the world of 68' micros

Tandy Color Computer, OS-9, OSK

15 December 1993 Vol. 1 Number 4 $4.50 Canada, $4.00 US

the editor and staff wish each of you a joyous holiday season and a happy new year

DON'T FORGET:

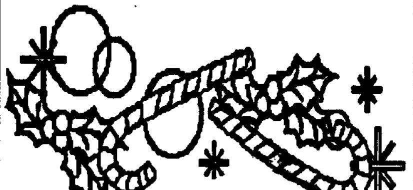
If your four issue subscription began with the first issue,

THIS IS YOUR LAST ISSUE!

Renew now to keep "68' micros" coming.

NOTE: The month/year of your last issue is after your name on the mailing label

CONTENTS



POSTMASTER:

If undeliverable return to:
FARNA *Systems* PB
Box 321
Warner Robins, GA 31099-0321

the world of 68' micros

Published by:
FARNA Systems
P.O. Box 321
Warner Robins, GA 31099-0321

Editor: F. G. Swygert

Subscriptions:
$23/year (8 issues) US; $30/year for Canada/Mexico ($12 US, $16 C/M for six months- four issues). Overseas $35/ year ($18 for four issues) surface. Add $8/year, $4 six months for air mail.

Advertising Rates:
Personal Classifieds- $0.20/word, $4 minimum. Name, address, and phone number free. FREE for subscribers.
Display Ads- $15 1/6 page, $20 1/4 page, $35 1/2 page, $60 full page, copy ready. Add $10 for special placement, $10 for typesetting ($5 1/4 or less). Dot matrix will be typeset if deemed unacceptable and submitter billed. 10% discount for four or more appearances.

Back issues are $4 per copy. Overseas add $1 each for airmail delivery.

Newsstand/bulk orders available. Dealers should contact the publisher for details.

Problems with delivery, change of address, subscriptions, or advertisers should be sent to the publisher with a short description.

The publisher is available for comment via Internet (dsrtfox@Delphi.com) or Delphi E-mail (DSRTFOX). The CoCo and OS-9 SIGs on Delphi are also frequented (The Delphi SIGs are sponsored by Falsoft).

The editor speaks... *F.G. Swygert*

As predicted last time, there were over 200 copies of the last issue sent out. Soon we will start shipping third class mail. What does this mean to "68' micros"? It means the publication will actually start making a worthwhile profit. What third class mailing will mean to the subscriber is about a wek delay in getting the magazine. I fully intend to shift production ahead by at least one week, so subscribers shouldn't notice more than a two or three day delay. Canadian and air mail subscribers won't be affected, as all mail to Canada must be first class.

The alternative to third class mailing would be to raise subscription rates by a minimum of $5 per year. Each copy currently weighs just under three ounces, I could add four pages at the most (two sheets of paper) and stay under three ounces. The cost to mail three ounces first class is 75 cents (80 cents to Canada). Mailing up to 3.5 ounces third class only costs 25 cents, plus stamps don't have to be placed on each copy, just mailing labels (third class mail imprint is printed on each copy). Why at least 62.5 cents each? So that additional pages can be added if needed without incurring *another* price increase.

The magazine was planned from the start to take advantage of third class mailing as soon as subscriptions neared the 250 mark. Without third class mailing or the $5 yearly price increase, there are not enough profits to justify the time and expense required to produce a good magazine. I just accepted the fact that there would be no profits for six to eight months- the cost of getting started. Do note that I haven't been losing money, however. The magazine DOES break even with first class mailing right now, I just work for free.

Now I don't want to alarm anyone. I promised to explain how things would be run when practical, and that's all I'm doing- explaining why the switch to third class mailing is necessary and what it means to you. I also promised to publish a magazine for a minimum of two years, more if possible. To date, all goals have been met. Third class mail is just the next step to building a profitable, long lasting magazine. If the current growth rate continues and third class mailing is successful, we will be around for quite a bit longer than two years!

Well, I FINALLY got the BASIC program listings right in the last issue. But there was at least one mistake that crept in. I accidently left the "Sub-Etha" logo out of the Sub-Etha Software ad on page 29. Sorry about that Allen! No excuses.. I just got in a hurry to get the issue out and forgot to put it in. Didn't even notice until all copies were printed. I just have to pay more attention to those little details!

DEMOGRAPHICS

Just out of curiosity, I counted subscriptions from every state when mailing the 01 November issue. This is how 268'm is spread over the country (and Canada):

LESS THAN 1%:	
Hawaii	North Dakota
Alabama	Maine
New Hampshire	Rhode Island
New Jersey	
1% TO 2%:	
Washington	Oregon
California	Utah
Idaho	Oklahoma
Louisiana	Nebraska
Missouri	Montana
Wisconsin	Iowa
Kentucky	Tenessee
North Carolina	West Virginia
Virginia	Delaware
Conneticut	Massachusetts
Puerto Rico	
3% TO 5%:	
Texas	Indiana
Ohio	Pennsylvania
New York	New Jersey
OVER 5%:	
Arizona	Michigan
Florida	Georgia
Illinois	Canada

Most of the states don't have a concentration of CoCo/68K users, but Arizona is an exception- 50% of the users there are in the Phoenix area!

If you can help to bring your state up by two percent, EVERYONE in that state will receive their choice of (1) a free issue of "microdisk"... their choice over the first year of "68' micros", (2) an issue added to their subscription, or (3) their choice of one back issue. So get out there and get friends to subscribe! Renewals for four issue subscribers will count this time also... so **if you haven't renewed that short subscription, now would be a good time.** Besides, if you don't, this may be your LAST ISSUE!

< 268'm >

Letters to the Editor

Dear Sirs,

I found out about your magazine at a meeting of the CoCo users group in Manistee, Michigan, and I really liked the issue they showed me there. I want to subscribe!

I have used my CoCo for entertainment and learning for about ten years, now. I remember when the CoCo 2 came out and when I saw the powerful graphics commands I knew that was the computer I would buy.

Most of my CoCo computing has been writing routines with Edtasm+, learning assembly language on a machine that restarts easily if it crashes. I've found cassette storage to be adequate for me, although I miss the random access of a disk.

Here are some topics that I would like to read about (or perhaps you might publish a reading list):

*Biography, tribute, or interview with either William Barden or Forrest Mims.

*Implementing the Peripheral Interface Adapter (PIA) chips.

*Use of the parallel port.

*Moving data between computers.

*Software that runs the Hi-Res adapter.

*Putting older CoCos to use.

The last one is a favorite theme of mine. Although I plan to move up to a CoCo 3, the early models are still out there, often cheap. Why waste what we've learned by limiting it to the desktop? I have set up a CoCo, cassette, and portable TV for less than most folks pay for their mouse. Why shouldn't systems like this be used for jobs like graphing, calculating, even robotics? How hard would it be to use even a 4K CoCo to count events and sound bells? So often it's cheaper and cleaner to use a small computer than it is to build a project, and much easier to test and change. The REAL computer revolution will come when we can start building machines out of modules as powerful as yesterdays TRS-80, but as cheap and easy to work with as the buffers and logic of today.

Thanks for providing a great magazine. Please put my money to good use. I was tempted to send payment for the book on the development of the CoCo, (one of the club members had it), but I thought I should find out if it's available. Please let me know.

Craig A. Wheeler
Box 136
Mears, MI 49436-0136

Craig,

Your letter was very well written and touches on a number of interesting subjects. William Barden disappeared from the CoCo scene several years ago, on a sour note I might add. He published a book (Connecting CoCo to the Real World) but didn't deliver all orders, leaving some people out in the cold. I know this for a fact, I was one of them! He refused to answer inquiries so was impossible to get in touch with. I did hear that a few who made complaints through the Postmaster General's office later received books, but by then it was to late for me.

I agree with you on using older CoCos for specialized jobs. Hopefully some readers will let us know what they have done with theirs. It is very easy to use the joystick ports to collect data from simple switches. What have you done with yours? Readers... any interesting projects?

I'll see what I can come up with on the PIAs I assume you mean the expansion port when you say "parallel port", or do you mean an add-on parallel printer port? Some interesting things have been happening on moving data between computers, and I hope to get someone to write an article on the subject.

Dear Mr. Swygert,

I was pleased to see my color-chart program published in the first issue of 268'm, and I'm also pleased that the magazine seems to be off to a good start.

I do have a few minor complaints/remarks: First, you must have been bleary-eyed getting out that first issue, because you gave me a different name in copyright line 15. Also, this line wasn't identified as a REMark, so run as is the program will give an "?SNERROR IN 15". In the second issue, I liked the sidebar about the current issue of "microdisk", but you should have included prices of subscriptions and single copies. Finally, you have a habit of putting an apostrophe in "it's" when it is not correct. A very minor point, but does detract from the professional polish of your publications.

If you are interested and I can find the time to write an accompanying article, I may submit for publication a BASIC spelling drill program. The cassette recorder is used to "say" each word so that a student could learn correct pronunciation as well as spelling. Many owners may be reluctant to drag out the cassette recorder and cable though.

Richard Bair
335 Jefferson Avenue
Glencoe, IL

Richard,

I'm sorry for giving you a new name! I don't know where I got "Steve" from... must have talked to a "Steve" on the phone while typing... (?).

Your points about the magazine have been taken. Hopefully you noticed the prices in the "microdisk" sidebar last issue. And I'll try to watch those "it's" also!

There is now a sound digitizer *for the CoCo. You might want to check out the FARNA Systems ad for it. It would allow you to record spelling words* on disk *rather than tape. Of course, many people still have their tape recorders too. Maybe you can write two versions?*

Hi Francis,

I've started to read 268'm fully, and I'm thoroughly enjoying it. It's by far the best alternative since Rainbow. Can you confirm for me if Farna systems is still offering the patch OS9 and Quick Reference Guide as a $12 combo? If so you will get an order in the mail.

I'll tell you what I would like to see in 268'm. My main interest is OS-9. The reason I still have a CoCo3 is because I did so much programming for my own uses -predominately in Basic09. Anything on OS-9 is of interest to me (though unfortunately, much has been over my head. Some day I'm really going to have to learn system calls and the like). I am interested in C programming but I have always been stumped by the differences between Microware CoCo C and present day K&R C. I have trouble making many simple functions work in C which are routine in Basic09, and much easier in present standard C. Simple C programs compatible with CoCo C would be appreciated. Also as I realize that some day I'll have to move up, all info on OSK and OS-9000 is of interest. Being a non hacker, I need a system that I can get supported, and I'm wary of small vendors.

Keep up the good work, & keep the issues coming. By the way, I hope you plan on billing the 4 month subscribers (like I believe me) for renewals. Else some will drop by the wayside unintentionally.

Joel Sherman
JSHERMAN, Delphi

Joel,

The Microware C compiler is a "standard" K&R compiler. What you refer to as K&R C is actually (most likely) ANSI C, which is used by most other systems. The K&R standard is oldest. Maybe a reader more familiar with C than I can explain the differences between the Microware CoCo C compiler and ANSI C.

I understand your wariness of small vendors, but they are THE BACKBONE of the OS-9 and CoCo communities. Without support, they will fall by the wayside also. Most of the vendors still in the market are the ones who intend to stay!

I am slowly getting more articles in on OSK systems and C. And yes, FARNA still offers Patch OS-9 and the QRG combo for $12 (+$2.50 S&H).

A Letter for Joel Hegberg...

Dear Joel,

I like your "OS-9/OSK Answer" column. Two observations:

1) In volume 1 #1, p15, you state: "Unfortunately, CoCo 3 Windows does not scale the position of the graphics cursor..." Scaling is a function of OS-9 and does not require WindInt. The program was using the absolute rather than the relative mouse pointers, Pt.WRX and Pt.WRY (or regs.x+28 through 31 in the context of the program). I have written programs which run on either 320 or 640 graphics screens without adjustment by reading the relative positions.

2) The BASIC signal handler mentioned in volume 1 #2, p17, was originally printeded in July 1988 Rainbow by Dale Puckett (pp174-181).

Robert Gault
832 N. Renaud
Grosse Point Woods, MI 48236

Thanks for passing this along! It is possible that Mr. Jaeger forgot the original source of the code. At any rate, I apologize to Mr. Puckett and Rainbow for not crediting the source.

Letters are printed on a space available and popular subject matter basis. If you don't want your letter printed, or wish to withhold your address or name and address, please state so when writing. In some cases, letters are slightly edited for space and/or clarity. If a personal reply is desired, please enclose an SASE.

Key Codes

Stephen Carville

Determining which key was pressed in "C" code.

While it is possible for a program to be designed that only uses control and other single byte keys to send commands this may seem a bit "unfriendly" to the user with a modern terminal. Using a control B to move a cursor left and a control F to move the cursor to the right may have made sense once but now we have perfectly good arrow keys on the keyboard.

Given that any useful application program will, probably, be interactive and require the user to communicate his desires by using some or all of these special keys, the programmer must have a way of determining exactly which key was pressed.

Most modern terminals send multi byte sequences for many of their "special" keys such as the page up, insert, arrow or function keys and the exact value of these sequences can vary from one terminal to the next. For example, the up arrow key on my System IV console system sends the three byte sequence $1b $5B $41 while a Digital Equipment Corp. (DEC) VT100 will send $1B $4F $41 for the same key. While this gives a terminal greater flexibility than trying to shoe-horn these special keys into the 255 characters available to a single byte it does pose a challenge to the programmer in determining what key was pressed.

One way to accomplish this is a simple technique that converts a multibyte sequence into a 32 bit unsigned integer. As each character is received, the current value of the integer to return is shifted left four bits and the new keypress added to it. This use of an integer value instead of a string greatly simplifies the lookup process.

The algorithm starts by initializing the number to be returned to $10 (#16). This value was chosen because it is the smallest initial value for the four bit shift used that guarantees the returned number will be greater than $FF (#255). It is important that any returned value that represents a "special" key be greater than $FF so that it can be distinguished from a printable character.

If "keys" is the value to return and "kp" is the value of each character in the sequence, the process looks like:

```
initialize keys to $10
get kp
shift keys left
add kp to keys
    while more input to read
    get keypress
    shift keys left
      add keypress to keys
  endwhile
  return keys
```

To use this technique in a program, first extract the needed information from termcap, convert these sequences into integers and store them in a table. The program can then read the sequence of characters from the keyboard, convert them into integers and use the table to determine which key was pressed. The entire process is demonstrated in the program hashkey.c.

The first step, extracting the needed strings from the termcap file, is accomplished in the routine getterm(). After extraction, a string is converted to an integer by the routine makekeycodes() and this value is saved in the array keycodes[]. Notice that each value is stored at the position one less than the offset defined in the header. This is done so that returning a zero can signify that the value under consideration is not in keycodes[].

Once the keycodes[] array has been initialized with all unused spaces set to a one, the last entry set to NULL and all the encoded sequences in their proper place, the program waits for a key to be pressed in the routine inkey(). To insure that printable characters are returned unmodified the first character received is checked to make sure it is a control character. If the value returned is greater than $FF (#255) it represents a control character or a multibyte sequence.

The routine delay() was suggested by Ed Gresick of Delmar Co. as necessary to insure that fast machines with slow terminals will have the chance to read the entire sequence without imposing any delay on fast terminals or on console systems.

Once a value is returned from inkey(), scankey() checks for the value in the array keycodes[]. If the value exists, the offset+1 is returned or, if the value is not in keycodes[], a zero is returned. Based on this offset program control branches in a switch() statement.

Some terminals do not have all of the special keys a designer may wish to have his program use. When this happens, a program can use a "default" set of commands where ^F, ^B, etc., replace the arrow keys and other special keys are replaced by two key sequences. If the first character in these sequences is the same for all and the variable "metachar" is set equal to it, the program will not have to do any special processing but can just respond to the two key sequence exactly as if it were from a single key. If all of the required keys exist then leave "metachar" set to the "impossible" value.

It may not always be easy to determine what value to assign to the variable "seqstart". I've set it to $1B (#27) in this demo program because most modern terminals transmit escape sequences for special keys as well as screen control. This variable is only important when the terminal can transmit faster than the program can process the information.

Some bright reader out there is bound to notice that a 4 bit shift will not generate a unique value for every sequence. The two byte sequence $02 $01 (^B^A), for example, will return a $1021 which is exactly the same as $01 $11 (^A^K) would return. I, personally, know of no terminals that will cause this kind of problem. The escape sequences that ANSI terminals use are a $1B (#27) followed by one or more printable characters and some terminals use a similar strategy but with a control A (^A) in place of the escape. This should not be a problem in reading the sequences sent from single keys but a programmer must be aware of it when setting up two key sequences.

Another important feature demonstrated in this program is the matter of restoring on exit any user settable values that the program may need to alter. The routines settmode() and restoretmode() demonstrate how this can be done easily for certain of the terminal settings.

Source code listings are both included in the file "LISTING.C" on microdisk. In the printed listings on the next page, a slash and asterics mark the beginning and end of comment lines (/* comments*/).

```
/*************************************
hashkey.c
   This program demonstrates a technique for
parsing keyboard input by converting the character
sequence from the terminal into a 4 Byte unsigned
integer.
compile: cc -xit=/r0 -l=/dd/lib/termlib.c -m=4k
*************************************/
#include  <stdio.h>
#include  <errno.h>
#include  <sgstat.h>
#include  <termcap.h>

#define  HASHIFT   4  /*shift this many bits*/
#define  KEYS_IVAL  0x10  /* intital value for
the encoded number */
#define  DELAYCOUNT 6    /* value for loop
counter in delay() */
#define  MAXCODES  10   /* reserved space
for this many keycodes */

/* keycode offsets */
#define  UA    1   /* up arrow (ku) */
#define  DA    2   /* down arrow (kd) */
#define  LA    3   /* left arrow (kl) */
#define  RA    4   /* right arrow (kr) */
#define  TB    5   /* tab key (ta) */
#define  BS    6   /* backspace key (bs) */
#define  XC    7   /* escape key */

/* default values */
#define  ESCAPE   0x1B  /* escape */
#define  TAB      0x09  /* tab (^I) */
#define  BACKSPACE  0x08  (*backspace(^H)

/* typedefs */
typedef char FLAG;          /* boolean defined */

/* global variables */
char tcstrings[1024],     /* store terminal control
strings here */
   PC_,               /* pad character */
   *CL;          /* clear screen and home cursor */

struct _sgs oldmodes;  /* storage for the original
tmodes */
unsigned int
  keycodes[MAXCODES+1],   /* storage for
encoded key information */
  metachar=0xFFFFFFFF;  /*metacharacter for
2 key sequences set here to "impossible"value */
int sigval=0;  /*global value for signal recieved*/
int seqstart=ESCAPE;       /* start of multi byte
sequence set to $1B for demo purposes */
/* function declarations */
int catchsig(),
  scankey(),
  delay(),
  main();
void initkeycodes(),
  settmode(),
  restoretmode();
char *getenv();
unsigned int inkey(),
       makekeycode();
```

```
/*************************************
MAIN
*************************************/
main()
{
unsigned int value;      /* value of key pressed */
int offset;  /*offset to key pressed in keycodes[]*/
char *term;              /* terminal name */

term=getenv("TERM");
if(term==0)
  exit(_errmsg(0,"Environment variable TERM not
defined\n"));

if(getterm(term))
  exit(0);

cls();
settmode();
intercept(catchsig);

printf("Hit the <Escape> Key to exit program\n");

while(1)               /* do this forever */
  {
  value=inkey();
  printf("key value= $%X, #%d. ",value,value);
  offset=scankey(value);
  switch(offset)
    {
    case UA:
      printf("That was the Up Arrow Key\n");
      break;
    case DA:
      printf("That was the Down Arrow
Key\n");
      break;
    case LA:
      printf("That was the Left Arrow Key\n");
      break;
    case RA:
      printf("That was the Right Arrow Key\n");
      break;
    case BS:
      printf("That was the Backspace Key\n");
      break;
    case TB:
      printf("That was the Tab Key\n");
      break;
    case XC:
      printf("Bye for now\n");
      restoretmode();
     exit(0);
      break;
    default:
      printf("\n");
      break;
    }
  }
}

/*************************************
read the INput KEY sequence
*************************************/
unsigned int inkey()
{
static unsigned char ch=0;
register unsigned int kp,
          keys=KEYS_IVAL;

if(ch==0) /* if no key waiting to be processed */
  read(0,&ch,1); /*get next character from stdin */
kp=ch;
if(kp>0x1F)           /* if not control */
  {
  ch=0;
  return kp;          /* send it back as is*/
  }

keys<<=HASHIFT;
keys+=kp;

if(keys==metachar)      /* if 2-key sequence start */
  {
  read(0,&ch,1);        /* wait for next character */
  kp=ch;
  keys<<=HASHIFT;
  keys+=kp;
  }

while(delay(DELAYCOUNT))
  {
  read(0,&ch,1);
  kp=ch;
  if(kp==seqstart)      /* if new sequence start */
    return keys;

  keys<<=HASHIFT;
  keys+=kp;
  }
ch=0;
return keys;
}

/*************************************
Key Press DELAY necessary for slow terminals returns
1 if data ready, 0 if not. The length of the delay can be
adjusted by changing the value cntr passed to the
program. Experimenting has determined that 6 is about
right. This routine was suggested by Ed Gresick of
Delmar Co. as neccessary to insure that fast machines
will read an entire sequence from slower terminals
before returning from inkey().
*************************************/
int delay(cntr)
register WORD cntr;
{
while(cntr--)
  {
  if(_gs_rdy(0)>0)      /* if data ready */
    return 1;       /* return 1 */
  tsleep(2);          /* otherwise sleep a bit */
  }
return 0;           /* if timed out return 0 */
}

/*************************************
SCAN the KEYcodes returns offset to key pressed
 0 if not in keycodes[];
*************************************/
int scankey(kp)
int  kp;
{
register WORD n=0;

while(keycodes[n])       /* until NULL is reached */
  {
  if(keycodes[n++]==kp)
    return n;
  }
return 0;
}
```

```
/************************************
GET terminal information from TERMcap returns
 0 on unrecoverable error otherwise 1
************************************/
int getterm(term)
char  *term;  /*name of termcap entry to read */
{
char  tcinfo[1024],       /* storage for extracted
termcap entry */
      defkey[2],          /* used to set keycode for
default values */
      *_tcptr,            /* points to storage for
extracted string */
      *temp;

FLAG  xflag=0;      /* set if missing critical info
from termcap */

if(tgetent(tcinfo,term)<=0)     /* get termcap info
*/
  {
  _errmsg(0,"Termcap entry %s not
available\n",term);
  return 1;
  }

_tcptr=tcstrings;         /* point to start of global
termcap storage */

PC_=0;              /* default pad character */
temp=tgetstr("pc",&_tcptr);         /* get pad
character if defined */
if(temp)
  PC_=*temp;

/* get needed control string */
CL=tgetstr("cl",&_tcptr);          /* clear screen
and home cursor */
if(CL==0)
  xflag=_errmsg(1,"Clear Screen (cl) is not
defined in Termcap\n");;

initkeycodes();   /* initialize keycodes array */
defkey[1]=0;         /* insure defkey[] has null
terminator */

/* set keycodes with defaults */
defkey[0]=TAB;   keycodes[TB-1]=makekey
code(defkey);
defkey[0]=BACKSPACE;keycodes[BS-1]=make
keycode(defkey);
defkey[0]=ESCAPE;  keycodes[XC-1]=makekey
code(defkey);

/* get keys without defaults */
temp=tgetstr("ku",&_tcptr);
if(temp)
  keycodes[UA-1]=makekeycode(temp);
else
  xflag=_errmsg(1,"Up Arrow (ku) is not defined
in Termcap\n");

temp=tgetstr("kd",&_tcptr);
if(temp)
  keycodes[DA-1]=makekeycode(temp);
else
  xflag=_errmsg(1,"Down Arrow (kd) is not
defined in Termcap\n");

temp=tgetstr("kl",&_tcptr);
if(temp)
  keycodes[LA-1]=makekeycode(temp);
else
  xflag=_errmsg(1,"Left Arrow (kl) is not defined
in Termcap\n");

temp=tgetstr("kr",&_tcptr);
if(temp)
  keycodes[RA-1]=makekeycode(temp);
else
  xflag=_errmsg(1,"Right Arrow (kr) is not
defined in Termcap\n");

return xflag;
}

/************************************
CLear the Screen
************************************/
cls()
{
write(1,CL,strlen(CL));
}

/************************************
MAKE a KEYCODE from a null terminated
character string
************************************/
unsigned int makekeycode(s)
unsigned char *s;        /* string to convert */
{
register unsigned int v=KEYS_IVAL;

while(*s)            /* until null terminator */
  {
  v<<=HASHIFT;
  v+=*s++;
  }
return v;
}

/************************************
INITialiize the KEYCODES[] array
************************************/
void initkeycodes()
{
WORD n;

for(n=0;n<MAXCODES;n++)
  keycodes[n]=1;

keycodes[MAXCODES]=0;       /* terminator! */
}

/*************************************
SET the Terminal MODEs - set the terminal
descriptor to do things the
   way we want them done
*************************************/
void settmode()
{
struct _sgs newmodes;

_gs_opt(1,&oldmodes);  /*get it once to save*/
_gs_opt(1,&newmodes);     /*and once to alter */

newmodes._sgs_case=0; /*upper & lower case*/
newmodes._sgs_echo=0;     /* set no echo */
newmodes._sgs_pause=0;    /* no pause */
newmodes._sgs_eofch=0;    /*esc doesn't mark
EOF */

/* uncomment these next two lines to turn off
keyboard interupt and abort */
/*newmodes._sgs_kbich=0;   /* no keyboard
interupt ^C */
/*newmodes._sgs_kbach=0; /*or keyboard abort
^E*/

_ss_opt(1,&newmodes);
}

/*************************************
RESTORE the Termimal MODEs - resets the terminal
back the way it started
*************************************/
void restoretmode()
{
_ss_opt(1,&oldmodes);
}

/*************************************
CATCH SIGnals
  just exits after resetting terminal
*************************************/
int catchsig(signal)
int signal;
{
sigval=signal;
restoretmode();
exit(sigval);
}
```

< *268'm* >

NEW PRODUCT ANNOUNCEMENT!

EDTASM6309 Version 2.02 for the CoCo (DECB) is now available. This is a major patch to Tandy's Disk EDTASM and offers many improvements over the original version and EDTASM6309 V.1.0. The program suppoorts all CoCo models. The CoCo 3 version uses an 80 column screen and runs in 2MHz mode. YOU MUST ALREADY OWN TANDY'S DISK EDTASM TO MAKE USE OF THIS PRODUCT. It WILL NOT work with a disk patched cartridge EDTASM.

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
$35 + $4 shipping & handling

Some Changes to EDTASM:
1) Tape no longer supported
2) Buffer increased to over 43K bytes
3) Enter V command directly from Editor & ZBUG
4) Multiple FCB & FDB data per line
5) FCS supported
6) SET command works properly
7) Screen color same as that set in Basic
8) Symbol table printed five per line (cc3 only)
9) Actual errors printed with /NL option
10) Warning on long branch where short possible
11) ZBUG defaults to numeric mode
12) Supports all RGBDOS drive numbers
13) Latest 6309 opcodes supported for assembly and disassembly.
14) Auto detects 6309, traps illegal opcodes
15) Lowercase support
16) Local names supported

many more improvements!

Maze Maker

Kenneth Reighard Jr.

Make your own mazes for "Mister Maze!"

Mr. Maze is a CoCo 3 game requiring a joystick. It was inspired by those wooden labyrinth games where you guide a marble through a maze avoiding the little holes cut out of the surface. Mr. Maze has one maze built right in to the program, and also allows you to load maze files created with this issue's MAZMAKR editor program. The editor creates mazes using a graphical editor, and also lets you enter maze code, as listed in the main program.

A maze consists of four basic things: Walls (red sections) are very neutral. They do you marble no harm, but they do not let you pass through them either. Holes (black circle) are bad. They always love to have you drop in. Your games is over when you fall in a hole.

There are two good things in mazeland, however. Checkpoints (dark blue squares) give you points when you cross them. The finish line (orange block) gives you points, and also the satisfaction of conquering the maze.

You control the marble by "tilting" the maze. The maze is tilted by pointing the joystick in the direction you want the maze tilted. The marble (white dot) will roll that way. The marble has inertia, and will not change directions immediately. Roll the marble over the checkpoints and to the finish line.

Scoring is dependent on each maze. The built in maze awards 100 points for each checkpoint passed, and 200 points for crossing the finish line. This can be changed in your new mazes.

To load a newly created maze, select LOAD MAZE at the main menu with the joystick. The maze files on the disk will then be listed. Use the arrow keys to select the maze you want, and press [ENTER]. Press [E] to return to the menu.

I hope you enjoy MR. MAZE and MAZMAKR. Any questions, comments, or suggestions can be sent to me at my school address:

Kenneth Reighard
3355 Dorr St.
Toledo, OH 43607
I can also be reached via the internet at:
reighard@jupiter.cse.utoledo.edu

NOTE: Send your creations in to the magazine and they will be featured on later issues of "microdisk",.

```
1 ' MR. MAZE MAZE MAKER
2 ' BY KENNETH REIGHARD, JR.
7 POKE 65497,0
8 CLEAR 2000
9 CLS:INPUT"MONITOR (R/C)";Q$:IFQ$="R"
THEN MN=-1 ELSE IF Q$="C" THEN MN=0
ELSE9
10 DIM MZ(38,22), P(7), PT(8), FL$(34)
15 GOSUB 5000
20 HBUFF 1, 500
23 HCOLOR 10,11
25 POKE &HE6C6, 18:POKE &HE6C7, 18
27 WIDTH 40
35 GOSUB 6000
37 END
40 GOSUB 4000
41 X=1:Y=1
42HGET(X*8,Y*8)-(X*8+7,Y*8+7),1
44HCOLOR 13:HLINE(X*8,Y*8)-(X*8+7,Y*8
+7),PSET,B:HCOLOR 2
45 Q$=INKEY$
50 IF Q$="" THEN 45
55HPUT(X*8,Y*8)-(X*8+7,Y*8+7),1
60IF Q$=CHR$(8) THEN X=X-1:IF X< 1THEN
X=38
70IF Q$=CHR$(9) THEN X=X+1:IF X>38 THE
NX=1
80 IF Q$=CHR$(10) THEN Y=Y+1:IF Y>22 T
HEN Y=1
90 IF Q$=CHR$(94) THEN Y=Y-1:IF Y<1 TH
EN Y=22
100IF Q$="W" AND MZ(X,Y)>9 AND MZ(X,
Y)<12 THEN MZ(X,Y)=10:GOSUB 3000
110IF Q$="B" THEN Q=MZ(X,Y): MZ (X,Y)
=11:HLINE(X*8,Y*8)-(X*8+7,Y*8+7), PRESE
T,BF:IF Q<8 THEN P(Q)=0 ELSE IF Q=8 TH
EN F=0 ELSE IF Q= 12 THEN S=0
111 IF Q$="F" AND MZ(X,Y)=11 AND F=0 T
HEN F=1:MZ(X,Y)=8:GOSUB3040
112 IF Q$="S" AND MZ(X,Y)=11 ANDS =0 T
HENS=1:MZ(X,Y)=12:GOSUB3020
113IF Q$="H" AND MZ(X,Y)>10 ANDMZ(X,
Y) <12 THEN MZ(X,Y)=9:GOSUB 3 010
114 IFASC(Q$)>=49AN ASC(Q$)<= 56THEN
IF MZ(X,Y)=11 THEN C=ASC (Q$)-49:IF P(C
)=0 THEN P(C)=1: MZ(X ,Y)=C:GOSUB 3030
115 IF Q$="Z" THEN HSCREEN 0: RETURN
116 IF Q$="P" THEN GOSUB 1000:GOSUB
3100:HSCREEN 2
120 GOTO 42
1000 HSCREEN0:GOSUB3200:ATTR3,2:CLS
1010 PRINT TAB(12)"Enter point value. ":PRI
NT TAB(3)"Press [ENTER] to ret ain old valu
e.":PRINT
1020 FOR Q=0 TO 8
1030 LOCATE 1,Q+4:IF Q<8 THEN PRINT"C
heckpoint"(Q+1)":"PT(Q) ELSEPRINT"Finish
:"PT(Q)
1035 NEXT Q
1037 FOR Q=0 TO 8
1040 LOCATE 25,Q+4:IF Q<8 THEN PRINT"
#"(Q+1)": "; ELSE PRINT"Fin : ";
1042 LINE INPUT Q$
1050 IF Q$<>"" THEN PT(Q)=VAL(Q$)
1060 NEXT Q
1070 RETURN
3000HCOLOR 10:HLINE(X*8,Y*8)-(X*8+7,Y*
8+7),PSET,BF:RETURN
3010XX=X*8+2:YY=Y*8:HDRAW"BM=XX;,
=YY;C9G2D3F2R3E2U3H2L3":HPAINT(X*8
+4,Y*8+4),9,9:RETURN
3020HCOLOR 12:HLINE(X*8,Y*8)-(X*8+7,Y*
8+7),PSET,BF:S=1:RETURN
3030HCOLOR C:HLINE(X*8,Y*8)-(X*8+7,Y*
8+7),PSET,BF:HCOLOR 9:HPRINT(X,Y), HEX
$(C+1):P(C)=1:RETURN
3040HCOLOR 8:HLINE(X*8,Y*8)-(X*8+7,Y*8
+7), PSET,BF:F=1:RETURN
3100 IF MN THEN FOR Q=0 TO 7:PALETTEQ
,8:NEXT Q:PALETTE 8,38: PALETTE 9,0:PAL
ETTE 10,36:PALETTE11,9:PALETTE12,18:P
ALETTE 13,27
3105 IF NOT(MN) THEN FOR Q=0 TO 7:PAL
ETTEQ,9:NEXTQ:PALETTE8,38:PALETTE9,0:
PALETTE10, 7:PALETTE11, 11:PALETTE 12,
18:PALETTE13,31:PALETTE14,63:PALETTE15,9
3110 RETURN
3200 IF MN THEN RGB ELSE CMP
3210 RETURN
3300 FOR Q=0 TO 15
3310 IF MN THEN PALETTE Q,9 ELSE PAL
ETTE Q,11
3320 NEXT Q
3330 RETURN
4000 GOSUB 3300:HCOLOR 10,11:H SCRE
EN 2:HCLS:HLINE(7,7)-(312,18 4),PSET ,B:
HPAINT(0,0),10,10:GOSUB 3100
4010 FOR Y=1 TO 22:FOR X=1 TO 38
4020C=MZ(X,Y)
4030 IF C=10 THEN GOSUB 3000
4032 IF C=9 THEN GOSUB 3010
4034 IF C<8 THEN GOSUB 3030
4036 IF C=8 THEN GOSUB 3040
4038 IF C=12 THEN GOSUB 3020
4040NEXTX,Y
4050 RETURN
4099 ' LOAD MAZE
4100 POKE 65496,0
4102 F=0:ER=0
4103 FOR Q=3 TO 11
4105 DSKI$ 0,17,Q,A$,B$
4110 FOR QQ=1 TO 127 STEP 32
4113 IF MID$(A$,QQ+8,3)="MAZ" AND MID$
(A$,QQ,1)<>CHR$(0) THENF=F+ 1:FL$(F) =
MID$(A$,QQ,11)
4114 NEXT QQ
4115 FOR QQ=1 TO 127 STEP 32
4116 IF MID$(B$,QQ+8,3)="MAZ" AND MID$
(B$,QQ,1)<>CHR$(0) THENF=F+ 1:FL$(F) =
MID$(B$,QQ,11)
4117 NEXT QQ
4118 NEXT Q
4125 HSCREEN 0:GOSUB 3200:CLS3:ATTR
3,2
4130 IF F=0 THEN PRINT"      No maze files
on this disk!":SOUND100,5:FORQ =1TO1000
:NEXT Q:ER=1:GOTO4190
4135 IF F<18 THEN FF=F ELSE FF=17
4137 FOR Q=1 TO FF
4138 LOCATE 2,Q:PRINT LEFT$(FL$ ( Q),8)
+"."+RIGHT$(FL$(Q),3)
4139 NEXT Q
4140 IF F<18 THEN 4150
4142 FOR Q=1 TO (F-17)
4144 LOCATE 22, Q:PRINT LEFT$(FL$ (Q+1
7 ),8)+"."+RIGHT$(FL$(Q+17),3)
```

```
4146 NEXT Q
4150 Q=1
4151 LOCATE 4,20:PRINT"Select maze with
the arrow keys.":LOCATE 9,21: PRI NT"Pres
s [ENTER] to load.":LOCATE 6, 23:PRINT"P
ress [E] to return to menu.";
4152 IF Q>17 THEN QX=22:QY=Q-17 ELSE
QX=2:QY=Q
4154 LOCATE QX,QY:ATTR 2,4:PRINT LEFT
$ (FL$(Q),8)+"."+RIGHT$(FL$(Q),3);
4156 Q$=INKEY$
4160 IF Q$="" THEN 4156
4161 LOCATE QX,QY:ATTR 3,2:PRINT LEFT
$ (FL$(Q),8)+"."+RIGHT$(FL$(Q),3);
4162 IF Q$=CHR$(10) THEN Q=Q+1:IF Q>F
THEN Q=1
4164 IF Q$=CHR$(94) THEN Q=Q-1:IF Q<1
THEN Q=F
4166 IFQ$=CHR$(8)THEN IFF>17THEN IFQ
> 17THEN Q=Q-17ELSEQ= Q+17
4167 IF Q$="E" THEN ER=1:GOTO 4190
4168 IF Q$=CHR$(9) THEN IF F>17 THEN IF
Q>17 THEN Q=Q-17 ELSE Q=Q+17
4169 IF Q$=CHR$(13) THEN 4173
4170 GOTO 4152
4173 ATTR 3,2:CLS:LOCATE
9,11:PRINT"Loading: ";LEFT$(FL$(Q),8) +"."
+RIGHT$(FL$(Q),3)
4175 OPEN "I",#1, LEFT$(FL$(Q),8)+"."+ RI
GHT$(FL$(Q),3)
4177 FOR Y=1 TO 22:FOR X=1 TO 38
4180 INPUT #1, MZ(X,Y)
4182 NEXT X,Y
4183 FOR Q=0TO8:INPUT #1,PT(Q):NEXT Q
4184 CLOSE #1
4190 POKE 65497,0
4191 GOSUB 3100
4192 RETURN
4200 ER=0
4210 PRINT"Filename (no ext.) (E - Exit)":
LINE INPUT">>> ";Q$
4211 IF Q$="E" THEN ER=1:GOTO 4280
4212 Q=INSTR(Q$,".")
4214 IF Q=0 THEN 4217
4216 Q$=LEFT$(Q$,Q-1)
4217 QQ$=LEFT$(Q$,8):IF LEN(QQ$)<8
THEN FOR Q=1 TO (8-LEN(QQ$)):QQ$=
QQ$ +" ":NEXT Q
4218 Q$=LEFT$(Q$,8)+EX$:QQ$=QQ$+ RI
GHT$ (EX$,3)
4219 EX=0
4220 FOR Q=3 TO 11
4222 DSKI$ 0,17,Q,A$,B$
4226 FOR QQ=1 TO 127 STEP 32
4228 IF MID$(A$,QQ,11)=QQ$ THEN EX=1
4230 IF MID$(B$,QQ,11)=QQ$ THEN EX=1
4232 NEXT QQ,Q
4234 PRINT
4235 IF EX=0 THEN 4280
4236 PRINT"File exists - Overwrite (Y/N)?"
4238 QQ$=INKEY$:IF QQ$="" THEN 4238
4240 IF QQ$="Y" THEN 4280 ELSE IF QQ$
="N" THEN ER=1:GOTO 4280
4250 GOTO 4238
4280 RETURN
4300 POKE 65496,0
4302 CLS:PRINT TAB(10)"Save maze ": PRI
NTSTRING$(40,"*");
4303 EX$=".MAZ"
4304 GOSUB 4200
4306 IF ER=1 THEN 4390
4307 PRINT:PRINT"Saving "Q$
4310 OPEN "O", #1, Q$
4320 FOR Y=1 TO 22:FOR X=1 TO 38
4325 PRINT #1,MZ(X,Y)
4330 NEXT X,Y
4340 FOR Q=0 TO 8
4345 PRINT #1, PT(Q)
4350 NEXT Q
4360 CLOSE #1
4390 POKE 65497,0
4395 RETURN
5000 FOR X=1 TO 38:FOR Y=1 TO 22
5010 MZ(X,Y)=11
5020 NEXT Y,X
5030 FOR Q=O TO 8:PT(Q)=0:NEXT Q
5040 RETURN
5100 CLS:PRINT"Current maze will be lost!!!"
5110 PRINT"Are you sure (Y/N)?"
5120 Q$=INKEY$:IF Q$="" THEN 5120
5130 IF Q$="Y"THEN5140 ELSE IF Q$="N"
THEN 5150 ELSE 5120
5140 PRINT:PRINT"Erasing maze..."
5145 GOSUB 5000
5150 RETURN
5200 ATTR 3,2:CLS
5210 PRINT:LINE INPUT"CMD: ";Q$
5212 Q=INSTR(Q$," "):IF Q=0THEN CD$=Q$
ELSE CD$=LEFT$(Q$,Q-1):PR$= RIGHT$ (Q
$, LEN(Q$)-Q)
5214 IF CD$="EDIT" THEN LN=VAL(PR$):IF
LN<1 OR LN>38 THEN PRINT:PRINT"LINE N
UMBER OUT OF RANGE.":GOTO 5210 ELS
E GOTO 5300
5216 IF CD$="EXIT" THEN RETURN
5220 IF CD$="SCORES"THEN GOSUB 1000
:GOTO 5200
5230 IF CD$="CHECK" THEN GOSUB 5400:
GOTO 5210
5240 PRINT:PRINT"INVALID COMMAND":G
O TO 5210
5300 PRINT:IF LN<10 THEN PRINT" ";
5305 PRINTLN": ";
5310 FOR Q=1 TO
22:PRINTHEX$(MZ(LN,Q));:NEXT Q:PRINT: P
RINT
5320 IF LN<10 THEN PRINT" ";
5322 PRINTLN": ";
5325 LINE INPUT Q$
5327 IF Q$="EXIT" THEN 5210
5328 IF Q$="" THEN 5345
5330 IF LEN(Q$) <> 22 THEN PRINT: PRINT
"INVALID LENGTH.":GOTO5300
5335 FOR Q=1TO22:IF VAL("&H"+MID$ (Q$
,Q,1))<0 OR VAL("&H"+MID$(Q$,Q,1))>12 T
HEN PRINT:PRINT"INVALID CODE.":GOTO
5300
5337 NEXT Q
5340 FOR Q=1 TO 22:MZ(LN ,Q)=VAL ("&H"
+MID$(Q$,Q,1)):NEXT Q
5345 LN=LN+1:IF LN>38 THEN 5210
5350 GOTO 5300
5390 RETURN
5400 PRINT
5405 S=0:F=0:FOR Q=0TO7:P(Q)=0:NEXTQ
5410 FOR X=1TO 38:FOR Y=1 TO 22
5420 C=MZ(X,Y)
5430 IFC=12THENIFS=1THENPRINT"EXTRA
STARTLINE:"X:PRINT:MZ(X,Y)=11:ELSES=1
5440 IF C=8 THEN IF F=1 THEN PRINT "EXT
RA FINISH LINE:"X:PRINT:MZ(X,Y)=11:ELSE
F=1
5450 IF C<8 THEN IF P(C)=1 THEN PRINT"
EXTRA CHECKPOINT #"+HEX$(C+1)+"LINE:
"X:PRINT:MZ(X,Y)=11:ELSE P(C)=1
5460 NEXT Y,X
5470 RETURN
5900 POKE 65496,0:CLS:PRINT TAB(15) "P
rint Maze Code":PRINT
5910 PRINT"To (D)isk or (P)rinter or (E)xit"
5920 Q$=INKEY$:IF Q$="" THEN 5920
5925 IF Q$="D" THEN 5930 ELSE IF Q$="P"
THEN5940ELSE IF Q$="E" THEN 5990
5930 DV=1:EX$=".TXT":GOSUB 4200:IF ER
=1 THEN 5990
5935 GOTO 5950
5940 DV=-2
5950 IF DV=1 THEN OPEN "O",#1,Q$
5955 FOR X=1 TO 38
5957 IF X<10 THEN PRINT #DV," ";
5960 PRINT #DV,X;": ";
5962 FOR Y=1 TO 22
5964 PRINT #DV,HEX$(MZ(X,Y));
5966 NEXT Y
5968 PRINT #DV,""
5970 NEXT X
5980 PRINT #DV,"SCORES: ";
5982 FOR Q=0 TO 8:PRINT #DV,PT(Q);: NE
XT Q:PRINT#DV,""
5984 IF DV=1 THEN CLOSE#1
5990 POKE 65497,0:RETURN
6000 ATTR 3,2:GOSUB 3200:CLS
6010 LOCATE 10,0:PRINT"Mr. Maze Maze
Maker"
6020 LOCATE 8,2:PRINT"By Kenneth Reigh
ard, Jr."
6030 PRINT STRING$(40,"*");
6040 Q=1
6045 LOCATE 15,5:IF Q=1 THEN ATTR 0,4
ELSE ATTR 3,2
6050 PRINT"Edit Maze";
6055 LOCATE 15,6:IF Q=2 THEN ATTR 0,4
ELSE ATTR 3,2
6060 PRINT"Load Maze";
6065 LOCATE 15,7:IF Q=3 THEN ATTR 0,4
ELSE ATTR 3,2
6070 PRINT"Save Maze";
6075 LOCATE 15,8:IF Q=4 THEN ATTR 0,4
ELSE ATTR 3,2
6080 PRINT"Type in Maze code";
6085 LOCATE 15,9:IF Q=5 THEN ATTR 0,4
ELSE ATTR 3,2
6090 PRINT"Print Maze code";
6092 LOCATE 15,10:IF Q=6 THEN ATTR 0,4
ELSE ATTR 3,2
6093 PRINT"New Maze";
6094 LOCATE 15,11:IF Q=7 THEN ATTR 0,4
ELSE ATTR 3,2
6095 PRINT"END program";
6100 Q$=INKEY$:IF Q$="" THEN 6100
6110 IF Q$=CHR$(94) THEN Q=Q-1:IF Q<1
THEN Q=7
6120 IF Q$=CHR$(10) THEN Q=Q+1:IF Q>7
THEN Q=1
6122 IF Q$=CHR$(13) THEN 6125
6124 GOTO 6045
6125 IF Q=7 THEN 6150
6130 ON Q GOSUB 40,4100,4300,5200, 590
0,5100
6140 GOTO 6000
6150 ATTR 3,2:CLS:PRINT"Are you sure (Y/
N)?"
6155 Q$=INKEY$:IF Q$="" THEN 6155
6160 IF Q$="Y" THEN CLS:POKE65496,0:R
ETURN ELSE IF Q$="N" THEN 6000 ELSE
6155
```

The Hardware Hacker

Dr. Marty Goodman

Keep your Color Computer alive and well.

With the CoCo 3 no longer in production or readily available, those of us who continue to use this remarkable machine are in an especially tricky situation when our system goes down.

The first thing one needs to do both to preserve the function of ones' system AND to assist with repair of a bad system is to try to "back up" the hardware by having at least TWO of everything, especially the CoCo 3 itself. Classified ads on Delphi, in this magazine, user's group newsletters, CoNect, and at times garage sales, flea markets, and ham fests should give one the opportunity to purchase a spare CoCo 3 for a reasonable price ($30 to $70, depending on condition, memory size, and extras included). It is always a nice feeling to be able to pop in a working unit and continue along when one first confronts a hardware failure. Then the primary unit can be repaired at the user's convenience. If you belong to a local user's group, or have friends locally that use CoCos also, consider a group purchase of one or two "backup" computers or complete systems.

The single irreplaceable chip in the CoCo 3 is the "GIME" chip, that 68 pin square socketed chip that was custom made for Tandy to be *the majority* of the CoCo 3's circuitry. This chip is now quite hard to get, and indeed usually the ONLY way one can get it is to purchase a used CoCo 3. Ideally, one would like to have a 1987 model of GIME chip, though the older 1986 versions, while flawed, will often do the job acceptably.

This chip is very tricky to get out of its socket, unless one has a $20 dedicated extractor tool. In the absence of that, if one is *very* careful, one can tease the chip out using two jeweler's screwdrivers at diagonally opposite corners. One can even fabricate an extractor tool of a sort starting with an ordinary DIP IC extractor tool and grinding its teeth down so it can fit inside the PLCC socket at diagonally opposite corners of the chip. If using anything other than the specialized custom extractor tool, one must be *extremely careful* to NOT crack the GIME chip socket while prying out the chip, and also to NOT damage the delicate pins on either the chip or the socket. Also, when removing the GIME chip, be sure to note its orientation in the socket, for it is possible to insert it rotated 90 or 180 degrees. Since the GIME chip is the only socketed chip in a stock CoCo (outside of the memory chips) it's often not a bad idea to try to swap GIME chips between a good and a bad computer to test for problems.

Fortunately, it's quite rare for the CoCo 3 to die with JUST the GIME chip out. More commonly, it's the 6809 or the memory chips or both that gets zapped during the usual mishap such as wiggling a Multi-Pak or disk controller in the system port slot. Indeed, when I encounter a "dead" CoCo, after I check the power supply (by checking voltages), memory chips, and GIME chip (by testing them in another computer), usually the very next step I take is to remove the old 6809, put a socket there, and insert a new one.

By FAR the best way to replace a 6809, whether one is getting ready to install a 6309 upgrade or just trying to fix a dead computer is to *DESTRUCTIVELY* remove the old 6809 chip. This procedure has the minor disadvantage of destroying the old 6809 chip, but has the overwhelming advantage of allowing even a relatively unskilled tinkerer to cleanly remove the old chip *WITHOUT* damaging the circuit board, to make way for installation of a socket and then a new processor chip.

First, have on hand a replacement processor. Either a 68B09E or a Hitachi 6309E is ideal. You can often "get away" with a 68A09E or even a 6809E chip, especially if you are just testing to see whether installation of a new processor fixes the problem. The 6809E is rated to no higher than 1 MHz operation, and the 68A09E is rated to no higher than 1.5 MHz operation. Both should work fine on initial boot up of a CoCo 3, for the CoCo 3 normally boots up at 1 MHz (unless you've told it to do otherwise using an ADOS 3 ROM!) But for long term replacement, I urge you to use the 68B09E, which is still available at reasonable ($5 to $8) prices from major chip suppliers.

The tools you will need will include a good quality, fine, diagonal cutting pliers, a temperature controlled OR low wattage soldering iron (20 watts or less if not temperature regulated) with a relatively fine tip, a solder sucker (I prefer those of the sort that resemble the Radio Shack catalog number 64-2098, which is a vacuum operated device), and a long nosed pliers. You will also need to have handy a 40 pin socket for the 6809. *DO NOT under ANY CIRCUMSTANCES* use a "single wipe" style socket. Use either a double wipe tinned socket (the kind that has metal contacts that touch BOTH sides of each of the 6809's pins) OR use a gold plated, machine pin style socket (round holes and pins). Single wipe sockets offer very poor reliability. Double wipe tinned sockets offer excellent reliability, and I recommend them. Gold plated machine pin sockets offer exceptionally high reliability, but are really desirable only in situations where the device will be subject to a lot of vibration or shock, for they have the down side of being a bit more difficult to insert chips into.

First remove the CoCo 3 circuit board from its case, and remove the groundplane from the bottom of the board by meticulously poking out all of those zillions of little hold-down clips. Be sure to save all the clips! Cut *EACH* pin of the 6809 *as close as possible* to the chip itself. If you own a Dremel tool, you might consider using an ultra thin cutting wheel to do this, for the result will be superior. If you do take the Dremel route, though, be CERTAIN to wear protective eye wear, for those rotating wheels DO easily break, and can easily take out an unprotected eye. Also, be careful not to push the cutting wheel down too hard or too quickly, least you damage the circuit board after cutting through the IC pin.

When you have cut off every pin, remove the (now leg-less) body of the 6809 and throw it away. Using your soldering iron and long nose pliers, heat each pad that has a leg of the 6809 in it, and when the solder is liquid, gently pull out the severed leg. Do this for all 40 legs. Now use the soldering iron and solder sucker to suck out solder from the holes in the pads. You'll probably find two or more pads that are quite difficult to clean. These will likely be the ground and Vcc pads, which have thick (heat- conducting) traces leading to them. For these pads, you will find that if you melt a little EXTRA solder into the pad, making a tiny pool of solder on the pad, then heat that pool thoroughly and completely, a good solid SUCK from your solder sucker will clean out the pad. You may have to try this a few times to make it work.

CoCo 3's with power supply problems may have a bad pass transistor (that TO-220 shape thing with the big heat sink that gets so bloody hot) or, on occasion, a bad SALT chip. The power transistor can be replaced by *any* general purpose NPN power transistor. Indeed, in the past I've recommended to those with overheating problems that they replace the weenie little TO-220 case pass transistor Tandy supplied with a big, fat, hefty TO-3 case NPN power transistor, such as the 2N3055 (Radio Shack cat no. 276-2041). Of course, you must mount that transistor on a BIG, WELL-VENTILATED heat sink if you do this.

Bad SALT chips also account for failure of the bit banger (serial printer) port. The SALT chip (IC8 or SC77527) is another custom chip, now no longer available. However, unlike the GIME chip, SALT chips are easier to find because all models of CoCo 2 used the *same* SALT chip, and so are a valuable source for that spare part.

Problems with the joystick ports are often referable to IC 7, SC77526, another custom IC made just for the CoCo. This chip is also found in all CoCo 2's, and so spares can be obtained by picking up a few CoCo 2's for $3 to $8 at garage sales, flea markets, etc.

I've heard of lots of problems relating to blown RS-232 paks. These are simple devices and usually can be easily repaired. Most commonly, the level converter chips (1488 and 1489, or a MAX232 chip) are blown. Occasionally the 6551 28 pin UART chip is blown. The address decoder chips (typically a 74LS133 and a 74LS00 or 74LS02 or 74LS04) almost NEVER are harmed. On one occasion I found a RS-232 pak whose sole problem was a dead 1.8432 MHz crystal. This pak had probably been dropped, and the crystal must have broke with the impact. Replacing the crystal cured the problem.

Repair of an RS-232 pack is a lot easier if one spends a little time doing just a bit of diagnosis. Check the level converter chips using a logic probe on their inputs and a volt meter or RS-232 port tester on the corresponding outputs, to make sure outputs follow inputs. Check 1488 chips for presence of + and - 12 volts that they require.

A quick test involves hooking a RS-232 tester (those cheapo little widgets that have several LEDs on them that monitor the status of the most commonly used RS-232 port lines, such as Radio Shack cat no. 276-1401) and watching what happens when you boot up a term program and then try to type data into the keyboard with the term program operating. As a DECB term program boots and initializes itself, it should cause the DTR line to go from low to high. If the pak is working, you should see the outgoing data light of the RS-232 tester flash as you type in data. It's often best to set a slow baud rate... perhaps 300 baud... when doing this test, so you can more clearly see the flashing of the data light.

I've noted in the past that dead disk controllers can often be fixed by replacing the 7406 (or 7416) chip that is connected to the HALT and NMI lines with a new 7406 chip. The symptom of such a dead controller will usually be that the CoCo 3 will NOT BOOT, but instead gives a blank green screen, if turned on with such a blown controller plugged into it. Note that it's desirable to replace a 7416 with a 7406 chip. The two are pin for pin compatible. The only difference is that the 7406 is capable of sinking higher voltages than the 7416 chip.

I hope you never have to use any of the above information but, if you do, I hope it helps you keep your system up and running!

Dr. Marty Goodman is an anesthesiologist . His many hobbies include electronics in general, the Color Computer in particular. He can be reached in care of this magazine, via Delphi or Internet (martygoodman@ delphi.com), or US mail (1633 Bayo Vista Ave., San Pablo, CA 94806).

For even more repair details for the CoCo, get a copy of Tandy's Little Wonder from FARNA Systems. This 140 page book contains repair info, schematics, upgrade and modification info, etc., for all models of the Color computer. There has never been an information source for the CoCo as complete as this! 2/3 of the book covers technical information, 1/3 history and support. Special price of only $20 (+ $2.50 S&H) extended through 01 Jan 1994.

FARNA Systems
Box 321
Warner Robins, GA 31099

THE UPGRADE

National Diskletter!

Mid Iowa & Country CoCo Club (non-profit) "Gathering the CoCo Community"!

Do you want support tommorrow? Stay with a group that keeps you in touch!

Say "I saw it in 68' micros" and get an UPGRADE disk and free bonus disk!

ANNUAL MEMBERSHIP:
$16 US, $21 Canada

Terry Simons (MI&CC Treasurer)
1328 48th Street
Des Moines, Iowa 50311
Phone 515-279-2576
after 8:00 PM CST

The Industrial OS-9 User

F.G. Swygert

Can hobbyist and industrial users help each other?

Editors note: This will be a regular column. Industrial users are encouraged to write!

Software and Pricing

OS-9 was originally developed for industrial control. It would seem that with a large base of industrial users, there would be an equally large base of OS-9 software. It would seem that some of these industrial users would be purchasing software from "hobby" and "personal" type vendors. There are reasons that both *are not true.*

Most of the industrial software is proprietary. A company has a need for a particular application so they hire a programmer or development company to design the application for them. The software remains the property of the company that purchased it. In many cases, the resulting software is highly specialized. Some of it even contains proprietary information or trade secrets that the company does not wish to release to the general public. This is not always true. In some cases the company just doesn't want to get into the software business with the accompanying support and distribution problems.

Industrial users rarely buy software that was written for the personal/hobby market. The main reason for this will not make sense to the average hobbyist- it doesn't cost enough. Industry is under the assumption that you get what you pay for, which in some respects is true. But let's not blame this issue all on industry, the software vendor is also to blame.

Hobbyist write software with the assumption that it will be used on a small multitasking, single user system. Only the occasional home user will have a terminal or two connected to their main system. Industrial users usually expect a license agreement that allows up to a certain number of users. What hobby people need to understand is that industry practice is to charge a fee per user. What many companies do is allow up to a certain number of users with the original package (1-3) then charge an additional fee for additional users. Some do charge a flat rate for any number of users, typically four to five times the single user rate. Remember, a company that sells a product may have to support that many users, so the charge is justified. Any of the 10 or so users may call for support. You should also provide that many copies of documentation and allow making that many copies of the disk(s) the software comes on if not providing that number of individual copies. Few vendors opt to supply the docs and only one disk copy, most supply a copy of the docs and disks per user (exception: network software. In that case, there would be one copy of the server software).

It does not sound very good when an industrial user calls a vendor and gets a confused answer when he asks about site licensing and multi-user pricing. Bob van der Poel has some of the best OS-9 software around, but he has had trouble selling to industrial users. Why? His prices are to low!

Software Support

A related problem is support. Someone who programs as a hobby but has a "day job" may not be able to offer the support an industrial user expects or needs. It may be a good idea to get a full time vendor to market a good, in demand product as they can offer some phone support during normal business hours. If the product holds promise, a vendor will pick it up. A good vendor with an established customer base should sell more copies of the product than the programmer trying to sell it alone. Royalties vary with vendors, but usually run 20-35% of retail cost. Vendors have to provide support, advertise, stock your product, and usually handle copying disks and documentation as well.

G-Windows

The major full system vendors are currently pushing the G-Windows windowing system. This is a very versatile system that will operate on a number of 68000 based machines, not just one as does K-Windows (MM/1). **GESPAC, FHL, and Delmar** all support G-Windows. Many industrial applications are being written under G-Windows, and vendors are looking for G-Windows applications. The MM/1 is a very good computer, but there is currently no G-Windows port for it.

Ed Gresick has stated that if 20 MM/1 owners will pay $200 in advance for a copy, he will port G-Windows over. This is a very good offer, as G-Windows for other systems cost around $300. Mr. Gresick has also offered to port software over to G-Windows provided there is no major work on his part. Generally, a complete package running on a terminal system or another windowing system (such as K-Windows) will easily port over to G-Windows. I'm sure someone will find an exception though- which is why Ed says "provided there is no major work.." **Contact Ed Gresick at Delmar Co.**

68000 Based Computers

In order to port or write software for an OS-9/68000 machine, one must have a 68000 based computer. For hobbyists, this can be expensive! A full OS-9/68000 system can cost well over $1000 *without* a windowing system. What would you say if there were a way to get a development system for less?

There are two companies willing to deal with a developer. **Dirt Cheap Software (attn: Mark Griffith)** has a couple MM/1s they are willing to loan an individual provided they can show a program, either completed or well along in development, that (in Mark's opinion) shows promise.

Frank Hogg Laboratories has a deal for software developers. They can purchase a complete G-Windows system for as much as 66% off retail! Submit the idea for a program under development to FHL. If the developer is willing to allow FHL exclusive marketing rights, the 66% discount applies. If not, FHL will still allow a discount of 50%. Some other restrictions apply- contact FHL for details.

FHL offers the high performance KiX computers. These use Hazelwood 68020 (KiX\20- 25MHz) or 68030 (KiX\30- 16-33MHz) motherboards. These systems, like all other OS-9 computers, can be run from a terminal. The speed capability of these machines really shines, however, when FHL's EK-VAK video/audio/keyboard board is installed. This is one of the fastest video boards on the market. It is rather expensive though. All the FHL systems use the very fast and reliable Euro-K 32 expansion buss (KiX\30 has four 32 and four 16 bit slots, the KiX\20 has one 32 and three 16 bit slots).

Another major 68000 based computer supplier is **Delmar Systems**. They currently offer two different platforms: the System IV (based on a Peripheral Technologies PT68K4 motherboard- 68000/16MHz) and the System V (based on the Computer Design Services CD68X20 motherboard - 68020/25MHz). Both of these computer systems feature standard eight bit ISA expansion slots, meaning the expansion cards are much cheaper than comparable VME or Euro-K buss cards. This DOES NOT mean, however, that any PC type card can be used. OS-9 drivers must be available for each card. Only certain VGA cards, for example, will work with the available drivers.

For those interested in the **MM/1 and K-Windows**, there are at least two dealers for these systems. **BlackHawk Enterprises** is currently working on production problems, as they now have rights to produce the MM/1.

< 268'm >

David Graham, the owner of BlackHawk, has promised to make every effort to deliver systems in a more timely manner than IMS. Many people didn't buy MM/1s due to the long delivery time. If you are interested in this machine, do contact David.

There is also at least one licensed MM/1 dealer. This is **William L. Wittman** in New York. BlackHawk is also looking for other distributors, contact them if interested.

Roll Your Own Platform...

The demise of the Burke & Burke "Rocket"(68000 processor board for the CoCo with OSK) was a let down for many CoCo owners. But **Peripheral Technologies** has an excellent entry level solution: a refurbished, guaranteed PT68K2 motherboard. This is a 10MHz 68000 motherboard with 1MB of memory, four serial ports, a parallel port, floppy controller, clock, and keyboard connector. There are also seven eight bit ISA (PC/XT) expansion slots. PT says that almost any CGA or monochrome card (except CGA/mono combo cards) will work. This allows CoCo owners with RGBA/RGBI monitors (such as the Magnavox 8CM515 or 1CM135) to use them. The cost of these boards (traded in from upgrades) is only $149.00. REX (a single user operating system similar to FLEX) is included. Professional OS-9 can be purchased for $299. PT will also allow credit on these boards if one decides to upgrade to a more powerful PT68K4 (16MHz) later. Add a PC case, power supply, keyboard, and a couple disk drives, and a complete OS-9/68000 system can be assembled for under $1000; provided one has access to good used or surplus components.

Peripheral Technologies also sells bare board kits for those who *really* want to assemble their own computer (K4 only). Bear in mind that this is a four layer motherboard that requires a good deal of soldering skill to successfully complete! Complete used systems and components as well as uncompleted kits may be found on the Star-K BBS (914-241-3307), which also supports STAR*DOS, an alternate operating system for PT68K based systems.

What Now?

Users, send in what you want/need for your OSK systems. Do this for personal, hobby, and industrial needs. Those with software ideas may send in a description of what they currently have under development. "Under development" means just that, something that has been started, with at least enough progress made to know if the project is feasible or not. Lists of these wants/needs and projects will be printed here. Just maybe we can match up some vendors with developers, and possibly match up some programmers who are working on similar projects so they can exchange notes. Maybe one has reached a roadblock that the other has solved and a cooperative venture can be worked out. *If we all try to cooperate and work together, we CAN bring OS-9 more into the mainstream of general computing.*

Addresses:

BlackHawk Enterprises
P.O. Box 10552
Enid, OK 73706-0552
Voice: 405-234-2347

Delmar Co.
P.O. Box 78
Middletown, DE 19709
Voice: 302-378-2555
Fax: 302-378-2556

Dirt Cheap Computer Stuff Company
1368 Old Highway 50 East
Union, MO 63084
Voice: 314-583-1168

FARNA Systems
904 Second Avenue
Warner Robins, GA 31098-1029
Voice: 912-328-7859
Fax: 912-328-8870
*Please put "Attn: Frank Swygert" on any fax messages.

Frank Hogg Laboratory, Inc.
204 Windmere Road
Syracuse, NY 13205
Voice: 315-469-7364
Fax: 315-469-8537

Peripheral Technologies
1480 Terrel Mill Road #870
Marietta, GA 30067
Voice: 404-973-2156

Star-K Software Systems
P.O. Box 209
Mt. Kisko, NY 10549
Voice: 914-241-0827
BBS: 914-241-3307 (8-N-1, 300-2400)

William L. Wittman, Jr.
873 Johnson Road
Churchville, NY 14428
Voice: 716-494-1506
Fax: 293-1207

NEXT MONTH: G-Windows- *is it what you want or need?*

< 268m >

If you just want OS-9...

GET DOWN

TO THE UNDERGROUND!

The "International" OS9 Underground Magazine

$18.00/Year (12 Issues) U.S.
($23. Canada, $27. overseas)
4650 Cahuenga Blvd., Ste 7
Toluca Lake, CA 91602
(818) 761-4135
Write or call for Info

Beginning With OS-9

NOTE: Though based on CoCo OS-9, this column is good for OS-9/68000 beginners also!

Rick Ulland

More of Tandy's utilities, looking over a startup file.

Last month, we started with OS-9's shell, and began working our way through the utilities. We'll get that done, then pick up a few odds-n-ends in preparation for next months dive into MultiVue.

List copies the contents of a specified path (or paths) to stdout (standard output). It's up to you to decide if this is a good thing to do. For instance, there is a good chance an executable file will contain codes used to control the window system. Listing one of these files to the screen can do interesting things.

Incidentally, careful study of this command's manual page illustrates why the Tandy manual must be taken with a grain of salt. First, it says list lists the contents of a text file. On the same page, their own example shows this isn't exactly true, when the device /term is listed to printer!

Load looks into the file specified in the pathlist, and loads any OS-9 modules it finds there into memory. This isn't what the manual says, and the difference is important- a file may have more than one module in it, the types can be mixed within a file, and last the name of the file doesn't need to have anything to do with the name of any of the modules included. In other words, you CAN'T specify the module to load! (see merge)

The only time the filename and a module name have to match is when you DO NOT load it- typing a command that isn't in memory causes OS-9 to look for a disk file with that name, which should contain a module of the same name.

Think of **MDir** as your commands RAM-disk. The extended version adds much more info, but takes a bit of work to decipher. One easy to read bit of news is a modules link count, which tells you how firmly it's attached. Sometimes. If a block of utilities is loaded from one file, the entire block lives and dies together. Only the first module in the block gets a count.

If it's not copy and it's not list, it must be **merge**! It's not as picky as copy and it doesn't add characters like list, so sometimes it's used to replace one or the other. It's biggest claim to CoCo fame is as a way to work around the GIME's 8K memory block size by combining smaller modules into files which more neatly fill them. For instance, almost all of the floppy disk utilities fit in a 16K file vs 96K loaded separately.

Mfree refers to total RAM left. It cannot comment on how much of each process 64K remains! Even if the process is OS-9 itself- which explains those sysram full errors with 200K free. You might be lucky enough to face this bug someday- mfree (even most patched ones) freaks out when it finds a 512K block open. It's been fixed, or just fire up a few extra things at boot to fragment RAM some.

Modpatch is the most basic patch utility one can imagine. Bang on the RAM in real time!

Montype is OK as far as it goes. Replacements include setype, which adds mouse port/res set, and control, which does everything. If you were wondering, setype comes from Color Computer Artist, and control from MultiVue. Both can be used separately, and their changes can be cobblered home.

Os9gen we've done to death. There are two faster/better alternatives, from B&B and GaleForce, but this one's $25 cheaper. That's half a word processor.

Procs fatal omission is the device the procedure is connected to. Frankly, if I'm looking at procs I'm thinking of killing somebody. It's hard to tell which of the 7 shells shown is the one to die without knowing the window it owns.

Pwd/Pxd I dumped. Sometimes wish I hadn't, but not often.

Rename breaks the command line mold of utility pathlist pathlist... followed by most. The second entry is just the name you now want. Any path info, it robs from the first file's path.

Setpr is your main real control over the OS-9 virus that has infected your CoCo. By selecting how much cpu time each process gets, you can influence your silicon.

Editor's note: No, there isn't an OS-9 virus!!! Rick is just referring to OS-9 itself here, so don't get alarmed!!!

Tmode and **xmode** are closely related. Since OS-9 could really care less about the machine, it counts on device descriptors for specifics. These utilities allow negotiating them. Think of the "t" as meaning temporary- tmode works on open paths, and it's effect goes away once they are closed. Xmode changes the master copy (still in RAM, but more permanent that tmode)

Any software timed port like the bit-banger should make allowances for cpu/program speed. By using **tuneport** you can, by inserting 0-255 delay counts in there.

Wcreate is an attempt to make the window system a little more friendly. The catch-all display codes can do this faster, but use hex input. So what? Memorize 80=50 and 24=18, then use display 1b 20........

That's basically the supplied Tandy utilities, less a few that didn't inspire comment. There are gaping holes, like save (a module to a disk file), and some third party utilities work better than the stockers- but these are owned by all. In the future, I will attempt to reduce any procedure to this level.

There are a few more 'official' Microware commands and utilities available as part of the "Development System". Of course, the big ticket items in this package are the assembler (RMA) and debug, but there are a few other goodies in there. The Development System was rather expensive (last sold by Tandy for $99.95 and is hard to find today. Luckily, the assembler and debug from Level I work fine with Level II. A few of the Development System utilities are:

Binex (and it's cousin **exbin**) convert binary data into text, or back. These text files (Motorola S Record files) can be sent through text-only telecom links. Also handy if you don't own a CoCo-specific EPROM programmer. Lots of other rs232 linked toys (logic analyzers and the like) understand them also.

Dump is the safe way to 'list' a binary file.

Login provides the security needed when allowing a second user to log on to your CoCo, by accepting a user name and password, then setting the new users priority, startup directories, and suchlike. Obviously there is more that needs to be done to make your system truly secure, like a few months with attr pulling pubic execute/read permissions, but it's a start.

Make is used to keep track of the rats nest of files that a large modular program becomes. If you are inclined to write large c, assembly, or pascal programs, you'll love this.

Save is hidden here. It's important enough Tandy had to hide a copy of it on the MultiVue disk for their install program to work. Just another reason to get MV even if you don't plan on using it.

Sleep gives you a legal way to do a timing loop or just let another process get some cpu time. Options range from give up this one timeslice to sleep until we wake you up.

Tee can be used to split a pipe, so one isn't limited to piping here to there, but can spread things around.

Touch simply changes the timestamp on a file. Not an essential, but handy.

Tsmon sits on an idle serial port, waiting for someone to try logging in to CoCo. After they leave, it resumes sitting and waiting. Without tsmon, connecting terminals to your CoCo would be much more difficult.

Verify checks a modules header and crc. It can also update a header after a module has been modified.

Virtual Disk Driver- This RAMdisk doesn't need formatting, but it has been superseded by more capable PD programs.

And that's about it. Now, on to some trickery using these utilities. One thing that OS-9 allows you to do with the command line is collect a group of them into a procedure file. Sometimes called a shell script or batch file, these files allow one to automate some fairly complex tasks. The usual way to invoke (run) one of these is to type the filename as a command-shell will look in memory and the cmds directory for a program with this name. Failing to find one, it will then assume you must mean a procedure file, and jump to the current data dir looking for one of those.

When invoking a procedure file, add the ampersand for concurrent execution. In addition, the commands inside may also have their own ampersands... think about it- your shell, the procedure file's shell, and a few dozen commands (with their shells) all churning away at once!

Which brings up a point I missed last month- *don't do a lot of disk tasks concurrently*. The CoCo stops. What happens is the drive steps to some track, writes a sector, then steps halfway across the disk to write another sector in a different file, then jumps back to where the first file is. When all this is finished, the CoCo will start again. Of course, if you have a no-halt controller (such as a Disto SCII), this problem isn't as prevalent.

There are a few nifty options that can be added when invoking a file this way (Typing procedurefilename option). Normally, any error causes the entire file to bomb, but a -x results in the bad line being skipped over. A -p will suppress the normal shell screen display for files you really don't need to watch. And last, a t will display each line in the procedure file before executing it- makes a handy debugging tool.

The most familiar procedure file is startup. Startup is ran at boot with the default options x p -t (the opposite of above). Probably the easiest way to cover the possibilities is to work our way through a fairly complex one. Here is mine:

```
  control -e
  getclk4 5&
  dmode /d1 stp=3
  merge /h0/sys/stdfonts /h0/sys/stdpats_16
/h0/sys/stdptrs >/w
  load sutils dboot xmode emacs conect
  mega2
  iniz /w7
  display 1b 32 03 >/w7
  shell i=/w7&
  tuneport /p -s=99
```

The very first line is a little trick I use to set up my system. By calling MultiVue's control utility with that -e, all 16 palettes, as well as mouse resolution, key repeat speed and delay are adjusted at startup time to match the env.sys file. Changing any value is as easy as firing up MVue and clicking about (another reason to get it!).

Next, setime has been replaced with a utility to read my smartwatch. Although the real time is still kept by the software clock in this system, it's reset every five minutes so we are never to far off.

Now, we get to the subject of laziness. Although the proper thing to do is cobbler changes like the dmode line into boot, it's so easy to just leave it to startup! In fact, there used to be a few 'modpatch diskfix.mpc' kind of lines in my startup... at least a few changes were eventually installed properly.

The required loading of the graphics system has been reduced to one line, simply because I like my startup to fit on one screen. Merge certainly doesn't mind, and it makes editing startup easier.

The next section is all RAM management. The first 2 files are actually several dozen utilities, taking up only 3 blocks. Xmode is a custom version of this utility, which sometimes has to be replaced with the stock form- so it's loaded as a single file in it's own block, the better to ditch later. Then comes the no patience bug. I hate to wait. So, a small text editor (emacs) and a LARGE chunk of Basic09 i-code are loaded. True, the boot process takes a while. But afterwards, there isn't near as much disk reading going on- perhaps 1/2 the stuff I do is already in RAM. What doesn't show in the file itself is the fact *the boot disk was filled in the same order*. If these files were spread all across the disk, things would be much slower.

Only after loading all this stuff is the 2Meg upgrade enabled (via mega2). No real reason, but it cleans up the mfree display a little bit here.

Then, it's off to bang a text window. Since /w7 is shipped as a 80 col. text window it's an easy choice. The foreground color is changed with display so it will be visibly different than /term. Then, we feed it an immortal shell (shelli=/w7&).

Lastly, I have to fix the bit-banger port for the 6309's extra speed. So we open and close with very machine specific commands in this file. Keep this in mind anytime you look at any procedure file, either in a book or on a buddy's system - they assume a lot in terms of both hardware and software, and so aren't very portable.

After booting and looking for and running startup (if present), OS-9 looks for one last file. This is an executable file named autoex, in /dd/cmds. Jumping ahead of myself a bit, the autoex you get with MultiVue is simply a copy of the file 'multistart' with a different name. You can rename or copy any executable file, for example procs, to autoex.

Next issue- on to MultiVue!

Rick Ulland can be reached in care of this magazine, via electronic mail (Delphi: rickuland; Internet: rickuland@delphi.com), or U.S. mail at 449 South 90th, West Allis, WI 53214. Feel free to send questions or comments, but include an SASE if expecting a personal reply. Items sent to the magazine will be considered for publication.

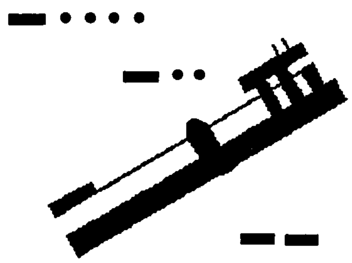
< 268'm >

Beginner's Showcase

MORSE listing, continued from page 21.

```
410 IF I$=B$ THEN PRINT "."; ELSE IF I$=C$
THEN PRINT "-";
420 GOTO 370
430 PRINT #-2, TAB(8), "THE MORSE CODE"
440 PRINT #-2
450 PRINT #-2, " To help you remember and
recognize the morsecode, press A on menu and
then  while not looking at keys or screen"
460 PRINT #-2, "press random keysand write
down the letter or number that you heard in
MorseCode, then check with screen."
470 PRINT #-2
480 PRINT #-2, "A .-,    B -...    C -.-.    D -.."
490 PRINT #-2
500 PRINT #-2, "E .    F ..-.    G --.    H ...."
510 PRINT #-2
520 PRINT #-2, "I ..    J .---    K -.-    L .-.."
530 PRINT #-2
540 PRINT #-2, "M --    N -.    O ---    P .--."
550 PRINT #-2
560 PRINT #-2, "Q --.-    R .-.    S ...    T -"
570 PRINT #-2
580 PRINT #-2, "U ..-    V ...-    W .--    X -..-
"
590 PRINT #-2
600 PRINT #-2, "Y -.--    Z --.."
610 PRINT #-2
620 PRINT #-2
630 PRINT #-2, "1 .----    6 -...."
640 PRINT #-2, "2 ..---    7 --..."
650 PRINT #-2, "3 ...--    8 ---.."
660 PRINT #-2, "4 ....-    9 ----."
670 PRINT #-2, "5 .....    0 -----"
```

< 268'm >



OS-9/OSK Answers!

Joel Mathew Hegberg

"Intelligent Entry" subroutines in BASIC and C.

Lately, there has been a lot of discussion on Delphi regarding simple yet helpful subroutines. During the discussions, Ted Jaeger mentioned he had written an "intelligent entry" subroutine for OS-9 BASIC. Essentially, you can specify what characters a user is allowed to enter, and any special characters, that may be required, such as the hyphen in a nine-digit zip code (30125-5114). This type of a routine is extremely useful for data entry applications, as it helps ensure valid data is being typed in.

I decided to take his OS-9 BASIC source and port it over to C to allow wider use of the routine. Listing 1, "entry.bas", is the commented OS-9 BASIC source for the Intelligent Entry subroutine. Listing 2 is the commented source for "testentry.bas", which is a simple program to demonstrate how to use the entry routines.

For C programmers, due to limited space, I've not included Ted's comments within the C source, but you can simply look to the OS-9 BASIC source if you have questions about the code. Listing 3, "entry.c", is the C source which can easily be compiled down to a relocateable '.r' file. If you're using the Microware C Compiler, you simply type "cc entry.c -r=<dir>" where <dir> is the directory you want the '.r' file to be placed in. After you have the relocatable file, any programs you compile in the future that use the entry routine simply have to be "linked" to 'entry.r' by the compiler. This also is fairly easy. For instance, to compile listing 4, "testentry.c", using the Microware C Compiler, you may use "cc testentry.c -l=entry.r". (The previous assumes that entry.r is in the current data directory. If not, the entire pathlist to entry.r would need to be specified.)

As always, anyone else with helpful OS-9 subroutines, please feel free to send them in!

LISTING #1: "entry.bas"

```
PROCEDURE entry
(* Nov 5, 1993
(* Intelligent entry routine
(* allows predefined characters in a string
(* and restriction of user input to
(* programmer specified characters
(* Cursor selection and input
(* positioning on screen also included
(* Inspired by a similar GWBASIC
program
(* written by Paul Lepato and published
(* in November PCM
(* return string to calling routine
(* adjust string length as needed
PARAM result:STRING[5]
(* initial value of input string
(* string length should match length of
"result" string
PARAM fields:STRING[5]
(* string of characters identified as
acceptable
(* to this routine
(* again, adjust string length as needed
PARAM allow:STRING[15]
(* asc specification of cursor character
PARAM cursor:INTEGER
(* location on screen where input will occur
PARAM row,col:INTEGER

DIM char:STRING[1]
DIM offset:INTEGER
DIM errnum:INTEGER
DIM ok:BOOLEAN
DIM k:INTEGER

(* check for legitimate values of row and
col
IF row>24 OR col>76 THEN
PRINT CHR$(7);
END
ENDIF

(* add BACKSPACE to allowable
characters
allow=allow+CHR$(8)

offset=1
ok=FALSE

ON ERROR GOTO 100

SHELL "tmode noecho"
(* cursor off
PRINT CHR$($05); CHR$($20)
PRINT CHR$(2); CHR$(31+col);
CHR$(31+row); fields
PRINT CHR$(2); CHR$(31+col);
CHR$(31+row); CHR$(cursor);

WHILE offset<=LEN(fields) DO
IF MID$(fields,offset,1)=" " THEN
GET #0,char
(* is it allowable character?
GOSUB 10

IF ok=TRUE THEN
(* go process the letter
GOSUB 20
ELSE
(* beep at user
PRINT CHR$(7);
ENDIF

ELSE
(* skip predefined characters
result=result+MID$(fields,offset,1)
PRINT CHR$(2); CHR$(31+col+offset);
CHR$(31+row); CHR$(cursor);
offset=offset+1
ENDIF

ok=FALSE
ENDWHILE
END

10 (* see if allowable character was
entered
k=1
WHILE k<=LEN(allow) DO
IF MID$(allow,k,1)=char THEN
ok=TRUE
ENDIF
k=k+1
ENDWHILE
RETURN

20 (* determine letter and take action
IF char<>CHR$(8) THEN
PRINT CHR$(2); CHR$(31+col+offset-1);
CHR$(31+row); char;
result=result+char
offset=offset+1
ELSE
(* erase cursor
PRINT CHR$(2); CHR$(31+col+offset-1);
CHR$(31+row); " ";
(* find fields field
WHILE offset>1 DO
offset=offset-1
EXITIF MID$(fields,offset,1)=" " THEN

ENDEXIT
ENDWHILE
PRINT CHR$(2); CHR$(31+col+offset);
CHR$(31+row); MID$(fields,offset+1,1);
result=LEFT$(result,offset-1)
ENDIF

(* display cursor
IF MID$(fields,offset,1)=" " THEN
PRINT CHR$(2); CHR$(31+col+offset-1);
CHR$(31+row); CHR$(cursor);
ENDIF
RETURN

100 errnum=ERR
(* user struck [ESC] key
IF errnum=211 THEN
END
ENDIF
```

LISTING #2: "testentry.bas"

```
PROCEDURE testentry
DIM result:STRING[5]
DIM allow:STRING[15]
DIM fields:STRING[5]
DIM cursor:INTEGER
DIM row,col:INTEGER

DIM sub:STRING[5]
sub="entry"

(* allow user to enter only numbers
allow="1234567890"
(* define original fields string
(* with predefined colon
fields=" : "
(* pick a cursor
(* examples from MM1 ascii codes
(* 63= question mark; 254=small block;
95=underline
(* 219=large block; dollar sign=36; number
sign=35
cursor=219
(* define position on screen
col=50
row=10
(* initialize return string
result=""
RUN sub(result,fields,allow,cursor,row,col)
PRINT
PRINT result; " returned"
SHELL "tmode echo"
(* turn cursor back on
PRINT CHR$($05); CHR$($21);
```

LISTING #3: "entry.c"

```
#include <stdio.h>

#define FALSE 0
#define TRUE  1
#define STDIN 0

int entry(result,fields,allow,cursor,row,col)
char *result,*fields,*allow;
int cursor,row,col;
{
   char chr[2];
  int offset,errnum,ok;

  if (row>24 || col>76)
   {
     putchar(7);

     return(0);
   }

   offset=1;
  ok=FALSE;
  *result=0x00;
  chr[1]=0x00;
  system("tmode noecho\n");
  printf("\x05\x20");
  printf("\x02%c%c%s",31+col,31+row,
fields);
  printf("\x02%c%c%c",31+col,31+row,
cursor);
  fflush(stdout);
  while(offset<=strlen(fields))
   {
    if (*(fields+offset-1)==0x20)
     {
       if (read(STDIN,chr,1)!=1)
return(FALSE);
        ok=check_valid_char(chr[0],allow);
        if (ok)
process_char(chr,fields,result,&offset,cursor,
row,col);
        else putchar(7);
     }
    else
     {
       chr[0]=*(fields+offset-1);
       strcat(result,chr);
   printf("\x02%c%c%c",31+col+offset,31+
row,cursor);
        offset++;
     }
    fflush(stdout);
    ok=FALSE;
  }
  return(TRUE);
}

int check_valid_char(chr,allow)
char chr,*allow;
{
  int k;

  if (chr==0x08) return(TRUE);
  k=1;
 for(k=1;k<=strlen(allow);k++)
  {
    if (*(allow+k-1)==chr) return(TRUE);
  }
  return(FALSE);
}

process_char(chr,fields,result,offset,cursor,
row,col)
char *chr,*fields,*result;
int *offset,cursor,row,col;
{
  if (*chr!=0x08)
   {
  printf("\x02%c%c%c",30+col+*offset,31+
row,*chr);
     strcat(result,chr);
     *offset+=1;
   }
  else
   {
  printf("\x02%c%c%c",30+col+*offset,31+
row,0x20);
     while(*offset>1)
      {
        *offset-=1;
       if (*(fields+*offset-1)==0x20)
break;
      }
  printf("\x02%c%c%c",31+col+*offset,31+row,*
(fields+*offset));
     *(result+strlen(result)-1)=0x00;
   }

  if (*(fields+*offset-1)==0x20)
   printf("\x02%c%c%c",30+col+*offset,31+
row,cursor);

   return;
}
```

LISTING #4: "testentry.c"

```
#include <stdio.h>

main()
{
  char result[6],fields[6],allow[15];
  int cursor,row,col,ok;

  strcpy(allow,"1234567890");
   strcpy(fields," : ");
   cursor=219;
   col=50; row=10;
   result[0]=0x00;
 ok=entry(result,fields,allow,cursor,row,
col);
   if (ok) printf("\n%s returned.\n",result);
   else printf("\nError occured while
reading keyboard.\n");
    system("tmode echo\n");
   printf("\x05\x21");
}
```

< 268'm >

Programming in "C"

P.J. Ponzo

Functions and How to Compile a Program.

FUNCTIONS()

```
main() {
float x,y,a;
printf("\n Enter two numbers (separated by a space) : ");
scanf("%f%f",x,y);
a=average(x,y);
printf("\n The average of %f and %f is %f",x,y,a);
}
```

When we begin our C program with main() we are defining a function. Execution of our C program will begin at the statements which make up this function (and it is this property which makes the name main special).

```
main() {
float x,y,a;
printf("\n Enter two numbers (separated by a space) : ");
```

Here we define three float variables called x, y and a and printf instructions asking for x and y. Note that the user must enter a space to separate the two numbers (pressing the tab key between numbers will also 'separate' them). printf() is a function, just like main(). When we invoke this function we pass to it a format string (between quotes) and (sometimes) a list of variables to be printed. In this example there is only the format string (which prints a newline first):

"\n Enter two numbers (separated by a space) : "

```
float x,y,a;
printf("\n Enter two numbers (separated by a space) : ");
scanf("%f%f",x,y);
```

Now we use the function scanf() to input the two numbers x and y. Note that scanf() also requires a format string. Can you pick out the ERROR in this statement? The function scanf() requires that we pass to it the addresses of x and y by using &x and &y...REMEMBER?

```
scanf("%f%f",x,y);
should be:  scanf("%f%f",&x,&y);
REMEMBER!!
```

```
scanf("%f%f",&x,&y);
a=average(x,y);
```

Here we see another function called average(). We pass to this function the values of two variables, x and y.

The C language recognizes a function by the fact that it is given a name (like main, scanf, printf or average) followed immediately by parentheses (....). Within the (and the) is information which is passed to the function. Unlike scanf() and printf() (which are included in the C library of functions and are available for our use), the function which we are calling average() is one which we must write ourself!

```
main() {
float x,y,a;
printf("\n Enter two numbers (separated by a space) : ");
scanf("%f%f",&x,&y); (note that we've
                         changed to &x,&y)
a=average(x,y);
printf("\n The average of %f and %f is %f",x,y,a);
}
```

The functions scanf() and printf() perform their task and return nothing, but our function average() is expected to return the average of the two variable values we passed to it.

In our program above we assign this 'returned average' to the float variable we are calling a... and pass to printf() a format string "\n The average of %f and %f is %f" indicating that we want certain text printed (after a newline) as well as 3 %float numbers. The variable list which we pass to printf() (namely x,y,a) tells printf() which 3 %float values are to replace the 3 %f which occur in the 'format' information.

Writing the Function Average()

Like the function main(), we begin with the name and an opening {. But unlike main(), the function average is to receive two variables....so, before our opening {, we write:

```
average(x,y)
float x,y;
{            the opening { occurs after
the declaration float x,y !
```

THE FIRST STATEMENT IN A FUNCTION (even before the {) MUST BE A DECLARATION OF THE ARGUMENT TYPES!

Let's continue writing our average() function:

```
average(x,y)
float x,y;
{
float z;
z=(x+y)/2;
return(z);
}
```

The float z; (within the body of our function) declares the variable z to be a float.

```
float z;
z=(x+y)/2;      z=(x+y)/2 assigns to z
                the average of x and y
return(z);      return(z); will return the
                value of z (so it can be used
                in our main() program
```

The whole thing, including main(), is now:

```
main() {
float x,y,a;
printf("\n Enter two numbers (separated by a space) : ");
scanf("%f%f",&x,&y);
a=average(x,y);
printf("\n The average of %f and %f is %f",x,y,a);
}
average(x,y)
float x,y;
{
float z;
z=(x+y)/2;
return(z);
}
```

Actually, this program won't (quite) work...but let's see how to get it to compile and run.

How to COMPILE

We save our program on disk and leave our word processor, giving our program a name: program1.c (note the necessary .c). Then we would ask the C-compiler to compile it, with the command:

```
cc program1
```

With this command the compiler looks for a file on the disk with the name program1.c (the extension .c being UNDERSTOOD!) and generates a file called program1.o (an object file).

After the compiler has done its thing it's our turn again. We ask to link the object file by issuing the command:

link program1

Again, the extension (.o in this case) is understood. The linker works on the program1.o file and generates an executable program called: program1.exe. Finally, we may issue the command: program1 to run our program.

NOTE: The commands necessary to compile and link, and the eventual name of the executable program, may vary from one C-compiler to another. On the IBM PC, the (final, compiled) program will normally have the extension .exe.

The reason for the 2-step process of compile then link is that we may write (for example) the average() function separately, and compile it separately (generating an object file, say average.o then link the main() function to the average function by issuing the command: link program1 average (where we have called the main() function program1.c when we saved it to disk before leaving the word processor text editor).

Now suppose we have (successfully) compiled and linked program1. (our program will actually compile...without any error messages!). We have on disk program1.exe which we execute by issuing the command: program1 and the program will print:

```
Enter two numbers (separated by a space) : _
```

We type: 21 22 (leaving a space between) then press the Enter (or Return) key and expect to get:

```
The average of 21.000000 and 22.000000 is 21.500000
```

Our program statement was:

```
printf("\n The average of %f and %f is %f",x,y,a);      the %f gives 6 decimal
                                                          places...by default
```

Alas, what we get is:

```
The average of 21.000000 and 22.000000 is 21.000000
```

Let's look at the function average() again:

```
average(x,y)
float x,y;
{
float z;
z=(x+y)/2;
return(z);
}
```

REMEMBER THIS: ALL FUNCTIONS WILL RETURN AN INTEGER UNLESS YOU SAY OTHERWISE!

So, the value of z which average() returned was changed from 21.5 (the floating point average of the two floating point numbers 21 and 22) to 21. The fractional part was truncated since (unless we say otherwise) our function returns an integer! And how do we tell the C-compiler that average() is to return a floating point number? We write the average() function with a type declaration built into its name!

```
float average(x,y)          Note the float!
float x,y;
{
float z;
z=(x+y)/2;
return(z);   Now return(z) gives a float!
}
```

FUNCTIONS HAVE A PRIVATE COPY OF THEIR ARGUMENTS.

```
main() {
float x,y,a;
printf("\n Enter two numbers (separated by a space) : ");
scanf("%f%f",&x,&y);
a=average(x,y);
printf("\n The average of %f and %f is %f",x,y,a);
}
average(x,y);
float x,y;
{
float z;
z=(x+y)/2;
return(z);
}
```

Although we called the two arguments of average() x and y (just as main() did), this is not necessary! The function average() only gets a copy of the variables which appear in its argument list and may give these copies any name it likes (they are NOT the original x and y which main() uses!). The above program might be changed to read:

```
MY copy
main() {
float x,y,a;
printf("\n Enter two numbers (separated by a space) : ");
scanf("%f%f",&x,&y);
a=average(x,y);
printf("\n The average of %f and %f is %f",x,y,a);
}
float average(sam,sally)
float sam, sally;
{
float george;
george=(sam+sally)/2;
return(george);
}
```

Since copies of char and int and float variables are passed to a function, the function may manipulate these variables as it sees fit. The original values remain unchanged. This mechanism for calling a function and passing variables is called : call by value.

The exception occurs when we pass a string variable to a function. In this case, since a string may be arbitrarily long (!), it seems inefficient to provide a copy of the string. So C will pass the address in memory where the string begins. This is called : call by reference.

Call by value is a mixed blessing. We cannot (for example) call upon an exchange(x,y) function to exchange the values of the integer variables x and y

```
exchange(x,y)
int x, y;        The values of x and y are
{                only exchanged within this
int temp;        function, NOT in main() !
temp=x; x=y; y=temp; return;
}
```

But don't despair, there are ways around this!

P.J.Ponzo
Dept. of Applied Math
Univ. of Waterloo
Ontario N2L 3G1 *<268'm>*

CoCo-C

A complete Disk BASIC based C development package for the Color Computer NOT requiring the OS-9 Operating System.
Contains:

- **Text Editor -**
- **C Compiler**
- **Assembler**
- **Library/Linker -**
- **Command Coordinator**

CoCo 1,2, & 3 versions included.

$59.95

+ $4 S&H (check, money order, or COD)

Infinitum Technology
P.O. Box 356
Saddle River, NJ 07458
Phone 914-356-7688

Repackaging the Color Computer

F.G.Swygert

The final installment! Putting the CoCo in a "PC" type case.

By far the easiest case to be found is a "standard" PC type case. There are several configurations, with the most popular being:

PC/XT or Baby AT: This case was made to hold a standard PC/XT size (12"x8.5") motherboard. They usually have four 5.25" half height drive bays, older ones with four open to the front and newer ones with two open and two concealed (for hard drives). Really old cases will only be drilled for two full height drives. Four half high units can be mounted by drilling the mounting brackets or adding side plates. A "Baby AT" motherboard is the same size as a PC/XT board. These cases typically measure 19"x16"x5" (WxDxH).

Full Size AT: When the AT first appeared it had a larger 14"x9" (approx.) motherboard. The case was also made to house three open half height devices plus two internal devices. The footprint (desk area taken) of this case is usually the same as the PC/XT, but it is about 8" tall, leaving enough room to mount an MPI if really needed.

Slimline: These cases are the smallest, having just enough room to mount two half height drives open and one internal drive- usually just a 3.5" hard drive. There is room to mount a CoCo in there with a Y-cable though. Typical measurements are 17"x16"x4" (WxDxH).

Tower: The tower case is really just a standard case mounted sideways. A large tower case will mount five to eight 5.25" half height drives. With a little work, an MPI and CoCo motherboard can be mounted. If you absolutely MUST have an MPI, then this is what you need. A full size tower is usually 17-19" tall, 16" deep, and 7" wide. A mini tower is usually only 13-14" tall... a CoCo just won't easily fit!

OEM Cases: Cases have been made in all shapes and sizes. Original Equipment Manufacturers (OEM) made cases specifically for their own motherboards and expansion needs (like the Tandy 1000 and 2000). You will have to evaluate your own needs and the size of the case. In general, if there is space for the number of desired drives and an open area the size of a CoCo motherboard, the case can be used.

You can typically count on plenty of room to mount a CoCo motherboard and peripherals in most cases, but NOT an MPI. Since a CoCo 3 motherboard is about 12"x7", a Y cable is necessary.

I got the chance to see two really good IBM cased CoCos at the Atlanta 'Fest this year. Alan Dekok and Colin McKay (both of Northern Exposure, from Canada) had their CoCo 3s in IBM style cases. Alan's was mounted in a genuine IBM PS/2 case. The only drawback to this is that it houses only 3.5" drives, making an external 5.25" drive necessary. Colin had his mounted in a "flip top" PC/XT case. This case is very easy to get into, but is also hard to find now, as the slide off top style is cheaper and easier to manufacture and not many people go into their systems very often.

I talked with both these gentlemen extensively about their setups. Alan had a Y cable about eight inches long but maintained that it did not give any problems even with three devices on it (floppy controller, hard drive controller, RS-232 Pak). Colin went about it the easy way- he had an FHL "Eliminator" mounted on a three inch Y cable.

There are two reasons there were no problems with the Y cable- shielding and a good connection to the CoCo motherboard. Both had removed the standard CoCo cartridge connector and installed a 40 pin double row male header in its place. A female crimp on connector was used on the Y cable and then 40 pin crimp on card edge connectors added. This is by far a more stable connection than the old card edge method. It is advisable to have an experienced person remove the 40 pin cartridge connector- you may want to keep it intact rather than destructively remove it. ANY TV/stereo repair shop can remove and replace the connector for you if they work on circuit boards. This usually costs under $25, or the stores minimum charge- it takes less than five minutes for an experienced person with a good vacuum desoldering station. Most shops will be much happier if you go ahead and remove the circuit board from the case. While you have the board there, go ahead and have them socket the 6809. It shouldn't cost you but a few dollars more, if any. Do call and make sure the shop carries the header before taking in your CoCo.

The other reason for a successful Y cable is proper shielding. This is very easy to accomplish when the CoCo is repackaged. In any case, the CoCo motherboard is grounded to the case itself through the power supply. Anything plugged into the Y cable should be grounded to the motherboard or the case also. This grounding will make the case itself a shield for the entire unit, eliminating many of the problems usually associated with Y cables. A cable length of 8-10 inches should not be a problem within a totally shielded environment.

The only place a Y cable must be kept as short as possible is between a CoCo and an MPI, even in a shielded case. The MPI and three to four devices adds quite a capacitance load on the cable. If using an MPI, keep the cable as short as possible- just enough to place the MPI within the case. Keep the length down to only three to four inches at the most. Two inches is enough to place the MPI 90 degrees or back to back with the motherboard.

The devices on a Y cable should be anchored inside the case in some way. Alan had his bolted to a bracket in the PS/2. He used threaded stand-offs between each card. The holes that normally mount the cards in their plastic cases were used. Long 1/8" bolts with plastic tubing spacers would be a good method also. Colin simply had a small metal bracket holding the Eliminator away from the motherboard. A very easy way to anchor devices is to just use picture hanging tape (double backed foam tape) between the device cases and to the computer case itself. If you do this, you might want to remove the case screws first. This will allow the cases to be snapped apart if removal becomes necessary.

Mounting an MPI takes a lot more work. The easiest way is to use a large toe . The MPI is tehn mounted on the bottom of the case and the motherboard on the side, 90 degrees to the MPI. Instead of using a Y cable, get a PCB mount straight 40 pin female connector and replace the CoCo's 90 degree connector. This allows the MPI to be plugged directly into the motherboard.

The MPI itself may need some surgery. Tower cases are usually only eight inches wide- to narrow for a full MPI. Well, since you're going to be using the PC power supply, cut the MPI power supply off! The majority of the circuit board after the fourth slot is only for the power supply. Examine the traces and find the three that go from pins 1, 2, and 9 of the fourth slot to the power supply portion of the board. These are the -12V, +12V, and +5V supply line, respectively. After making sure no other traces cross over, cut the power supply portion off with a hack saw or heavy shears. Attach power from the PC power supply to the lines mentioned above and you're ready to go. Don't forget to place a piece of heavy cardboard or some type standoff between the MPI and case bottom, also between the case and motherboard.

The other known method of mounting a CoCo and MPI requires a full size AT case. This is needed mainly for the height , as a long cartridge (older disk controller, RS-232 Pack, Speech/Sound Pack, etc.) and the Burke & Burke CoCoXT hard drive controller is about six inches long. The CoCo motherboard is placed near the back of the case with the ports facing the front and the cartridge port to the left. The MPI is placed in the front section of the case as far to the left and with the connector end as close to the motherboard cartridge connector as possible. A Y cable four to five inches long will be needed. If the internal hard drive bays won't be used, cut the frame out and place the MPI in that area instead. This will allow the CoCo to be placed with the connectors facing the back of the system while maintaining a short cable between the two components and eliminating any extension cables (cut a slot in the case for the CoCo connectors). Some of the newer small footprint PC/XT cases have open mountings for three 5.25" half height devices. These might be suitable for mounting a CoCo and MPI in provided they allow the motherboard to mount under the power supply. The connectors on the CoCo would have to face inside the computer and there may not be enough clearance between the case bottom and power supply to mount a 2MB upgrade.

There is one product currently on the market which was made specifically to make mounting a CoCo and peripherals in the same case very simple. This is the CoNect "XPander". The XPander was designed to flip two expansion devices over the CoCo motherboard inside the CoCo case. This is done with the aid of a new sheetmetal bottom half for the case that lowers the motherboard substantially. With this new bottom half installed, the CoCo is only a little taller overall than with the original case bottom. Not only are there two slots inside the CoCo, but there is a built in RS-232 port that works just like the Tandy RS-232 Pak and there is an external cartridge port in the original location . A fully decoded device can be mounted in the external port or the internal slots can be switched out and a game cartridge used. The entire XPander /CoCo unit only take up an area 12"x7"x3". This makes the XPander ideal for mounting in almost any PC/XT style case, even the slimline types. Odd OEM cases should have plenty room for the XPander also.

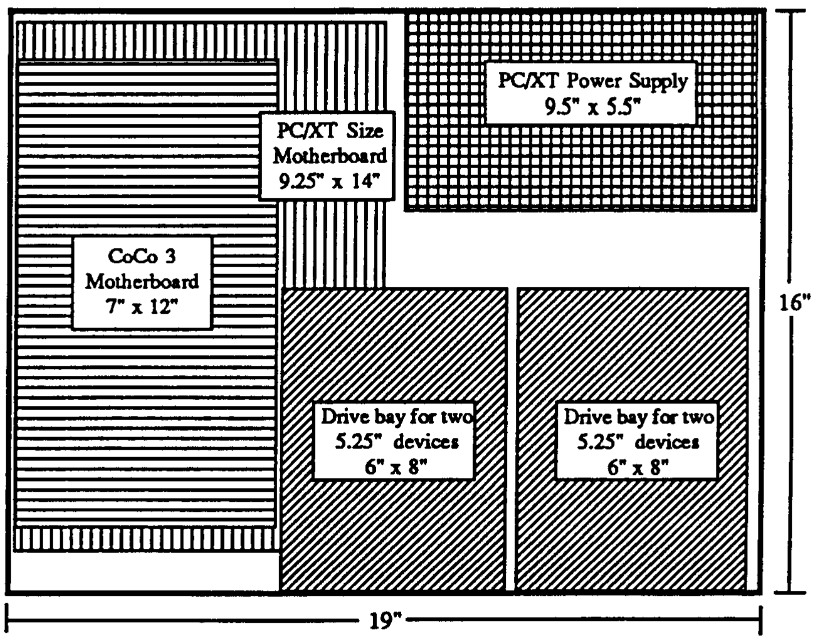
A new back plate will have to be made for most cases. One will also need to make extension cables for all the CoCo connectors and extend them to the back plate. Be creative on the back plate! Some people use existing openings for DB-9 and DB-25 connectors, others get the original style DIN connectors. All the slot dividers can be removed and a single large plate used, or just one or two of the dividers cut and a smaller plate used. I've even seen people make longer cables for devices and simply run them through one of the open PC expansion slots and plug directly into the CoCo. Then others get really creative and make switched slots on the back of their newly created systems. This all depends on the individual and their needs. If a hi-res pack and standard joystick are switched often, then consider mounting a switchable interface (maybe from HawkSoft or CoNect) inside and remotely locating the switch on the front or back panel. That may be a good use for the "turbo" or keyboard lock switch. There was one advantage to Alan's PS/2 case- it could be mounted with the CoCo connectors to the back, where IBM had a convenient slot already, eliminating the need for extension cables.

An external drive connector is easy to add to the existing drive cable. Use a DC37 (RS-449 37 pin) or Centronics printer (36 pin) crimp on connector. The drive cable is only 34 pins, so when making an external cable, remember which end you left the extra pins on! The extra pins can be used to carry power to the external drive (many IBM type external drives supply power within the connector). The connector can be attached to the backplate with the other connectors.

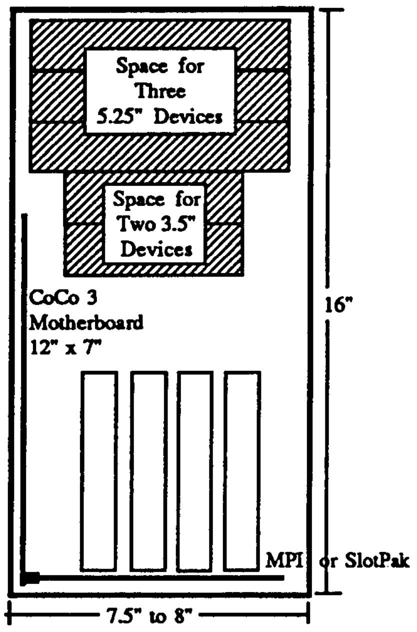
To recap once more from the previous two articles, power the motherboard by: 1) cutting the +5V regulator out. 2) Add +5V to the end of R19 closest to the +5V regulator. 3) Add -12V to the unbanded (anode) end of D4. 4) Add +12V to the banded (cathode) ends of D2 and D14.

Well, that brings this series to an end. I do hope that these four repackaging articles provided enough information to get you headed in the right direction. Hopefully, they also let you know that repackaging doesn't have to be an ordeal- it can be as complicated or as simple as the end user desires.

< 268'm >



Typical PC/XT Size Case. Floppy controller and other devices must be mounted on a Y cable in the left drive bay or above the CoCo motherboard. It may be possible to mount one drive in the left bay above the Y cable mounted devices. The PC/AT case has the same dimensions but is taller and the motherboard is a little wider and longer (10.5" x 14"). The standard AT power supply is nearly a cube (6.5" x 6" x 6"), leaving more room for the motherboard. The PC/XT case is only 5" tall inside. Some OEM and aftermarket manufacturers (other than IBM) use a PC/XT size power supply raised off the floor of the case, allowing the motherboard to extend slightly beneath it.



In a medium or large tower case (medium shown), mount the CoCo motherboard on the left side (looking from the front) with the connectors facing the rearif possible. It may be possible to cut put theback of the case for access to the original connectors, eliminating the expense of extending the connectors to the back plate.

Beginner's Showcase

Programmers of all skill levels will appear here. The emphasis is on short, easy to type in programs that illustrate programming techniques. Typing in examples is a great learning tool! If you have a short program or subroutine in any language, drop us a line! Any program/subroutine printed may be used by anyone within their programs, even commercial programs, as long as credit is given the author and magazine within the code (REM statements) and documentation.

Morse Code Trainer

This little program is rather self explanatory. It trains one on Morse Code! Dots and dashes can be printed on screen and heard, just printed on screen, or dumped to a printer. We can thank Gary Holder of Australia for this one! (reprinted from Jul/Aug 93 CoCo-Link) The printer baud rate must be set in line 45 if using other than the CoCo default of 600 baud.

```
30'MORSE CODE
40'BY GARY HOLDER, 1991
45 POKE 150,1
50 CLS: WIDTH 40
60 LOCATE 26,0:PRINT"MORSE
CODE"
70 LOCATE 26,2:PRINT STRING$
(10,"-")
80 LOCATE 1,6:PRINT"A - PRINT
AND HEAR":LOCATE 1,9: PRINT
"B - DOTS AND DASHES"
90 LOCATE 1,12:PRINT"C - THE
MORSE CODE TO PRINTER"
95 LOCATE 1,15:PRINT"Q - QUIT
THE PROGRAM"
100 I$=INKEY$:IF I$="" THEN 100
110 IF I$="A" THEN 120 ELSE IF I$
="B" THEN 320 ELSE IF I$="C" T
HEN 430 ELSE IF I$="Q" THEN 800
120 CLS:PRINT"PRESS LETTERS
AND NUMBERS":PRINT:PRINT"
PRESS - DASH FOR CT":PRINT: P
RINT" PRESS / FOR AR"
130 PRINT:PRINT" PRESS @ TO R
ETURN TO MENU"
140 PRINT STRING$(32,"-")
150 I$=INKEY$;IF I$="" THEN 150
155 REM *** LETTERS
160 IF I$="A" THEN PLAY "T2003
L3GP2L1G" ELSE IF I$="B" THEN
PLAY "T2003L1GP2L3GP2GP2G" E
LSE IS I$="C" THEN PLAY "T200
3L1GP2L3GP2L1GP2L3G" ELSE IF
I$="D" THEN PLAY "T2003L1GP2
L3GP2G"
170 IF I$="E" THEN PLAY "T2003
L3G" ELSE IF I$="F" THEN PLAY
"T2003L3GP2GL1P2GL3P2G" ELSE
IS I$="G" THEN PLAY "T2003L1G
P2GL3GP2G" ELSE IS I$="H" THE
N PLAY "T2003L3GP2GP2GP2G"
180 IF I$="I" THEN PLAY "T2003L
3GP2G" ELSE IF I$="J" THEN PLA
Y"T2003L3GP2L1GP2GP2G" ELSE
IF I$="K" THEN PLAY "T2003L1G
L3P2GL1P2G" ELSE IF I$="L" THE
N PLAY "T2003L3GL1P2GL3P2GP
2G"
190 IF I$="M" THEN PLAY "T2003
L1GP2G" ELSE IF I$="N" THEN P
LAY "T2003L1GL3P2G" ELSE IF I$
="O" THEN PLAY "T2003L1GP2G
P2G" ELSE IF I$="P" THEN PLAY
"T2003L3GL1P2GP2GL3P2G"
200 IF I$="Q" THEN PLAY "T200
3L1GP2GL3P2GL1P2G" ELSE IF I$
="R" THEN PLAY "T2003L3GP2L
1GP2L3G" ELSE IS I$="S" THEN P
LAY "T2003L3GP2GP2G" ELSE IF
I$="T" THEN PLAY "03T20L1G"
210 IF I$="U" THEN PLAY "T200
3L36P2L1G" ELSE IS I$="V" THE
N PLAY "T2003L3GP2GP2GL1P2G
" ELSE IF I$="W" THEN PLAY "T
2003L3GL1P2GP2G" ELSE IF I$="
X" THEN PLAY "03T20L1GL3P2G
P2GL1P2G"
220 IF I$="Y" THEN PLAY "T200
3L1GL3P2GL1P2GP2G" ELSE IF I$
="Z" THEN PLAY "T2003L1GP2G
L3P2GL1P2G"
240 REM *** NUMBERS
250 IF I$="1" THEN PLAY "T2003
L3GL1P2GP2GP2GP2G" ELSE IF I$
="2" THEN PLAY "T2003L3P2GL
1P2GP2GP2G" ELSE IS I$="3" TH
EN PLAY "T2003L3GP2GP2L1P2G
P2G" ELSE IS I$="4" THEN PLAY
"03T20L3GP2GP2GP2GP1P2G"
260 IF I$="5" THEN PLAY "T2003
L3GP2GP2GP2GP2G" ELSE IF I$=
"6" THEN PLAY "T2003L1GL3P2
GP2GP2GP2G" ELSE IS I$="7"
THEN PLAY "T2003L1GP2GL3P2
GP2GP2G" ELSE IF I$="8" THEN
PLAY "03T20L1GP2GP2GL3P2GP2G"
270 IF I$="9" THEN PLAY "T2003
L1GP2GP2GP2GL3P2G" ELSE IF I$
="0" THEN PLAY "T2003L1GP2G
P2GP2GP2G" ELSE IS I$="-" THE
N PLAY "T2003L1GP2L3GL1P2GL
3P2GL1P2G" ELSE IS I$="/" THE
N PLAY "T2003L3GP2L1GL3P2GL
1P2GL3P2G"
290 IF I$=CHR$(64) THEN 50 ELSE
IF I$=CHR$(12) THEN CLS
300 PRINT I$;
310 GOTO 150
320 CLS
330 PRINT:PRINT:PRINT"PRESS
PERIOD . FOR DOT"
340 PRINT:PRINT"PRESS SLASH
/ FOR DASH"
350 PRINT:PRINT"PRESS SPACE
BAR TO SEPARATE"
360 PRINT"PRESS @ TO RETURN
TO MENU"
370  A$=CHR$(32):B$=CHR$
(46):C$=CHR$(47)
380 I$=INKEY$:IF I$="" THEN 380
390 IF I$=B$ THEN SOUND 100,1
ELSE IF I$=C$ THEN SOUND 100,3
400 IF I$=CHR$(64) THEN 50 ELSE
IF I$=CHR$(12) THEN CLS ELSE IF
I$=A$ THEN PRINT CHR$(32)
```

LISTING CONTINUES ON PAGE 14

announcing the

68340 Accelerator Card for the MM/1!

Produced by Kreider Electronics
Marketed by BlackHawk Enterprises
(and other IMS representatives)

The Accelerator Card brings 68020 power to users of the MM/1!

The 68340 mpu features :

- A 68020 core minus bitfield instructions
- 3 serial ports with hardware handshaking vs. 2 serial ports (one without handshaking) on the 68070 board.
- No more 64K limit on DMA access.
- Eprom boots from the hard drive,

MORE SPEED!

Programs compiled on an accelerated MM/1 (MM/1a) with ULTRA C (not included) benchmarked at 5770 dhrystones. An MM/1a with MW C 3.2 yielded 2777 dhrystones, while an unmodified MM/1 yielded 1000 dhrystones from the same code. So, the 68340 upgrade gives you a speed increase ranging from 2.7 to 5.7 times faster than a stock MM/1!

Call today, and order your 68340!
At $325, this is a real bargain!

Great MM/1 Software:

From HyperTech:

DeskTop 1.0	$79	Paint 1.0	$79
PixUtils	$24	Fontasee	$34

From Bob van der Poel

VED 2.0	$59.95	VPRINT	$59.95

Call for other products today!

BlackHawk Enterprises

P.O. Box 10552
Enid, OK 73706-0552
(405) 234-2347

Reviews

68340 Accelerator

Upgrade for the MM/1

I am the proud owner of an MM/1A, that is an MM/1 which has been upgraded with the 68340 Accelerator upgrade. This is a description of the upgrade, what is involved in installing it, what it is supposed to give you, some gotchas I ran into installing it on my system, and some other observations.

What it is:

A "daughter board" which has the 68340 and a few support chips on it.

Two replacement ROMs.

One replacement PAL.

Floppy disk with new kernel and drivers.

Cost: $325 introductory price. Soon to be regular price of $350.

Optional:

If you are one of the unfortunate ones whose I/O board was equipped with 74HC257s (there are three of the in a row on the IO board) instead of 74AC257s, then you will need to replace the HCs with three ACs. Ouch! virtually all I/O boards with the HCs have them soldered in! I watched Kevin Pease replace these three chips in Bill Wittman's MM/1 in less than 20 minutes, so it is not an impossible task!

Note: If your I/O board does have the 74HC257s, you can still upgrade without replacing them. You can use a PAL which supports a four cycle memory access (instead of the three cycle) and put up with a 20% loss in speed gains.

(Editor's Note - Any TV/electronics repair shop with the capability of desoldering chips can replace the 74HC257s for you, usually at the minimum charge as it doesn't take but a couple minutes with the right equipment and experience. They WILL NOT usually warrant anything except the solder joints though!)

Installation:

1. MAKE A NEW BOOT DISK FIRST!!! Instructions on how to do this is in the install docs.
2. Remove the I/O board. Replace the PAL on it. Replace the 74HC257s if necessary. Re-jumper the I/O board for "no wait states".
3. Replace the two ROMs on the mother board.
4. Remove the 68070 and treasure it forever. If you ever have problems with the speed-up board, you'll need it again!
5. Set the speed DIP switch on the daughter board accordingly.
6. Plupgrade the daughter board into the empty '070 socket.
7. Reinstall the I/O board.
8. Boot up from the new boot floppy.
9. Customize your previous bootlist to use the new modules where needed.

The entire process (if you do not have to replace the 74HC257s) should take no more than a half an hour to an hour, depending on how long it takes you to recustomize your previous bootlist.

What does the 340 upgrade give you?

1. More processing power. The 68070 was a backwards engineered 68000 and therefore had numerous situations where the microcode was less efficient than an original 68000. The 68340 is a genuine Motorola product and is an enhanced 68020. Most instructions take fewer (usually half as many) clock cycles to complete plus since the 68340 was based on the 68020 core, it has most of the 68020 instructions. (There are only a few of the 68020 instruction set which are not supported.) Even though the "normal" clock speed of the 68340 will be 16.59 MHz for most MM/1 users, it still rates about twice as fast as a 15MHz 68070.

According to Mark Griffith, using the 68020 code will make many programs twice again as fast. Scott McGee reported a dhrystone result of 5770 (or close to that) for his accelerated MM/1 using the Ultra-C compiler while a dhrystone compiled with the 68000 compiler does about 2777 dhrystones on an accelerated MM/1. Each of these are compared to about 1000 dhrystones on a non-accelerated MM/1.

The 68340's clock (as used in the upgrade) is adjustable from 11.98 MHz to 16.59 MHz using either the DIP switch or a system state program which pokes the appropriate value in the correct memory location. There are a couple of models of the 68340, but the model which is faster than 16.59 MHz would be outside the price range of any MM/1 owner.

Brian White wrote a utility to modify the clock rate and did some dhrystone tests. He found that the 68340 had to be set down to an undocumented speed of 8 MHz to get a value equivalent to his 15 MHz 68070.

If you have slow DRAMS (i.e., 100ns) you will only be able to reliably run at about 12 MHz. 80ns DRAMS will allow a clock speed of about 16 MHz.

2. Better system I/O since the DMA transfers are not limited to a size of 64K. The new SCSI driver which comes with the upgrade apparently takes this new feature into consideration.

3. More and better serial ports. The 68340 has three serial ports, being used as /t0, /t1 and /t5. These are improvements over the '070s two serial ports in that not only do you get another port, but they all three look more like 68681 ports- full ports with hardware handshake- but CD is not currently implemented. A standard /t3-/t4 style paddle is required to use /t5. The header for it is on the bottom side of the '340 daughter board. /t0 and /t1 still connect at the same place as before.

4. The new ROMs support booting your MM/1A from your hard disk.

5. The new serial drivers now will time out on close, so that a modem port that is stopped by xoff can still be closed.

All in all, these advantages add up to a MM/1 which is roughly twice as fast as it was before the upgrade. All of the "benchmarks" I tested held true to this.

Gotchas which hit me

There is an elusive bug in OSK which Carl and Kevin Pease (the creators of the '340 upgrade) are both aware of which causes you to potentially have problems making floppy boot disks. If your floppy device is not initialized (or has some path in the system open to it), then when os9gen tries to rename the file TempBoot to OS9Boot, it fails. I forget the exact error message.

There are two workarounds. One is to simply rename the file yourself and do a -q option on another os9gen command (I found that this workaround was intermittent). The other is to simply put an iniz /d0 command in your start-up. This was my final solution.

Why did I even need to make a floppy boot disk when the new boot ROMs support hard disk booting, you may ask? Well, problem two is that apparently the boot code in the new ROMs perform ONLY "multi-sector reads". My hard drives, Maxtor 7213S', apparently do not support multi-sector reads, as I had to turn this bit off even with the '070 to get them to work. The result is when I try to boot from my hard drive, the boot code dies with a sector read error. Kevin has assured me that he will be able to fix this.

My third and last major problem was that I could not format my hard drives. Since I have two drives exactly alike, I decided some good tests would be copying files back and forth, etc. So the first thing I did was to reformat /h1 (of course I had done a backup first!!) But when format wanted to write LSN0, it would die. Again, I forget the exact error message. The solution here was easy. The original OSK disks from IMS included the format command from V2.3 and said that it was needed under certain circumstances, formatting floppies or something like that. So, my format command in my CMDS directory was version 2.3. Version 2.4 was in ROM, so whenever I ran format, it was coming from ROM anyway. Well the replacement ROMs which come with the 340 upgrade do NOT have ANY of those additional modules and programs which were in the original ROMs. So, when I replaced my ROMs for the upgrade, I was now running version 2.3 of format. Scrounging up my original floppies and getting the version 2.4 solved this problem. So the situation was caused by my not having the original V2.4 format command on my hard drive and the V2.4 format command (along with a bunch of other modules) were no longer present in the boot ROM like they were in the original ROMs. Best to save off any modules in memory which got loaded from the boot ROMs before upgrading just in case!

Other observations:

1. There is a mouse driver for Microsoft mice for the /t2 port, but there is not one yet for /t0.
2. There is NO sound driver AT ALL, yet. supposedly there will be one soon, one which takes the larger DMA transfer size into account.
3. The DIP for adjusting the clock speed is on the bottom of the board, and cannot be changed without actually removing the daughter board. But one can write a system state program and adjust the clock rate on the fly, the address and formula is provided in the docs. Maybe soon, Brian White will post his utility for adjusting the clock to Delphi or somewhere.
4. I use the new /t0 for my modem port and /t1 (with a /mi device descriptor and a MIDI paddle) and both seem to function just fine. I haven't tested /t5 yet.
5. Ulti-MuseE OSK seems to work fine.
6. dEd (OSK) will not run without a Stack Overflow. One can work around this problem by increasing the stack space given to dEd on the Shell command line, i.e.:

```
        $ ded #100 filename
```

7. Autofollow mouse seems *much, much* smoother and responsive.
8. One can take advantage of the included 68020 instructions by one of the following methods:

   a. Use GNU-C with appropriate switches. GNU C will utilize the Bit Field instructions but not callm/retm instructions.

   b. Use Ultra-C with the appropriate switches.

Using the 68020 instructions will make the MM/1A faster still. I would be interested in seeing some benchmarks with this. Remember that the compiled code will only run on a 68320 or higher processor, and possibly some other Motorola processor I am unaware of.

9. Oh yeah, one last thing. The upgrade comes with a new shell. It doesn't seem to be very different from the old shell, but I have noticed two differences. One is that cd and cx are used instead of chd and chx; in fact, if you try to run a chd or chx command you'll get a 216 error! The other difference is that when you fork a process at the shell level with a & on the end of the line, the PID of the child process which gets displayed does not have the plus sign (+) in front of it. I would have really liked it if there had been SOME documentation on this new shell.

Conclusion:

I feel that the investment was well spent. I am very pleased with the speed increase in my MM/1 and will soon take advantage of the additional serial port. I can live with the minor problems which have not been solved yet.

The 68340 Accelerator Upgrade, standard Serial Paddle boards and MIDI Paddle boards are available for immediate delivery from:

BlackHawk Enterprises
P. O. Box 10552
Enid, OK 73706-0552
(405) 234-2347 from 9am to 2pm CST
and
William Wittman, Jr.
873 Johnson Road
Churchville, NY 14428
(716)494-1506

BlackHawk is also taking $40 deposits to go towards a $95 8M backplane. As soon as he gets enough deposits (I think that he now has that) he will have the boards stamped and Kevin Pease will populate them. *See the BlackHawk ad in this issue.*

Note: I am not affiliated with BlackHawk Enterprises or William Wittman at the present time other than being a satisfied customer.

Zack C. Sessions, Color Systems

Tandy's Little Wonder

A book about the Color Computer

This book, I believe, is definitely a must for all CoCo owners be it beginner or experienced. The plethora of information contained within the pages of this book is incredible.

The book is 140 pages long double sided, 8.5"x11", two columns per page. It contains information on the CoCo one, two, and three. It starts off with an addendum for us good folk Down Under written by Fred Remin (Australian edition only) and then its straight to the Table of Contents.

The main topics covered in the book are as follows: Introduction, History, CoCo Hardware Prices: 1980-1991, CoCo Clones, Operating Environments and Programming Languages, Microware's OS-9, and Support. Then its followed by a Technical Reference that covers Peripherals, Hardware Upgrades and Modifications, Repairs, and Tid-Bits. Plus there is an Appendix, Schematics, Advertisers, and Index.

The information in this book must be worth at least three times its selling price. I purchased a service manual for my CoCo 3 from Tandy for $35 and this book is worth as much or more!

The book is an exceptional value for the money. It explains everything about how all the CoCos work and how Tandy went about producing it. But the book did not mention the fact that the CoCo 1 was originally sold to wheat farmers in the U.S.A. back in 1981 before Tandy ever released it! How many of you knew that? And I challenge someone out there to find which OS-9 book I got that from! Any way, back to the book.

If anybody can remember back to the October 1993 edition of *CoCo-Link* (an Australian publication), I asked about CN3 and CN4 in my CoCo 2. This book told me that the connectors were for a planned memory upgrade which was to be released from Tandy with 64K on it. But they never released it. I had suspected it might have been a prototype for the OS8 Microwave technology that was to eliminate the need for a modem. Who remembers that?). It also has some interesting stories on how to do software and hardware modifications from double siding your disk drives to repairing your Multi-Pak Interface, and everything in between.

This book is fantastic for value for money you just cannot go past this beaut of a book. I have read it from cover to cover and I thought it was great.

I'd like to finish off with this little question for you: Who first coined the word "CoCo" for our favorite computer? You will find the answer to this question plus many others in this book so why not buy it!

Desmond Rae
From Nov/Dec 1993 *CoCo-Link*
Reprinted with permission.

Tandy's Little Wonder is available from:

FARNA Systems
Box 321
Warner Robins, GA 31099-0321
912-328-7859
$25 including shipping and handling.
and
REMCOMS
11 Corcoran Cres
Canungra, QLD 4275, AUSTRALIA
075-435-821
$35 AUS. plus actual shipping charges.

Micro News

IBM has launched (early Nov.) a lower priced version of OS/2. Previously, OS/2 contained Windows code to run Windows applications. IBM had to pay license fees for use of this code, and charged $150 direct mail for OS/2 (resellers were charging $224). The "new" version takes into account that most systems are shipped with Windows and simply moves that code into a directory and incorporates it in the OS/2/Windows subsystem. What's the relevance to 68K systems? Professional OS-9 costs at least $299 from resellers, and doesn't come with a graphical interface like Windows (but it does come with a C compiler instead of BASIC). G-Windows is an additional amount (approx. $300). One reason not many people consider OS-9 as a general purpose operating system. You can get a near equal OS for much less with the capability of running many more applications. And OS-9000 is around $1,000! I wonder if that had anything to do with the name (OS-9 x 1000)...

IBM and Motorola announced that they have made their first MPC603 chips. Of more importance, however, is that the performance of the now available MPC601 has been upgraded - to a maximum of 80MHz. The 66MHz max (original) version is now used in the IBM 250 POWERStation and offers performance of 62 SPECint92 and 72 SPECfp92 (measurements of integer and floating point performance, respectively). The 80MHz version offers performance of 77 SPECint92 and 93 SPECfp92 and will sell for $500 ($50 more than the 66MHz) in lots of 1000 beginning in January. In comparison, a 90MHz Intel Pentium P54C (the fastest model, not available until early '94) has a performance of 78 SPECint92 and 85SPECfp92. The 603 is smaller than the 601, having only 1.6 million transistors compared to the 601's 2.8 million, and consumes only 2 watts compared to the 601's 8. The 603 is also a little slower, having only two 8K caches rather than two 32K. To refresh your memory, the 603 was designed for laptops and other energy efficient applications.

Just how fast is the PowerPC? Apple ran a demonstration at the October Sybold San Francisco computer show. A prototype Macintosh with a 60MHz MPC601 was loaded with native based applications. A Quadra 700 (68040) was loaded with the same applications in 680x0 code. The 601 model was obviously faster- Apple says two to four times faster depending on the application. Where the 601 took as little as two seconds to complete a complex graphics effect, the Quadra typically took an extra ten seconds. Apple plans on introducing MPC601 Macs in mid March.

Has anyone seen the Apple Newton commercials? Well, Tandy has released a pen based PDA (Personal Data Assistant) also. Called the Z-PDA or "Zoomer", it features 1MB of RAM (640K for system, 348K for RAM disk) and 4MB of ROM. The new PDAs have no keyboard... everything is input through a pen that interacts with the screen. You use the pen much as you would a mouse, except that you actually touch the screen. To input data, simply write on the screen. It does take a while for the handwriting recognition software to "learn" how you write, and you do have to learn to write clearly. But after a few pages of text, errors are few and it gets better with use (don't loan it out a lot... multiple users could create problems!). Unlike the Apple Newton, the Zoomer only takes printed characters, not cursive. This forces a user to write slower and consequently improves the recognition and lessens time for the machine to "learn" how one writes. Tandy uses the GEOS operating environment and MS-DOS in ROM, along with several applications (address book, notepad, Quicken, world-time map, America Online interface, dictionary, language translator, etc.). Along with the pen, there is also a Nintendo style pad and buttons below the screen (maybe games will become available?). There is no disk drive interface, just one RS-232C port (subminiature 10 pin connector) and an infrared transceiver capable of 9600bps. Hopefully this means a transceiver will be available to connect to a desktop later. The unit only weighs 16 ounces and will last about 100 hours on three AA cells. The PDA was jointly developed by Tandy and Casio, who will also sell it as the Z-7000. Similar in concept, the Z-PDA should eventually replace the aging Model 102 (it won't replace a good laptop though!). Would be great for taking notes and transferring through the serial port to an OS-9/OSK machine. Price is around $700.

Kodak's PhotoCD format has become a standard in the computer industry. The reason is that it is now easy to get high quality images in a computer compatible format. Simply pick up your camera, send the film to Kodak, and a week or two later you have your images. High quality images take up 7-8 megabytes of space on average, so you'd need BIG hard drives otherwise. The Phillips CD-I machines will allow viewing the CDs as well as 80% of the CD-ROM drive available now. If someone wants more info on writing software and maybe even a device driver, Kodak can be contacted at 1-800-243-8811. Many hardware companies have sample code that can be used without license available. If Kodak has C source available, porting to OS-9 will be easy. assembly will of course be harder, but possible. Dirt Cheap Computer Stuff Co. (1368 Old Highway 50 East, Union, MO 63084; 314-583-1168) currently sells drivers that allow accessing information on a CD-ROM as well as CD-ROM drives, but unless the info is ASCII text special database/application software is usually required. This is usually contained on the CD, so the code can be accessed for later disassembly...

The Clinton administration has called for more public access to government documents and resources. The White House can be sent E-mail over Internet, and now the Commerce Department's National Technical Information Service has connected it's large Fedworld bulletin boards into the Internet. Eight Internet connection lines are available at fedworld.gov, and have become so popular that eight more will soon be added... the existing ones are almost always jammed! 32 dial in lines are available at 703-321-8020. Simply use telnet to connect to any number of government BBS'. There is separate Internet access to the Commerce Dept. Economic BBS at ebb.stat-usa.gov (telnet can also be used at fedworld.gov). Other government BBS' available through Internet are the Future Studies Unit of the EPA (futures.wic.epa.gov), Library of Congress (marvel.loc.gov), and the President's and Vice-President's offices (president@whitehouse.gov and vice-president@whitehouse.gov receipt of message sent but no personal replies). E-mail can also be sent to the House of Representatives by requesting the E-mail address of your rep from comments@gr.house.gov.

I bet you thought the MC68040 processor was pretty fast, huh? Well, Cray's new T3D super computer has up to 256 Alpha AXP 225MHz 64 bit processors (the near equivalent of 57,600MHz!!!!). This is about the pinnacle of speed in computers today. The internode input/output is only 200 megabytes per second in each direction... both at the same time (400MB/sec). The entry level 128 processor system is only $2 million. Hmm... I wonder if we could interest them in a port of OS-9...

I recently got a letter from Mr. William Wittman, Sr., who is an authorized MM/1 dealer. Anyone in the New York state area interested in an MM/1 can contact him at 873 Johnson Road, Churchville, NY 14428; phone 716-494-1506. I wish Mr. Wittman well in his new MM/1 dealership! Others interested in becoming MM/1 dealers should contact David Graham at BlackHawk Enterprises (see ad in this issue).

Have you ever wished there was a complete OSK computer system for under $100? Sure, a KiX\20 can be purchased for around $700, but add a case, power supply, hard drive, video/audio/keyboard card, keyboard and monitor, and you have spent over twice that amount. Even the MM/1 is over $1400 with all peripherals!

Would you be interested in a basic OSK system, complete with everything needed to start computing, for under $1000? That's what FARNA Systems wants to know! What you would get for around $900 is a 10MHz 68000, 1MB of RAM, monitor (mono or CGA color), floppy and hard drives (360K & 20-30MB, respectively), keyboard, case, and power supply. State of the art and super fast this isn't, but the main board can be traded in for partial credit on a faster one later, and the system is easily upgradeable to VGA and a larger hard drive later.

If enough people are interested, a completly assembled system will be made available for under $1000 and a component kit for a little less. This sytem won't become available unless enough people express interest before 01 February 94. If enough interest is shown, systems and kits will be available by the end of February for immediate shipment. This approach was decided since the Burke&Burke "Rocket" was dropped. If you're interested, write FARNA Systems at 904 Second Avenue, Warner Robins, GA31098-1029.

< 268'm >

If you have new soft or hardware products, let us know! We will gladly print a free blurb for you here in MicroNews whether you advertise or not (though we will be happy to have your ad also).

THE SWAP SHOP

Buy, sell, trade hard/soft ware! If you have something you no longer need, need something you can't find, or just have some extra "stuff" lying around, this is the place to find or get rid of it!

Ads are free. Only those vendors **with at least a 1/4 page ad may place classifieds without special permission.** For these, short ads will be free as space permits. PLEASE drop us a post card or E-mail if/when an item sells or is found!

FORSALE

Multi-Pak, 26-3124 (small), upgraded for CC3- $70; OS-9 Level II- $35; Multi-Vue-$10; Koronis Rift-$10; Rogue-$10. Shipping included to US. John Gar, RR1 Box 141, Newel, SD 57760

Replacement 68B09E for all CoCos. $7 each includes S&H. Timothy D. Boos Sr., Rt. 1 Box 155, Peculiar, MO 64078

SALEORTRADE

CoCo 3, 512K, CM-8 RGB monitor, FD-501 disk drive (2 drives), Ken-Ton SCSI hard drive interface, Y-cable, RGB-DOS in ROM, 40MB hard drive (factory RGB-DOS setup for DECB), lots of software (send SASE for listing). Will take best offer. Alan Clarey, 1012 Borrette Lane, Napa, CA 94558

Extra CoCo 3s! Have 5 CoCo 3s, 6 CM-11 monitors, 3 5 1/4" drive systems. Call and make offers on one or all. Arnold Stark 813-654-4198 (day) or 813-621-4987 (evenings)

CoCo 2, 64K; CoCo 3, 128K; CM-8 monitor, dual 5 1/4" floppy system. Call and make an offer! Jenny 510-779-1102

WANTED

Tandy OS-9 C Compiler with documentation. Contact Joe Charbonneau 527 Jarvis St., Windsor, Ontario N8P 1C8 (Canada); Phone 519-735-8630

Used CoCo hard/software. Will buy by piece or entire collections. Rick Ulland, 449 South 90th St., West Allis, WI 53214

DISTO Products

Quality hardware for your Color Computer!

2MB Upgrade (no RAM) - $99.95
Mini Disk Controller - $70
Super Controller I - $100
Super Controller II - $130
MPROM Burner - $50
3-N-1 (parallel, RS-232 RTC) - $75
SASI/SCSI Hard Disk Adapter - $75
Full Turn of the Screw - book with all Tony DiStefano's Rainbow articles, January 1983 to July 1989 - $20
Complete Schematic Set - all DISTO product schematics except 2MB upgrade - $20

Include $2 S&H for book or schematics. Hardware S&H is $4.50 for one item, $6.50 two or more. Certified check or International Money Order only!

DISTO
1710 DePatie,
St. Laurent, QC H4L 4A8
CANADA

Bob van der Poel Software

Great stuff for your OS-9 Level II system!

Ved Text Editor	$24.95
Vprint Text Formatter	$29.95
OS-9 character Set Editor	$19.95
OS-9 Disk Mailing List (DML9)	$24.95
Basic09 Subroutine Package	$24.95
Cribbage	$19.95
Ultra Label Maker	$19.95
Magazine Index System	$19.95
RMA Assembler Library	$19.95
Stock Manager	$24.95
OS-9 Public Domain Disk	$9.95

(see DML9 review in this issue!)

All our programs are in stock for immediate shipping. Please include check or money order with your order. Sorry, no credit cards; but will ship COD to US and Canada (we add a small additional charge to cover the post office COD fee). Mention this ad and get FREE SHIPPING (normally 5% or $2 minimum)! All orders are shipped via first class mail, usually the day received. Write or call for free DECB or OS-9/6800 catalogue.

P.O. Box 355
Porthill, ID
US 83853

P.O. Box 57
Wynndel, BC
Canada V0B 2N0

Telephone (604) 866-5772

for all your CoCo hardware needs, connect with

CoNect

449 South 90th Street
Milwaukee, WI 53214
414-258-2989 (after 5pm EST)

Mini RS-232 Port: Don't let the name fool you! This is a full featured serial port, supporting the signals needed for flow control as well as the basic 4. Jumper blocks allow readdressing or swapping DSR/DCD. No custom cables or hardware widgets needed here! Y cable users will need to add $9.95 for a power supply. **$49.95**

XPander: Don't you think the CoCo would be a lot nicer without all that mess hanging off the right side? Of course it would! Our XPander allows mounting two SCS decoded devices (like a floppy and hard drive controller) inside your CoCo. Built-in no-slot RS-232 port is similar to our "Mini" described above. The external cartridge connector is still present, and can be configured to run games or as an additional hardware slot. Kit includes new lower case shell and 12V power supply. Board only is great for use in a PC case!
Kit: $124.95 Board Only: $99.95

Hitachification: CoNect will install a Hitachi 63B09E CPU and a socket into your CoCo. Machine MUST be in working condition! The 68B09E will be returned unharmed. 90 day limited warranty. Chip and installation only **$29.95**

REPAIRS: We can repair most damaged CoCos, even those with bad traces where a 68B09 was removed. Costs vary with damage. Bad 68B09 sockets repaired for only $40! **Inquire BEFORE sending your computer.**

New Lower Prices!
from ColorSystems

Variations of Solitaire

Includes FIVE variations; Pyramid, Klondike, Spider, Poker and Canfield. Complete documentation shows how to create your own games boot disk using the special menu program which is included.

CoCo 3 Version $29.95
MM/1 Version $39.95

WPShel

A Word Processing Oriented Point and Click Shell for all your word processing needs. Requires WindInt from your Multi-Vue disk. Does not include Editor, Formatter, or Spelling Checker.

CoCo 3 Only! $20.00

We accept Personal Checks or Money Orders drawn from US Banks or International Postal Money Orders. NC residents please add 6% Sales Tax. Call or write for FREE catalog! Please add $3 per item for shipping outside of the Continental United States.

NEW!
Using AWK With OS-9

A description of the AWK Programming Language with an emphasis on GNU AWK for OSK. Includes the latest version of GNU AWK.

OSK ONLY! Just $19.95

OS-9 Game Pack

Includes FIVE complete games; Othello, Yahtzee, Minefield, Knights Bridge, and Battleship. Includes special menu and step by step instructions on creating your own games boot disk.

CoCo 3 Version $29.95
MM/1 Version $39.95

All CoCo 3 Programs require at least 256K of memory.

Coming SOON! Indexed Files for OS-9 Level 2, OS-9/68000, and OS-9000!

ColorSystems
P.O. Box 540
Castle Hayne, NC 28429
(916) 675-1706

Quality OS-9 Software for the Color Computer 3 and the MM/1 from IMS

Quality CoCo Software and Publications from FARNA *Systems...*

ALL NEW! – **DigiTech Professional Sound Recorder**
Record ANY sound on disk with your CoCo! Easily add sound effects/music to BASIC or M/L programs. Sound input through joystick port with easy to make cable (cable instrucs. included, or order separately). **$20** cable- **$6** w/software, **$8** later

OS-9 Quick Reference Guide II & PatchOS9 Disk Set
New, revised edition of the popular QRG now has Basic09 error codes added to it. Also contains commands, system calls, error codes, everything needed in a handy desktop size. The PatchOS-9 disk set includes the most popular and useful patches and replacement commands for CoCo OS-9 Level II.
QRGII/PatchOS9 set-$13.50; QRGII- $8.50; PatchOS9 - $7.50

ADOS - the premier Disk Basic enhancement! Adds support for double sided drives, 40/80 tracks, faster formatting, many more items. Extended ADOS3 has built in RAM disk, even more!
ADOS (CC 1 & 2) - $15; ADOS 3 - $25; Ext ADOS 3 - $30 (ADOS3 required); ADOS3/Ext ADOS3 Combo - $50

CoCo Family Recorder - absolutly the *BEST* for Genealogy! Prints many different reports. Tracks up to 550 persons, any number of generations. Requires 80 column monitor and two disk drives. No special order for information to be in, easy to use. **$20**

Programming the 6809 - Zaks & Labiak, teaches m/l programming for the 6809. Recommended by Motorola. $23

FARNA *Systems*
Box 321, Warner Robins, GA 31099-0321
Phone 912-328-7859; E-mail dsrtfox@delphi.com

add $2.50 S&H, $4 Canada, $7 Overseas

INTRODUCING: CoCoTop,
Desktop Program for OS-9 Level II.

Now it has become much easier to use OS-9 on your CoCo.Most of the functions that you carry out by typing command lines at the OS-9: prompt have been built into this program that sports some 60 functions in a text based point-and-click interface.

If you have to copy or delete a file, you no longer have to type a filename or pathlist but you highlight the name on screen with you mouse or joystick, press a buootn, and watch your CoCo do the work.

There are also a number of functions you will not find in standard OS-9: a "safe" format along with an "unformat" command that will let you recover data from a formatted disk. For files: undelete, move, a file finder, and overwrite capabilities for the copy command. For directories you will find sorting of contents and you can move a directory if you wish.

There are functions to track your system: configuration and changing device desriptors for disk and printer. Maps of how your computer uses it's memory and disk space. CoCoTop can set up windows for you and launch other programs. It even has a menu where you can link your favorite programs to it. Of course it also has a built in calculator and notepad.

For people who have problems with OS-9's disk structure the good news is that the program presents that directory structure as a series of point-and-click menus: keeping track of where you are and where you can safely go. It also displays the pathlists used so you can learn as you go along.

Requires 512K CoCo 3. There are two versions: 1.0 is the complete package. 1.1 is for those who already have TOOLS 3 since CoCoTop uses some of the TOOLS 3 utilities.

CoCoTop 1.0 : $24.95
CoCoTop 1.1 : $19.95
CoCoTop 1.1 + TOOLS3 : $34.95
TOOLS3 1.1 : $29.95
QUICKLETTER 2.0 : $19.95
ACCOUNTING LEVEL II : $34.95
INVESTING LEVEL II : $24.95
LEVEL II GRAPHICS : $34.95
UPGRADES from earlier versions are only $5 (return original disk)

Christmas Sale: 20% off two packages, 40% off three or more!

Level II Graphics Demo: $5 (+$3 S&H). Full amount credited toward purchase of full working version.
RScopy/OScopy: Transfer between OS-9 and Disk BASIC with no special disk drivers or patches. $10 (+$3 S&H).

C. Dekker ... *User-friendly Level II Programs!*

RR #4 Centreville, NB
E0J 1H0, CANADA
Phone 506-276-4841

Shipping+handling: US/Canada $3.00 all others $5.
Prices in US dollars Send cheque or money order NO COD'S.
Call or write for Canadian dollar prices.

C. Dekker

This Issue's "microdisk"

OS-9 Content:
- LISTING.C
- ENTRY.BAS
- TSTENTRY.BAS
- ENTRY.C

Disk BASIC Content:
- MAZEMKR.BAS
- MORSE.BAS
- RTLONG.BIN

*Note: RTLONG is a 512K memory test that should have been on Vol. 1 #3 "microdisk". This is a true memory test which checks for ghosting and shorted address lines... the only CoCo test to do so!

"microdisk" is available for $40 per year, $21 for six months, or $6 per issue. Overseas must add $10/year, $5/six months for air mail delivery.
"microdisk" is NOT a stand-alone product, but compliments "the world of 68' micros" magazine.

Disk BASIC users, don't criticize the quality of programs and articles... DO SOMETHING ABOUT IT!
See "Submitting Material", this page

Submitting Material to 68' micros

FARNA Systems retains rights to print and distribute any and all contributions. The submitter retains rights to distribute, but not to print in another publication without consent of FARNA Systems unless other arrangements are made.

We accept program submissions in any programming language for DECB and OS-9 (6809 & 68000) of any type (games, utilities, etc.). Articles are accepted covering any aspect of Motorola 68xx and 68xxx processors. This includes microcontroller projects as well as alternate operating systems. If there are enough subscribers interested, we will begin accepting programs for alternate operating systems as well.

Submissions should be sent on disk in ASCII and executable formats. A printed listing should also be included if possible. A letter describing the program or article is also necessary. Submissions can be made to DSRTFOX on Delphi, or dsrtfox@delphi.com via Internet.

Media accepted: 5.25" disk in CoCo OS-9 (35/40T, SS/DS), IBM (DD/HD), or DECB (35/40T). 3.5" in IBM only (DD/HD)

Your Ad Should be here!

Advertising budget tight? Don't think you can afford to advertise? Well, you can't sell if you don't!
You can advertise in "the world of 68' micros" for as little as $10 per issue !
NOTE: These rates for first time advertisers only. Limited to three insertions. See inside front cover for regular rates.

MultiBoot by Terry Todd & Allen Huffman
Now have up to SIXTEEN bootfiles on your startup disk!
Hot off the assemblers and compilers is a great must-have utility which lets you have up to 16 bootfiles on one disk! No more boot disk floppy-swapping! MultiBoot will install itself to a cobbled boot disk and, upon typing "DOS", will greet you with a scrolling menu of available bootfiles!
OS-9 Req: CoCo3, OS-9 Level 2........................$19.95

Towel by Allen C. Huffman
The first EthaWin program - a disk utility for OS-9.
A program no intergalactic hitchhiker should be without! Use a mouse or keyboard hot-keys to perform common file and disk commands from pull-down menus. Tag multiple files for Delete, Copy, Rename, etc., and even have point 'n click disk Backup, Cobbler, Dcheck and other commands. Usermenu lets you specify up to seven of your own commands to execute. Runs under the EthaWin interface on a high-speed text screen. All commands/colors configurable.
OS-9 Req: CoCo3, OS-9 Level 2..............................$19.95 OS/K Req: MM/1 or K-Windows Compatible...........$24.95

1992 CoCoFest SIMULATOR by Allen C. Huffman
Graphics "adventure" based on the 1992 Atlanta CoCoFest
The next best thing to having been there! Digitized graphics of the event and a text command parser (ie, "get the box of disks") let you see all the vendors and even run into some famous faces of the CoCo Community. The show area, seminar room, and portions of the hotel are all represented. No true "goal", but you do have to figure some things out, like how to get into the show and how to buy items from vendors. Runs on a 640x192 hi-res graphics screen.
OS-9 Req: 512K CC3, OS-9 Lvl 2, 490K Disk Space...$9.95
OS/K Req: MM/1 or 100% K-Windows Compat.......$14.95

Worlds at War: *A complete wargame simulator package!*
Finally, this Canadian masterpiece is available. Icon editor lets you build full color game pieces. Map editor lets you put together a multi-screen playfield. Options such as pass, move, attack, status, cargo, search, and build make this game a real "blast".
OS-9 Req: 512K CoCo3, OS-9 Level 2........................$SOON!

SUB-ETHA SOFTWARE

P.O. Box 152442,
Lufkin, TX 75915
Please include $2.50 S&H per order

ADVERTISER'S INDEX:

BlackHawk Enterprises	21
Bob van der Poel Software	25
C. Dekker	26
Color Systems	26
CoNect	25
Delmar Company	BC
DISTO	25
FARNA Systems	10, 20, 26
Infinitum Technology	22
Mid Iowa & Country CoCo	10
OS-9 Underground	12
Robert Gault	6
Sub-Etha Software	27

[illegible]

68' micros

[illegible]

For superior OS-9 performance, the

SYSTEM V

Provides a 68020 running at 25 MHz, up to 128 MBytes of 0 wait state memory, SCSI and IDE interfaces, 4 serial and 2 parallel ports, 5 16-bit and 2 8-bit ISA slots and much more. The SYSTEM V builds on the design concepts proven in the SYSTEM IV, providing maximum flexibility with inexpensive expandability. Now available at 33 MHz.

AN OS-9 FIRST - the MICROPROCESSOR is mounted on a daughter board which plugs onto the motherboard. This will permit inexpensive upgrades in the future when even greater performance is required.

G-WINDOWS benchmark performance index is 0.15 seconds faster with a standard VGA board than a 68030 running at 30 MHz with ARTC video board (85.90 seconds vs 86.05 seconds).

Or, for less demanding requirements, the

SYSTEM IV

The perfect low cost, high-quality and high performance OS-9 computer serving customers world-wide. Designed for and accepted by industry. Ideal low-cost work-station, development platform, or just plain fun machine. Powerful, flexible, and inexpensively expandable. Uses a 68000 microprocessor running at 16 MHz.

Both computers provide flexible screen displays in the native mode with the optional VGA card.

Eight text modes are supported: 40 x 24, 80 x 25, 80 x 50, 100 x 40, 132 x 25, 132 x 28, 132 x 44, and 132 x 60. Foreground, background, and border colors are user selectable from up to 16 colors.

Eleven graphics modes are supported: 640 x 200 x 16, 320 x 200 x 256, 640 x 350 x 16, 640 x 350 x 256, 40 x 480 x 16, 640 x 400 x 256, 800 x 600 x 16, 640 x 480 x 256, 1024 x 768 x 16, 800 x 600 x 256, and 1024 x 768 x 256.

Text and graphics modes may be selected by a utility provided, MODESET, by software using SetStt calls, or by termcap entries. In the text mode, the screen responds to standard VT100 control sequences. The full character set from Hex 20 through Hex FF is supported in the text modes up to and including 100 characters wide. The upper 128 characters follow the 'IBM Character Set 2' popular with many terminals and printers. These may be displayed on the screen by using the 'Alt' key and one or two other keys (software permitting).

Optional G-WINDOWS provides three screen resolutions: 640 x 480 x 256, 800 x 600 x 256, or 1024 x 768 x 256. You can have two full size 80 x 25 windows with room to spare. Or, a window as large as 122 x 44 using the large fonts or one over 180 x 70 using the small fonts

delmar co

PO Box 78 - 5238 Summit Bridge Road - Middletown, DE 19709
302-378-2555 FAX 302-378-2556