C.C.O.G.
COLOR COMPUTER OWNERS GROUP

NEWSLETTER AUGUST 1991

OFFICERS:	
PRESIDENT	Bernie Patton
VICE-PRESIDENT.	Chuck Goodwin
SECRETARY	Marcine Glowicki
TREASURER	Bob Waite
CORRESPONDING SECRETARY . .	Robert Gault
LIBRARIAN	Bob Humphrey
BULLETIN BOARD.	(313) 292-4713

INDEX:

PRESIDENTS COMMENTS:

My grandmother had a favorite saying: "You had better be careful about what you ask for, you just might get it!!!" Well we asked and we got it!! What did we ask for? Vendors to relpy to our offer to place their ad in our little newlsetter for free.

Well, thus far 11 vendors have responded (ads included in this volumn) and also most of them have opted to subscribe to our newsletter. I think this is great!! Many of them indicated that they liked reading our newsletter and had good things to say about the articles.

The september Rainbow has an article on "The Delmar 68000 based System IV". The headline is "A New Era". I hope that Frank Hogg and IMS will also send machines to the Rainbow for review. We have been two years waiting for these "NEW" machines to come to market. Many from the CoCo community have tired of waiting and gone on to other machines. Let's hope that those of us who have stuck it out will be rewarded with machine capability that will surpass that of the current MS-DOS world.

This meeting it will be almost Labor Day. We only have three more meetings after this until we are into the holiday season. We have to give consideration to what we are going to do about a December meeting (If, where, and when). We also need to have more discussion at meetings about future topics and demonstrations. We need to have more of our membership step up and show us the software you are currently using on your COCO. It dosn't matter how new or old the software is, if it works, if it gets the job done, if you have fun with it then it's not obsolete.

It's soon going to be time for you to be thinking about running for office in YOUR club. We will be having nominations in October and elections in November. Be prepared to help to make this the BEST club anyone could belong to. Bernie

COLOR COMPUTER OWNERS GROUP

This newsletter is a periodic publication of the
COLOR COMPUTER OWNERS GROUP
of Metropolitan Detroit.

Contact: Bernie Patton (313) 283-2474 BBS (313) 292-4713

C.C.O.G Meeting Minutes July 23, 1991

The meeting was called to order at 7:20 pm. Sixteen people were in attendance. The minutes from last month were accepted.

Old Business--Bernie went through past issues of COCO CLIPBOARD and RAINBOW to formulate a list of CoCo advertisers. He had a list of 57 advertisers. Next he composed a coverletter to send to each advertiser. The letter basically stated that our club, C.C.O.G., would accept advertising from them for inclusion in the club newsletter. There were several conditions but nothing overwhelming. C.C.O.G. would accept their copy of advertising for a time period of 3 months, and this is renewable if they continue to send an updated copy. Also for $3 the advertisers would receive 3 months of our newsletter. $6 would get them a one year's subscription to our newsletter. This fee is only to cover the cost of mailing the newsletter since we are a non profit organization. Bernie mailed the letters approximately 2 weeks ago. One letter came back "moved and no forwarding address". Three responses have already arrived, (see this month's newsletter for their advertising), Color System, MV Systems and Howard Medical. Howard Medical also sent a check for $6. By next month there should be more responses.

Gus also asked if we were going to write a letter to the editor of RAINBOW stating what our club was doing. Bernie sited his closing statement in his coverletter to advertisers, "Accept this in the spirit it was intended." This idea of accepting advertising is not to usurp RAINBOW but it is an offer for those companies unable to pay the high advertising costs to get their product out to the public. Remember our newsletter goes out to other clubs, Radio Shack's and our members.

After all the mailings a postage bill totalling $17.80 for advertising mailings, $2.36 postage for last month, and probably another $2.36 for this month. Bob was not at the meeting to give a treasurers report.

New Business--Robert Gault noticed that we have been dropped from RAINBOW'S Club List. He will be contacting RAINBOW to find out why.

Bob Humphrey resigned as the club librarian. Larry Schneider will be taking over. Thanks Bob for a job well done.

Demonstrations for the evening included Bob Gault working with random number generators and checkers to see how really random they are and Jim Snider showed off his latest hardware project. He used a mini tower AT case and put all his COCO toys in it.

RAINBOWS theme next month is graphics and normally it is games. Both will be August's theme.

Next meeting Aug. 27, 1991.

OS-9 by Tom Napolitano

Robert Gault last month wrote a benchmark test program for comparison of the processing speed of the Coco3 running a basic looping routine under RSDos BASIC and under BASIC09. The results were interesting, and he suggested that I do the same for C. Actually I started the project then found there were quite a few different ways to express the same algorithm in the C language.

The first version ran through an empty loop 10000 times. This is inconclusive of anything, since it took about 1 second using the internal clock. Since the granularity of the clock is only on second, all this says is that it takes a "very short time". So back to the drawing board.

Here is the program.

```
/* loop1.c   */
/*  Benchmark test of c compiler  */
/*    < 1 second @ 2mhz  */

#include <stdio.h>
#include <time.h>

struct sgtbuf buff;
main()
{
int i;

    printime();

    for(i = 0; i < 10000; ++i)
        ;

    printime();
}
```

Notice that I declared the loop variable i as an integer. This should be much faster than RSDos BASIC using a float for all numbers. It is hard to tell, so I slowed the program down by stretching out the loop. By the way, printime() is a routine to print the current time on the os9 clock. I'll get into that at this time only to say that how it is written doesn't affect the results much.

```
/* loop2.c   */
/*  Benchmark test of c compiler  */
/*    10 seconds @ 2 mhz  */
main()
{
unsigned int i, j;

    for(i = 0; i < 10; ++i)
        for(j = 0; j < 65535; ++j)
            ;
}
```

In loop2 I extended the loop to 655350, in order to get a better measure of running times. From here on, I'll skip printing some of the "overhead" code, and just show the parts of the program that change. I had to unroll the loop with two variables. Why? Because an os9 C integer is only 16 bits of binary data. As you would expect, this loop takes ten times as long. But why use two loops? Why not use just one by declaring the loop variable as a long integer? Surprise, surprise. It takes six times as long!

```
/* loop3.c   */
/*  Benchmark test of c compiler  */
/*    59 seconds @ 2 mhz  */
main()
{
long i;

    for(i = 0; i < 655350; ++i)
            ;
}
```

The reason, when you think about it is that in order to compare two long ints, and to increment a long integer, it takes over twice as long. Just shows the inefficiency of C versus assembly. Now lets see if we can speed things up from the ten second time.

```
/* loop4.c   */
/*  Benchmark test of c compiler  */
/*    6 seconds @ 2 mhz  */
main()
{
unsigned int  i, j;

    i = j = 0;
    while(++i <= 10)
        while(++j )
            ;
    printime();
}
```

Here I used a "while" loop instead of the "for" loop. It has the same effect, but runs faster. Sometime we should examine the resulting output from the c compiler to see how it handles the two constructs differently.

```
/* loop5.c   */
/*  Benchmark test of c compiler  */
/*    10 seconds @ 2 mhz  */
unsigned int  i, j;
main()
{
    i = j = 0;
    while(++i <= 10)
        while(++j )
            ;
}
```

When you declare int i,j within a function, these are called "automatic" variables. Space is reserved on the temporary stack area of a function as the function is entered, and when the program leaves off executing the function, the automatic variables go away. Instead, in loop5 I declared i and j outside the function "main" as "global" variables. These keep there value as long as the program runs, irrespective of what function is being executed at any given time. It appears that accessing globals is slower that accessing data off the stack.

```
/* loop6.c   */
/*  Benchmark test of c compiler  */
/*    7 seconds @ 2 mhz  */
main()
{
double  i;

    printime();
    i = 0;
    while(++i <= 10000)
            ;
}
```

As I mentioned, RSDos BASIC declares all numerical data as floating point numbers. So to test how this compares in C, I declared i as a double width floating point number. This slowed things down somewhat, but now I can go back in loop6 to only one loop variable, and a loop of 10000.

```
/* loop7.c   */
/*  Benchmark test of c compiler  */
/*    6.5 seconds @ 2 mhz  */
direct int  i, j;
main()
{
    i = j = 0;
    while(++i <= 10)
        while(++j )
            ;
}
```

In loop7 we took advantage of a trick of the 6809 architecture, direct variables. What this means is that we are using special 6809 instructions that allow us to access direct data with one less byte of addressing. What this shows here is that it is faster access direct variables than regular globals, but still is faster to access off the stack. Another trick is to declare certain variables in C as residing in registers of the CPU. This only works for one variable on the 6809 processor, but we might as well take advantage of it. It shaves another half second from the previous best time.

```
/* loop8.c   */
/*  Benchmark test of c compiler  */
/*    5.5 seconds @ 2 mhz  */
```

```
main()
{
register unsigned int j;
unsigned int  i;

    i = j = 0;
    while(++i <= 10)
        while(++j )
            ;
}
```

For those of you who are interested, here are two versions of printime(), with no comments from this person. See if you can decode the somewhat cryptic programming.

```
/* Version 1 of printime;  obscure  */

int printime()
{
    getime(&buff);

    putchar(buff.t_minute>9?(putchar(buff.t_minute/10+'0'), \
    (int)buff.t_minute%10+'0') : (int)buff.t_minute%10+'0');

    putchar(' ');

    putchar(buff.t_second> 9 ? (putchar(buff.t_second/10 + '0'), \
    (int)buff.t_second%10+'0') : (int)buff.t_second%10+'0');

    putchar('\n');
}

/* Version 2; neater */

int printime()
{
    getime(&buff);

    if(buff.t_minute > 9)
        putchar(buff.t_minute/10+'0');

    putchar((int)buff.t_minute%10+'0'));

    putchar(' ');

    if(buff.t_second > 9)
        putchar(buff.t_second/10 + '0');

    putchar((int)buff.t_second%10+'0'));

    putchar('\n');
}
```

"Viewpoints" by Gus Korte"

Recently I contacted club member Jim Snider on the electronic bulletin board system (BBS) the CCOG club supports and arranged to install a FD502 disk drive and a FD500 disk drive into one unit. Not only did he do this with his usual efficiency but also replaced my disk controller ROM with a much more efficient ROM he has developed called an EPROM (electrically programmable read only memory). What a difference!! Now I have 3 disk drives compressed into one space-saving unit instead of 2 separate disk drive units as I had initially. In addition the EPROM does a lot of additional things the previous ROM did not do. When the disk drive is turned on, an 80 column screen appears automatically with white letters on a dark blue background instead of the 32 column green screen with black letters. When I type the command DIR to see the disk directory, instead of the contents scrolling by so fast that they are virtually unreadable, now the information pauses after the screen almost fills. This allows one to read the screen for a change at the reader's pace with a screenful of information each time. Previously I had to use club member Bob Gault's program (in the CCOG club library if you want to use it) to slow down the scrolling. Now each disk drive including a double-sided disk drive is available within one ROM. The step rate using the DSKINI command is now increased too. The VERIFY command is now constantly on which is very handy when copying other diskettes. Before you can "kill" a file, you must confirm it first. There are many other items Jim has in the EPROM. It also was interesting to see Jim's electronic work space. He is well equiped to work on the COCO. Many thanks to Jim for his assistance in this situation and in others where he has given considerable assistance to us CCOG members. As has been mentioned previously, having a multiple disk drive system is a necessity to get full use of the BASIC09 software and related products.

It was interesting to view the VCR that was made at the 6/25/91 CCOG meeting. All of the comments were clear but for an unknown reason the video became unclear and could not be viewed. The sound portion was clear and any of you who missed or want to reveiw the June meeting may want to run this VCR tape which Bernie Patton owns. Thanks to Bernie for producing VCR tapes like this of selected useful presentations at the CCOG meetings for us to view and hear especially where we missed the meeting or want to reveiw them.

On page 11F of the Detroit News dated 7/22/91, appeared an article entitled "New software sparks a CD-ROM boom". It stated that "the home market for CD-ROM -- the computer version of the audio CD, or compact disc -- is in the early stages of a boom as personal computers move onto the shelves of mainstream retailers such as Sears."
It also stated "Computer users can also play regular music CDs

on their system, controlling volume and other controls by clicking with a mouse on the picture of a CD player on the screen." "Among some of the new offerings: Microsoft Bookshelf, a collection of reference books including a dictionary, thesaurus, almanac and ZIP code directory. National Geographic's Mammals: A Multimedia Encyclopedia, with 700 color photographs, 600 pages of text, 155 animal vocalizatins and 45 video clips."

That indicates the ever expanding items available to the computer hobbyist. The memory required for such CD's must be extremely larger than anything now available to us COCO users. Maybe the future COCO4 models will one day be able to include such items but at a very high price. This does indicate the future trend for those willing to invest more into such items.

Karl Sefcik's message in last month's club newsletter is again repeated here for those of you who wish to subscribe to the Rainbow magazine. You can use a free telephone number to order by credit card. It is 1-800-847-0309. Since some of you do not use the BBS yet, this information is included here once again for your information.

A reminder especially for new members of the CCOG, the VFW uses your money donations from our club in it's many charitable activities especially concerning unfortunate hospitalized veterans. It's all going for a good cause. The Livonia VFW also collects used books and magazines to give hospitalized veterans. So if any of you want to contribute such, which you would otherwise discard, give them to the VFW bartender to make some unfortunate veterans happy. Also we owe some thanks to the VFW bartender because he volunteers his time to open the VFW Post for us and serve us drinks. He gets nothing for providing us this service so let him know you appreciate his efforts for us whenever you can.

Expert's Corner by Robert Gault

Here are some opinions I have about the C language. They probably are controversial but I will be glad to field counter arguments.

There are few (very few) advantages to programming in C versus using in a quality high level language like Basic09. In most cases, Basic09 (for example) is faster, easier to learn and use, and has more built in features than C. But there are reasons for using C. For me, the most important one is the ease with which assembly code can be incorporated into a C program. In fact, I DON'T THINK YOU CAN WRITE QUALITY C PROGRAMS UNLESS YOU KNOW ASSEMBLY LANGUAGE!

I am going to present a program in C which will demonstrate some of C's strong and weak points. The program is my version of "Demon" a graphics algorithm presented in the August, 1989, Scientific American in "Computer Recreations."

First we will look at a pure C version which probably can't be made faster by a rewrite; it takes about 1.4 seconds per pass. Then we will look at techniques for hand optimization that remove some of the horrible code produced by the Tandy C compiler. The modest improvement to 1.3 seconds per pass would have been much greater but for the emphasis on speed in the original C code. Finally, I will present the code for an assembly version of the main loop. This caused a boost in speed to about .33 seconds per pass resulting in a watchable program.

Here is the jist of the algorithm. Picture an array, say 100x100 pixels. It is filled with 15 different colors in a random pattern. Then at each clock pulse the old pattern is tested to give a new pattern as follows:

```
repeat until break
for i = 1 to 100
     for j = 1to 100
          for each neighbor pixel(k,l) of (i,j)
               if old(k,l) = old(i,j) + 1
                    then new(i,j) = old(k,l)
               else
                    new(i,j) = old(i,j)

for i = 1 to 100
     for j = 1 to 100
          display new(i,j)
          old(i,j) = new(i,j)
```

The values of the old/new array are just the palette color values. One important point. Both the palette colors and the array itself wrap around in a circle. That is both color and location are modulus values. For example with 15 colors, 15+1=0 and old(100+1,0) means old(0,0).

The above algorithm produces some amazing patterns ending in three possible states. Some patterns lock up and go nowhere. Some eventually become all one color. Some reach a "demon" state. Read the Sci.Am. article or type in the program.

We are going to use a version of the random number generator from last month. We will also again use a VDG screen because then we can directly poke pixels for maximum speed.

```
/* Demon: see Scientific American, Aug. 1989 for info on algorithm
   This program must be run from a VDG screen. */

#include <stdio.h>
#include <time.h>

#define I_SetStt 0x8e
#define I_GetStt 0x8d
#define SS_AScrn 0x8b
#define SS_DScrn 0x8c
#define SS_FScrn 0x8d
#define SS_ScTyp 0x93

struct registers{       /* used by getime */
    char rg_cc,rg_a,rg_b,rg_dp;
    unsigned rg_x,rg_y,rg_u;
    };

direct long next;       /* used by random number */

direct char i,j,k,l,set,*screen,seq1,seq2;
direct char keypress;
char rnd();
direct int n,sig,temp,temp2,temp3;
int interrupt();

/* RGB color scheme */
char palette[]={1,8,9,2,16,18,6,48,4,32,36,34,54,62,63,0};

/* set palette */
char pal_set[]={0x1b,0x31,0,0};

/* default palettes */
char default_pal[]={0x1b,0x30};

struct registers reg;

/* work space for screen data */
direct char *new,*old;  /* pointers to data space */
char data0[64][64],data1[64][64];

main()
{
    unsigned getscreen();
    char ans;
    struct sgtbuf buffer;
    intercept (interrupt);
    seq1=14;
    puts("\xcWelcome to the world of");
    puts("Griffeath's Demons.\n");
    puts("If you want instructions");
    puts("or information press \"y\"");
    puts("\n<hit any key to continue>");
    while (!read(0,&ans,1));
    if(tolower(ans)=='y'){
      puts("\xcGriffeath Demons were described");
      puts("in \"Computer Recreations\"");
      puts("Scientific American, Aug., 1989");
      puts("\nThis version written in C by");
      puts("Robert Gault.");
      puts("\nOnce a 64x64 array is made,");
      puts("resulting patterns depend on");
      puts("the number of colors used.\n");
      puts("<<shift>><<esc>> = restart");
      puts("<<esc>>          = quit\n");
      puts("<hit any key to continue>");
      while (!read(0,&ans,1));
      }


    puts("\xcPlease wait while I initialize\nthe screen with a
random");
    puts("pattern....\n\n");

    /* get random seed for rnd from system time */
    getime(&buffer);
    next=buffer.t_year*buffer.t_month*buffer.t_hour;
    next*=buffer.t_minute*buffer.t_second;

    if(next<0)
      next=-next;

    /* test screen type for VDG screen */
    reg.rg_a=1;
    reg.rg_b=SS_ScTyp;
    if(_os9(I_GetStt,&reg)==0)
    {
       puts("Sorry, this program requires\na VDG screen to run.");
       exit(0);
       }

    /* map high res screen into VDG space */
    screen=getscreen();

    restart:

    keypress=0;

    /* clear graphics screen to palette 15 which will be black */
    for(n=0;n<0x3c00;++n)
      *(screen+n)=0xff;
    /* set palette colors */
    for(i=0;i<16;++i){
      pal_set[2]=i;
      pal_set[3]=palette[i];
      write(1,pal_set,4);
      }

    /* display the graphics screen */
    reg.rg_b=SS_DScrn;
    reg.rg_y=1;
    _os9(I_SetStt,&reg);

    /* set screen pointer to start of graphic image */
    screen+=5064;

    /* setup a random screen pattern */

    seq2=seq1-1;
    old=&data0[0][0];
    for(j=63,n=5040;j>=0;--j,n-=80){
      temp=j<<6;
      for(i=62;i>=0;i-=2){
        if((k=rnd()&0xf)>=seq1)
          k=seq2;
        *(old+temp+i)=k;
        if((l=rnd()&0xf)>=seq1)
          l=seq2;
        *(old+temp+i+1)=l;
        *(screen +(i>>1)+n)=(k<<4)+l;
        }
      if(keypress)    /* break out if shift esc indicated */
        break;
      }

    /* setup for main loop */
    set=1;
    for(;;){

      if(keypress){
        screen-=5064;
        goto restart;
        }

      set=set+1&1;
      if(!set)          /* this test must be included in the main
loop */
        {               /* for the ml. version; see text */
        old=&data0[0][0];
        new=&data1[0][0];}
      else    /* swap data banks */
        {
        old=&data1[0][0];
        new=&data0[0][0];}
/*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
#include mainloop
/*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/

  }
 }

interrupt(sig)
int sig;
{
  char ans;
  int temp;
  if(sig==2)
    sig=0;
  reg.rg_a=1;
  reg.rg_b=SS_DScrn;
  reg.rg_y=0;
  _os9(I_SetStt,&reg);
  write(1,default_pal,2);
  puts("\x0c");
  if(sig==3){
    keypress=1;
    puts("Shall I change the number");
    puts("of colors? <y or n>");
    while (!read(0,&ans,1)¦¦ans==3);
    if(tolower(ans)=='y'){
      temp=0;
      while(temp<5¦¦temp>15){
        printf("\x0cCurrent selection: %d\n",seq1);
        puts("Enter the number of colors:\n");
        printf("   4< n <16   ?=>");
        scanf("%d",&temp);
        }
      seq1=temp;
      }
    puts("\x0cPlease wait....");
    return;
    }

  reg.rg_b=SS_FScrn;
  reg.rg_y=1;
  _os9(I_SetStt,&reg);

  exit(sig);
  }


unsigned getscreen()
{
  reg.rg_a=1;
  reg.rg_b=SS_AScrn;
  reg.rg_x=2;  /* 160x192 16 colors 16K screen */
  if(_os9(I_SetStt,&reg)!=0)
    interrupt(reg.rg_b);
  return(reg.rg_x);
  }

char rnd()          /* returns psuedorandom numbers 0-134456 */
{
 long m=134456, a=8121, c=28411;
 next=(a*next+c)%m;
 return((char)(seq1*next/m));
 }
```

This is a pure C version of the main loop.

```
        for(i=63;i>=0;--i){
          temp=i<<6;
          temp2=(i-1&63)<<6;
          temp3=(i+1&63)<<6;
          for(j=63;j>=0;--j){

            if((l=*(old+temp+j))==seq2)
              k=0;
            else
              k=l+1;

            if(k==*(old+temp+(j-1&63)))
              l=k;
            else if(k==*(old+temp+(j+1&63)))
                   l=k;
                 else if(k==*(old+temp2+j))
                        l=k;
                      else if(k==*(old+temp3+j))
                             l=k;

            *(new+temp+j)=l;
            }
          }

      for(i=63,n=5040;i>=0;--i,n-=80){
        temp=i<<6;
        for(j=62;j>=0;j-=2){
          *(screen+(j>>1)+n)=(*(new+temp+j)<<4)+*(new+temp+j+1);
          }
        }
```

Weak point #1 for C is that the random number code should return a value from 0<=rnd<val. Most of the time it does but Tandy C does not reliably do integer divide math operations; sometimes it rounds off, then rnd equals val.

Problem #2 involves arrays. It would be nice to say new = old and have the entire array be changed without a loop as is done in Basic09. C does not allow either elements of an array nor the entire array to be swapped by readdressing because the name of an array is not a variable. It is also NOT nice to use a loop (too slow) nor to duplicate code using a new variable name (too much code.) I have chosen to declare the arrays char data0[64][64],data1[64][64] and access them via the pointers char *new, *old.

Each time through the main loop, *new and *old get reassigned:

```
 if(!set)
    {
    old=&data0[0][0];
    new=&data1[0][0];}
  else    /* swap data banks */
    {
    old=&data1[0][0];
    new=&data0[0][0];}
```

Then I can use *(old+i<<6+j) as equivalent to data0[i][j]. In fact, the code produced by the C compiler is slightly better with the former version even if it is harder for humans to read. The choice of a 64x64 array is deliberate as it allows the i<<6 rather than i*64. This can't be done unless the array size is a power of two.

Now let's look at the main loop. Everything there has been chosen for speed. We need all the speed we can get because we have at least 64x64x4 operations to perform each pass; and that does not include language overhead!

The loops count down to zero instead of up. The assembly code BGE (branch if greater than or equal to) takes 3 cycles while CMPA #64, BLO (compare A to 64 and branch if less than) takes 5 cycles.

Syntax such as:

```
        if((l=*(old+temp+j))==seq2)
```

which says let L equal old[i][j] then if L equals seq2, is faster than seperate statements although harder to read. It is faster because C won't STA L,y and then LDA L,y but only do the former. After all why load a register if the value is already there? Calculations based on i are done before entering the j loop, ie. temp=i<<6.

Our next group of problems are the worst ones. When C decodes i<<6 which means shift the value of i to the left 6 times, ie. i x 2**6, the code produced is:
PSHS D, LDB #6, LBSR CCASL where:

```
* CCASL means C language char arithmetic shift left

 ccasl tstb
       beq exit
 ccasl2 asl 3,s
       rol 2,s
       decb
       bne ccasl2
```

```
     exit  ldd 2,s
           pshs d
           ldd 2,s
           std 4,s
           ldd ,s
           leas 4,s
           rts
```

which is truly ugly code, but that is the price you pay for flexibility.

I compiled the source code with CC1 -ac demon.c which left the output with comments at the assembler source level. Then I removed all pshs d, ldd #6 and changed lbsr ccasl to lbsr casl. Then I wrote my own version as:

```
     casl lslb
          rola
          lslb
          rola
          lslb
          rola
          lslb
          rola
          lslb
          rola
          lslb
          rola
          rts
```

The above and a few more odds and ends at the assembly level (ex. LDB SEX CLRA to LDB CLRA) gave a some increase in speed, but not enough.

What techniques should be used if the MAINLOOP is written in pure assembly? Well get rid of as much y relative addressing, unnecessary stacking, and unnecessary subroutine calls as possible for starters. Also get rid of integer operations when byte operations will do and use as many direct page variables as possible for speed. Some code is duplicated, sacrificing compactness for speed. Usually this code would be made into a subroutine.

The largest speed increase seemed to result from converting the screen update routine to assembly. With the C version, you can see the color changes scroll. Changes appear instantaneous in the ml. version. I attribute this to the use of multiple register variables; Tandy C allows only one per function.

The code for the main loop is shown below. It is instructive to compare the code generated by the C compiler against the version below.

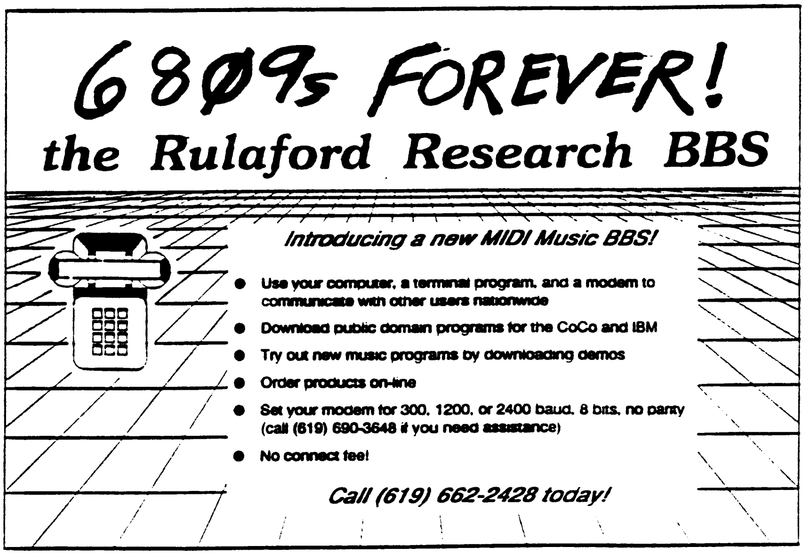
```
#asm
      tst <set                      * see note above for why this
test was not *
```

```
     bne flop                    * left as part of the C code *
flip leax data0,y
     stx <old
     leau data1,y
     stu <new
     bra process
flop leax data1,y
     stx <old
     leau data0,y
     stu <new

process pshs y   * save the C data index *
     ldy #64  * number of rows *
ml1 ldb #64 * number of columns *
ml2 lda ,x
     cmpa <seq2   * largest palette color; 15-black is background
*
     blo ml211
     lda #$ff
ml211 inca  * set for next higher color *
     cmpb #64   * first column ? *
     beq ml21
     cmpa -1,x
     beq store
     bra ml212
ml21 cmpa 63,x
     beq store
ml212 cmpb #1   * last column ? *
     beq ml22
     cmpa 1,x
     beq store
     bra ml222
ml22 cmpa -63,x
     beq store
ml222 cmpy #64   * first row ? *
     bne ml3
     cmpa 4032,x
     beq store
     bra ml4
ml3 cmpa -64,x
     beq store
ml4 cmpy #1     * last row ? *
     bne ml5
     cmpa -4032,x
     beq store
     bra ml52
ml5 cmpa 64,x
     beq store
ml52 lda ,x     * no change; get old data *
store sta ,u+   * update data array *
     leax 1,x
ml6 decb        * next column *
     bne ml2
     leay -1,y      * next row *
     bne ml1

endml ldu <screen   * display new data on screen *
     ldx <new
     ldy #64
show1 ldb #32
show2 lda ,x+
     lsla           * two pixels per byte *
     lsla
     lsla
     lsla
     ora ,x+
     sta ,u+
     decb           * next column *
     bne show2
     leau 48,u
     leay -1,y      * next row *
     bne show1
     puls y
#endasm
```

One unexpected idiosyncrasy of the Tandy C compiler showed up during this project. If the main loops are written as separate files and added to the mainline using the #include "mainloop" call, there had better not be an IF statement preceding the #include for assembly versions. C causes a FALSE IF statement to branch to the first next C statement present. #Asm does not qualify as a C statement for PASS1 nor PASS2 so your entire assembly source code will be skipped over during False IF conditions during run time.

That is why the assembly version moved the if(set==0) into the assembly code.

Club members with OS-9 but not the C language can get a compiled copy of the ml. version from the BBS and eventually our library.



C.C.O.G.

COLOR COMPUTER OWNERS GROUP

Oldest COCO Club in Metro Detroit
Meetings the ~~third~~ FOURTH Tuesday of each month
7:00 PM – 9:30 PM
LIVONIA VFW – 7 MILE & MIDDLEBELT
SIDE DOOR – "THE FOXHOLE"

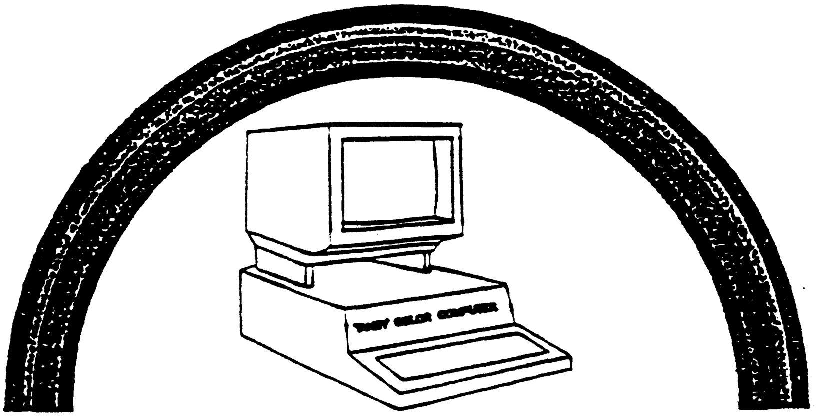
Join other COCO owners for Hardware and Software demos, Public domain Library, Discounts, and general knowledge exchange.

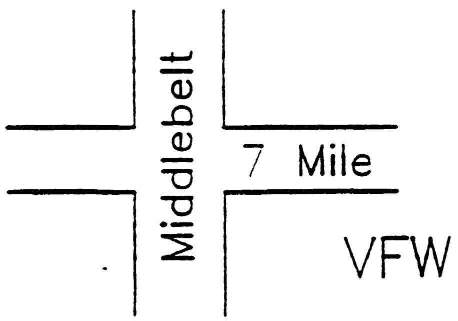
1991 Meetings

Jan	***15***	***Jul***	~~***16***~~ 23
Feb	***19***	***Aug***	~~***20***~~ 27
Mar	***19***	***Sep***	~~***17***~~ 24
Apr	***16***	***Oct***	~~***15***~~ 22
May	~~***21***~~ 28	***Nov***	~~***19***~~ 26
Jun	~~***18***~~ 25	***Dec***	~~***17***~~ ?

MEETINGS INFORMAL AND OPEN TO THE PUBLIC

Contact: Bernie @ (313) 283-2474
or
BBS @ (313) 292-4713





COLOR COMPUTER OWNERS GROUP
c/o BERNIE PATTON
388 EMMONS BLVD.
WYANDOTTE, MI 48192