

AUGUST 1990

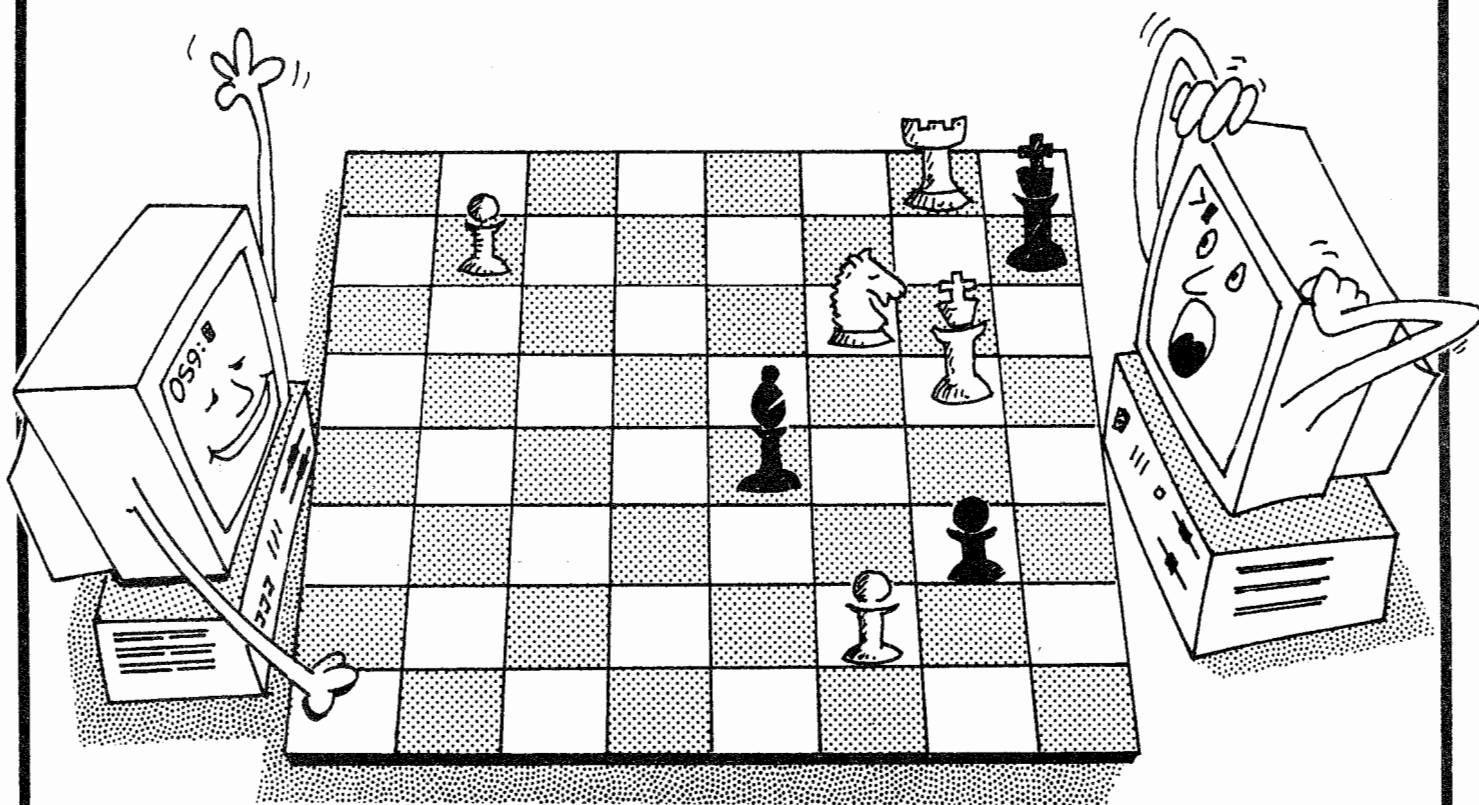
VER 01.02

\$2.00

THE OSK'ER[®]

News and Views in the World of OS9/68000 and 6809

● PLAYING CHESS IN "C"



- WHAT IS MULTI-MEDIA ANYWAY?**
- HOW TO BUILD A CDI**

- SPECS ON THE 2 NEW MACHINES**
- NATION-WIDE OS9 BBS LISTING**

Contents

Directory of /dd/OSKer/Aug90 14:16:04

Owner	Last modified	Attributes	Sector	Bytecount	Name
OSKer	90/08/13 2355	----r-wr	1	3855	BBS_List
OSKer	90/08/13 2243	----r-wr	2	2095	Doc_OSKer
Editor	90/08/08 1431	----r-wr	2	7596	Editor_Rambles
Editor	90/08/07 2023	----r-wr	4	4757	Flame_ON
StG	90/08/07 2023	----r-wr	7	6038	How_to_Build_a_CDI
Editor	90/08/13 2350	----r-wr	8	12537	New_Machine_Specs
Miller	90/08/16 2023	----r-wr	11	9626	OS9_in_Industry
Editor	90/08/03 1853	----r-wr	13	14983	Playing_Chess_in_C
Editor	90/08/11 0335	----r-wr	16	1498	Syscall_for_C
Editor	90/08/12 1712	----r-wr	17	4296	What_is_Multi_Media_Anyway

"the OSKer" is published monthly by:

StG Computers inc.
P.O. Box 24285
Speedway IN 46224

President: Scott Griepentrog (Editor)
Vice-Pres: Jim Hutchins
Secretary: Doug Dalton
Treasurer: Dave Henk

Cover Art: Alan Sheltra

Subscriptions to the OSKer are \$12 per year in the U.S., \$15 in Canada, and \$20 overseas.

Advertising Rates:

4/4 Page	7.5"w X 10.0"h	\$ 100
3/4 Page	7.5"w X 7.5"h	\$ 80
1/2 Page	7.5"w X 5.0"h	\$ 60
1/2 Page	3.5"w X 10.0"h	\$ 60
1/4 Page	3.5"w X 5.0"h	\$ 40
1/8 Page	3.5"w X 2.5"h	\$ 20

Pre-pay for two issues, get one free.
Ad copy must be received before last day of month previous to issue for inclusion.

Editing for the OSKer was done on an Atari Mega 2 ST running OSK, using uMacs. The text was fit together and pre-formatted with a custom program and layed out using Ventura on a Turbo PC. Address labels are printed with DB9 under OSK.

All submissions must be in the Public Domain to be considered for publication. Authors of printed articles will receive the following 6 months of the OSKer without charge.

StG Computers inc., as publisher of the OSKer and having ownership of software will not directly advertise in the OSKer to prevent a conflict of interest. Additionally, the editor will not use his position to promote said products.

The hacker contest has been extended to the following issue to allow time for receipt of all entries. The software list has not received any entries, and will be postponed until it does.

BBS List

Okay, everybody wants to know where the local boards are. So, here's a list in order by phone that has been compiled from several sources. However, there's a LOT of information missing! Yo Sysop's! Get your system listed (correctly!) here so that people know where to find you. I know there are lots of RCIS and Fido nodes out there, so speak up and make yourself known! I don't have the time to go around calling all these, so if anybody finds anything wrong (or left out), either call (317-241-6401 (leave message on machine), e-mail to me, or s-mail to address given inside front cover.

Phone	Name	Max Bd	Net	Sysop
201-494-3649	MicroFone TBBS	9600		
201-967-1061	R.C.I.S. Hq.	2400	RCIS	
203-399-7394	CoCo Kingdom	1200		
203-444-1597	One-Byte	2400		
204-582-3593	Alpha Software	2400		
206-285-8335	Farpoint Station	2400		
206-425-5804	Columbia Hts	2400		
213-461-7948	Zog's Cavern BBS (ZOG)	2400	StG	Alan Sheltra
215-375-8814	Garf's BBS	2400		
215-376-1819	Glass Menagerie	2400		
216-221-6809	OS9 Connection	2400		
301-863-5312	CoCo County Airport II	2400		
303-343-6707	Ribbs Hq.	2400	Fido	
303-757-8197	USS liberty	2400		
306-373-5029	Computer Link	1200		
312-745-1387	CoCo Rame	2400		
313-268-0028	Mega CoCo	2400		
313-879-2318	Arthur BBS	2400		
317-244-3159	Indy OS9 (ROOT)	2400	StG	Scott Griepentrog
319-362-6577	Color Connection	2400		
403-320-6481	Anarchy Intl' BBS (AIB)	2400	StG	Corey (CHIN-Lee
403-329-6438	So. Alberta BBS (SAB)	2400	StG	Dieter Rossman
404-446-6336	Online With Hayes	9600		
404-951-1540	INDEX System	2400		
405-752-8955	Micro-Link	2400		
419-829-4825	Colorama PBBS	2400		
501-661-0527	Workshop (WORKSHOP)	2400	StG	Al Fleagle
504-649-5761	Alpha Software	2400		
508-792-0381	Gravayard BBS	2400		
515-432-7853	The Tomb	2400		
516-795-5874	Long Island CoCo Club	9600		
518-372-2694	Fishline BBS	2400		
519-753-7420	Multi-Net BBS	2400		
602-844-7840	The Pub	2400		
608-655-3806	Tandy Users Group	2400		
612-780-8936	TCOS9UG BBS	2400		
614-965-1527	LastOutpost BBS	2400		
615-265-2629	Old Folks Home	9600		
615-383-0727	Nashville Exchange	2400		
616-884-1283	Zone BBS	2400		
617-661-0776	Firehouse	2400		
619-571-6366	8-bit Tandy	2400		
703-323-7654	PBBS	2400		
708-352-0948	S and V BBS (SANDV)	2400	StG	Paul Jerkatis
714-781-5825	Pegagus	2400		
804-744-9260	Tree House BBS	2400		
818-772-8890	Plain Rap	2400		
904-245-6585	Nobody's Home (HOME)	2400	StG	Scott Proctor
904-595-2184	Snoopy's DogHouse (GJOE)	2400	StG	John Swinson

Doc_OSKer

The Doctor is IN - but his waiting room is empty?

FROM: Frank@Zog (Frank Delaratta)

TO: SysOp@ROOT

SUBJ: I Gotta Question Doc?

DATE: 90/08/01 23:47:55

RCVD: root 90/08/04 01:29:26

I just downloaded Bruce Isted's patch files to use a serial mouse with a coco3 and an rs232 pak or other serial port. I need to know the pin to pin connections from the 9 pin connector on the digitech p5 mouse to the 25 pin connector on the rs232 pak. I also uploaded the file to this bbs.

Frank.

....

The DB9 connector has become an alternative standard to the DB25 connector for RS232 communications. Mostly because it's a lot smaller. But of course 'they' moved the pins around - more so than was necessary anyways. Only pins 2-8 and 20 of the DB25 are most commonly used, so this fits fairly nicely into the DB9 connector.

These days serial mice all come with DB9 connectors, but often include a DB25 adapter as well. The adapter works the same for CoCo as it would for PC's, except that the DB25 on the adapter is female. It will plug into the backwards RS232 connector on a PC, but you'd need to use a Male-Male gender changer to use it on the RS232 Pak. Actually, our RS232 Pak adheres to the older standard where all DB25's mounted on machines are Female, and all cables have Males at both ends. Because people often got the cables backwards, as well as a few other reasons like having to use gender changers every time you wanted to add another cable as an extension, IBM decided to put Males on the Computer side, and use Male-Female cables. Interestingly enough, this has become the 'new' standard. One has to admit that they actually did something that makes sense. Just this once though.

Anyways, on to your question. By analyzing a DB9-DB25 adapter with a meter, the following connections are found:

DB9 DB25 SIGNAL

1	8	Carrier Detect (CD)
2	3	Receive Data (RD)
3	2	Transmit Data (TD)
4	20	Data Terminal Ready (DTR)
5	7	Signal Ground (GND)
6	6	Data Set Ready (DSR)
7	4	Ready to Send (RTS)
8	5	Clear to Send (CTS)
9	22	Ring Indicator (RI)

Editor_Rambles

'On and On it Goes and Where it Stops, Nobody Knows...'

Well, here I am again hacking away into the middle of the night trying to get another OSKer issue (version?) done. As with the first one, it appears I'm running right on 7 days behind - that is, behind where I would like to be. The idea is to have the end of a month the deadline for submissions (which usually turn up late if at all), spend a few days splicing the layout together and take it to the printer, and start mailing it out about the 7th or 8th of the month. Well, as I write this it's the 7th, and two pieces are missing. But then, it'll be tomorrow before I'm done stuffing the pages together, providing I don't run out of crazy glue. But it'll be there, and it'll be worth the effort. To those of you who received the last issue just this month, I had so many requests piling up it took me a while to get them all put in the database and run through another mailing. As it is I've already got about 50 more I'll be mailing while waiting for this one to come back from the printer. Ya know, I started last month with 10 projects to work on - finished 2, and started this month with 13. It just aint fair!

I've gotten my mailboxes (snail mail, internet, stg-net, even cis) stuffed with people responding to the premiere. Although not all pertaining to the OSKer, I've had about 30 messages a day total! A few people have critiqued parts of layout or content, but as a whole everybody seemed to enjoy the magazine. One guy went so far as to say that the one issue had more interesting info in it about OS9 than the Rainbow had all year! It's very refreshing having so many people appreciate something you do, and I hope that all of you reading this can get the chance to feel the same by creating something for OS9 and spreading it around. There is a lot of software to write, but there's a lot of brains out there. Okay, off my soap box, and onto some of the remarks.

It seems everybody liked the cover art (Smile Zog!), and thought giving the pages sector numbers and the OS9 directory format were 'cute' (to paraphrase). But the general consensus is that the Allocation Map and Attributres on every single article was pushing it. Hey, I couldn't agree more. I thought it was a good idea at the time. I also got a nudge about the font that was used, and the lack of full justification. Yup, I agree there too. I used that font because I was trying to get the stuff that would line up on a normal text display to look right. And I didn't have the time to go through and mark just those sections using PageMaker. It was a pain to get it to look even half-way decent as the only font that would work was that 8 point super-bold. I'm working on a routine to take of the printing (beginnings of a DTP) in a much nicer format, not to mention under OSK, but it won't be ready for another issue or two. I think I'll go back to Ventura (ick) for the meantime.

Another highly complimented area was the interview with Kevin Darling, although I had a couple of people comment about it's length. I had thought about splitting it into two pieces - in hindsight I realize this would have been a good tactic, as well as saving the eyesight of more than a few readers. Oh well. And yes, Kevin, I need a spelling checker, as well as a brain transplant so I can spell your name right (SORRY!). I can't believe I did that. Well, maybe I can.

I'm eager to hear all of your comments about format, content, etc. Quite frankly, I'm suprised nobody said anything about the doc's inane mumbblings, which was not only a very poor attempt at bad humor on my part, but also the first draft. I had completely re-written that section (on paper) to actually be slightly humorous, and forgot to key it into the file. Oops. That's what I get for not having a portable. Seriously folks, flame at me all you want about the mag - I need to know if I'm doing something even slightly unworthy of my readers. And hey, I've a fire proof suit laying around here somewhere.

And now for a few messages from our sponsors:

....
FROM: SPENT@GJOE (Darrell Spencer)
AREA: SIG
SUBJ: OSKer
DATE: 90/08/01 19:38:11
RCVD: root 90/08/01 20:34:50

Scott, thanx for the complimentary copy of the premier issue of OSKer! It was a very informative collection of goodies. I particularly enjoyed the interview with Kevin Darling...very interesting material. I have been using Microsoft Windows V 3.0 at work on an IBM Model 80 lately and have been considering moving to an AT machine based on the many features available with 3.0. (Movable windows and icons and a few other niceties)

The first issue of OSKer has made me stand back and say, "Wait a minute! First, the switching between tasks (as Microsoft calls them) is SLOW and very jerky with weird flashes of graphics trash in between. Secondly, unless an application is completely recoded (written for Windows), multi tasking is not occuring, just task switching! I believe you and Kevin discussed this having to re-invent the wheel every couple of years. Thirdly, this graphical interface hogs around 4 Meg of disk space and 1 Meg of system ram! What an efficient jewel OS9 LII is.

The bottom line is this: I am holding onto my CoCo a bit longer, listen to the pros and cons of TC70's and MM/1's and stick with the Motorola/Microware combination. We've been on the right track all along. The rest of the world will catch up sooner or later!

Darrell Spencer

P.S. My subscription request is on the way!

....
FROM: Eric@Zog (Eric Levinson)

AREA: os9
SUBJ: The OSKer
DATE: 90/08/03 22:35:41
RCVD: root 90/08/04 01:29:35

I recieved the OSKer magazine today. It looks like a great magazine.

I have one quirk though. In sector 6, a question was asked about power on the COCO bus of the TC9. The Doctor OSKer answered: The TC9 can handle more power than the COCO3 could, because it feeds on a 200W power supply.

This statement is not true. I am an electronics technician and have worked many years in the field.

The term "power" is not handled. "Power" is drawn from a component, not handled. Even though a 200 Watt power supply can supply up to 200W, all 200 watts are not sent into the components, rather a component has a specific power consumption, and it only draws what it needs to use. The rest can be used for other components.

Both the TC9 and the COCO3 have the same power consumption, but since the TC9 has a more sophisticated architecture, it does tend to draw a little more than the COCO3. I would recommend that you mention this in your next issue of the OSKer, so it doesnt confuse users that aren't familiar with power as a whole.

Eric Levinson

....
FROM: sysop@WorkShop (System Operator)
TO: sysop@root
SUBJ: OSKer
DATE: 90/08/03 19:01:13
RCVD: root 90/08/03 23:58:05

Scott, ignore any messages you might have received about my not having a copy of OSKer. It came yesterday. One heck of a magazine! That must have taken forever to transcribe the conversation with Kevin Darling. I'll put my check in the mail to you tomorrow. That's not the same as saying "The check is in the mail."

I was interested to note that Alan Sheltra made the suggestion to drop the apostrophe from OSKer. I sent you the same suggestion on June 2nd.

Keep up the great work!

Al - sysop@WorkShop

....
Sorry Al, you're the one that suggested dropping the apostrophe. I got you confused with Alan on that one. Nothing like hundreds of people reading what you write to bring out the idiot in you eh?

StG

Flame_ON

by Scott Griepentrog

Well, nobody else seems up to building a fire, so I'll start one...

There are a lot of ways of formatting your C program code to make it look readable, but there's only one that should be used. Mine! Let me explain myself.

In the very beginnings of C, code was often formatted like this:

```
main()
{
  if (foo) {
    bar;
    bar;
  }
}
```

But this is ugly. Note that the {}'s line up only on the function, but not in the section of code after the if. The idea I guess was to put the { in place of a single expression to indicate the start of a section of code to be executed conditionally. I've also seen people recently doing this:

```
main()
{
  if (foo)
  {
    bar;
    bar;
  }
}
```

This is ugly too. I don't have a logical reason, it just is. Actually, I do have a reason against it - the uMacs editor automatically indents for you, you can't write code that way using it. Now the proper way to format the same piece of code is:

```
main()
{
  if (foo)
  {
    bar;
    bar;
  }
}
```

Doesn't this just look so much better? The {}'s line up, stick out, you can find them easy, you can see what they're enclosing, it's just beautiful. Well, almost. I've even seen people follow this form, and STILL manage to make an ugly looking program! Here's an example in the form of a code segment from my PostMan module:

```
main(argc,argv)
char **argv;
{
  chkarg(argv);
  if (!usr) _wxt(_ls_scf(2),1,'E','not logged in',0);
  _strass(&u,usr,sizeof(u));
  chuid(0);
}
```

SPACE FOR
RENT

Your ad here!

Reasonable
Rates!

Call (317)

241-6401

```

*to = *area = *rply = *file = *subj = Ø;
hdrcnt = hdrlop = Ø;
dash(argv)
{
  case 't': *(*argv + 1) = Ø; strcpy(to, to ); strcat(to, *argv + 2); break;
  case 'a': *(*argv + 1) = Ø; strcpy(area, area); strcat(area, *argv + 2); break;
  case 'f': *(*argv + 1) = Ø; strcpy(file, file); strcat(file, *argv + 2); break;
  case 's': *(*argv + 1) = Ø; strcpy(subj, subj); strcat(subj, *argv + 2); break;
  case 'd': dflag ++; break;
  case 'z': debug ++; break;
  default: _wxi(_is_scf(2), 1, Ø, "Invalid option ", *argv);
}
if (*argv)
{
  close(Ø);
  if (open(*argv, 3)) _wxi(_is_scf(2), 1, 'E', "cant open ", *argv);
}
close(1);
close(2);
setpr(getpid(), 16);
strcpy(b, cia -> news);
strcat(b, "/news.read");
if ((nfn = open(b, 3)) = EOF && (nfn = creat(b, 3)) = EOF)
  _wxi(Ø, 1, 'E', "cant creat ", b);
if (chrser(u.o, NETOPT) net(rcvd ++); else user());
close(nfn);
if (debug) _wxi(Ø, Ø, Ø, "DONE", Ø);
close(Ø);
if (*argv && dflag) if (unlink(*argv) = EOF)
  _wxi(Ø, 1, 'E', "cant del ", *argv);
}

```

Now here's the exact same code, but well commented and with blank lines between sections of code to separate them. A heck of a lot easier to read, isn't it? I wrote this nearly a year ago now, but I can read through the code and it makes perfect sense. I can't say the same for some of the things I didn't take the time to format carefully.

```

main(argc, argv)
char **argv;
{
  /* check for , -? and get usr info */
  chkarg(argv);
  if (!usr) _wxi(_is_scf(2), 1, 'E', "not logged in", Ø);

  /* copy the d*mn usr rec to u to keep from loosing it due to logout */
  _strase(&u, usr, sizeof(u));

  /* 9Ø/Ø4/3Ø StG - chuld to Ø instead of openØ/creatØ */
  chuld(Ø);

  /* clear option buffers */
  *to = *area = *rply = *file = *subj = Ø;
  hdrcnt = hdrlop = Ø;

  dash(argv)
  {
    case 't': *(*argv + 1) = Ø; strcpy(to, to ); strcat(to, *argv + 2); break;
    case 'a': *(*argv + 1) = Ø; strcpy(area, area); strcat(area, *argv + 2); break;
    case 'f': *(*argv + 1) = Ø; strcpy(file, file); strcat(file, *argv + 2); break;
    case 's': *(*argv + 1) = Ø; strcpy(subj, subj); strcat(subj, *argv + 2); break;
  }
}

```

```

case 'd': dflag++; break;
case 'z': debug++; break;
default: _wxi(_ls_scf(2),1,0,"invalid option ",*argv);
}

/* open input file */
if (*argv)
{
close(0);
if (open(*argv,3)) _wxi(_ls_scf(2),1,'E',"cant open ",*argv);
}

/* close all i/o other than input so we dont get killed */
close(1);
close(2);

/* set priority down */
setpr(getpid(),16);

/* open global news.read file */
strcpy(b,cia->news);
strcat(b,"/news.read");
if ((nfn=open(b,3)) == EOF && (nfn=creat(b,3)) == EOF)
_wxi(0,1,'E',"cant creat ",b);

/* process either a net transfer or user transfer */
if (chrser(u.o,NETOPT)) net(rcvd++); else user();

/* close news.read */
close(nfn);

if (debug) _wxi(0,0,0,"DONE",0);

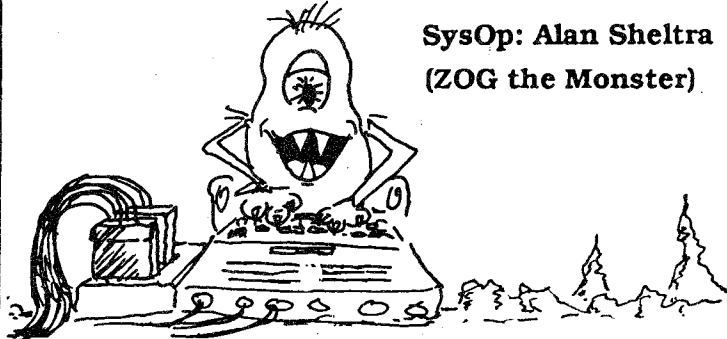
/* then delete input file if needed */
close(0);
if (*argv && dflag) if (unlink(*argv) == EOF)
_wxi(0,1,'E',"cant del ",*argv);
}

```

Note that I didn't bother to comment the section of code starting with the dash(). It's pretty obvious what that does. The code looks twice as long, but it's more than twice as readable. I always write things as if somebody else is going to be reading them - and when I go back to read them several years later, I might as well be somebody else, cause I'm surely not going to remember how I wrote the program the first time!

Visit
Zog's Cavern BBS
(213) 461-7948
2400/1200/300 8/N/1
Now 24 hrs.

SysOp: Alan Sheltra
(ZOG the Monster)



Features NetMail, On-Line Games
OS9 & Coco SIGs & File Transfer
Discussion Boards...
Your Late Night Meeting Place!

How to Build a CDI

by Scott Griepentrog

No, I'm not talking about Compact Disk Interactive (CD-I) that's in an upcoming issue. The following is the technical info on how to build a Carrier Detect Interface. What do you need a CDI for? Read on.

Technical Specifications for the Carrier Detect Interface

Released into the Public Domain

The CDI was developed as an answer to answer a problem with the 6551 ACIA chip used in the Tandy RS232 Pak and most replacements when used in a BBS or other dial-in (ismon) situation. The 6551 chip will ignore incoming data from the modem if no carrier detect (CD) signal is present. Hayes modems output the CONNECT XXXX message BEFORE turning on the CD signal, thus a ACIA device will never receive the CONNECT message unless the CD signal is forced to an ON state. Of course, if CD is always on regardless of the actual carrier detection state, it is difficult to detect when a user disconnects. The modem output the message "NO CARRIER" but writing every single program to look for that message is a bit too much. The idea for this unit actually came from the MULTITECH 224E modem that I had purchased for its internal baud rate adjust feature. That is, the modem talks to the computer at a fixed baud rate regardless of what rate the user is connected at. This modem has three options for carrier detect: forced on, normal, and drop for 2 secs. The third mode allows detection of the connect messages and loss of CD because it leaves CD high except when a user first disconnects from the modem. About 2 seconds after CD was initially lost, it brings the CD signal back to the ON state so that the next CONNECT message can be heard.

The CDI applies this same feature to modems that do not have it built in. The unit is actually a very simple R-C (resistor-capacitor) circuit, borrowing voltage from the DTR signal to feed the CD signal. The parts required and instructions for assembly follow. As it is fairly difficult to damage RS232 connections even if the unit where to be assembled incorrectly, this project can be put together even by somewhat inexperienced persons without fear of destroying anything. The only skills required are identifying the parts listed and limited soldering capability.

Assembly is best done by attaching the parts to a small piece of perfboard, the kind with .1" holes and tin plating on the bottom for soldering to. Attach the larger capacitor to one side of the board so that it will fit in the space between the two DB25's. The easiest way to mount the resistor and diode going to DB25 connections is to stick only one end in the board and leave the other floating in the air. Then connect the wires between pins 1-8 & 20 before wiring in the rest of the circuit.

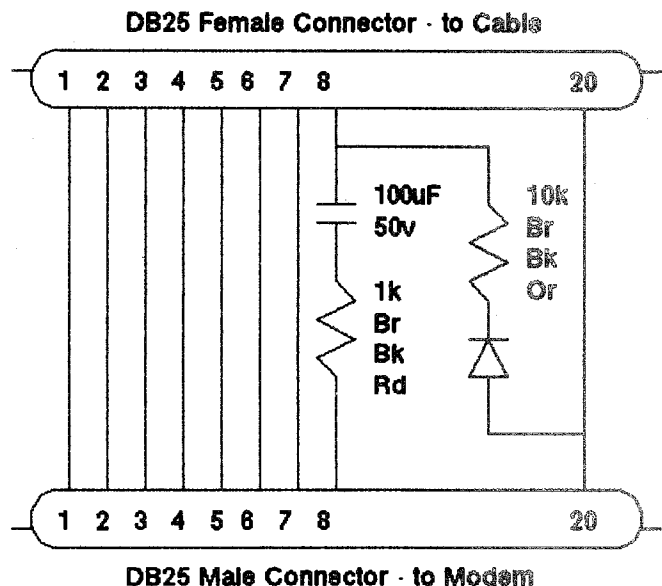
Before closing it up with some hot-melt glue to hold all the pieces in place, test to insure that your new CDI

works. To do this install it on the back of a modem (Male on the modem, cable to modem on Female connector), and start up a terminal program. Assuming your modem supports the AT&C command to turn CD on and off, first issue the AT&C0 to force CD on. You should get a "OK" back from the modem. Then enter the AT&C1 to turn CD back off (normal). You should not see the "OK" message this time. Then enter AT and you should get an "OK" again. When the CD signal goes off, the ACIA misses the following OK while the CDI force CD down for about a second. But following commands work. If it passes the test, you have a functional CDI!

QTY PART DESCRIPTION

- 1 DB 25 Connector, Male, Solder Lugs
- 1 DB 25 Connector, Female, Solder Lugs
- 1 DB 25 Hood, Double-ended (Null Modem Type)
- 8 Wire, about 2" long
- 1 Diode (any generic diode will do)
- 1 Resistor, 1K OHM (Br Bk Rd)
- 1 Resistor, 10K OHM (Br Bk Or)
- 1 Electrolytic Capacitor, 100uF, 50v
- 1 Small piece of perf board

SCHEMATIC DIAGRAM



New Machine Specs

Here they are, the technical information about all three of the new machines, straight from the source. This is mostly to benefit those readers who are overseas or otherwise disconnected from the flow of information about the new machines.

Most of you are already familiar with these files, but check the latest changes the MM1 - a palette controller and two additional serial ports.

As Frank got to go first last time, Paul gets to this time. The following is Paul's latest writeup about the MM1 (with some minor editing) and pricing info attached.

THE MM1

The MM1 comes in a slim-line PC style case with 200 watt power supply, and room to expand to two 5.25" half heights devices and three 3.5" drives.

There's room in case for a total of six boards with optional bus. Attaches to a 16-bit implementation of a 32-bit bus...

MM1 FEATURES AT A GLANCE

Main CPU board (usable alone): Signetics 68070 running at 15 MHz, One meg of RAM for video/cpu included, OS-9/68000 in EPROM (boot from EPROM/disk/HD), Graphics chip (Phillips VSC) with palette controller, RGB Analog port with sync jumpers, DMA normal/high-density floppy controller, Two serial ports, IBM PC keyboard DIN port.

Second I/O board: DMA SCSI high-speed interface, Up to eight megabytes additional cpu hi-speed SIMM RAM, One DB-9 serial port powered for PC serial mouse, Two DB-9 serial ports for modem/printer/etc expansion, CoCo 8-bit analog joystick port, Two Centronics parallel ports (bidirectional), Realtime battery-backed clock w/nonvolatile RAM, Dual channel DMA-able 8-bit analog line-level D/A and A/D ports, for stereo sound output AND input digitization of voice, music, data.

MM1 FEATURE DETAILS

CPU

The MM1 uses the CMOS multifunction Philips-Signetics SCC68070 processor, running at its full rated speed of 15 MHz.

The 68070 is built around an internal bus consisting of a 16/32-bit Motorola 68000-compatible CPU which has been enhanced to recover gracefully from bus faults, a built-in MMU, two DMA channels, a UART serial interface, two counter registers, and an interprocessor hi-speed serial bus interface.

Graphics and math benchmarks place the MM1 at approximately 60% faster than a 10Mhz QT+, and about the speed of a VAX 11/780.

DMA

Direct memory access is used by the floppy disk, hard disk, and sound subsystems. DMA aids in moving data

around a computer system, allowing the system to continue multitasking without problematic interruption. For example, when accessing a floppy disk drive, you won't lose any keystrokes in your type-ahead keyboard buffer.

GRAPHICS

The CMOS Philips 66470B Video and System Controller (VSC) integrates a high resolution color display controller and a 680xx family system controller on one chip, providing the best solution at the lowest cost to you, the customer.

The graphics modes which will be used in the American (NTSC video) version include the following:

320 x 210 - 256 colors/pixel
320 x 420 - 256 colors/pixel (interlaced)
360 x 480 - 256 colors/pixel (overscan)
640 x 210 - 16 colors/pixel
640 x 420 - 16 colors/pixel (interlaced)
720 x 480 - 16 colors/pixel (overscan)

A VGA-derived palette controller allows each of the 16 or 256 displayable colors to be chosen from over 16.7 million shades. Or in other words, 8-bits (256 levels) each of Red, Green, and Blue information.

The output is analog RGB, with onboard jumpers to adapt to most any analog monitor's sync requirements. The standard connector was designed for plugin compatibility with the Tandy CM-8 used by most CoCo-3 owners.

The VSC also has onboard Pixel Accelerator (PIXAC) manipulation logic, which the software uses to greatly speed up text output and pixel combining on the screen.

Pictures stored on disk in the RLE format (an efficient storage format for pictures such as business graphics, animation cels, and so on), can be decoded on the fly by the VSC, making animation extremely easy to implement.

The VSC also includes ROM, RAM and I/O decoding, memory access control, and a watchdog timer to prevent system lockup in case a programmer should accidentally access a non-used area of memory.

Sound

The MM1 implements sound on the CPU board and its I/O add-on board.

IBM PC-style sound is possible from the CPU board, adding utility to programs that need to prompt the user, and to video games. Music, special effects, and simple beeps are possible from the timer-based sound generator, which normally would go through your monitor.

A more complete sound implementation is on the I/O add-on board. Dual (stereo) 8-bit A/D/A converters run to a standard 5-pin din, allowing a simple interface both for recording and playing back sound. Sample rates can be set to far exceed CD sample rates, making excellent sound easy to reproduce. A DMA channel may be used to do either input or output (to both 8-bit converters) without cpu intervention.

Utilities for converting hundreds of existing sound files will be available before the end of 1990.

These ports can also be used for data acquisition, of course.

Parallel and Serial Ports

The Parallel ports on the MM/1 are configured to be compatible with most popular printers. One can be used for the OS/Gateway (due in 1991), a high speed interface to the Color Computer 3.

The MM/1 can have up to five serial ports, three of them with hardware handshaking, making remote communication easier. This helps with uucp and other telecommunications facilities such as BBSs and proprietary messaging systems.

One of the ports can be configured for MIDI. The MIDI board will be ready before the end of 1990.

PC Keyboard port

Any XT detachable keyboard can be used with the MM/1. IMS can also supply you with one. This adds to the professionalism and customization of the MM/1.

1.4 Megabyte floppy

The DMA floppy controller can handle up to 4 drives of any popular type (5" or 3.5" normal or high-density).

The base MM/1 includes a 3.5" floppy drive capable of either 720K or 1.44Meg operations. That means a disk can hold up to four times the storage of double sided CoCo drives. Quick access and data transfer times make OS-9/68000 much easier to use on a floppy disk-based system. IMS will be providing extra floppy disk drives.

SCSI interface

The DMA hi-speed SCSI host controller allows use of up to seven devices. (Examples are hard disk drives, tape drives, CDR/WORM drives)

Memory Expansion

MM/1 includes 1 megabyte of 256K x 4 dynamic RAM on the main CPU board. Memory is easily expandable on the second board using SIMM technology, a cost-effective means of expanding your system to 3 or 9 megabytes.

The 1 or 4 meg SIMMs may be either Mac or PC style; and will be available at low cost from IMS to our customers.

Software

The MM/1 operating system is OS-9/68000, with portions in ROM to save memory. Also included is the compiler, networking software, PC-DOS file manager for reading and writing MS-DOS diskettes, print spooling, tape backup support, Microware Basic, graphics editor, text editor.

Mouse/Joystick Ports

One of the serial ports may be used to connect an inexpensive and accurate PC-style serial mouse.

For games, there is a CoCo-style analog joystick port, with support for two fire buttons, and up to 8-bit resolution on two axis.

Internal Clock

Realtime clock, battery-backed with 56 bytes of non-volatile memory.

Networking

The MM/1 has an inter-cpu networking interface that will later be used with appropriate cables (available separately, requires installation). This interface can be in either master or slave mode. A master can have up to 127 slaves.

Data is transferred over the network at up to 100 KBaud, or the equivalent of five printed pages of text per second. When a computer on the network is not being addressed, there is no overhead on the computer system.

This networking is ideal for low-cost educational and business applications, and can be used in the home to connect several MM/1s.

PRICING INFO

MM1 first board, case, power, floppy: \$779

MM1 as above with keyboard: \$859

MM1 as above with keyboard, monitor: \$1149

MM1 with 2nd board, case, power floppy: \$1125

MM1 2nd board upgrade: \$399

Financing is available, and the machine is due to be released Sept. 15.

Interactive Media Systems Inc.

1840 Biltmore St.

Washington DC 20009

(800) 866-9084 (9-5 EST)

And now, for Frank Hogg's TC70 machine, a close competitor to the MM/1.

Frank Hogg Laboratory is pleased to announce the TC70, the 68K computer of choice for Tomcat/Color Computer/68K users.

The TC70 is the latest in our line of K-Bus compatible products, providing the greatest flexibility and expansion for the OS9/OSK community.

The TC70 is a stand-alone system that can also be used with the TC9 Tomcat for complete OS-9 Level 2 compatibility. It is fully expandable via the K-Bus to over 14+ megabytes of RAM and 60+ ports and is the lowest cost of any system available. These TC70 in conjunction with the TC9 provides both CoCo compatibility as well as OS9/68K. The Tomcat is the most flexible and expandable of any computer system available today.

The TC70 has 50% more built in RAM, a better AT keyboard interface, is more cost effective, and is more

standard with K-Bus compatibility than other 68070 based single board computers announced or on the market.

The Technical Specs

Signetics 68070 CPU (Motorola 68000 compatible) at 15 MHz, 1.5 MB RAM (1,536K), Memory upgradeable to 14+ MB via K-Bus, Graphics resolution from 320x200 to 720x540 (interlaced), From 16 to 256 colors on-screen, depending on resolution mode, Three serial ports expandable to 60 via K-Bus, PC keyboard port for 101-key AT-style keyboard, RGB-Analog output for CM-8 Style monitor and RGB TTL for PC monitors, OS9/68K Professional Version with C and Basic included, Direct Memory Access (DMA) floppy disk controller, DMA SCSI host adapter built in for hard drives and tape, K-Bus compatible, TC9 compatible (CoCo 3), 8-bit D to A port, 8-bit port A to D (CoCo joystick), 1 parallel port for parallel printer expandable to 60 via K-Bus, Serial mouse port, Real-time battery-backed clock.

CPU

The Signetics 68070 is a Motorola compatible CPU running at 15 MHz

I/O Support

The PC keyboard port is designed for standard AT-style keyboards. The AT-style keyboards are available in a better quality than XT keyboards and also provide bi-directional control of the keyboard LEDs from the computer. This way CAPS lock etc can be tied into each window.

Floppy disk controller is included at no extra charge. Supports both 3.5 and 5.25 drives and ALL OS9-OSK disk formats including CoCo, Mizar, Atari, Motorola etc etc. Also supports our PC Utility for using PC DOS disks.

The TC70 floppy controller uses separate DMA from the SCSI port allowing very fast transfer from hard disk to floppy, great for backups. Our SCSI drivers, proven by over 6 years of use supports all SCSI hard drives, tape drives and most SASI/SCSI controllers including XEBEC, OMTI, Adaptec, Western Digital etc.

Software support

Microware's OS9/68000 Professional version with C and BASIC is included. Our port of OS9/68K is a mature port with over 6 years of proven reliability. Additional utilities only available for the Tomcat system extend OS9/68K to the utmost.

Expansion

The TC70 can be expanded with K-Bus cards.

Physical specs

The TC70 is 5.25 X 8 (The same size as a 5.25 disk drive) and has mounting holes that allow mounting to a 5.25 drive. This allows very flexible mounting. The TC70 will fit in and is an upgrade to the QT, QT Plus and QT 00x. The TC70 also mounts in the K-Bus

and will work with the TC9 board and other K-Bus cards.

Pricing

The preliminary price is \$999.99 for the TC70 board and software. Complete system prices and final specifications will be uploaded later. Consult the Tomcat brochure for TC9 pricing.

Availability

The TC70 will be available early September.

For more information or to be placed on the waiting list for any of our Tomcat computers contact:

Frank Hogg Laboratory, Inc.
204 Windemere Rd.
Syracuse NY 13205
315/469-7364
FAX 315/469-8537

Unfortunately, Frank's price list is rather large and won't fit in the space I have, and he has a lot of options with his bus scheme. Please contact him directly for price information.

the OSKer

Official Standard Sub-Standard Subscription Program

- 10 PRINT "YOUR NAME"
- 20 PRINT "YOUR ADDRESS"
- 30 PRINT "YOUR CITY, STATE, ZIP"
- 40 IF (IN USA) INCLUDE \$12
- 50 IF (IN CANADA) INCLUDE \$15
- 60 ELSE INCLUDE \$20
- 70 MAIL TO:

the OSKer

P.O. Box 24285

Speedway IN 46224

OS9_in_Industry

by Robert Miller

As most home users of OS9 know, Radio Shack (aka Tandy) no longer supports our beloved Color Computer. They claim to, but the truth is evident. The CoCo is obsolete, and there are no plans for an upgrade apparent on the horizon. For us OS9ers, this means that, until the MM/1 and the TC9 are released, there is no easily available, inexpensive avenue for putting OS9 in the home. It is available for the Mac, Amiga, Atari ST, and even the PC clone (by means of an AT card with an on-board 68000 processor) running DOS concurrently! But all of these implementations are not widely supported, very expensive, and, as is obvious, not very popular.

This brings up some interesting questions. Consider these facts: OS9 has not been a highly successful operating system (along the lines of DOS and Mac's Finder) in the home. It's high point in sales was probably at least three years ago. It is not used extensively in offices, nor in schools. But Microware, an international company with offices throughout the world, is still in business. Where is OS9 being sold?

Having eliminated most other options, and realizing that it must be getting used in an area that is not readily visible to the casual computer user, it becomes obvious that OS9's major sales must be made in the industrial world. This is indeed the case. As surprising as it seems, our trusty little disk file organizer is being used by scientists and engineers the world over in laboratories and R&D rooms to control machines, acquire data from the outside world and process it, operate as the user interface to complex networks of incredibly fast, expensive machines, and even to prepare presentations that mix stereo sound, graphics, animation, and even true filmed sequences of reality!

Before I go any further, let me introduce myself and give myself some credibility. My name is Robert James Miller, I attend University of California, Riverside as a Computer Science major working towards a B.S., and I have been using OS9 at home for about 3 1/2 years. But I am also a sales engineer for a representative company here in Riverside, that deals in the selling of primarily board level products (as opposed to chips or systems) and some peripherals all throughout Southern California. I personally sell in the Los Angeles County territory. Two of the companies that we represent, Radstone Technology and GreenSpring Computers, develop boards for the VME bus structure. (More on VME later.) These companies both have OS9/68000, hereafter to be referred to as OSK, ported to their processor products. What this means is that not only do I have a great deal of interest in OS9 as a personal operating system, but I have exposure to it in the real computer industry. And thus, here I am, humbly providing you the reader with information in an attempt to bridge the gap between OS9 use at home and in the industrial world.

The gap that I am referring to is significant. To make my point, I would like you to ask yourself right now if you think that this is the only OS9 specific magazine currently in publication. I would bet that many of you do. But there is another OS9 specific magazine that has been in print since before 1980! Of course it didn't start as an OS9 specific magazine. It began as a magazine dedicated to the state of the art line of processors developed by Motorola: the 6800 and the 6809. It is called "68 Micro Journal," and it has followed the Motorola line of CPU's throughout the last ten years. It is known as an industry mag, read mostly by engineers during coffee breaks. The major contributors are hardware developers, writing about technologies that they have designed. And it has recently, due to the overwhelming use of OS9 as the operating system for Motorola based industrial machines, dedicated itself to our operating system. It refers to itself as an "OS9 Systems Integration" publication. Not casual reading material, unless you happen to be one bright enough to consider techniques of shell development and designs of integrated circuits casual.

So now you know. OS9 is being used in industrial applications by major institutions. Wait, did I say major institutions? I guess I haven't mentioned some of Microware's best customers yet. NASA's in there. They like OS9's real time capabilities. (Real time? Again... I'll explain later.) Sony and Phillips IMS like OS9 a lot, too. They enjoy its portability, and ease of interface to new, innovative devices. Other companies use OS9 scattered among the thousands of projects that they are involved with. These include the likes of Hughes Aircraft Company, TRW, McDonnell Douglas, and Jet Propulsion Laboratories, as well as other aerospace companies. In all of these organizations I have either seen or heard of the use of OS9. And I only work in Southern California.

Now comes probably the most important question of all that I have presented. What does this mean to us, the home users of OS9 who hope to be able to afford our next hard disk when we run out of space. It means that there is a lot more support for us out there than we think. But it also means that we are not central to Microware's universe, as they are to ours. Right now, we are small fish to the big boys in Des Moines. They are more than happy to let us do our own development and build our own machines. They are ecstatic to see a new interest arising in their operating system. But until we start to turn into some actual, significant revenue for them, they will do little more than encourage us and watch from the sidelines. Not that they don't want to help, but they are a corporation, and they do have to watch the bottom line. However, as those of us who have had a chance to talk to them know, they are supportive and would definitely like to see our recent efforts come to fruition as much as we would.

I realize that all of this talk is very vague. I have been referring to what are (to some of you) unknown,

SYSTEMS such as real time operations, and advanced bus structures. And I have been setting the stage for interesting examples of actual projects that involve OS9. Hopefully, I have alerted the curiosity of one or two people out there enough to warrant the time I have spent writing this. What I plan to do is write a column for "The OSKer" every month, pertaining in some way to the use of OS9 or (primarily) OSK in industrial applications. I will start next month by writing a tutorial about bus structures and real time operating systems, explaining the terms that will be used quite often throughout my discussions. (I like to call them discussions. They're actually ramblings, but I like to dream.) After that, I will start with a few examples of actual applications that are currently using OS9. And from there... that's a ways off. We'll decide when we get there.

I truly mean "we" when I say that, as well. I would like to get some feedback from people who are reading this magazine. Comments, criticisms, and kudos (especially kudos :-)) are more than welcome. In fact, I will only continue to write these articles if I really feel that there is an interest. I would hate to be taking up space in Scott's wonderful, if young, magazine when there isn't anyone who wants to hear what I say. So if you like it, or don't, feel free to write. I'd even appreciate grammatical corrections. At least I'd know you're reading. Don't be too hard on me, however. I have an easily damaged ego. :-)) (--- For those of you that are unfamiliar with this little symbol, it is a sideways happy face. In computer lingo, it means that a humorous or sarcastic remark was just made, one not to be taken seriously, or at least without a grain of salt.)

I realize that this column is geared towards people who are unfamiliar with the industrial world of OS9. However, both myself and Scott would like to see engineers and others reading "The OSKer" as well. So if you aren't a beginner, but had an interest in something I referred to, or would like to see some more advanced topics being discussed in a different forum, please let me know. We don't want anybody to be excluded from having a magazine to call their own.

I can be reached by U.S. Mail at:

Robert James Miller
500 Hibiscus Dr #E201
Redlands, CA 92373

If you have access to a facility that allows internet, (internet is another topic I might discuss in the future) I can be E-mailed at rmiller@ucrmath.ucr.edu. I do not as of yet have a CIS or Delphi account, but hopefully I will in the fairly near future. I'll let you know. I won't release my phone number for now, because it's likely to change within the next three months (as will my address), and I'm rarely at home anyways. That wraps this up, now to wait for those comments.

DISCLAIMER: While I mention a lot of public and private organizations in this column and will continue to in future columns, most of them I am not affiliated with. The ones that I am affiliated with I am not

representing in writing this article. All opinions are my own, not those of my employer or any other organizations mentioned. And, finally, while I get my information from reliable sources, and I do my best to write only factual information, I cannot accept responsibility for any incorrect information or for anyone else's use of incorrect information that has been obtained from this column. I will, however, publicly apologize and correct my error if someone knows information to be incorrect. I would like to know as much as my readers. Whew! Thank you.

Nine Central L.A.'s Outlet for OS9/OSK !!!

SPECIAL CLOSE-OUT ON THE StG Login Package V3.0 !!!

Included in the StG Login Package V3.0:

- Run a powerful multi-user BBS without losing use of your computer.
- Exchange messages and files through a nation-wide network automatically.
- Package contains modules: TSMON, LOGIN, MAIL, NEWS, POSTMAN, MENU, HELP, CHAT, NEWUSER, OPTION, PASSWORD, and many more.
- Run up to 8 lines at once - Any OS9 program can be run remotely (except for mouse or direct graphics).
- Binary files can be sent as either Mail or News to other Machines.
- FREE updates/improvements are distributed through the network.
- Upgrade to V4.0 when released - FREE!
- Assorted utilities also included.
- Anlmajk's Games PAK for the StG Login Pakg. included FREE (Exclusively from Nine Central).
- Includes printed manual.

Special Close-out Price \$40.00

Supply is limited, so ORDER NOW!

Make Payable to:

Wayne Campbell
P.O. Box 85043
L.A., Ca. 90072

Call (818) 753-9864 (Voice)
10am. to 4pm. (PST)
or (213) 461-3872 (BBS)
7pm. to 7am. (PST)

Please allow 1-4 Weeks for delivery
Include \$3.00 S&H (Check, M.O.)
\$7.00 for COD. - U.S. Currency Only

Playing Chess in C

by Scott Griepentrog

Hey, I'm not normally into games, but writing one can be a learning experience. And one of the most commonly done, as well as most difficult, is chess. Although the idea of pieces on a board can be easily translated into a computer language, developing the rest is quite a challenge. Each of the pieces has different rules governing its movement, and then there's the problem of making the computer understand the game well enough to be a good opponent. It all begins to sound very difficult - but it is exactly that kind of challenge that I live for, or rather, program for. As soon as somebody tells me something can't be done, you can bet I'm hard at work thinking up a way to do it.

I also want to use this program as a way to introduce some of you to the C language that before now have not taken the time to learn it. We're going to start out real slow, by first doing some planning on how to make this thing work. Each month a section of the program will be developed, adding together until we have something useful. Before we're done, we'll have added termcap support for doing full-screen display on different terminals, the capability to play against the computer as well as another user, and maybe even get into some hi-res 3d graphics. But one step at a time, so you have a whole month to digest the new code before we move on. And, you can give me feedback on alternatives to my own methods - maybe we can make this a group effort?

Some tools you will find handy during this are 1) a C compiler, either OS9, OSK, or even PC or Unix (until we really get into it). 2) The manual that came with the C compiler so you can look up what certain functions do, how to compile a program, etc. Very necessary. 3) The book "The C Programming Language", by Brian w. Kernighan and Dennis M. Ritchie, the guys who wrote C. This isn't a must have, that is, if you already know what's in it. My copy is only about 4 years old, but is already falling apart. You can still buy the book in stores, though you may have to order it, and the newer edition covers ANSI C. For what we're doing, ignore the ANSI stuff, we either don't have it, won't use it, or can write around it. And finally, the most important: 4) your brain online, and lots of time. Many a project has been started without one of the two, but never finished. Now that you're ready, on to the program...

The first thing that needs doing would be a routine that displays the board layout. The output routine is usually the first thing that is written, because it shows you what's going on in the program. Input can be pre-set, and the output routine can be completely debugged before going on to the next step. Splitting up the task into smaller pieces not only makes the job seem easier, but keeps you from going through the whole program to find a bug. If each piece was tested as it was developed, the bug is usually in the section you just wrote. But writing a program in this fashion

also means you have to plan how the pieces interact via common variables and so forth - so that they will work with each other as they are assembled into a finished product.

Before just sitting down and writing out some code, it's important to sit back and consider for a moment what would be the best way to store the board layout in memory. In C, there are several 'types' of variables for storing things. The type char is essentially a byte except that it has a sign. That is, instead of storing the number zero through 255, it stores -128 through 127. How and why is not important right now, to explain that I have to get into two's complement. There are a number of different pieces, let's see, the whole bottom row of pieces totals eight, plus the pawn would be 9 pieces for each side.

No, wait a sec..., there's less than that. There are two Rooks, two Bishops, and two of those Horseys. So there's really only 6 distinct pieces: the King, Queen, Bishops, Knights, Rooks, and Pawns. But then times 2 for the two sides makes 12 different pieces. But then computers have this nasty habit of calling 0 a number. Even if there's nothing there, that nothing counts as something - because it has to be something if the computer is going to think about it. Confusing isn't it? A square that has no piece on it has to be considered. If a particular square on the board can be occupied by any one of 12 pieces, or no piece at all, that makes a total of 13 possibilities, or 'states'. And each state can be represented by a number; numbers are the only language a computer really understands anyways.

So it's pretty obvious that a type char will work for storing which piece is sitting on a particular square of the board. Now for the question of how to number them. What are the options here? We could start at 1, counting upwards for each piece. That would make 1-6 for one side, and 7-12 for the other. Or what about using 1-6 for one side, then maybe 9-14 for the other. But why skip 7 and 8 you're asking? Thinking forwards to a time when we'll be writing a routine to test for a valid move, for instance, it would be nice if the routine worked the same for both sides of the board. Each type of piece moves in a certain way, so it's important to know which one of the types of pieces occupies a certain square, regardless of which side it belongs to.

A sneaky way to do this (otherwise known as a neat hack) is to use the bit-wise AND operator of C. The binary for a 6 would be 110, and the corresponding piece for the other side would be number 14, or 1110 in binary. Note that the last three bits are the same, and that two sets of numbers, 1-6 and 9-14, are offset by 8, which happens to be a significant position in binary. Given a particular number representing a piece, say foo, the equation (foo & 8) would tell what side the piece is on, and (foo & 7) would tell which kind of piece it is. By placing different information in separate bits (known as bit-mapping), the needed numbers can be easily separated using the AND, which

is a lot faster than doing a compare and subtracting. Just a little planning ahead like this can save a heck of a lot of coding. Of course, it also helps to know how binary works.

Oh, and don't mistake the operator & with the && operator in C. The single ampersand is bit-wise, and the double && is logical. I.E. to pull out certain bits, you use (number & number), but in an if this and the other situation you use (true && false). If this doesn't make sense just yet, look at some code where both are used for examples.

Okay, back to our choosing number to represent the pieces. Are there any other ideas? Hmm... What if you used ASCII characters (which are assigned numbers) to represent the pieces? Lessee... 'K' for King, 'Q'ueen, 'B'ishop, 'K'night, 'R'ook, 'P'awn. Oops, that's two K's. Hmm. Maybe 'N' for Knight? Hey, it's better than 'H' for Horsey. Not very elegant, but it might work. But then what about the other side? I know- use lower case. That way the toupper() function of C can be used to look at both sides the same, and a simple comparison (foo-'a') will yield which side the piece is on.

Because we want to keep this as easy to understand as possible, maybe going with the ascii form would be the best way. It will make the program easier to read, not to mention write. And for display purposes, we can always convert the letters to something better looking - like eventually a neat 3d graphics display. But right now it will make life easier to just print out those letters, instead of having to convert from arbitrary codes. Okay, everybody agreed? What else do we need to figure out. I know - what are we going to use to represent a square that does not have a piece on it? Going with the Ascii convention, it would be logical to use a space. But is there maybe a better choice? What about using zero? That would allow us to say "if (piece)" instead of "if (piece==' ')" all the time.

C allows just a variable in an if statement that will be tested for zero or non zero. In fact, it also allows us to use the NOT operator (exclamation point, often called 'bang') which then lets you write "if (!piece)" which is read as 'if not piece'. Or, if you're really into C programming, you say 'if bang piece'. The bang changes a zero to non-zero, and non-zero into zero. Sorta handy eh? Oh, by the way, C uses the 'equals equals' for testing something, instead of just one equals sign like in basic. If you were to say "if (piece==' ')" then C would make piece equals to a space, and then test it for being non-zero. It's easy to forget to put in the extra equals when doing comparisons, and every now and then I find a bug where I've done just that.

Ugh, where were we? Oh yah, so the question is, space or zero. If we use space, we can just print it, but it puts all that extra muck in our program. But if we use zero, we can put just one extra line in the display routine to convert it to a space, and leave a lot of extra tests for space out of the code. And believe it or not, C programs run faster if you omit the extra test too - which means that when we get down to the

time critical routines to figure out the best move for the computer... well, you get the idea. It's amazing how just sitting back for a few minutes and thinking over some of the minor details can make a big difference later on.

Okay, now we need some storage. We decided to use a char for each board position to remember what piece is on it, but we need an 8 by 8 arrangement of these chars. So we declare an array. But we also need to name this array variable. Let's see, it's a n image of the board, so board would probably be a good name. The code to declare it then is:

```
char board[8][8];
```

Which makes 64 chars. But it's very important to realize that this array is declared exclusively, but referenced inclusively. I know, what the %\$#@ did I just say, right? Okay, exclusively means to exclude, or not count, the zero. There are 8 columns and rows on the board, 1 through 8. Inclusively means to include the zero when counting. So our 8 becomes 0 through 7 instead. Referenced is a big word that just means 'look at'. So when you look at an element (one char out of the array) in C, you have to use 0-7. Basic of course does things exclusively, 1-8, which tends to create an odd syndrome where new C programmers who started programming in Basic have been discovered to have a substantial decrease in hair. Aint this just so much fun?

Okay, looks like we're all ready to start writing code. First thing that we need to do is write a function (small section of code with a name) to put the pieces in the right places on the board. We'll make use of a while loop to put some of the pieces in easier. First things first, though, let's name the function. What does it do? It puts the pieces on the board. Hmm. Normally I like to name functions with a combination of two words - a verb that describes what we're doing, and a noun that names what we're doing it to. As an example, taking a 10 pound sledge hammer to your computer because it won't compile your program would be called "hammer_computer(10)". But that gets a bit much to type every time you want to do it, which is quite often. So naming functions becomes an art. Naming it "hamcom()" would be a little easier on the fingers, as long as you don't forget what it means years later and start thinking that you're having a conversation with a ham. How about "setbrd()?"

```
setbrd()  
{  
  int x,y;  
  
  /* first row */  
  board[0][0] = 'R';  
  board[0][1] = 'N';  
  board[0][2] = 'B';  
  board[0][3] = 'Q';  
  board[0][4] = 'K';  
  board[0][5] = 'B';  
  board[0][6] = 'N';
```



```

board[0][7]='R';

/* row of pawns */
x=0;
while (x<8)
{
    board[1][x]='P';
    x++;
}

/* 6 empty rows */
y=2;
while (y<6)
{
    x=0;
    while (x<8)
    {
        board[y][x]=0;
        x++;
    }
    y++;
}

/* another row of pawns */
x=0;
while (x<8)
{
    board[6][x]='P';
    x++;
}

/* and the last row */
board[7][0]='r';
board[7][1]='n';
board[7][2]='b';
board[7][3]='q';
board[7][4]='k';
board[7][5]='b';
board[7][6]='n';
board[7][7]='r';
}

```

Oh, that `x++` is the same thing as saying `x=x+1`, only shorter (and faster). The type `int` is either 2 bytes, or 4 bytes, depending on which machine you're on (6809 vs. 68k), and is also signed. For our purposes, either will do, as we're only going as high as 7. In fact, a `char` would do, but is more conventional to use `int` for indexing (selecting one of the elements). `setbrd()` goes through the whole array in order, and sets the piece types into `board[][]`, and sets the empty elements to 0. Now to display the board on the screen:

```

dspbrd()
{
    int x,y;

    /* y,x nested loop */
    y=0;
    while (y<8)
    {
        x=0;
        while (x<8)
        {

```

```

            if (board[y][x]) putchar(board[y][x]);
            else putchar(' ');

```

```

                putchar(' ');
                x++;
            }
            putchar('\n');
            y++;
        }
    }
}

```

This function is pretty similar to `setbrd()`. I bet you're wondering what `\n` is, right? It's the the carriage return, or next line char. Each board position is checked to see if there's a piece on it. If so, that character is put on the output path for display. If not, a space is. Another space is displayed between each piece, and after every line it puts the carriage return out to go the next line. That was actually pretty easy right? Okay, now put the two functions together, and write a `main()`.

```
#include <stdio.h>
```

```
char board[8][8];
```

```
/* put setbrd() here */
```

```
/* put dspbrd() here */
```

```
main()
{
    setbrd();
    dspbrd();
}

```

The line `#include <stdio.h>` is always put at the top of every C program (well, almost always). It loads some stuff into your program that the `putchar()` function needs, as well as some other handy stuff we'll discuss later. For now, just put it in and ignore it. Then we declare the board. Because this is done outside all the functions, all of them can use the board array without having to pass it to them. The `x` and `y` `int` in each of the first two functions can only be used within the function, and each pair are separate. And lastly, the `main()` function is put in. C automatically calls this function when the program runs, and it calls the other two in order. Piece of cake, right? Not bad for a day's programming, eh?

Oh, and in case you don't know how to compile this, just type it all into a file called `'chess.c'`, and then compile it with the command `'cc chess.c'`. If you're using OS9, you may have to use `ccl` instead of `cc`. Run it by entering `'chess'`, and it should display the board. All that work and you're not all that impressed huh? Well, just wait, next month we're going to add routines to let you move pieces around!

And if you get bored in the meantime, see if you can figure out a way to condense (simplify) the `setbrd()` function. I can see a number of things that would bring the size of the function down to about 10 lines or less even.

Syscall_for_C

by Scott Griepentrog

Mike Haaland called me up and wanted the equivalent to the SYSCALL function in Basic for OSK C. So I wrote one. End of story.

```
#include <types.h>
#include <machine/reg.h>

/*
 * _osk(code,reg) - call an OSK system call directly
 *
 * code = OSK system call code number
 * reg = pointer to register struct a la <machine/reg.h>
 *
 * NOTE: passes/returns registers D0-D7, A0-A4 ONLY!
 *       also returns SR (status register, condition codes)
 *
 * PD by StG 90/08/15
 */

_osk(code,reg)
int code;
REGISTERS *reg;
{
  #asm
  * push a5 on stack (can't modify it)
    move.l a5,-(sp)

  * put passed vars in registers for use later
    movea.l 4(sp),a0          a0 = code;
    movea.l 8(sp),a5         a5 = reg;

  * shove program on stack (sp) backwards
    move.w #$4e75,-(sp)      *--sp = instruction("rts");
    move.w a0,-(sp)         *--sp = a0;
    move.w #$4e40,-(sp)      *--sp = instruction("trap #0");

  * grab registers from &reg, execute stacked program, put regs and stack back
    movem.l (a5),d0-d7/a0-a4 with (d0-d7 && a0-a4) do that_register=*a5 ++;
    jsr (sp)                (*sp)();
    movem.l d0-d7/a0-a4,(a5) with (d0-d7 && a0-a4) do *a5 ++ = that_register;
    move sr,64(a5)          *(a5 + 64) = status_register;

  * restore stack, and check for errors
    lea.l 6(sp),sp          sp += 6;
    bcc.b no_err            if (error)
    move.l d1,errno(a6)     errno = d1;
    moveq.l #-1,d0         d0 = -1;
  no_err:

  * restore original a5
    move.l (sp)+,a5
  #endasm
}
```

What is Multi Media Anyway

Who wants to know?

Now that's a loaded question. There has always been a trend in computing to have more memory, faster speed, better resolution graphics, and interface to anything and everything. This is the eternal truth of computers and their programming capabilities will expand.

So, what you're sayin' then is, some dude just come up with this crazy name, right?

Well, that's the question before us, yes. Is there really more this industry buzz-word than the mere concept of a computer being able to do more than it was before? The definition is actually very simple. Multi Media means that more than one 'Media', that being text, graphics, animation, or sound, are used together in a program.

Hey, wait a sec, does 'dat mean my Tetris game that's got all them funny blocks and bleeps at me is one of these MultiMedia thingles?

Well, yes, in a way all video games these days could be considered MultiMedia, because they combine both graphics and sound.

So what's the big deal then, eh?

The idea behind MultiMedia is that these combinations can be used together in ways that were not possible before. For example, your average, ordinary, everyday business computer is suited more towards crunching numbers, keeping track of data, and doing word processing than producing an animated demonstration of a roller coaster - with a view from the front seat including the clackity clack sound and screaming. And it would have controls allowing you to stop it at any time, run it backwards, slow it down, speed it up, and even modify the construction of the track. A machine that can happily keep the corporate office functioning correctly does not need the speed, high resolution graphics, and sound output capabilities required by such an application.

Wow! That would be really rad, man! Hey, I could make it take a turn too fast and blow off the track - that'd be cool, watch it fly through the air...

Yes, well, you could do that I suppose, depending on how flexible the program was. Actually, a better demonstration of this would be to add a history of roller coasters to our example. Have a database of all existing roller coasters, linked to a map of their locations. Include stories about them, and pictures. Pick out a city you're going to be in soon, and it'll show you a picture of the roller coaster there. Select more information, and it'll give you a complete text description, including the builder. Select the builder and get a list of coasters build by them, and a description of their work. Select another coaster by the same people and go take a ride on it. Even though different

types of information are stored, they are interlinked so that you can jump from one to another easily. Actually, this gets more into another buzzword, HyperMedia.

What, you got all yur medias all hyper and running around buzen or something?

No, HyperMedia means that you have different information 'linked' together, as in our example. The information came from different people, in different media forms, but is all tied together in this HyperMedia form to make it easy to explore at will. Actually, the term comes from HyperText.

Geez, man, you lost me...

HyperText is the same principle of different information linked together, only just with text. For example, say you are reading about Shakespeare, come across mention of his story Romeo and Juliet, and decide to read the story. You select Romeo and Juliet, ask for the text to the story, and begin reading. But you keep running across words you don't know. So you select the word, ask for a definition. After a few minutes of this, finding yourself spending more time in the dictionary than reading the story, you happen across a word that is so obscure that the definition doesn't make sense. So, you select that word again and ask for any other files that have the same word, and so on. You can explore the text files you have stored by jumping around at will - this capability is called HyperText. But add graphics, sound, and animation to the same, and you've got HyperMedia. But to be able to do HyperMedia, you have to have a machine that can handle MultiMedia. So, does it all make sense now?

Naw, man, it's all Greek to me!