

PRICE  
\$5.00/COPY

# FORUM

INTERNATIONAL

VOL.3 NO.1  
FINAL EDITION  
1983

A MAGAZINE FOR OWNERS OF  
COMPUCOLOR/INTECOLOR COMPUTERS

<u>ARTICLE</u>	<u>PAGE NO.</u>
Editor's Corner .....	1
Spelling Correction For The Compucolor? .....	3
Serial Line Print Buffer Needed! .....	3
Disk Alignment For The Compucolor .....	3
Basic Compilers For The Compucolor .....	9
Instructions For Using "FASBAS" .....	11
ROMPACK Development For The Compucolor .....	26
First Aid For Compucolor Disk Drives .....	28
Assembly Language Subroutines (Part 7) .....	29
Membership Listing (Past and Present) .....	34

103 SANDYHOOK SQUARE, SCARBOROUGH, ONTARIO, CANADA M1W 3N6

"SOURCE" - TCM101 (416) 499-7177

EDITOR'S CORNER

As you have probably already noticed from the front cover, this issue of FORUM will be the last produced by us. This issue was supposed to be the March/April edition. Once again, we had to wait until enough revenue arrived in order to print it. Membership has dropped from over 325 users down to 182, with less than 20 users subscribing from Canada. It seems that the majority of CompuColor/Intecolor owners reside in either Australia or the United States.

If it wasn't for the CUVIC USERS GROUP of VICTORIA, AUSTRALIA, we would probably be still waiting to get this issue out. Their generosity in unselfishly donating \$200 (US) turned out to be the extra monetary requirement to pay off the printer one last time. I would personally like to thank Mr. Keith Ochiltree and CUVIC for that donation and apologize that it wasn't able to keep FORUM alive on an on going basis. Perhaps the members of CUVIC would like to adopt FORUM INTERNATIONAL as an official part of their club activities. As most of you are aware, I was the founding user of FORUM as well as being its Editor and chief cook and bottle washer. As a result, I think that I can speak on behalf of all members that CUVIC's interest and financial donation more than adequately gives their group the rights to perpetuate FORUM if they so desire. So Keith, please consider this official permission for CUVIC to take over and use FORUM and any of its past published columns and articles as your group sees fit.

I guess the technical terminology for our financial state is bankruptcy, however, the only obligations outstanding are to the members who have paid their \$20.00 (US) and haven't received a full six issues as yet. To those members, I would like to apologize that we weren't able to meet that full obligation. There is simply not enough funds left to mail back a part of the subscription even on a pro-rated basis. The small balance of our bank account that is left will be sent to CUVIC to assist them in recovering at least part of their substantial donation.

As far as the club library goes, I have spoken to Mr. Dave Grice who is willing to continue operating the library on a completely independent basis. Those members in Australia may contact CUVIC for library requirements as they were sent in January a complete copy of all library programs. Any U.S. member who would be willing to act as a U.S. distribution point for the library should contact Dave for further information. His address as well as the CUVIC mailing address are as follows:

Mr. Dave Grice  
R.R. # 1, Hastings  
Ontario, Canada  
K0L 1Y0

CUVIC (CompuColor Intecolor Users of Victoria)  
Box 420, Camberwell,  
Victoria, 3124  
Australia

In order to assist in keeping some form of contact open for FORUM members, we have included in this issue a complete listing of all users who presently subscribe to FORUM or who at one time in the past maintained a membership which has since expired. This listing also includes their known equipment configuration. You may find other CompuColor users who live close to you or perhaps want to develop a pen-pal relationship with users in other countries. You will notice that we have deliberately left the user's phone number off the listing as we feel that this type of information should only be given out by the individual member himself. We did leave you enough room to fill it in on the listing once you obtain it. We hope that with the demise of FORUM that this will help members feel less alone and enable them to develop individual contacts with users of similar interests.

On a personal note, it is with deep regret that I see FORUM folding after two years of efforts. I am quite sure that if FORUM had been based upon another computer produced by a company that had the slightest clue about marketing, we wouldn't be in the present state. However, now is not the time to get bitter about ISC or its personnel. That is history and we hope that we are able to all learn from history and not repeat the same mistakes. I would like to thank all members who contributed to FORUM in the form of articles, regular columns and product reviews. There are simply too many to name individually here, however, I'm quite sure that you all know their names.

I would like to close by saying that I have enjoyed tremendously the producing and editing of FORUM, but mostly, I have enjoyed the people whom I have met and exchanged a friendship with that I will always cherish and remember in the years to come. To all I wish best of luck on the future and perhaps we may meet again through some other such vehicle as FORUM.

Sincerely,



D.R. PEEL  
EDITOR

SPELLING CORRECTION FOR THE COMPUCOLOR??  
=====

Does anyone know whether there is a spelling correction and/or a proofreading program that will run on a Compucolor? If so, please contact:

MR. JOHN TOWLER  
185 PINEGROVE CRESCENT  
WATERLOO, ONTARIO  
CANADA, N2L 4V2

\*\*\*\*\*

SERIAL LINE PRINT BUFFER NEEDED!!!  
=====

Are there any technical wizards out there that can/will design a cheap serial line buffer with handshaking? An 8K buffer would be great, 16K of buffer would be outstanding? Also how about in kit form. Please contact:

Mr. M.F. Pezok  
1382 Ignacio Blvd.  
Novato, California  
U.S.A. 94947

\*\*\*\*\*

DISK ALIGNMENT FOR THE COMPUCOLOR - By Mr. Tom Devlin  
=====

The following several pages include a program, text and diagram on handling disk drive alignment for the Compucolor computer. For those of you who do not feel up to tackling this project, one of the members of FORUM who is also a computer technician on ISC products will completely re-align a drive for \$35.00 (US). His name is:

Mr. Gary Sipple  
222750 Golfview  
Southfield, Michigan  
U.S.A. 48034  
(313) 356-5140

```

00010 REM      "STEP"  PROGRAM FOR DISK ALIGNMENT
00020 REM              BY TOM DEVLIN 1983
00030 REM              FOR V6.78 ONLY
00040
00050 PLOT 12,6,2
00060 INPUT "DRIVE 0 OR 1?";CD
00070 IF CD<0 OR CD>1 THEN 50
00080 CD=CD*16+16
00090 XX=0:WG=0
00100 PLOT 12
00110 INPUT "TRACK NUMBER? ";TR
00120 IF TR<0 OR TR>40 THEN 110
00130 PLOT 12
00140 PRINT "<SHIFT> = STEP OUT, STEP IN"
00150 PRINT "<CONTROL> = STEP IN, STEP OUT"
00160 PRINT "<REPEAT> = CHANGE TRACK":PRINT
00170 PRINT "TRACK ="TR
00180 OUT 8,0:REM DISABLE INTERUPTS
00190 OUT 7,128+CD AND 247:REM START THE MOTOR
00200 FOR I=1 TO 500:NEXT:REM WAIT A WHILE BEFORE MOVING HEAD
00210 IF XX=0 THEN GOSUB 470:REM HOME THE HEAD FIRST TIME
00220 GOSUB 400:REM STEP TO DESIRED TRACK
00230 GOSUB 290
00240 OUT 4,9:REM RESET TMS 5501
00250 OUT 11,255:REM START KEYBOARD TIMER
00260 OUT 8,255:REM ENABLE INTERUPTS
00270 GOTO 110:REM GET NEW TRACK NUMBER
00280
00290 K=INP (1) OR 128:REM SCREEN OUT CAPS LOCK
00300 IF K=255 THEN 290:REM WAIT FOR CONTROL, REPEAT, OR SHIFT
00310 IF K=191 THEN RETURN:REM REPEAT
00320 TT=TR:WG=4:REM WG=MAX # OF STEPS. (MAY BE LESS)
00330 IF K=239 THEN TR=0:GOSUB 400
00340 IF K=223 THEN TR=40:GOSUB 400
00350 TR=TT:WG=0:GOSUB 400
00360 WAIT 1,127,127:REM WAIT FOR NO KEY DOWN
00370 GOTO 290
00380
00390 REM MOVE TO TRACK (TR) FROM TRACK (CT)
00400 IF TR=CT THEN RETURN
00410 IF TR>CT THEN CT=CT+1:PH=PH*2:IF PH>4 THEN PH=1
00420 IF TR<CT THEN CT=CT-1:PH=PH/2:IF PH<1 THEN PH=4
00430 OUT 7,128+CD+PH AND 247
00440 IF WG<>0 THEN WG=WG-1:IF WG=0 THEN RETURN
00450 GOTO 400
00460
00470 REM HOME HEAD TO TRACK 0
00480 TT=TR:TR=0:CT=42
00490 GOSUB 400
00500 PH=4:REM HEAD MUST HOME ON PHASE 4
00510 OUT 7,128+CD+PH AND 247
00520 CT=0:XX=1:TR=TT
00530 RETURN

```

One of the more troublesome features of the CompuColor has been the disk drives, especially the internal one. This is not due to any basic design flaw, the drive mechanism itself is the widely used Siemens/Wango model 82 and the CompuColor designed disk controller, while inexpensive, is fundamentally sound. The main problem seems to be a general lack of knowledge regarding proper set-up and adjustment of the drives, a lack that seems to stretch all the way back to the factory.

While this article isn't overly technical it does presume some basic familiarity with an oscilloscope. Adrian Donkin has published an innovative 'no tools' alignment method in CompUKolour that was reprinted in the April-July 1982 issue of Data Chip. His article goes into far more detail on some of the steps covered and is recommended reading if this is your first foray into this area.

The two most common problems with the drives are the speed adjustment and noise pickup from the deflection coils and the Analog board. The speed is easy to set, simply remove the back and adjust the control on the rear of the disk drive circuit board while viewing the strobe disk glued to the drive spindle under fluorescent light with the drive running. (Hit AUTO with no disk in the drive) If you have 60 cycle per second A.C. power (U.S. and Canada) adjust the control until the outer band of stripes is stationary. If you have 50 cps power use the inner set. COM-TRONICS has a speed adjustment program that uses a screen display to indicate the speed. This allows checking the speed without removing the back.

The noise problem is more difficult. The best way to solve it is to move the internal drive to an external cabinet. Intelligent Computer Systems has a conversion kit for this available, write them for details. If you want to keep CDO: inside (as I have) you may have to play with the shielding. If you look at the drive belt side of the drive you will see a small piece of foil partially covering a rectangular hole in the frame. This foil is MU Metal (a magnetic shielding foil) it will probably be bent out so that it doesn't lay flat against the frame. This, believe it or not, is your adjustment! Move the head out to track 40 with the drive mounted in its normal position (but not running) and monitor TP1 on the rear of the drive circuit board with a scope. Note the amount of noise present. Use a wooden dowel or piece of plastic (a metal tool would mess up the reading) to bend the foil around. You will find that you can 'tune' the noise level up and down, you want to find the position that gives the least amount of noise with the best symmetry above and below the D.C. reference level.

Most other problems can be solved by a general clean up and adjustment. It's easiest if the drive is partially disassembled. (the 4 phase drives are tough to put back together again, I would not recommend tearing one of these units apart unless you know what you are doing.)

Use a sharp, hard, pencil to make an index mark on the stepper motor and another corresponding one on the disk drive frame. You will need these marks to reassemble the drive properly. You may have to make a second set of marks for the alignment procedure later on so don't use a sharp tool as a scribe.

Mark the positions of all connectors to the printed circuit board and remove it from the frame. Disconnect the connector that supplies power to the stepper motor and remove the two hex-head screws and washers that hold the stepper motor to the drive frame. You can now unscrew the stepper motor and lead screw assembly out the rear of the drive. The Read/Write head assembly and the anti-backlash nut and spring will be free when you have done this so don't let them fall on the floor. Use an old toothbrush to loosen any 'dirt' imbedded in the lead screw grooves and carefully polish it clean with a paper towel. Then polish it again with some waxed paper to lubricate it.

Inspect the R/W head for signs of wear and take a good look at the felt pressure pad that holds the disk against the R/W head, if it is loose it can render the drive unuseable. If the felt pad is dirty you can use a stiff brush (not your fingers) to brush it clean, one of the small brushes sold at audio stores for stylus cleaning is ideal. New pads are available from Heathkit stores or Radio Shack service centers.

When you have everything looking shipshape you can put it all back together again. The only thing to worry about is the gap between the anti-backlash nut and the R/W head. Siemens calls for a .01/.03 inch gap, I have found that the drives are far quieter and more reliable if this gap is increased to about 1/16 inch. Run the lead screw through the Anti-Backlash nut until the screw sticks out the end slightly and then into the R/W head. If the gap doesn't look right then back the lead screw out of the R/W head, rotate the lead screw slightly, and try again. The proper position will be obvious as it is the only one that will be anywhere close to 1/16 inch.

The next step is to adjust the stepper motor. This will insure that your R/W head is indexing to the proper position for each track. You will need a Dysan 282 alignment disk (\$25.00) for this step. The small booklet that comes with the disk makes reference to all sorts of signals and switches that we don't have and all of the track numbers listed will be off by one. The only tracks of interest to us are track 17 (for stepper motor alignment) and track 35. Track 35 is supposed to be used to adjust the little felt pad for the maximum signal level. I have rotated the pad all over the place to no effect, I think this adjustment can be safely ignored. Track 35 can be used to check the output level from the R/W head and input amplifier, it should be about 3 volts peak to peak measured at TP1.

NOTE: The adjustments that follow apply to both the 3 and 4 phase drives used by Compucolor but the "STEP" program works only on the three phase drives used in the V6.78 systems. Compucolor has published a BASIC program "DSKALN" for the 4 phase drives.

1. Connect the scope to TP1, low side to ground. load and run the 'Step' program. Insert the Dysan 282 alignment disk into the drive (label to the door side) and index to track 17. If, by some chance, you see a pattern on the scope that closely resembles the Ideal Alignment drawing don't go any further, you are all set! Everybody else procede to Step 2.

2. Slightly loosen the two screws that hold the stepper motor in place. Rotate the stepper motor a very slight amount in either direction while watching the pattern on the scope, if things get worse rotate it the other way. When you get close to the proper point snug the screws down slightly.

3 Step in-out. This will probably shift the alignment so go back to step 2 and try it again. It is important to tighten the screws and step in-out after every adjustment until you find the right point. When you get to the "very close" point you may want to add a second pair of index marks. Since you are going to be moving the stepper motor in progressively smaller amounts make this new set as narrow and accurate as possible, the rotation in the final stages may well be less than the width of a pencil line. (If you rotate the stepper motor a slight amount and see no change in the scope pattern you have probably gone through the proper setting, back up about 1/2 the distance you moved it and try again.)

4. Now loosen the two screws holding the track 0/40 stop. Step the head to track 0 and look at the scope screen to make sure that no signal is present. Adjust the stop until there is just a slight (about the thickness of 1 page of FORUM) amount of clearance between the stop and the boss cast into the drive frame when the head is on track 0 and retighten the screws. The idea is to leave a little room to be sure that the stepper motor is able to pull the head all the way to home without being held back by the stop. Step out-in a couple of times to make sure. Because Compucolor track 0 is Siemens/Wango track -1 the stop probably can not be adjusted far enough to give the required gap. If this is the case, and it usually is, remove the stop and file the edge to give enough clearance. Be sure that you clean the stop before you reinstall it, a speck of aluminum dust could ruin a disk in a hurry.

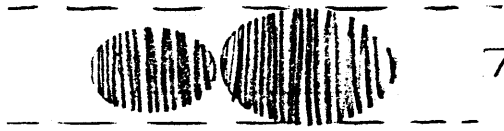
Make sure that the Carriage Guide Shaft is seated firmly against the side of the drive frame, if it is not it can cause the R/W head to bind up.

Re-assemble the drive. Make sure that all of the screws you loosened are snugged down properly and mount the drive back in the box.





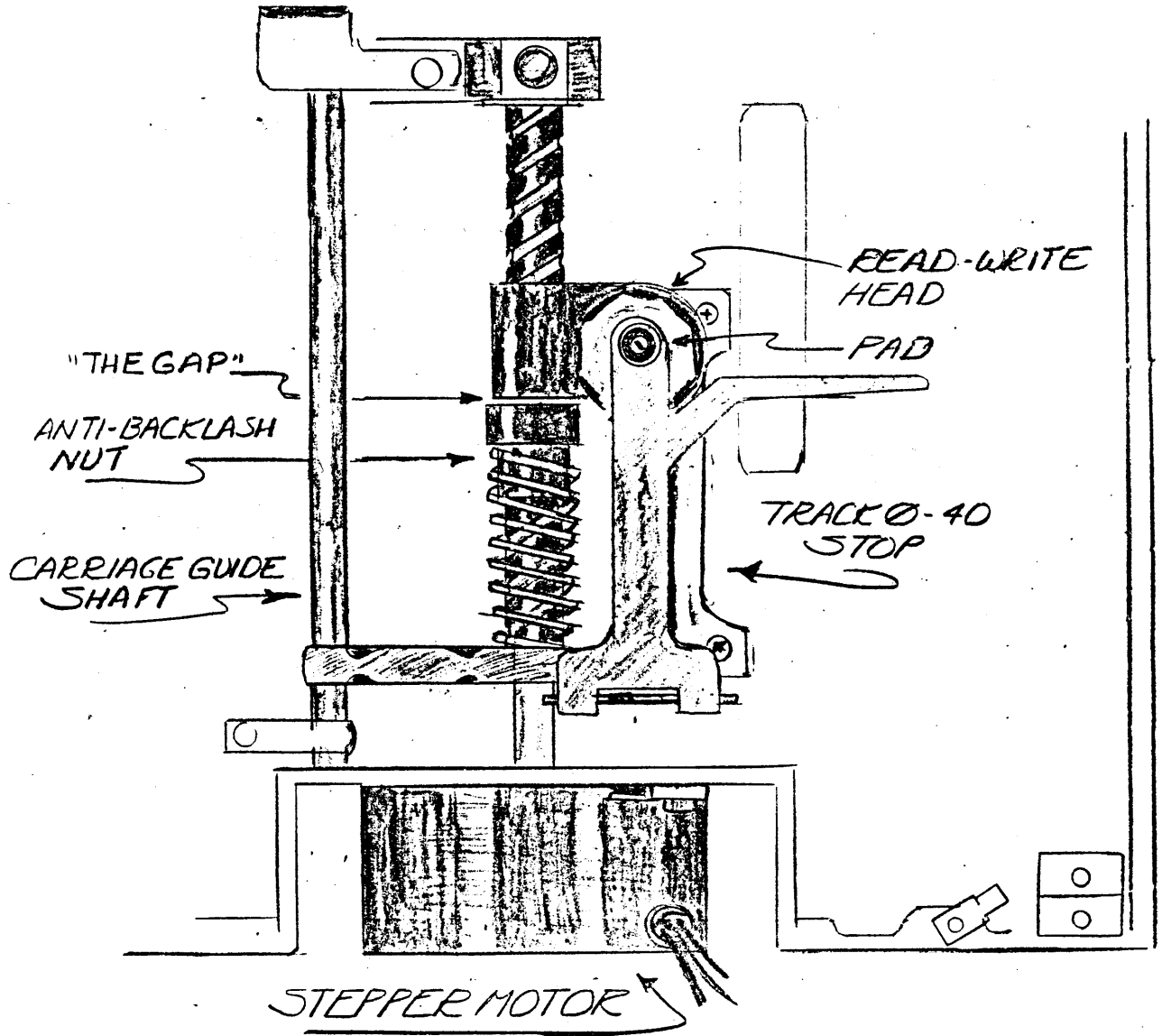
TRACK 17 - IDEAL ALIGNMENT



TRACK 17 - SOMEWHAT OFF



TRACK 17 - ONE FULL TRACK OFF



Wouldn't you know it. Just at the time when the CompuColor seems to be winding down, not one but two Basic Compilers appear on the market. First, we're informed by Mr. Keith Ochiltree of CUVIC, Victoria, Australia, that a Basic Compiler was about to be released for the CompuColor over there. Apparently it will be selling for about \$100.00 Australian and be on an "uncopiable" disk to protect against piracy. We don't have any further information on this product, so perhaps if you're interested, contact CUVIC at the address listed in the EDITOR'S CORNER for more data.

The second compiler we do have a lot of information on including the compiler itself. Not only is the product itself superb, but the pricing will blow your socks off. It is called FASBAS and we have included the full instruction manual in this issue of FORUM on the following pages. We did this rather than trying to go into long explanations in explaining how to use it.

The following letter was received from Mr. Peter Hiner, the author of FASBAS, and I am reprinting it here because I think Peter better explains how he wishes to handle the distribution than I could.

" Dear Mr. Peel,

Thank you for publishing in FORUM the letter from Dave Thomas, describing my Basic Compiler.

I am taking the liberty of sending you the program and documentation, in the hope that you will be pleased with the performance of the compiler and that you will review it in FORUM and distribute it within your user group.

I am asking a price of \$25 (US) for the compiler and documentation ordered direct from me, but I would prefer to reach agreement with user groups, for them to make and distribute their own copies. In this case I would ask user groups to send me a fee of \$15 (US) per copy made.

I am well aware of the impossibility of enforcing payment of fees. This is why I am offering the compiler at such a low price (\$15 compared with a typical market price of \$150-\$300). I hope that people will be fair to me in return.

I should like to comment on the wide readership of FORUM. I have received two enquiries from the USA and two from Australia, following the publication of Dave Thomas's letter, and that did not give information on price etc. Clearly FORUM moves far and fast.

I look forward to hearing your comments on the compiler and on my offer.

Yours Sincerely,  
Peter Hiner "

Well, I can only say that I'm extremely impressed with what Peter has been able to accomplish. I wouldn't have believed that a Basic Compiler could be written for the CompuColor that would run in 32K let alone on a 16K machine.

The price is incredible value for what the program is capable of doing. I would encourage any user who orders it directly from Peter to request NO documentation, since the entire manual is published here in FORUM. The reason for this is that it cost Peter over \$5.00 in postage to mail the compiler to me plus the disk. Give him a break and leave him something extra for this fantastic programming effort.

The following pages are the actual documentation for FASBAS. They will give you all the information that you could possibly require about the program. FASBAS needs the following requirements to run and may be ordered directly from Peter at the address listed below.

FASBAS  
=====

MINIMUM REQUIRMENTS: 16K MEMORY  
SINGLE DISK DRIVE  
V6.78 or V8.79 BASIC

(C)COPYRIGHT 1982

MR. PETER HINER  
11, PENNY CROFT  
HARPENDEN  
HERTS, AL5 2PD  
ENGLAND

TELEPHONE 05827 64872

## FASBAS

### 1. INTRODUCTION

FASBAS (Fast Basic) is a program which compiles Basic programs to make them run faster. It produces output to disk in a modified form of Assembly Language, for subsequent assembly into machine code using the FBASM assembler program.

After compilation and assembly, Basic programs will run up to 5 times faster, though the increase in speed will vary considerably, depending on the contents of the original Basic program. FASBAS interprets the Basic program and generates addresses for fast access to variables, subroutines, etc., but it uses the same ROM subroutines as Basic for trigonometric and other functions. So a minimum speed increase of 50% is almost certain, but further speed increases will depend on the proportion of time used by the Basic program for the fundamental jobs of interpretation and searching. Many articles have been written about how to speed up Basic by putting common subroutines at the beginning, by declaring common variables early in the program, and by using variables to replace constants. These particular techniques will speed up a Basic program but will not affect the speed of a compiled program. On the other hand, normal good practices (such as minimising activities within a FOR...NEXT loop) will speed up both.

On the subject of speed, it is interesting to consider a real example. In the September, 1981 issue of Data Chip, Ken Jenkins published a Basic program for comparing three routines for sorting variables, using the internal clock to measure time taken. I compiled this program and the table shows time taken before and after compilation. In every case the number of items sorted was 100.

Type of sort routine	Random data		Already in order		Reverse order	
	Basic	Compiled	Basic	Compiled	Basic	Compiled
Bubblesort	253	70	2	1	229	83
Shellsort	58	14	11	3	28	6
Quicksort	21	5	12	3	14	3

Times are in seconds, and exclude fractions of a second, so comparisons for the shorter times are not accurate. This program is quite a favourable test for FASBAS, as it makes heavy use of single dimension numerical arrays, which are given special treatment by FASBAS. A program using two-dimensional arrays would not have shown such good results.

A demonstration program SNAKE is included on the disk for practice in the FASBAS operating procedures. In Basic the SNAKE is so slow that you should beat it every time, but after compilation the increase in speed makes it a much more challenging game.

The price that has to be paid for speed is program size. The final compiled version will be about twice the size of the original Basic program, with a minimum size of about 750 bytes (containing a run time library of routines). This is not likely to be a practical problem, because there is a more severe limitation on the maximum size of Basic program that can be compiled. FASBAS generates a tokenised Assembly language file (SRC file) as an intermediate stage, and for a large program the SRC file will be 4 or 5 times the size of the Basic program. Therefore, due to the limited storage capacity of a disk, the maximum size of Basic program (excluding REM statements) that can be compiled is effectively about 8 Kbytes with one disk drive (or 12 K with two drives).

This problem can sometimes be overcome by splitting out part of the Basic program (e.g. instructions for a game) and then chaining the two parts of the program.

A lot of care has been taken to ensure that FASBAS is capable of compiling every possible combination of instructions that programmers might use in writing Basic programs. There are a few limitations imposed (which are detailed later), but within these limits there should be no problems. However, it would be a brave man who declared his program to be free of bugs, particularly a program of this complexity. So please let me know if you hit problems.

## 2. DEMONSTRATION PROGRAM

The FASBAS disk contains a Basic game SNAKE.

I recommend that you try compiling this first, to see what FASBAS can do to speed things up and to gain confidence in how compilation ought to work before you try something more complicated.

1. Put the FASBAS disk in the default drive (say drive 0)
2. Enter Escape D (for FCS mode) and RUN FASBAS
3. In response to 'Select Version', just hit RETURN
4. Leave the FASBAS disk in the drive
5. In response to FILESPEC >, enter SNAKE
6. When compilation is complete, control returns to FCS mode
7. Enter RUN FBASM
8. In response to the prompt, enter ASM SNAKE to 0:
9. The last address indicated by the assembler should be 98CB
10. Enter LOAD SNAKE
11. Enter SAVE SNAKE. PRG 82A0 - 98CB

Now you can test the difference between the Basic and compiled versions.

I suggest you try the Basic version first (you will need to enter Escape W to initialise Basic).

When you have beaten the snake, enter Escape D RUN SNAKE, and see if you can still win.

### 3. INSTRUCTIONS FOR USE OF FASBAS

#### 3.1 Debugging Basic

First of all, debug your Basic program thoroughly, so that it runs consistently, without any error messages. Compiled programs save time by omitting many of the error checks provided by Basic, so programs with errors in them may crash in a cloud of dots and lines, produce a meaningless error message or simply disappear into thin air.

#### 3.2 Check for unacceptable Basic statements

FASBAS will accept nearly all the Basic instruction set, but there are a few exceptions, which are detailed later. You may prefer to let FASBAS flag up unacceptable statements, rather than scan through the Basic program yourself. However, I do recommend that you check for premature exit from FOR...NEXT loops, as this will not cause FASBAS to indicate an error but may cause unexpected results when you run the compiled program. This is the most common problem encountered (see section 5.1 for more details).

#### 3.3 Running FASBAS

Load disk containing FASBAS into the default drive, enter FCS mode (Escape D) and key in RUN FASBAS. (Note that FASBAS will automatically configure itself to run on V6.78 or V8.79 machines.)

FASBAS loads to address 82A0 and you can not re-run it using Escape T etc. FASBAS uses overlay techniques to reduce memory requirements, and you must always start from scratch.

#### 3.4 Select target machine

It is not feasible to generate compiled programs that will automatically run on either V6.78 or V8.79 machines, as that would increase the size of the final program considerably. However, FASBAS allows you to select the version on which the compiled program is to run.

To compile for the same version of machine as you are using, enter 0 (or Return).

To compile for the other version of machine, enter 1.

FASBAS will now load the appropriate table of addresses and routines.

### 3.5 Load disk containing Basic program

With a single disk drive, there are no options, and FASBAS will output the Assembly language file to the disk containing the Basic program (so make sure there is plenty of space).

With more than one disk drive you can select the drives for Basic input and Assembly language output (see next section).

### 3.6 Response to FILE SPEC >

The minimum response required is the name of the Basic program to be compiled (e.g. TEST). This will cause the Basic program to be read from the disk in the default drive and the compiled version to be written onto the same disk (as an SRC file with the same name as the Basic program).

The maximum response allows input and output drives, program names and versions to be specified. For example:

```
CD0 : TEST. BAS; 01   TO   CD1 : DOODLE. SRC; 02
```

Any legitimate variation or subset of the above can be entered (but if you insist on making the output file type anything other than SRC, remember to specify file type during assembly).

### 3.7 Compilation

FASBAS makes two passes through the Basic program. The first pass is quite fast since FASBAS is only looking for DATA and DIM statements. During the second pass the compiled program is generated and written to disk.

The Basic program line numbers are displayed, so that you can see progress. If FASBAS detects any Basic statements that it can not accept, it will display an error message against the offending line number, and will continue compiling at the next line.

At the end, FASBAS exits to FCS mode.



### 3.8 Correction of errors

Inspect the Basic program line number(s) indicated as containing an error, and check for functions not accepted by FASBAS. As well as obvious items such as DEF FNA(X), FASBAS will detect undimensioned numerical arrays with more than one dimension, and unsupported FILE statements (e.g. FILE "T").

Some of the more subtle points (such as RESTORE to a line number not containing data) will not cause FASBAS to generate an error message. In the above example of RESTORE, an error will be indicated during Assembly, since there will be a pointer to a non-existent data reference (e.g. (Shift)XHD200 means LXI H,D200 and refers to data supposedly stored in line 200). Some errors may not cause any indication at all until the compiled program is run. For example, jumping out of FOR...NEXT loops prematurely may even work satisfactorily when the compiled program is run (but it will probably cause the program to give peculiar results or to crash).

A particular point to watch for is the use of POKE to load machine code programs in what you believe to be spare RAM (for subsequent use by the CALL function). Since the compiled version of a program will be much larger than the original Basic program, the addresses being POKED may no longer be spare. The compiled program uses memory space in a very similar way to Basic, that is to say:-

82A0	Start of compiled program (which includes space for storing variables and single dimension numerical arrays).
VVVV	{ End of compiled program Start of storage space for string pointers, multi-dimension numerical arrays and string array pointers
WWWW	{ End of storage space Start of spare space
XXXX	{ End of spare space Start of stack space
YYYY	{ End of stack space Start of string manipulation space (50 bytes unless altered by a CLEAR statement).
ZZZZ	{ End of string manipulation space End of available memory (unless memory space reserved during Basic initialisation)

As you can see, the start and end of spare space are not very well defined. The start depends on size of program and storage space for strings and arrays, and the end depends on both the amount of space allocated for string manipulation and how big the stack grows during program run. Note that the stack grows from end towards start, and although it should decrease as often as it grows, so that it finishes back at the end point, it will reach peak size during deeply nested subroutines.

The above remarks apply equally to Basic and compiled programs. The safe way to obtain spare space is to reserve it after the end of string manipulation space. This can be done by initialising Basic (Escape W) and, when asked 'Maximum RAM available?' entering a number which is n bytes less than the actual memory available. That is not practicable, since you would have to remember to do it every time you ran the program.

The same result can be obtained by POKE and CLEAR instructions at the start of the Basic program. Here is an example for 16K and 32K machines (values for (a) and (b) can be found from the table):

O POKE 32940, (a) : POKE 32941, (b) : CLEAR 50 (or more)

Number of bytes to be reserved	16K Machine		32K Machine	
	(a)	(b)	(a)	(b)
128	127	191	127	255
256	255	190	255	254
512	255	189	255	253

This is too complicated for many people and so the usual practice is to use spare space after the Basic program, variables and arrays, allowing generous margins for errors in estimation, and then to run the program and see if it works. The same procedure can be applied to compiled programs, allowing twice as much space as for the original Basic program.

### 3.9 Editing

FASBAS produces a modified form of assembly language output (using single byte tokens to save space). The SRC file, which is generated by FASBAS, can be edited using the Com-tronics CTE program (version 2.23 on disk). For some unknown reason the ISC editor (EDT) loses the last block of the SRC file. No doubt this is caused by a deficiency in the way FASBAS writes the SRC file to disk, but I have not been able to discover the reason. The CTE editor increases the size of the SRC file by automatically inserting carriage return characters (which have been omitted by FASBAS to save space), so you may not be able to edit long programs.

Although there is no normal reason for editing the SRC file the identities of tokens are given in the following table. In each case the token is a letter together with shift key operated. Note also that FBASM assumes that all numbers are hexadecimal and the H postscript is always omitted, as are spaces and commas.

Shifted Letter	Hex Value	Significance
A	61	LDA
B	62	STA
C	63	CALL
D	64	DCX
E	65	DAD
F	66	CPI
G	67	XCHG
H	68	PUSH
I	69	INX
J	6A	JMP
K	6B	ANA
L	6C	LHLD
M	6D	MOV
N	6E	DB
O	6F	DS
P	70	POP
Q	71	XRA
R	72	RET
S	73	SHLD
T	74	XTHL
U	75	INR
V	76	MVI
W	77	DCR
X	78	LXI
Y	79	JN <del>Z</del>
Z	7A	J <del>Z</del>

### 3.10 Assembly

The FBASM assembler program is derived from the ISC Assembler and is used in the same way as the ISC Assembler is used for programs in normal assembly language. The size of assembled program will be between a third and a half of the size of the SRC file (in other words, about twice the size of the original Basic program). With a large program and a single disk drive you may need to delete the original Basic program from disk to leave enough space for assembly.

In response to the prompt (>), enter ASM.... TO 0: (or TO 1:). Entering ASM.... only will cause assembly without writing to disk.

If FBASM finds an error it will stop (hit return to continue assembly). You should assume in the first instance that errors result from some unacceptable practice in the original Basic program (e.g. RESTORE to a line number not containing data, or a redundant line containing instruction to GOTO or GOSUB a line number which does not exist). You can identify the line number of the Basic program from the reference labels in the SRC file (e.g. L999 : precedes the compiled contents of line 999. Ignore auxiliary references such as L999A: or L999A0:).

If you believe an error has actually been caused by a bug in FASBAS, then please let me know.

### 3.11 Running the compiled program

At last the fruits of your labour will be revealed. The program can be run as normal for LDA programs, or you can load it and then save it again as a PRG.

At the end of a program run, control will return to Basic, with a READY message. Programs can be interrupted only by using the RESET key. To run a program again, use ESC ^.

### 3.12 Changing an LDA to a PRG type program

The FBASM program produces an LDA file which can be run by entering (in FCS mode):-

```
RUN TEST. LDA
```

The only drawback is that LDA files are slow to load, and therefore it is preferable to convert them to PRG. This can not be done by simply renaming the file in the Directory, as LDA files contain extra information (load addresses for each block of data) which must be removed.

The simplest way requires that you note the last memory address displayed during the FBASM run. If this last address were for example 9872, then the method would be:-

```
FCS > LOAD TEST
```

```
FCS > SAVE TEST. PRG 82A0 - 9872
```

(Note that the start address for compiled programs is always 82A0)

If you forgot to note the last address during the FBASM run, you can carry out assembly again (omitting the instruction 'TO 0:' or 'TO 1:', as the LDA file has already been written to disk).

#### 4. BASIC STATEMENTS ACCEPTED FREELY BY BASIC

The following categories of Basic statement may be used without any abnormal limitation:-

Multiple statements per line

Equivalence statements (e.g.  $A = 1, B = C$ )

Mathematics (e.g.  $B = A + (2/3^2) - 32E2$ )

Logical connectives (e.g.  $C = X \text{ OR } 3$ )

Trigonometric and other functions (e.g.  $\text{COS}(X), \text{ABS}(X)$  etc)

PEEK and POKE (e.g.  $\text{PEEK}(A + B), \text{POKE } X, Y$ )

INP and OUT (e.g.  $\text{INP}(X), \text{OUT } X, Y$ )

PRINT (e.g.  $\text{PRINT TAB}(X); A\$; "OK"; A+B, Y$ )

PLOT (e.g.  $\text{PLOT } 12, 3, X, Y$ )

IF...THEN... and IF... GOTO ... (e.g.  $\text{IF } X \text{ THEN } 200$ )

GOTO and GOSUB

REM

END (Restart program by using Escape ^)

ON... GOTO... and ON... GOSUB

FRE(X) and FRE(X\$)

WAIT X, Y and WAIT X, Y, Z

Strings and functions (e.g.  $A\$, \text{CHR}\$(X), \text{MID}\$(A\$, B, C)$ )

String arrays (any number of dimensions)

CLEAR (e.g.  $\text{CLEAR}, \text{CLEAR } 500$ )

CALL (e.g.  $Y = \text{CALL}(X)$ )

INPUT (variables, arrays, strings, single or multiple)

This list contains practically everything and it is much easier to look at the lists of statements accepted with limitations or not accepted at all.

5. BASIC STATEMENTS ACCEPTED WITH LIMITATIONS

The following categories of statement have some limitation imposed on their use:-

FOR... NEXT loops

DATA, READ, RESTORE

NUMERICAL ARRAYS

DIM statements

LOAD, RUN

FILE, GET, PUT

5.1 FOR....NEXT loops

All normal FOR....NEXT loop statements are permitted, except for premature exit from loops. This is not good practice but is tolerated by Basic, although in some cases it could cause the stack to grow so large that it overwrites part of the program. Type b) below will cause compiled programs to loop back to the wrong place.

Premature exit can occur in three forms:

a) Straightforward jump out of a loop, e.g.:

```
10 FOR X = 1 TO 10 : READ A(X)
20 IF A(X) = 0 THEN 200
30 NEXT
```

This should be rewritten as follows:-

```
10 FOR X = 1 TO 10 : READ A(X)
20 IF A(X) = 0 THEN X = 10 : NEXT : GOTO 200
30 NEXT
```

b) Skipping part of the inner of two nested loops, by using a conditional NEXT to go back to the outer loop e.g. :

```
110 FOR A = 1 TO 3 : FOR B = 1 TO 10
120 READ X(A,B) : IF X (A,B) = 0 THEN NEXT A
130 PRINT X(A,B) : NEXT B,A
140 .....
```

This form of premature exit will certainly not be compiled correctly and it must be rewritten to close the loop, e.g. :

```
110 FOR A = 1 TO 3 : FOR B = 1 TO 10
120 READ X(A,B) : IF X(A,B) = 0 THEN B = 10: NEXT
B,A: GOTO 140

130 PRINT X(A,B) : NEXT B,A
140 .....
```

- c) Sometimes the value of the variable controlling a FOR ...NEXT loop must be preserved at exit as it is used afterwards, and in this case setting the variable to its final value will give the wrong results. So a new variable must be used to store the exit value temporarily while the loop is closed, e.g. :-

```
50 FOR X = 1 TO 10
60 IF A(X) = 0 THEN 80
70 NEXT
80 PRINT X
```

This could be rewritten as:

```
50 FOR X = 1 TO 10
60 IFA(X) = 0 THEN Y =X : X =10 :NEXT:X = Y:GOTO 80
70 NEXT
80 PRINT X
```

Unfortunately some Basic programs are so badly structured, with jumps out of a loop and back in again, that it is difficult to tell whether there has been a premature exit or not. A useful method of finding instances of type b) above is to edit the Basic program, changing suspected statements like NEXT X to simply NEXT, and then running the Basic program to see if it still works. It is best not to change too many at once, or you will not be able to work out where the problems are. This technique will not find the cases where extra bytes are left on stack, but these may not be critical ( and Basic can not stop the stack growing in such cases either).



## 5.2 DATA, READ, RESTORE

These statements can be used freely except that if the statement RESTORE to a line number (e.g. RESTORE 200) is used, then the referenced line must contain some DATA. (Basic allows RESTORE 200 to cause reading of data from the next DATA statement at or after line 200.)

This error will not be found by FASBAS itself, but will cause an error indication during assembly, due to reference to non-existent label D.....

## 5.3 Numerical Arrays

Numerical arrays with two or more dimensions (e.g. A(1,2), B(1,2,3) etc) are handled exactly the same as in Basic, using most of the same access routines, and as a result they do not give much increase in speed. Single dimension numerical arrays such as X(3) are treated as a special case to reduce accesstime, but this does cause slight complications.

Essentially all numerical arrays can be used freely, except that multi-dimension arrays must be declared using a DIM statement (for single dimension arrays the DIM statement is optional).

Failure to include a DIM statement for a multi-dimension numerical array will cause FASBAS to generate an error message against every line containing the array.

(Note that no such limitations apply to string arrays, which are all handled as in Basic.)

A second peculiarity of numerical arrays is that single dimension arrays can not be loaded using the LOAD"....ARY" statement. This can be fiddled by turning them into two dimensional arrays with one dimension set to zero (e.g. DIM A(20,0)), with consequent loss of speed.

## 5.4 DIM Statements

Dimensioning of multi-dimension numerical arrays is described in section 5.3.

Dynamic dimensioning, by using variables instead of constants (e.g. DIM A\$(X,Y)), is not allowed, and will cause an error indication from FASBAS. This is because when running the compiled program, variable X and Y would be found to have value zero, and therefore a BS ERROR would result if this were not corrected.

## 5.5 LOAD and RUN

The LOAD statement can be used freely, with the sole exception that LOAD....ARY can be used only for multi-dimension arrays (see section 5.3). This is because compiled programs use a different method for accessing single dimension arrays.

The RUN statement can be used by itself or following a LOAD statement in the same line. Used by itself, the RUN statement is taken as an instruction to run the compiled program from the beginning, and any attempt to specify a line number will be ignored (e.g. IF A\$ = "Y" THEN RUN 20 will cause the program to run from the beginning rather than from line 20).

When following a LOAD statement, RUN is assumed to be an instruction to run a Basic program that has been loaded, and in this case a line number can be specified. The following example will work the same as in Basic:

```
1500 IF A$ = "N" THEN LOAD "MENU": RUN 100
```

Note that LOAD "MENU": GOTO 100 would probably not give the results expected, since this technique is normally used in Basic when it is required to carry forward variable values from one program to another, and a compiled program does not store variables in a suitable location or format for subsequent use by Basic.

Compiled programs can also be chained, but not using the LOAD statement. For these you must use PLOT 27,4 : PRINT "RUN TEST. LDA" (or "RUN TEST" for PRG type programs).

## 5.6 FILE, GET, PUT

FASBAS does not accept FILE "A", "T" or "E" statements, and will give an error indication. FILE "N", "R", "C", and "D" statements, GET and PUT can be used freely without any limitations. However, with a large program you might possibly find that, because the compiled program is larger than the original, the memory space available for file buffers is not sufficient.

## 6. STATEMENTS NOT ACCEPTED BY FASBAS

The following statements are not accepted by FASBAS and will cause an ERROR indication.

```
DEF  
FN  
LIST  
SAVE  
CONT
```

ROMPACK DEVELOPMENT FOR THE COMPUCOLOR  
=====

Having just completed my last fiddle with hardware modifications for the Compucolor (multiple character sets), I turned my attention to the normally unused 8K of memory space in the address range 4000H to 5FFF. As my machine is forever in pieces, I often wondered why it was fitted with the 8K EPROM board which contained a single 2K EPROM. It seems that many of the early V8.79 models required this extra memory to fix a bug or two in FCS.

As we have nine Compucolors (plus nine 3651s) at the Western Australian Institute of technology, I soon found some of the V8.79 models running quite happily with the normal four x 4K FCS ROMS and without the extra EPROM. The next step was to check the ISC numbers printed on the ROMS. Only one was different; in location UA5.

Four ROM set

-----  
UA4 #100652  
UA5 #100695 \*  
UA6 #100654  
UA7 #100655

Five ROM Set (? Earlier)

-----  
UA4 #100652  
UA5 #100653  
UA6 #100654  
UA7 #100655

I borrowed the #100695 ROM to try in my machine in an attempt to recover a free EPROM board. No luck! The Compucolor wouldn't start up. After considerably more head scratching and IC comparisons I found that the Bipolar PROM in location UB3 was marked with a different number. (#100690 on mine, #100635 on the borrowed logic board.) Exchanging this UB3 PROM was the answer. When I checked the logic diagram, the UB3 PROM provides the required Chip Select signals, hence there are differences if FCS/BASIC is spread over five ROMS instead of four.

Following this was a long saga of trying to copy the #100695 mask ROM into a 2532 EPROM and then the #100635 bipolar PROM. The bipolars in particular are a rat's nest of so called "equivalents". Every brand seems to require its own special programmer.

Having finally a free 8K EPROM board, I modified it to hold four by 4K EPROMS (16K total of type: 2732 or 2532). Using a panel switch I could select either bank of 8K memory. In one bank I put the superb BASIC editor by Merv. Lynden, containing DOS, variable lister and a printer driver. The other bank held Jim Helm's Editor and Assembler.

This was demonstrated to a group of local Compucolor owners, where a number of them expressed an interest in obtaining one for themselves. As the ISC EPROM boards were in somewhat short supply in this remote part of the world (as are lots of things), I began designing my own.

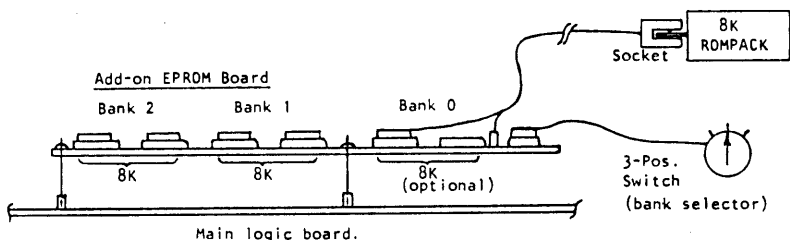
Initially this was to be a simple copy of the original, but the first design improvement was to place the EPROMS on the top of the board to allow easy exchange. The next step was to see how many 4K chips would fit in the space available. Currently six can be plugged in giving 24K in total.

This was followed by adding some selection logic so that each bank could be selected by a program instead of a switch. (If this is used it also involves a cable to the 50-pin bus.) Even with these improvements, to change an EPROM involved removing the back of the cabinet and all of the associated hassles.

Thus the ROMPACK was born. I simply extended one of the 8K banks via a ribbon cable to a 26-pin socket on top of the CompuColor cabinet. The ROMPACK itself is a miniature gold-plated PCB holding two 4K EPROMS fitted into a small plastic box. As the 2732 EPROMS use only a single 5 volt supply, the ROMPACK can be easily exchanged without having to turn off the power.

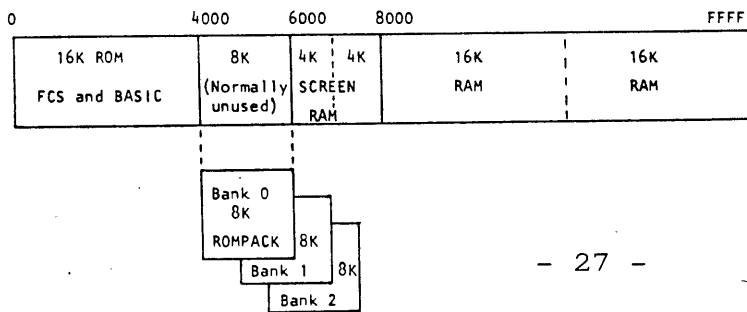
The whole system can be configured in a variety of ways. I currently have two fixed 8K banks holding the editors and Assembler mentioned above, plus the 8K ROMPACK for a number of other programs. A three-position switch selects the required bank. For V8.79 machines, to enter 4000H just press (ESC) then F. For V6.78 a jump instruction to 4000H will have to be entered first so that (ESC) ^ can be used. (FCS can be modified to overcome this.)

So what's next. Probably an 8K RAM board somehow integrated with this system. (I'm then going to have to find a 4-position switch!)



John Newman,  
18 Feb 83

EPROM/ROMPACK System - Physical layout



MEMORY MAP - COMPUCOLOR II

FIRST AID FOR COMPUCOLOR DISKDRIVES by Thomas J. Herold

Is your disk drive giving you error messages, particularly when you are loading a new disk for the first time? Does it fail to read or write as you think it should?

In almost all cases these problems are caused by a mis-match between the current disk drive speed and the original speed used to record the disk. Both the internal and external disk drives for the CCII are notorious for changing speeds without prior notice. For this reason, the speed of all CCII disk drives should be checked and adjusted as necessary, at least once a month. The allowable operating range is only 299 to 300 r.p.m. If your drive's speed is out of this range, you will have problems loading programs, exchanging disks with other CCII owners and possibly even reading your own disks.

Both internal and external drives can be adjusted visually, by turning the potentiometer on the disk drive's internal printed circuit board while watching the stroboscope indicator lines on the bottom of the disk drive rotor (under artificial light, not sunlight), in much the same way a stereophonic turntable is adjusted to rotate at exactly 33-1/3 r.p.m. This method provides reasonable accuracy, if performed carefully.

For the internal disk drive, it is first necessary to remove the back housing from the computer. The potentiometer can then be adjusted by using a small screwdriver, and the indicator lines can be watched by using a small mechanic's mirror; or the disk drive can be dismantled from the computer frame and allowed to sit beside the computer during the adjustment. **BE EXTREMELY CAREFUL WHEN WORKING WITH A SCREWDRIVER ON A POWERED-UP COMPUTER. DON'T TOUCH ANYTHING ELSE.**

Although, there is more reliable way to keep your disk drive speed properly adjusted. This is with the FORMATTER program, which allows you to check the speed of your disk drive at any time, graphically, right on the screen, without removing or dismantling anything and without using any tools. Of course, the drive will still have to be dismantled and adjusted with a screwdriver, if the speed control program indicates that the drive speed is outside the allowable operating range. This same FORMATTER program also allows you to Format and Initialize your new diskettes.

There is also a much easier way to gain access to, and adjust your drive, when necessary. Since an external disk drive is considerably easier to adjust than an internally mounted drive, many CCII owners have purchased an external disk drive housing, dismantled their internal disk drive from the computer, and installed it in the external housing where it is very easy to adjust. These factory provided metal and plastic external housings provide protection from physical damage, electronic radiation, spilled liquids and dust, and come complete with LED and factory ribbon cable to plug into the computer. You can order a complete housing at Intelligent Computer Systems in Hunstville, Al. The housing comes with a good documentation on how to take the drive out of the unit and how to install it into the housing. The price of the housing is US\$ 79 and the Formatter with Speed Control is US\$ 24. Good luck, and be careful!

ASSEMBLY LANGUAGE SUBROUTINES, part seven

by Myron T. Steffy, 10833 Brookside Drive  
Sun City, Arizona, 85351

=====

One of the fundamental functions of a computer is the acceptance of keyboard input. When operating in Basic, we take it for granted. When it comes to assembly language, it's another ball game and is one of the more complex things we do. Actually, the 'CHRINT' routines have been published more than once in 'COLORCUE' so they won't be repeated here.

The simplest form of keyboard input is the single character. In the first installment of this series we presented a sub-routine named 'GETANS' which performed this function. Frequently however we need to input a hex address or a numerical quantity. The 'CHRINT' and its ancillary operators is designed to take alphanumeric characters from the keyboard, provide backspace editing, echo them to the screen and place them in an input buffer for further processing. We don't really need anything quite that elaborate for hex addresses. What is required here is simply conversion of ASCII entries into hex values and storage for use later in the program. To print them on the screen, they must be converted back into ASCII.

The routine named 'HEXINP' that follows is designed to take a series of four hex characters as a 'starting' address, followed by an 'end' address. It then calculates the number of bytes between the two addresses and prints the result on the screen. The addresses and byte count are stored for use elsewhere. 'HEXINP' could be used to enter quantities or other values with slight alterations. It rejects non-hex characters and prints an error message if the start address is higher than the end address. No backspacing is provided but if you make an error in entering a value, simply type a non-hex character such as 'H' and the question will be repeated.

The annotated source code is not too difficult to follow so we won't elaborate here. The subs 'LBYT' and 'B2HEX' are utilites in the CCII ROM and can be called as such. They are printed out in full to illustrate the conversion process. If you use the ROM subroutines, add 'MOV C,A' after calling B2HEX. Please note also that the keyboard flag must be set to '14' or the keyboard itself will echo to the screen and you will have double printing of the input characters.

There is a seldom used function of 'OSTR' that we have tacked onto the end. It allows a string to be repeated a number of times. All that it does here is repeat the '=' sign 62 times and underline a statement. It could be used to draw a grid on the screen or write "I will not tell a lie" one hundred times. It would have been handy when we were eleven years old and had to do it on the blackboard.

This hacker has just about run out of ideas for subject matter. If there is something pertaining to assembly language that you want covered you had better let me know. There is a very good possibility that I won't have the answer but I'll sure try to dig it out for you. And as usual 'Have Fun'.

```
;      ROUTINE TO ENTER HEX CHARACTERS IN A PROGRAM
```

```
;      by Myron T. Steffy, Sun City, Arizona 85351
```

```
;=====
```

```

      LO      EQU      3392H      ;(17C8H)
      OSTR    EQU      33F4H      ;(182AH)
      CRLF    EQU      338BH      ;(17C1H)
      ESC1    EQU      053AH      ;(2420H)

      KBDL    EQU      81DFH
      KBCHA   EQU      81FEH
      READY   EQU      81FFH
      KEYTST  EQU      0024H

      ORG     9000H

START:  LXI     H,00
        DAD     SP
        SHLD   FCSSP
        LXI     SP,STACK
        XRA    A
        MVI    A,14      ;DISABLE ECHO TO SCREEN
        STA    KBDL

RSTART: LXI     H,MSG01 ;START ADDRESS MSG
        CALL   OSTR
        CALL   GETHX
        JNC    RSTART
        MOV    H,B
        MOV    L,C
        SHLD   MEMST     ;STORE START ADDRESS

ST02:  LXI     H,MSG02 ;END ADDRESS MSG
        CALL   OSTR
        CALL   GETHX
        JNC    ST02
        MOV    H,B

```

```

MOV      L,C
SHLD    MEMND      ;STORE END ADDRESS

;TEST FOR START > END ADDRESS & COMPUTE NUMBER OF BYTES

XCHG
LHLD    MEMST
MOV     A,H
CMA
MOV     H,A
MOV     A,L
CMA
MOV     L,A
INX     H
DAD     D
JC      ST03
LXI     H,MSG03
CALL    OSTR
CALL    CRLF
JMP     RSTART      ;RESTART HEX INPUT

ST03:   INX     H      ;STORE THE NUMBER
SHLD    NBYTE
LXI     H,MSG04
CALL    OSTR
LDA     NBYTE+1      ;LIST HIGH BYTE
CALL    LBYT
LDA     NBYTE      ;LIST LOW BYTE
CALL    LBYT
MVI     A,'H'
CALL    LO
CALL    CRLF
LXI     H,MSG05      ;ILLUSTRATES REPEAT FUNCTION
CALL    OSTR      ;OF 'OSTR' SUBROUTINE
JMP     EXIT

GETANS: MVI     A,32H
STA     READY
GTCHA:  CALL    KEYTST
JNZ     GTCHA
RET

CNVBN:  MOV     A,C      ;CONVERTS ASCII CHARACTER TO BINARY
SUI     30H
CPI     10D
RM
SUI     7
RET

GETCH:  CALL    GETANS      ;RETURNS NEXT CHARACTER TO CALLING
ANI     7FH      ;ROUTINE WITH PARITY BIT OFF
MOV     C,A
RET

GETHX:  PUSH    H
LXI     H,00

```



```

GHX05:  MVI      E,0
        CALL    GETCH
        CALL    LO
        MOV     A,C
        CPI    0DH      ;SEE IF C/R
        JNZ    GHX10
        PUSH   H
        POP    B
        POP    H
        MOV    A,E
        ORA   A
        JNZ    SRET
        JZ     FRET
GHX10:  CALL    VALDG    ;SEE IF DIGIT
        JNC    GHX15
        CALL    CNVBN
        MVI    E,0FFH
        DAD    H        ;* 2
        DAD    H        ;* 4
        DAD    H        ;* 8
        DAD    H        ;*16
        MVI    B,0      ;CLEAR UPPER 8 BITS OF BC PAIR
        MOV    C,A
        DAD    B
        JMP    GHX05    ;GET NEXT CHARACTER
GHX15:  CALL    CRLF
        POP    H
        JMP    FRET

LBYT:   PUSH   PSW      ;CONVERTS 8-BIT INTEGER IN A REGISTER
        RRC
        RRC
        RRC
        RRC
        CALL   LHXD     ;LIST FIRST HEX DIGIT
        POP    PSW
LHXD:   CALL   B2HEX    ;CONVERT SECOND HEX DIGIT
        JMP    LO       ;LIST IT AND RETURN

B2HEX:  ANI    0FH      ;MASK OUT UPPER 4 BITS
        ADI    90H
        DAA
        ACI    40H
        DAA
        MOV    C,A
        RET

VALDG:  MOV     A,C      ;VALIDATES HEX DIGIT
        CPI    '0'
        JM     FRET
        CPI    '9'
        JM     SRET
        JZ     SRET
        CPI    'A'
        JM     FRET
        CPI    'G'

```

```

        JP      FRET
        JMP     SRET

SRET:   STC           ;SETS CARRY TRUE
        RET

FRET:   STC           ;ROUTINE RETURNS CARRY SET TO FALSE
        CMC
        RET

EXIT:   CALL    CRLF
        LXI    H,KBDFL
        SHLD  8200H
        LHLD  FCSSP
        SPHL
        MVI   A,44H
        JMP   ESC1

MSG01:  DB      13,10,10,19,9,'ENTER START ADDRESS IN HEX '
        DB      22,239
MSG02:  DB      13,10,10,19,9,'ENTER FINISH ADDRESS IN HEX '
        DB      22,239
MSG03:  DB      13,10,10,17,9,'ERROR: MEMORY START ADDRESS '
        DB      '> THAN MEMORY END ADDRESS ',13,10,239
MSG04:  DB      13,10,10,18,9,'THE NUMBER OF BYTES IS ',239

MSG05:  DB 19,13,10,9,'THIS IS A TEST OF THE REPEAT FUNCTION.'
        DB 13,10,18,237,62,'=',238,13,10,10,239

; 237 indicates next quantity is number of repeats, '62'.
; '=' is string to be repeated.
; 238 signals end of repeat string.
; 239 marks the end of entire legend.

        ;DATA   STORAGE

MEMST:  DW      1           ;START ADDRESS
MEMND:  DW      1           ;END ADDRESS
NBYTE:  DW      1           ;NUMBER OF BYTES
FCSSP:  DW      1           ;FCSSP STACK POINTER

        DS      50H        ;STACK AREA

STACK:

        END      START

```



EPPS

# Word Processor

v5.2



for the COMPUCOLOR II (V6.78, 8.79), 3621 and INTECOLOR 3651. (32K RAM)

**only \$55 (u.s.)** Incl. airmail

- \* Full screen, fast operation with 20K byte buffer. (Assembler written)
- \* Can be used with any level keyboard. (101 key is recommended.)
- \* Automatic word wrap on screen and printer with justification. (30-199 col)
- \* Block and character Move, Copy, Delete, Save and Print.
- \* String search with optional replace. (Both up and down file.)
- \* Operates with or without lowercase character set. (Selectable).
- \* HELP facility. Full command summary on screen. (Two pages.)
- \* Automatic repeat on all keys.
- \* Imbedded control codes allows operation of any printer function.
- \* Screen preview of printout at any time.
- \* Compact file storage in FCS format. Can process existing .SRC files.

**PPI**

PROGRAM PACKAGE INSTALLERS,

P O Box 37,

DARLINGTON,

WESTERN AUSTRALIA 6070 (Ph.092996153)

Please include  
payment with order.

## ◆ CompuColor Hardware Options ◆

- ★ LOWER CASE Character set. (Switchable) MSC12 \$29 U.S.
- ★ MULTI-CHARACTER sets. (Lowercase, Electronic, Music etc.) \$39
- ★ REMOTE DEVICE CONTROLLER. Switch ON/OFF 8 devices. PSC1 \$45
- ★ 16K RAM Upgrade. (Increase from 16K to 32K.) \$99
- ★ ROMPACKS. Easy exchange of 8K EPROM MODULES. The interface board can hold an additional 8 or 16K EPROM.
  - Interface board, cable and ROMPACK socket: \$48
  - Each blank ROMPACK (Including EPROMS): \$25

(Please write for full details of ROMPACK system and available software.)

**PPI**

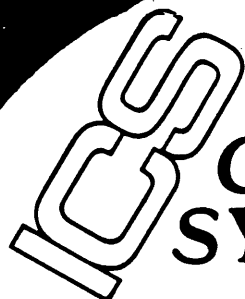
PROGRAM PACKAGE INSTALLERS,

P O Box 37,

DARLINGTON,

WESTERN AUSTRALIA 6070

All prices  
incl. airmail.



# INTELLIGENT COMPUTER SYSTEMS INC.

UNDERSTANDABLE SOFTWARE \*\*\*\*\*RELIABLE HARDWARE  
GOOD SUPPORT \*\*\*\*\* AND ALWAYS DISCOUNT PRICES

## NASHUA DISKETTE

\* single sided \* double density \* 40 tracks \* soft sector \* hub ring \*  
\* \* AND free exchange on all defective diskettes within 30 days \* \*

10 blank diskettes	US\$ 26	10 formatted diskettes	US\$ 31
20 blank diskettes	US\$ 50	20 formatted diskettes	US\$ 60

## SPECIAL

FORMATTER WITH SPEED CONTROL program  
Formats your CCII diskette, and dis-  
plays the speed of the diskdrive  
graphically on the screen   ORDER# SP845   US\$ 24

EXTERNAL HOUSING FOR DISKDRIVE  
complete with cable and detailed  
instructions (only 6 left)   ORDER# IM790   US\$ 79

COMPUCALC  
The Superversion of the Visicalc  
for the CCII                   ORDER# BG115   US\$ 120

SOFTWARE CATALOG WITH OVER 150 PROGRAMS AVAILABLE ON REQUEST  
\*\* SEND US\$1 FOR COMPLETE SOFTWARE AND HARDWARE CATALOG \*\*  
\*\*\* MASTER CHARGE, VISA AND AMERICAN EXPRESS ACCEPTED \*\*\*

INTELLIGENT COMPUTER SYSTEM, 12117 COMANCHE TRAIL  
HUNTSVILLE, AL 35803 USA, PHONE 205-881-3800

PRICES FOR COMPUCOLOR AND ISC COMPATIBLE EQUIPMENT & SOFTWARE  
MARCH, 1983

ADD ON 16K RAM BOARD (INCREASE A 16K CCII OR 3621 TO 32K RAM)  
ASSEMBLED AND TESTED WITH RAM CHIPS 109.00  
UPGRADE 8K OR 24K TO 32K CALL OR WRITE  
8K PROM BOARD (INTEL 2716 TYPE EPROMS FOR USE IN 4000H-FFFFH)  
ASSEMBLED AND TESTED-NO PROMS INCL. 49.00

64K BANK SELECTABLE ROM BOARD

SELECTS VIA SOFTWARE CONTROL UP TO 56K (D) OF EPROM IN 8K SEGMENTS.  
PLUGS INTO THE ADD-ON ROM SOCKETS INSIDE THE COMPUTER OR WITH THE  
EXTERNALIZER BOARD IT WILL OPERATE OUTSIDE THE COMPUTER. IT UTILIZES  
TI 2532 TYPE EPROMS. BUILT IN IS A SOCKET FOR ADDITION OF AN 8K BOARD  
E.G., 8K SINGLE BANK BOARD OR DEVLIN R A M BOARD. THIS COMBINATION  
GIVES YOU THE FULL 64 K OF PROM. 50 PIN BUS CONNECTOR ALLOWS FOR  
INSTALLATION WITH NO SOLDERING TO LOGIC BOARD.

ASSEMBLED AND TESTED---WITHOUT PROMS 249.00  
ABOVE IN KIT FORM 199.00  
50 PIN BUS CONNECTOR 10.00

BUFFERED EXTERNALIZER BOARD FOR CCII, 3621 OR 3650

THIS BOARD AND CABLE COMBINATION ALLOWS USE OF YOUR PLUG IN BOARD ON  
THE EXTERIOR OF THE MACHINE. IT IS MANDATORY FOR THE 3650 AND 3621.  
ASSEMBLED AND TESTED ONLY 59.95

'THE' BASIC EDITOR (SEE FORUM VOL 2 NO 1 PP 11-12 FOR REVIEW)

PROM VERSION IN 2532 OR 2716 PROMS 89.00  
PURCHASED WITH 8K PROM BOARD (A&T) 109.00  
PURCHASED WITH 64K BANK BOARD (A&T) 269.00

LOWER CASE PROM WITH STANDARD CCII OR ISC GRAPHICS

(CAN USE CAPS LOCK SWITCH OR ADD ON TOGGLE SWITCH)  
EPROM WITH LOWER CASE 35.00

ENHANCED OPERATING SYSTEM ROM FOR 6.78

ADDS 4 NEW JUMPS TO ACCESS EPROM AREA WITHOUT POKES TO USER  
VECTOR. PERSONALIZED WITH THREE INITIALS AT NO EXTRA CHARGE  
EPROM WITH NEW OPERATING SYSTEM 29.00

THE FOLLOWING PROGRAMS CONSIST OF ONE BANK OF 2532 OR 2716 EPROMS  
COM-TRONICS (tm) SOFTWARE IN NEW EPROM VERSIONS

1) TERM II COMMUNICATIONS PACKAGE 89.95  
2) CTE, FORMATTER, SPEED 144.95  
3) NEWBUG, CTA 109.95  
4) CRC, DFM 79.95  
5) PRINT2, CLIST, LLIST, LDAFIL 109.95  
6) TERMII, SRC/BAS, BAS/SRC, FILMRG 154.95  
7) CTA 69.95  
8) NEWBUG, SORT, RENUM 109.95

BILL GREENE SOFTWARE IN EPROM

1) SUPER MONITOR PLUS 79.95

JIM HELMS SOFTWARE IN EPROM VERSIONS

1) EDITOR/ASSEMBLER 90.00  
2) WISEII 8080 EMULATOR 75.00  
3) DISK EDITOR 60.00  
4) GENERAL LEDGER SPREADSHEET 110.00  
5) SOURCE DISASSEMBLER 130.00

## FREPOST COMPUTERS, INC (tm) EPROM VERSIONS

1) AT LAST! DIRECTORY PROGRAM BY BILL POWER	59.95
2) DISK BASED VERSION	39.95

RICK TAUBOLD & BILL GOSS' NEW REAL TIME STAR TREK  
FEATURING GAME SAVE AND ALL NEW GRAPHICS 25.00

A FIRST FOR THE COMPUCOLOR/INTECOLOR! DOUBLE PRECISION MATH!  
CAN BE USED TO GIVE UP TO 16 DIGIT PRECISION MATH OPERATIONS ON ADD,  
SUBTRACT, MULTIPLY AND DIVIDE. TRANSCENDENTAL FUNCTIONS TO COME.  
AVAILABLE IN ROM FOR YOUR BANK SELECT OR 8K PROM BOARD. CAN BE ADDED  
TO MOST PROM PACKAGES. THIS FAST MACHINE LANGUAGE MODULE IS CALLABLE  
FROM BASIC, DO AWAY WITH THE PENNY ERRORS FOREVER!

STANDALONE IN ROM PACK	59.95
INCLUDED IN ROM PACK WITH OTHER PGMS	39.95

THE QUADRAM LINE OF PRINTER SPOOLERS FREE YOUR COMPUTER FROM NEEDLESS  
WASTE OF TIME WHILE EVEN FAST PRINTERS PRINT. DUMP YOUR DATA AT 9600  
BAUD AND LET THE MICROFAZER HANDLE THE PRINTING CHORES WHILE YOU  
CONTINUE CRUNCHING. STANDALONE UNITS POWERED FROM YOUR PRINTER'S  
PARALLEL INPUT, ALLOW RESET AND RECOPY FROM FRONT PANEL.

64K SERIAL IN/PARALLEL OUT	269.95
8K SERIAL IN/PARALLEL OUT	189.95

ALL CONFIGURATIONS OF INPUT AND OUTPUT MODE ARE AVAILABLE. CALL!

THE ANGEL IS ANOTHER PRINT SPOOLER, AND IS LIKE THE MICROFAZER, BUT  
HAS UNIVERSAL INPUT/OUTPUT, MORE FRONT PANEL CONTROLS ALLOWING PRINT  
INTERRUPT, REPRINT FROM PAGE X, AND MUCH MUCH MORE

64K SERIAL OR PARALLE IN AND OUT	289.95
----------------------------------	--------

## MULTI COMPUTER USERS NOTE :

ALL QUADRAM, STB, AST, PRINCETON GRAPHICS, AMDEK, TECMAR, AND RELATED  
MANUFACTURERS PRODUCTS ARE AVAILABLE AT GREAT SAVINGS FROM FREPOST  
FOR OTHER TYPE SYSTEMS.

WE SAVED THE BEST FOR LAST.....

NEW FOR 1983!! THE OKIDATA MICROLINE 92 9X9 LETTER QUALITY PRINTER  
COMES WITH THESE STANDARD FEATURES ---->

2000 BYTE BUFFER, PARALLEL INTERFACE, 7 CHARACTER FONTS PLUS  
CORRESPONDENCE FONT (NOT JUST A DOUBLE STRIKE DP FONT). ALSO,  
A SMART VERTICAL FORMAT UNIT, REAR OR BOTTOM PAPER FEED,  
FRICTION OR 9 1/2" PIN FEED, 6 CHARACTER WIDTHS, ENHANCED  
PRINT MODES AND

\*\*\*\* DOT ADDRESSABLE GRAPHICS INCLUDED AT NO EXTRA CHARGE! \*\*\*\*  
YOU CAN ALSO CREATE YOUR OWN FONT AND DOWNLOAD IT TO THE  
M/L 92! 160CPS PRINT SPEED IN DP MODE, SHORT LINE SEEKING  
BIDIRECTIONAL PRINTING.

MICROLINE 92 80 COL 9X9 MATRIX PARALLEL	540.00
MICROLINE 92 AS ABOVE, SERIAL INTERFACE	620.00
MICROLINE 93 132 COLUMN 9X9 MATRIX PAR'LL	900.00
MICROLINE 93 AS ABOVE, SERIAL INTERFACE	980.00
MICROLINE 80 80 COL 7X9 MATRIX 80CPS	350.00
FREPOST SCREEN DUMP PROGRAM (CCII GRAPHICS CHAR SET)	39.95

FOR INFORMATION ON THESE AND OTHER PRODUCTS, PLEASE CALL OR WRITE  
TODAY. DELIVERY ON MOST ITEMS IS FROM STOCK.

INSTALLATION ASSISTANCE AND SERVICE IS AVAILABLE AT MODEST COST FOR  
YOUR COMPUCOLOR OR ISC COMPUTER.