the character combination that goes like this (in HEX) 01020440. This is the drive select table bit pattern. Change the byte 40 to 07. Don't forget to write the new module to disk, and verify to correct the CRC. You can do this from within DED. Please read the DED docs FIRST, and do it on a BACKUP of your normal system disk. Next, you'll need to add two new device descriptors /D2 and /D3 to your OS9Boot file. You will of course do this in the normal way (whatever that is). The /D3 descriptor you'll have to create yourself, or modify the one on the Boot/config disk named d2_35s.dd. Here is a dump of one of my D2 descriptor to show you what they look like:

```
ADDR  0 1  2 3  4 5  6 7  8 9  A B  C D  E F   0 2 4 6 8 A C E
---- ---- ---- ---- ---- ---- ---- ---- ----  ---------------
0000 87CD 0030 0021 F181 D400 2300 26FF 07FF  .M.0.!q.T.#.&_._
0010 400F 0102 0320 0300 5002 0000 1200 1203  @.... ..P.......
0020 0844 B252 42C6 4343 3344 6973 6B83 4376  .D2RBFCC3Dis;.Cv
```

Don't forget... I am using the CRC-Disto Super controller II in NO-Halt mode. The descriptors may be different from yours! CHECK THIS FIRST.

Club Activity Report

Bellingham OS9 Users Group - Longview CoCo Club
Mt. Rainier CoCo Club - Port O'CoCo Club - Seattle 68xx Mug

Bellingham OS-9 Users Group

The meeting agenda was loaded for this meeting and progressed at a quick clip.

1. PNW CoCoFest update and projections was reported and a discussion about our club's presentation/representation at the event followed. A display much like our display last year was decided with the assurance that the display areas will be up and maintained all day Saturday and perhaps Friday night.
2. Discussion about the 6309 upgrade advertisement in the Rainbow and how it could be possible to achieve the higher efficiency rating. Does Chris Burke use the chip completely in it's native mode or switch between 6809 emulation mode and native mode?
3. Pat Pleuard's Hard Drive Interface project was shared with hand outs and a display of all of the parts. Unfortunately there were some unknown bugs in the hand-outs that have been corrected and are correct in this issue of the Newsletter.
4. Programming Problem. It was suggested by Seattle's 6809 Club President, Scott Honaker, to rewrite the Search Procedure presented last month so that the found record would be displayed on the menu screen. Everyone was give a copy of the original program listing. PROBLEM: How would you modify the program listing to achieve the "short cut" suggested by Scott? Several methods were tried and some undocumented facts were discovered. One problem encountered was that the parameter variable **Current** coulb not be used in a **FOR - NEXT** statement. We tried several ways around this but to no avail. Finally we gave in by using a REAL variable in the FOR - NEXT loop and then reassigned the value of **Current** with the result of the FOR - NEXT loop. Worked great and eliminated almost half of the original code.
5. Topic suggested last month was to address several preferred methods of using the icons with Multivue. For the remainder of the meeting we created shell+ script files and then used **datamod** to meke the script files into memory modules in the CMDS directory. The script files would change the default directory to the desired application data directory and then call up the application program. The associated AIF files called the script files rather than the application file. The desired result was to create a Multivue Application Directory that would display ICONS off all of the application programs on the hard drive. When the desired ICON was double clicked, the associated script file would change the data directory and execue the application.

Next meeting will be June 24th with Craig DuBois demonstrating how to use derived fields for automatic data manipulation.

--Rodger Alexander--

Mt. Rainier CoCo Club

May's meeting started off with the introduction of a new member, Rick Quzts. He is from Reedsport, Oregon and is not only a CoCo user, but also a ham radio operator. Welcome aboard, Rick!

John Schliep gave a presentation of the file management program JTFM. He showed us many of the features including selectable copying and mass attribute changes. This public domain program is a must for OS9 users.

Donald Zimmerman from the Port Orchard Club gave us another update on the upcoming CoCo Fest II and showed off the new CoCo T-shirt, which will be available at the Fest. Everyone is sure to want one of these great looking T-shirts. Donald also informed us of the dropping prices of Coco drives including 5-1/4 and 3-1/2 inch floppies and hard drives.

Michael Stokes then gave a very interesting demo of the "C" language and graphics for the coCo. He showed us four different ways to draw lines and explained the advantage and drawbacks of each method. He showed how his "failed" attempts lead to better methods and in the end how he created a library of graphics functions.

The meeting wrapped up with presentation by Alan Johnson of the program File Indexing System by Bob Zaker. This set of programs, found on most BBS's under the title INDEX.AR, allows you to create a catalog of your disk files that can be searched for a specific program, printed out, or help you find duplicate programs that need to be eliminated to free up disk space.

Our next meeting will be June 11th.

--Alan Johnson--

Port O-CoCo Club

The May meeting was something a little different. Our focus was on regional transportation, our future with the CoCo and the

individual. The Washington State Department of Transportation Committee is deciding where to build launching facilities for a passenger only service between the peninsula area and downtown Seattle. The two sites being considered are Kingston and Southworth. Southworth is the east edge of Port Orchard. After a little discussion we as an organization endorsed the idea of having Southworth as the next direct service with downtown Seattle. Why get involved? From the club's point of view it would ease and promote exchange between the Seattle club and ours by making it only a 35 minute trip across the Sound.

As a piece of trivia we briefly joked about the fact that IBM is going to be making a PC without their name on it. In other words, IBM has decided to clone itself! If you can't fight them, join them!

Speaking of hardware, we discussed the most recent Radio Shack flyer which offers the CoCo3 for $49.95. Several members have already tried to locate a machine and apparently there is NOT ONE machine in the district. So we as an organization are calling on Radio Shack to locate machines for us. We voted to purchase one machine for the club and there were about 5-7 more machines spoken for. Because ALL company stores are tied together by computer it is possible, at some level of the company, to know EXACTLY what is in each and every store in the nation. There must be hundreds of CoCo3 out there some place for Fort Worth to devote the ad space to this item. We are going to promote an aggressive search for those machines. If you are looking for another machine for any purpose (parts, backup, other family members, etc.) get in touch with us. We may be able to assist you.

The next large block of time was devoted to the BIG EVENT OF THE YEAR, the NW CoCoFest II. The "beta test" of the T-shirt was in hand. Two colors were used because I wasn't sure which would be most more pleasing to the eye. The choice was unanimous. And the acceptance of the artwork was overwhelming. Registration forms were handed out to all and some were filled out on the spot. Speakers and scheduling is pulling together nicely. Several people had particular questions about finances and best case vs worst case scenarios. I attempted to answer all the questions and convey the idea that everything is looking stronger than last year and last year was a surprising success.

If you or anyone you know has anything to do with the CoCo, June 26-27th Port Orchard IS THE PLACE TO BE.

Mark Kulien of Des Moines, who regularly attends our meetings, volunteered to take the various releases we have been sending out over FidoNet and send them over packet radio. This is a use of computers that is foreign to me, but for some reason the CoCo is well suited for this use. He will take those releases and send them out over packet radio so that world learns about the CoCoFest II in June.

The remainder of the meeting was devoted to individual questions, problems and small group discussions. Dan Statham has been successful in downloading C and it's library from MS-DOS. He talked about how he did it and the problems he is currently having. Phyllis Armstrong brought her friend Wonda who is just weeks old in computing. It's great to see Phyllis become the teacher when she started computing just about a year ago herself! Terry Laraway was his usual bag of tricks. Tom Brooks returned after his family tragedy and will be heading into surgery himself soon. We wish you the best, Tom.

After everything was packed up a post meeting took place in the parking lot. It was warm and the conversation flowed freely! The next meeting is June 15th, the earliest it can be in a month, and our focus will be on making the NW CoCoFest II the GREATEST for all those who attend!

--Donald Zimmerman--

Seattle 68xxx Mug

May's meeting featured Pat Pleuard's hardware projects. Several months ago he showed us a hard drive interface that cost less than $10. This time Pat showed off a multipak of sorts. The first slot was specifically designed for a standard Tandy Disk Controller, but the remaining four slots were IBM type XT Slots. He had astandard XT parallel printer I/O card plugged into one slot and a XT hayes compatible intergan modem card plugged into another slot.

Pat uses point to point soldering technique which was so neat and precise that it compared in appearance to etched circuit traces. Very impressive. Unfortunately, Pt did not have hand outs of his schematic diagrams but they are included in this issue of the Newsletter.

Rodger distributed the floppy drives that were ordered as a club purchase and demonstrated his **Search Procedure.**

The June meeting should be the last session dealing with the **PDS Database**. Scott will demonstrate the Pak Procedure.

--Rodger Alexander--

Washington State BBS List

FAR POINT BBS (Seattle)
RiBBS (Fido NET)
(206) 285-8335

COLUMBIA HTS. BB (Longview/Kelso)
RiBBS (Fido NET)
(206) 425-5804

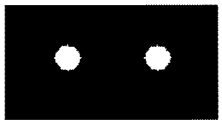
DATA WAREHOUSE BBS (Spokane)
RiBBS (Fido NET)
(509) 325-6787

BARBEQUED RIBBS (Bellingham)
PC-Board (PC-Net)
(206) 676-5787 - CoCo Conference #5

OS-9 TACOMA BBS (Tacoma)
RiBBS (Fido NET)
(206) 566-8857

COCO EXPRESS BBS (Anacortes)
RiBBS (Fido NET)
(206) 293-1057

Color Computer Video Library



Fixing the Multipak "IRQ"
Installing a 2nd floppy drive
Installing 512K Memory Board
Installing a Burke & Burke Hard Drive

$10

Bellingham OS-9 Users Group
3404 Illinois Lane; Bellingham, Wa. 98226

The *OS-9 Newsletter* is published by the Bellingham OS-9 Users Group. Rodger Alexander, Editor. Publishing software is Microsoft Word for Windows 2.0, printing to a Hewlitt Packard Desk Jet 500.

OS-9 Newsletter
3404 Illinois Lane
Bellingham, Wa 98226