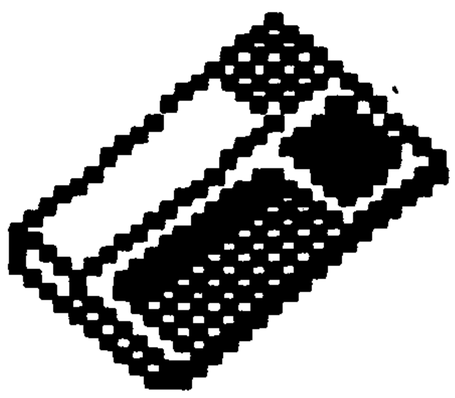

HUG

HOUSTON USERS' GROUP

JANUARY

1986

HUG TIBBS - (713) 475-8909
24-hour BULLETIN BOARD

MEETING SCHEDULE

FIRST SUNDAY OF EVERY MONTH
(2nd Sunday if 1st Sunday is on a holiday weekend)

****

THE NEXT MEETING IS

SUNDAY, MARCH 02, 1986 2:00 P.M.

St. John's School - 2401 Claremont

****

IN THIS ISSUE

****

1986 HUG OFFICERS

President --- MARK CRUMP .... unlisted	Secretary - JIM HUMPHREY . 465-6525
VP/Membership DON LEWIS 353-5295	Treasurer - JERRY ILLING . 664-7059
VP/Program -- DAVID SHOLMIRE 482-0186	Librarian - LARRY PIPKIN . 499-9991
VP/S.I.G. --- ROGERS MILLS .. 930-0810	TIBBS/SysOp BILL KNECHT .. 473-5713
Exec. Asst. - TOM JAY 850-0222	Editor ---- PHIL POXON .. 973-2362

****

EDITORS RAM

As newsletter editor, I would like to bring to the membership what they would like in their newsletter. Therefore I ask that if there is any type of article, program or other information pertaining to the TI-99/4A computer that you would like to see in **your newsletter**, please let me know and I will try to bring you what you want. If you would like to submit an article, product review, program or tips and suggestions, please feel free to send them to me care of the address on the front of the newsletter. If you wish you can also send the files to me by modem if you have one. All help in keeping this newsletter one of the best in the country will be greatly appreciated.

CREDITS

I would like to thank Johnson Users Group for permission to use an article from the Texas Bull bulletin board. I also want to thank the Sydney News Digest,the newsletter of the TI Sydney Users group for the article used in last month's newsletter (ASCII listin) and the remainder of the article used in this month's newsletter which explain the logical set-up of the code and some interesting qualities of the code. I would also like to thank Jim Peterson for sending us his Tigercub tips.

NOTE: NEW BBS

STARSHIP EXODUS
713 479-0466

FIFTH DIMENSION TI BBS
300/1200 BAUD (713)277-0459
X-MODEM .

4A BBS
713 492-0018

Lets have a very close look at the ASCII Code. It has more to it than meets the eye! It is a very logical code. As we all know ASCII is the abbreviation for American Standard Code for Information Interchange. It is the most commonly used microcomputer code, although other codes are used, mainly by complex word processors used by printers for typesetting.

The notes explain the binary logic of the code. This is used by the computer for internal checks and has no implications in any program written in a high level language. We can use some of the features of the ASCII code in programs written in Basic. Some of these are:

- if it is smaller than 48 or greater than 57 then it is not a number;
- if it is greater than 64 and smaller than 90 then it is a capital letter;
- if it is greater than 96 and less than 123 then it is a lower case letter;
- adding 32 to the code of a capital letter converts it to its lower case equivilent;
- conversley, subtracting 32 from the code of a lower case letter converts it to its upper case equivalent;
- subtracting 48 from the code of a numeral equals the value of the numeral;
- sorting from 65 to 90 or 97 to 122 in ascending order will sort a list in alphabetical order.

As I said before it is a very logical code and one of the most widely accepted industry standard.

N O T E S

1* The ASCII code uses 7 bits. The 8th bit is always 0. This bit may be used as parity bit. (see the information on the computer architecture).

2* >00 to >1F are control characters. These have no printable characters on the VDU (Screen) or the printer. These are used by the computer to communicate with the printer. Historicallly, these were developed for the teletype machines in the past. Bits 5 and 6 are equal to 0.

3* >20 to >2F are signs and non letter characters, also mathematical operators. Bit 6 = 0; bit 5 =1 bit 4 = 0

4* >30 to >3F are numerals, logical comparators and special characters.
Usually, special program functions are assigned to the characters ':',';' and '?'(TI did not assign any special function to the '?') this is the main reason for their presence in this group.

5* >40 to >5F are letters, -upper case, and carriage control characters.
Bit 6 = 1 means letter, further if bit 6 = 0 and bit 5 = 0 then letter is upper case (capitals).

6* >5B to >5F are optionally defined. These may vary, depending on the maker of the computer and printer.

7* >60 to >7E are small case letters and secondary printer characters. Bit 6 and 5 are equal to 1.

8* Delete (rub-out) code. It is used by punched tape readers. A code is deleted (rubbed out) if all coding holes are punched out.

A* TI allocated this code to the cursor character.

B* TI allocated this code to the edge character.

C* TI allocated this code to the space character.

TIPS FROM THE TIGERCUB

#31

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Over 130 original programs in Basic and Extended Basic, available on casette or disk, only $3.00 each plus $1.50 per order for PPM. Entertainment, education, programmer's utilities.
Descriptive catalog $1.00, deductable from your first order.
Tips from The Tigercub, a full disk containing the complete contents of this newsletter Nos. 1 through 14, 50 original programs and files, just $15 postpaid.
Tips from the Tigercub Vol. 2, another diskfull, complete contents of Nos. 15 through 24, over 60 files and programs, also just $15 postpaid. Or, both for $27 postpaid.
Nuts & Bolts (No. 1), a full disk of 100 Extended Basic utility subprograms in merge format, ready to merge into your own programs. Plus the Tigercub Menuloader, a tutorial on using subprograms, and 5 pages of documentation with an example of the use of each subprogram. All for just $19.95 postpaid.
Nuts & Bolts No. 2, another full disk of 100 utility subprograms in merge format, all new and fully compatible with the last, and with 10 pages of documentation and examples. Also $19.95 postpaid, or both Nuts Bolts disks for $37 postpaid.
Tigercub Full Disk Collections, just $12 postpaid! Each of these contains either 5 or 6 of my regular $3 catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am NOT selling public domain programs - my own programs on these disks are greatly discounted from their usual price, and the public domain is a FREE bonus!
TIGERCUB'S BEST
PROGRAMMING TUTOR
PROGRAMMER'S UTILITIES
BRAIN GAMES
BRAIN TEASERS
BRAIN BUSTERS!
MANEUVERING GAMES
ACTION GAMES
REFLEX AND CONCENTRATION
TWO-PLAYER GAMES
KID'S GAMES
MORE GAMES
WORD GAMES
ELEMENTARY MATH
MIDDLE/HIGH SCHOOL MATH
VOCABULARY AND READING
MUSICAL EDUCATION
KALEIDOSCOPES AND DISPLAYS
For descriptions of these send a dollar for my catalog!

A few people have asked for a program that they could use to encode personal messages on a BBS. considering the current legal threats to BBS's, I doubt that a SysOp will allow coded messages, but here is a coder/decoder to create code that should be quite difficult to crack. First we need another of those programs that write a program -

```
100 !CODEPRINT by Jim Peterson - creates a random code in a MERGE format program CODESTRING to be MERGEd into CODEMAKER
110 FOR J=1 TO 254 :: N$=N$&CHR$(J):: NEXT J
120 FOR J=1 TO 254 :: RANDOMIZE :: X=INT(RND*LEN(N$)+1):: C$=C$&SEG$(N$,X,1):: N$=SEG$(N$,1,X-1)&SEG$(N$,X+1,LEN(N$)):: NEXT J
130 OPEN #1:"DSK1.CODESTRING",VARIABLE 163,OUTPUT :: PRINT #1:CHR$(0)&CHR$(1)&"C$"&CHR$(190)&CHR$(199)&CHR$(127)&SEG$(C$,1,127)&CHR$(0)
140 PRINT #1:CHR$(0)&CHR$(2)&"C2$"&CHR$(190)&CHR$(199)&CHR$(127)&SEG$(C$,128,127)&CHR$(0)
150 PRINT #1:CHR$(0)&CHR$(3)&"C$"&CHR$(190)&"C$"&CHR$(184)&"C2$"&CHR$(0):: PRINT #1:CHR$(255)&CHR$(255):: CLOSE #1 :: END
```

And now the coder/decoder -

```
100 !TIGERCUB CODEMAKER written by Jim Peterson
110 !The MERGE format program CODESTRING created by the program CODEPRINT must be MERGEd into lines 1-3 of this program
120 DIM A$(254):: DISPLAY AT(3,6)ERASE ALL:"TIGERCUB CODEMAKER" :: DISPLAY AT(12,1):"Do you want to": :"(1)Encode":"(2)Decode"
130 CALL KEY(0,K,ST):: IF K=49 THEN 140 ELSE IF K=50 THEN 290 ELSE 130
140 OPEN #1:"DSK1.CODE",VARIABLE 254,OUTPUT
150 DISPLAY AT(5,6)ERASE ALL:"Type message in segments of":"not more than 254 characters":"and Enter. When done, type"
160 DISPLAY AT(9,1):"END and Enter. Type slowly":"to avoid skipped characters.":"Backspace with FCTN S to":"correct.": :"Press any key"
170 CALL KEY(0,K,ST):: IF ST=0 THEN 170
180 CALL CLEAR :: CALL LONGACCEPT(0,M$):: IF M$="END" THEN 280
190 DISPLAY AT(20,1):"WAIT, PLEASE - ENCODING"
200 FOR J=1 TO LEN(M$)
210 A$(ASC(SEG$(C$,J,1)))=SEG$(M$,J,1)
220 NEXT J
230 FOR J=1 TO 254 :: RANDOMIZE
240 IF A$(J)="" THEN A$(J)=CHR$(INT(26*RND+65))
250 CODE$=CODE$&A$(J)
260 NEXT J :: PRINT CODE$
270 PRINT #1:CODE$ :: CODE$="" :: FOR J=1 TO 254 :: A$(J)="" :: NEXT J :: GOTO 180
280 CLOSE #1 :: END
290 OPEN #1:"DSK1.CODE",VARIABLE 254,INPUT :: CALL CLEAR :: DISPLAY AT(12,10):"DECODING"
300 LINPUT #1:CODE$ :: FOR J=1 TO 254 :: M$=M$&SEG$(CODE$,ASC(SEG$(C$,J,1)),1):: NEXT J :: PRINT M$;:: M$=""
310 IF EOF(1)<>1 THEN 300 :: CLOSE #1 :: END
320 SUB LONGACCEPT(L,M$):: X=0 :: IF L<>0 THEN R=L ELSE R=R+1
330 M$="" :: C=3 :: CH=140 :: CALL CHAR(140,RPT$("0",14)&"FF")
340 CALL HCHAR(R,C,CH):: CH=CH+5+(CH=160)*25 :: CALL KEY(0,K,ST):: IF ST<1 THEN 340
350 IF K<>8 THEN 370 :: X=X-1 :: C=C-1 :: IF C=2 THEN C=3 :: R=R-1
360 M$=SEG$(M$,1,LEN(M$)-1):: GOTO 340
370 IF K=13 THEN 410
380 X=X+1 :: M$=M$&CHR$(K):: CALL HCHAR(R,C,K):: IF X=254 THEN 410
390 C=C+1 :: IF C=31 THEN C=3 :: R=R+1 :: IF R=25 THEN CALL CLEAR :: R=1
400 GOTO 340
410 R=0 :: SUBEND
```

Here is a simple little game I call Cover-Up. Use the #1 joystick, try to cover the white square with the black square. Press the fire button to speed up, release it to slow down.

```
100 CALL CLEAR :: CALL CHAR(96,RPT$("F",64)):: CALL SPRITE(#1,96,5,92,124):: CALL MAGNIFY(4):: CALL SPRITE(#2,96,16,100,100)
110 X=INT(20*RND)-INT(20*RND):: Y=INT(20*RND)-INT(20*RND):: CALL MOTION(#2,X,Y):: T=T+1 :: IF T=250 THEN 300
120 CALL JOYSPEED(1,1):: CALL COINC(#1,#2,8,A):: IF A=-1
```

```
 THEN 130 ELSE 110
130 Z=Z+1 :: DISPLAY AT(1,1)
:Z :: CALL SOUND(-50,500,5):
: GOTO 120
300 CALL DELSPRITE(ALL):: DI
SPLAY AT(12,5):"YOUR SCORE I
S "&STR$(Z):: DISPLAY AT(20,
1):"PRESS ENTER TO PLAY AGAI
N"
310 CALL KEY(0,K,S):: IF S=0
 OR K<>13 THEN 310 :: T,Z=0
:: GOTO 100
2111 SUB JOYSPEED(N,A):: CA
LL JOYST(N,X,Y):: CALL KEY(N
,K,ST):: S=S+K/9-1 :: S=S*AB
S(S>0):: IF S>30 THEN S=30
21111 CALL MOTION(#A,-(Y*S),
X*S):: SUBEND
```

For a one-handed BREAK, if you can't reach FCTN and 4, try FCTN with J and the space bar together.

If you like to call BBS's, try the TIBBS Spirit of 99 BBS in Columbus, Ohio on (614)451-8880 and leave me a "hello!"

Probably useless info - holding down FCTN and CTRL together and typing 1, 2, 3 and 5 will give ASCII codes 145, 151, 133 and 148, which are the codes obtained from CTRL Q, W, E and T, the keys diagonally below the 1, 2, 3 and 5.

Occasionally someone sends me a program they have keyed in from my newsletter, and asks why it won't run, so I wrote this routine to help find the errors. It is also useful to check whether two copies of a program are identical, but only if they have not been resequenced.

```
100 !CHECKER by Jim Peterson
 - to compare two programs a
nd list all differing lines
to the printer
110 DISPLAY AT(12,1)ERASE AL
L:"1st program DSK/filename?
":"DSK" :: ACCEPT AT(13,4):F
1$
120 DISPLAY AT(12,1)ERASE AL
L:"2nd program DSK/filename?
":"DSK" :: ACCEPT AT(13,4):F
2$
130 OPEN #1:"DSK"&F1$,INPUT
:: DIM M$(500),CH(500):: OPE
N #2:"PIO",VARIABLE 255 :: P
RINT #2:CHR$(15)
140 X=X+1 :: LINPUT #1:M$(X)
:: M$(X)=M$(X)&" " :: IF EOF
(1)<>1 THEN 140 :: CLOSE #1
:: OPEN #1:"DSK"&F2$,INPUT
150 IF EOF(1)=1 THEN 230 ::
LINPUT #1:X$ :: X$=X$&" "
160 FOR Y=1 TO X
170 IF X$=M$(Y)THEN CH(Y)=1
:: GOTO 150
180 NEXT Y
190 P2=POS(X$," ",1):: P2$=S
EG$(X$,1,P2-1)
200 FOR Y=2 TO X :: P1=POS(M
$(Y)," ",1):: P1$=SEG$(M$(Y)
,1,P1-1)
210 IF P2$=P1$ THEN PRINT #2
:"1st program = ";M$(Y):"2nd
 program = ";X$ :: CH(Y)=1 :
: GOTO 150
220 NEXT Y :: PRINT #2:"2nd
program = ";X$ :: GOTO 150
230 FOR J=1 TO X :: IF CH(J)
=0 THEN PRINT #2:"1st progra
m = ";M$(J)
240 NEXT J
250 CLOSE #1 :: CLOSE #2
```

Here's a great idea that was printed and reprinted in several newsletters -
At the beginning of a program that will run only in Basic, add the lines -

```
1 IF PI=0 THEN (first line
of program)
2 PRINT "YOU ARE IN EXTENDED
BASIC":"THIS PROGRAM RUNS
ONLY IN BASIC"
3 STOP
```

The idea is that PI is a function in XBasic with the value of pi, but is just a variable name in Basic with an undefined value of 0.

The trouble is, it doesn't work! If PI is keyed in from Basic and saved, it is saved in token format as a variable name, and when loaded back into XBasic is still just a variable name. And if PI is saved from XBasic, it is tokenized as a function, loads back into Basic as an unrecognized function and crashes! Can anyone come up with a way around that?

The above is the answer to the Challenge in Tips #3. Lines 100 and 110 were keyed in and saved from Basic, and loaded back into XBasic, then lines 120 and 130 were keyed in.

Here is a handy PEEK that hasn't been published as widely as most of them -

```
100 CALL INIT
110 CALL PEEK(8192,X)!Thanks
to Dale Loftis in the Orange
County UG newsletter!
120 PRINT X !If X=32 you are
in Extended Basic; if X=165
you are in Basic with the
Editor Assembler or
MiniMemory module inserted.
```

And another 3-D sprite demo, just to make all the Apple polishers jealous. See if you can figure out how it works.

```
100 CALL CLEAR :: CALL SCREE
N(5):: CALL CHAR(100,RPT$("F
",64)):: CALL MAGNIFY(4):: F
OR S=5 TO 9 :: CALL COLOR(S,
16,1):: NEXT S
110 DISPLAY AT(3,3):"TIGERCU
B SPRITE SHUFFLE" !by Jim Pe
terson
120 DATA 70,116,2,75,121,7,6
9,124,11,78,115,16
130 FOR J=5 TO 8 :: READ P(J
,1),P(J,2),L(J):: CALL SPRIT
E(#J,100,L(J),P(J,1),P(J,2))
:: NEXT J :: N=45
140 DATA 5,6,7,8,8,5,6,7,7,8
,5,6,6,7,8,5
150 RESTORE 140 :: FOR Y=5 T
O 8 :: READ A,B,C,D
160 FOR J=1 TO N :: CALL LOC
ATE(#A,P(A,1)-J,P(A,2),#B,P(
B,1),P(B,2)-J,#C,P(C,1)+J,P(
C,2),#D,P(D,1),P(D,2)+J):: N
=N-9 :: NEXT J :: GOSUB 180
170 NEXT Y :: GOTO 150
180 FOR J=5 TO 7 :: CALL POS
ITION(#J,P(J+1,1),P(J+1,2)):
: NEXT J :: CALL POSITION(#8
,P(5,1),P(5,2))
190 T=L(8):: L(8)=L(7):: L(7
)=L(6):: L(6)=L(5):: L(5)=T
200 FOR J=5 TO 8 :: CALL SPR
ITE(#J-4,100,L(J),P(J,1),P(J
,2)):: NEXT J
210 FOR J=5 TO 8 :: CALL SPR
ITE(#J,100,L(J),P(J,1),P(J,2
)):: NEXT J :: CALL DELSPRIT
E(#1,#2,#3,#4):: RETURN
```

Do you need some really REAL BIG letters on the screen? Just type your letter at the beep.

```
100 DIM X$(96):: CALL CLEAR
:: FOR CH=33 TO 89 STEP 8 ::
 FOR A=0 TO 7 !REAL BIG LETT
ERS by Jim Peterson
110 CALL CHARPAT(CH+A,X$(CH+
A-32)):: CALL CHAR(CH+A,"0")
:: L$=L$&RPT$(CHR$(CH+A),3):
: NEXT A
120 FOR T=1 TO 3 :: R=R+1 ::
 DISPLAY AT(R,4):L$ :: NEXT
T :: L$="" :: NEXT CH
130 CH$(1)=RPT$("0",16):: CH
$(2)=RPT$("F",16)
140 CALL SOUND(100,500,0)
150 CALL KEY(0,CH,S):: IF S=
0 OR CH>96 THEN 150
160 CALL HEX_BIN(X$(CH-32),B
$):: FOR J=9 TO 64 :: CALL C
HAR(J+32,CH$(VAL(SEG$(B$,J,1
))+1))
170 NEXT J :: GOTO 140
180 SUB HEX_BIN(H$,B$):: HX$
="0123456789ABCDEF" :: BN$="
0000X0001X0010X0011X0100X010
1X0110X0111X1000X1001X1010X1
011X1100X1101X1110X1111"
190 FOR J=LEN(H$)TO 1 STEP -
1 :: X$=SEG$(H$,J,1)
200 X=POS(HX$,X$,1)-1 :: T$=
SEG$(BN$,X*5+1,4)&T$ :: NEXT
 J :: B$=T$ :: T$="" :: SUBE
ND
```

Thought for the day. The excuses for piracy are exactly the same as the excuses for shoplifting, but you probably won't have to tell them to the judge - in this world, at least.

And that is almost

MEMORY FULL

Jim Peterson

DOWNLOADED FROM THE TEXAS BULL BBS
JUG 12/85

THE MAKING OF THE

CORCOMP 9900 DISK CONTROLLER CARD

By W. R. Moseid

When the decision was made to provide a new disk controller card to the TI-99/4A world, we began what turned out to be a long and ardous trip. The metal case (clamshell) was easy. Just use the one that had been designed for the CorComp 32K and RS-232 cards. Then, modify it slightly for a slot to accommodate the new circuit board which would project throught the back of the case. This part would hold the connectors for the cables to the internal and external floppy disk drives.

Since all disk controller integrated circuits (chips) can control up to four floppy disk drives, we decided not to restrict people to three drives. With some careful planning, design and care a card could be produced which would support up to four disk drives with any combination from the list below:

Single or Double Sided Single or Double Density 35 or 40 Tracks per Diskette Side Any Combination of the Above Automatic Density Recognition

In order to allow the use of a variety of disk drives which people had available, a feature was selected which allowed the owner to set the Head Step Time (Head step time is the time it takes the disk drive read/write heads to step from one track to another). You can select one of four step times for each disk drive in your system. The times selected were 15 milliseconds (ms) 10 ms, 6 ms, 3 ms. This timing is set by positioning a set of DIP (Dual-In-line Pack) switches to various on/off settings. The decision was made to place the DIP switches inside the case. Even though this meant the owner would have to remove the case to set all the DIP switches, this approach was selected for several reasons.

* Safety for the Card (Makes sure power is OFF when the DIP switch is set)

* Safety for the DIP switch (Chances of something hitting it or changing the settings were minimized)

* Lower costs to the Consumer (Assembly Time and Material Costs were lower)

* The 10ms Factory Setting works with most Drives

When the 99/4A Power Up sequence was examined, several interesting things were discovered:

* Plato, TE II, and other Command Modules have a special sequence that runs at that time. (once they get control they do not give it up and the power up scan is not completed.)

* In order to allow the owner the ability to select the CorComp Disk Manager from the Title screen, a special power up Screen had to be made to allow our rapid loader to execute on the single key press.

* Because of the way Plato, TE II and other Command Modules operate during PowerUp, a choice of two different Menu Screens had to be provided.

When the early timing studies were done, they were conducted with direct I/O using an Assembly Language program (NO GROM). At that time we calculated that the CorComp Disk Controller in double density could run 2 to 4 times faster than the TI Disk Controller. A fact proven by the speed at which the 98 sector Disk Manager program loads into expansion memory. This is based on the way that the CorComp Disk Controller card accesses the diskette and transfers the information into the computer. However, when TI designed their disk memory system, they made a decision that the memory expansion would not be required. This way, with the controller (old stand alone disk controller) and a console, you could utilize BASIC with the disk system. In order to do this, the TI Disk System and **ALL** of the Command Modules which use disk expect the information read from the disk to be in the console (VDP RAM) memory. For example, when BASIC, EXTENDED BASIC or the EDITOR/ASSEMBLER load from disk, they expect the information to be in the console memory. If memory expansion is attached then EXTENDED BASIC will move the information to the memory expansion after it is loaded. This moving process is very time consuming. Remember, each sector (256 characters) read or written must be passed through VDP RAM in order to be compatible with TI firmware/software. In the following table are some of the latest timing tests using GROM and VDP RAM:

Time To Load In Seconds

FILE	TI		CORCOMP			
TYPE	CONTROLER		SINGLE		DOUBLE	
SIZE	L/O	READY	L/O	READY	L/O	READY
	---	----	---	----	---	----
BASIC						
47	7.2	24.5	7.2	24.5	5.0	22.3

X-B
39 6.3 8.8 6.3 8.8 3.6 6.3

I/V 254
52 --- 18.2 --- 18.2 --- 14.9

E/A
25 --- 6.3 --- 6.3 --- 4.2

D/F 80
181 --- 55.8 --- 54.3 --- 47.8

L/O = CONTROLLER OUT
READY = CURSOR BACK ON SCREEN

NOTE: The tests were conducted with default interlace selections. Timing may be improved with different interlace selections for the various modules and languages.

With the speed increase indicated in the previous table, we naturally were curious how time is consumed having to use the VDP RAM as an intermediate storage area. The table below shows the time required to copy a disk. When the CorComp Disk Controller copies a disk, VDP RAM does not have to be used as an intermediate storage place saving time.

Time To Copy 360 Sectors (3 Files)
in Seconds

Type of copy	TI Cont.	CorComp Controler W/O TURBO	W/TURBO
SDtoSD	151	143	70
SDtoDD	---	135	61
DDtoDD	---	123	51

Measuring performance increase figures is always a challenge. This is due to the fact that the "statistics people" can make them do what they want. But, you can see a performance increase in using the CorComp Disk Controller Card and DiskManager of up to approximately 296% depending on several factors which are:

Diskette Density
The Operation Being Done
Diskette File Type (DIS/VAR-worst)
Language Used (GROM - worst case)
Kind Of Loader (X/B, Ed/Asm, etc.)
File Sector Location On Diskette

When the Utilities were designed we found that TI changed Extended Basic to prevent its scanning the peripheral DSR's for CALL's (ie; CALL FILES) from running a program. In order to allow the utiities to function with Extended Basic, a list of the utilities had to be provided in the Link Table in extended memory. Thus the syntax for using the CorComp Utilities in X/B is CALL LINK(utility name)(etc.....).

Console Basic did not possess this constraint. In Basic, the Utility syntax is CALL (utility name, etc.)

The Disk Manager was written especially for the CorComp Disk Controller. We sat down and thought of all the features one would like in a Disk Manager. Just about all of the ideas are in the current Disk Manager. A decision was made to publish the Disk Manager on diskette for the following reasons:

Easy to Release an Update
Lower Cost to the Customer

The task of figuring out all the technical details of how to achieve compatibility with the TI hardware, software and firmware was a very hard and time-consuming effort. At times, some of the issues seemed almost too much to overcome. In the end, our perseverence and determination were rewarded and the Disk Controller Card reached the market. All the known problems, as of this writing (11/10/84), have been resolved. All other 3rd Party cards are fully compatible with the CorComp 9900 Disk Controller Card.

While writing the manual, we decided to try and present the material in a manner that would allow the beginning user to sit down and follow the guide in a logical step-wise manner. This method allows the user to learn how to utilize the card from its manual in a very straight-forward manner. The manual was also designed for the technically minded individual who could read to the level of detail desired. It required 700 hours of effort preparing the manual before these objectives were achieved.

LOADING CORCOMP DISK MANAGER FROM EDITOR/ASSEMBLER

This bit of information is reprinted from Miller's Graphics publication "The Smart Programmer". It was printed on page 5 of the July, 1984 issue. For everyone with a CorComp Disk Controller card, Tom Knight sent us the following program. This short program allows you to load the CorComp Disk Manager from the Editor/Assembler module with option 3 - Load and Run. Thanks Tom.

```
IDT 'LOADMNGR'
```

```
       AORG >2700
       DEF MGR
MGR    LWPI >83E0
       MOV  R11, >8300
       LI   R12,>1100
       SBO  >0000
       SBZ  >0008
       BL   >44F2
       NOP
       SBZ  >0000
       MOV  >8300,R11
       B    R11
       END
```

HOUSTON USERS GROUP
PHILIP POXON--EDITOR
9122 HAMMERLY
HOUSTON,TX 77080

HUG LIBRARY CATALOG ADDENDUM
February 1986

0102 BUZZARD##EA/3 Joysticks rqd
This cute "Freeware" program is a maze game similar to "Frogger". 66 sectors

0103 BINGO##XB Printer optional
This program plays "Bingo" automatically or manually. Has option to print out Bingo cards and call list. 38 sectors

0104 DRAGON STORM##XB Joysticks rqd
A "Freeware" program by Howard Unan and L.E. Benson. Cute game where you fight the dragon and rescue the princess.
49 sectors

1073 CREATE##XB
Create your own pictures and save them to disk. Another excellent program by Andrew Kirase. 26 sectors

4147 EXTENDED BASIC EDITOR ASSEMBLER##XB
An excellent program that will allow you to load your E/A Programs through Extended Basic without an EA cartridge. Permission to be used in HUG Library granted by Paolo Bagnaresi. 360 sectors
REQUIRES 1 SSSD DEDICATED DISK

4148 MINI-MEMORY E/A & TI/WRITER Mini Memory cartridge rqd.
This program written and released to HUG Library by Jean Delestienne will allow you to use your Mini-Memory cartridge to access Editor Assembler and TI/Writer wthout cartridges. File names to be loaded into Mini-Memory are ME/A for Editor Assembler and MWRITER for TI/Writer. Program names are UTIL. 345 sectors.
REQUIRES 1 SSSD DEDICATED DISK

4149 EDITOR ASSEMBLER PILOT##E/A rqd.
Now you can run PILOT programs without a P-Code card using only your Editor Assembler. Comes with documentation that can be printed out. Programs can be written with either TI/Writer or E/A and executed with this program.
694 sectors
REQUIRES 2 SSSD DISKS##1 FOR PROGRAM & 1 FOR DOCUMENTATION

4150 SYSTEM##XB Printer optional
This program, written by Barry Boone, allows you to save your XB machine language programs as XB program files. Comes with complete documentation.
23 sectors

5229 ORGAN II##XB Printer optional
This "Freeware" program by Jerome Trinkl plays some interesting music. Program written in assembly language. Comes complete with documentation that shows you how to program your own songs. 156 SECTORS

5230 SINFONIA##XB
Beautiful music & cute graphics written by Chris & Andrew Kirase. A nice rendition of Bach's Sinfonia #15.
90 sectors

5231 TEXAS OUR TEXAS##XB
Bill Knecht's contribution for Texas's Sesquicentennial. Comes complete with bluebonnets and sing-along-words.
29 sectors.