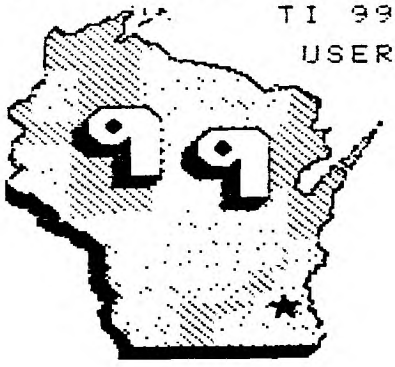
HOME COMPUTER USERS SPOTLIGHT

A MONTHLY PUBLICATION
OF THE MILWAUKEE AREA
TI 99/4A
USER GROUP



4122 No. Glenway - Wauwatosa WI 53222

President.....	James Schroeder	264-4735
Vice-President.	Donald Walden	529-2173
Treasurer.....	Phil Norton	462-8954
Secretary.....	George Kasica	321-7558
Librarians.....	E.J. VonDerEhe	549-0593
.....	Fred Pabian	327-3618
Newsletter.....	Gene Hitz	535-0133
.....	Jerry Trinkl	327-0170
Forth Info.....	Gene Hitz	
Assembly SIG...	Jerry Trinkl	

Membership in the Milwaukee Area 99/4A U.G. is open to all interested in the solid performing Texas Instrument's 99/4A computer and the shared knowledge and good fun it provides.

Annual Dues.....Individuals - $10.00
.....Families - $15.00

We meet on the SECOND SATURDAY each month in the lower level of WAUWATOSA S&L located at 7500 W. State Street 1:00 to 4:00 P.M.

MARCH 86

SERVICE CENTER CLOSING

The T I service exchange center in Bishops Woods, Brookfield will close on March 21 1986.

Any further service should be referred to the center in Lubbock Texas. Their rates are said to be cheaper but no doubt will be offset by postage and handling costs.

Their address is:

T.I. Inc.
2305 N. University Ave.
Lubbock,TX 79415

or contact the toll-free 1-800-TICARES.

U P D A T E 4 A / T A L K

The long awaited version 1.4 of the popular terminal program 4A/TALK is now available.

It has the following features:

- Loads from all three disk controllers
- Auto logs to disk with the option of changing file names to keep one file from getting to large.
- Autoload from Corcomps option 1 (just press 1 from Corcomp screen as if you were loading the disk manager.)
- From the delete file option you can now change file protection and filename.

All this can be yours if you send your 4A/TALK disk along with $5.00 to:

DataBioTics Inc.
22411 Moutian Laurel Way
Diamond Bar, CA 91765

NLQ FOR THE GEMINI 10X

Johnson Space Center

A new product from Germany has arrived on the American Market and is proving to be a big hit with Gemini 10X owners. It's a plug in chip that allows the 10X to produce Letter Quality Print that rivals the SG-10. The chip has been available in Europe for over a year, so you can be assured that all the bugs have been worked out. I have one on my 10X and couldn't be happier with it's performance.

The NLQ mode can be involked by changing Dip-switch settings or by simple printer commands in your program. I had some samples at the last meeting and everyone who saw them thought they were super. If you missed it, here is a sample of what the chip can do.

The letters "w" and "p" are fabulous. Print is very near the true typewriter. You would be hard pressed to tell the diference. Letters are round, not square. A plus for readability.

The letters are formed during two passes across the paper. Of course, this reduces the print speed to about half. The second pass completes the distenders and emphasizes the print. The print quality is remarkable.

Just about anyone can install it. It takes about 20 minutes. The chip replaces two integrated circuits found on the board right behind the carriage.

The NLQ type face resides where the ITALIC face used to be. In fact, the codes that involked ITALIC print now involk NLQ print. SO GOODBYE ITALIC PRINT. Ihave tried the chip with TI-WRITER and have experienced no problems.

Now you are asking yourself, how much is this chip? The answer is $57.50 each or a group discount is available if we buy several at one time. It becomes much cheaper than $230.00 for a new SG-10.

The NLQ chip is sold by:

E.S.P. CORPORATION
7900 NORTH TAMIAMI TRAIL SARASOTA, FL. 34243 PHONE 813-355-6797

About two years ago, Star Micronics changed one of the chips in the 10X. This change makes it necessary for you to open your printer to determine which chip is needed in yours. Look over the board in your printer. If you find a chip labled D78016176 then you need chip number G10M. If you find a chip labled D78006 then you need NLQ chip number 610. You will have one or the other in your printer. It may sound confusing but, a call to the company will result in instant help. Once your printer has the chip in place, you will be very pleased with the enhancement.

CORE ON SG-10, TRY THIS LITTLE 10X ON FOR SIZE.

CONVERTING RF MODULATOR FOR UNIVERSAL USE

by Harold Hoyt

The RF Modulator can easily be modified to input video and audio signals to a video monitor or a VCR.

Pop the top cover off of the RF Modulator. Position an RCA stereo phono jack, Radio Shack part #274-332, on the corner of the cover where the cable enters the Modulator. Drill 2 holes for the pins of the connector. Make the holes large enough to allow position adjustment. Verify that the position is correct and drill 4 #29 drill holes using the phono jack as a template. Attach the phono jack to the cover. Chop off screws and cut the partition if required to avoid interference.

Find where the Video and audio wires enter the printed circuit board (These are marked with the words video and audio on the top of the board). Drill a #43 hole next to each of these leads from the copper side, being careful not to cut a trace. Cut and strip the ends of two short pieces of wire. Push the wire ends through the top of the board and solder the ends of the wires to the jacks. Replace the cover.

Plug in a stereo phono cord with male RCA phono plugs on each end. Code red for video and black for audio. This cord may then may then be plugged into either a VCR or Video Monitor.

This device may then be used either in the Modulator mode or as a Monitor/VCR input without changing cords.

TINYCAL

The program can probably be modified to operate using any dotmatrix printer that includes super/subscript characters. Although it is designed for RS232 operation, users may use parallel printers simply by changing the I/O characteristics in line 280. It is in line 280 that the super/subscriptcharactes are accessed. This line may be used as the basis for "miniaturizing" printer output for many programs, including disk catalog programs.

The program requires Extended Basic.

Ed's note: If your interested in more programs of this type the name and addres of the author is included in the program. You should write to him for his freeware offerings.

Also if you do not have a subscription to MICROpendium you are missing out on a lot of news, views, programs, and information. I've been a subscriber since they first started publishing, and I consider it to be a best buy for 99/4A users.

CONTACT: MICROpendium
P.O. Box 1343
Round Rock, Tx 78680

```
100 !*******************
110 !*      TINY       *
120 !*EPSON/TI CALENDER*
130 !*       BY        *
140 !*RICHARD J. BAILEY*
150 !*68A CHURCH STREET*
160 !*GONIC, N)H) 03867*
170 !*******************
180 DIM T(12),D(12),MO$(12):
: CALL CLEAR :: CALL SCREEN(
2):: FOR I=0 TO 14 :: CALL C
OLOR(I,16,2):: NEXT I
190 FOR I=1 TO 12 :: READ T(
I),D(I),MO$(I):: NEXT I
200 DATA 7,31,JANUARY,30,28,
FEBRUARY,8,31,MARCH,32,30,AP
RIL,9,31,MAY,32,30,JUNE
210 DATA 9,31,JULY,31,31,AUG
UST,6,30,SEPTEMBER,30,31,OCT
OBER,7,30,NOVEMBER,30,31,DEC
EMBER
220 DISPLAY AT(5,14):"TINY":
"       EPSON/T.I. CALENDAR":
"":"":"**THIS PROGRAM WILL P
RINT A":"  CALENDAR FOR ANY
YEAR FROM":"  1776 TO 2099."
230 DISPLAY AT(13,1):"**SET
TOP OF FORM AND ENTER":"  TH
E YEAR AS A FOUR DIGIT":"  N
UMBER   (ex. 1985)   OR":"
JUST ENTER TO EXIT PROGRAM"
240 DISPLAY AT(19,1)BEEP:"**
ENTER CALENDAR YEAR" :: ACCE
PT AT(19,24)SIZE(4)VALIDATE(
DIGIT):Y$
250 IF Y$="" THEN CALL CLEAR
 :: END ELSE Y=VAL(Y$):: IF
Y(1776 OR Y)2099 THEN 240
260 IF INT(Y/4)*4=Y AND NOT(
INT(Y/100)*100=Y AND INT(Y/4
00)*400()Y)THEN D(2)=29
270 DI=Y-1906+INT((Y-1901)/4
):: D(0)=DI+1-(INT(DI/7)*7)
280 M2=0 :: OPEN #1:"PIO" ::
 PRINT #1:CHR$(27);"S";CHR$(
1);CHR$(15);CHR$(27);"3";CHR
$(14):TAB(19);Y
290 FOR I=1 TO 12 STEP 2 ::
PRINT #1:TAB(T(I));MO$(I);TA
B(T(I+1));MO$(I+1)
300 J,K=1 :: A,M1=D(I-1)+M2
:: B,M2=M1+D(I)
310 PRINT #1:CHR$(27);"3";CH
R$(8);"S  M  T  W  T  F  S
  S  M  T  W  T  F  S":CHR$(
27);"3";CHR$(14);"-  -  -  -
  -  -  -  -  -  -  -  -
-  -"
320 IF J)D(I)THEN 330 :: IF
A)7 THEN A=A-7 :: GOTO 320 E
LSE PRINT #1:TAB(A*3-2);STR$
(J);:: IF A=7 THEN 330 ELSE
A=A+1 :: J=J+1 :: GOTO 320
330 IF K)D(I+1)THEN 340 :: I
F B)7 THEN B=B-7 :: GOTO 330
 ELSE PRINT #1:TAB(21+B*3);S
TR$(K);:: IF B=7 THEN 340 EL
SE B=B+1 :: K=K+1 :: GOTO 33
0
340 IF J)D(I)AND K)D(I+1)THE
N 350 ELSE PRINT #1:"" :: A=
A+1 :: B=B+1 :: J=J+1 :: K=K
+1 :: GOTO 320
350 PRINT #1:"" :: NEXT I ::
 PRINT #1:"":CHR$(27);"@" ::
 CLOSE #1 :: RESTORE :: GOTO
 190
```

BINARY SEARCH

by DAVID ROMER
NEW HORIZONS

One of the best uses of the power and speed of a computer is in the manipulation; storing, sorting and searching, of sets of information. Data base or file management programs are standard pieces of software for any computer system. Those kinds of programs, and others, always include some way to search the list of information or files to retrieve a particular item. The time required to execute a search depends greatly on the search method used, as certain methods are much faster than others. The speed of a search method is determined by the number of comparisons required to find a match or determine that a match does not exist. The fastest, most efficient method of searching a list is called a BINARY SEARCH.

Suppose, for a moment, that you are in a strange city (fairly small, just 50,000 telephones) and you find it is necessary to look up a phone number in the phone book. You open the phone book to find that there seems to be no rhyme or reason to the various entries; A's are mixed with U's, D's with S's. The entries seem to be in random order, certainly not the nice neat alphabetical by last name then first name listing you are used to seeing. (you conclude that this is due to the breakup of AT&T). At that point, to find the number you want, you must do a LINEAR SEARCH, the slowest search method. You must start at the beginning of the list and check EACH entry in succession until you find what you want. If you are lucky you will find what you want near the beginning of the list. However, it is entirely possible that you might have to make 50,000 comparisons to locate a match. You definitely will have to look at every entry to determine that there is NO match. Certainly a time consuming job.

Now had that phone book been in the usual alphabetical order, you would have estimated about where in the book the name you want might be. Then, based on the ORDER of the alphabet, you would go forward or backward narrowing the pages then names to be searched until the name and number you want is found. In effect, you were performing a BINARY SEARCH.

To use a binary search, the list to be searched must be in some kind of order. Alphabetical or numeric, ascending or descending is of no consequence as long as the order is known. If a computer is to perform a binary search you also need to know the number of items in the list. The basic concept of the binary search is to successively reduce the size of the list by eliminating, based on the order, large parts of the list were the item can not be until a match is found or the list is exhausted. The computer performs this task by dividing the list by 2 (thus the name binary) finding the midpoint. It then checks the item at the midpoint for a match. Based on whether a match is above or below the midpoint, the midpoint becomes either the start or the end of a new list HALF the length of the original. The same procedure is followed with the new list. This successive division of the list by 2 continues until a match is found or not found. Using this method any number in our imaginary 50,000 item phone book can be found by making a maximum of 17 comparisons.

Languages not presently covered at our S.I.G. .....

(Reprint from APL SIG newsletter via MSP 99 newsletter)

APL, BASIC, COBOL, FORTRAN, PASCAL, these programming languages are well known and more or less loved throughout the computer industry. There are numerous other languages however, that are less well known yet still have ardent devotees. In fact, these little known languages generally have the most fanatic admirers For those who wish to know more about these relatively obscure languages, and why they are so obscure, we are publishing the following catalogue.

C-
This language is named for the grade received by its creator when he submitted it as a class project in a graduate programming class. C- is best described as a "low level" programming language. In general, the language requires more C- statements than machine code instructions to execute a given task. In this respect, it is very similar to COBOL.

DOGO
Developed at MIDT Massachusetts Institute of Obedience Training DOGO heralds a new era of computer literate pets. DOGO commands include SIT, HEEL, STAY, PLAY_DEAD and ROLL_OVER. An innovative feature of DOGO is "puppy graphics" a small cocker spaniel that occasionally leaves deposits as it travels across the screen.

FIFTH
FIFTH is a precise mathematical language in which data types refer to quantities. The data types range from CC, OUNCE, SHOT and JIGGER to FIFTH (hence the name of the language), MAGNUM, LITER and BLOTTO. Commands refer to ingredients such as CHABLIS CABERNET, GIN, VERMOUTH, VODKA, SCOTCH, BOURBON, CANADIAN, BUD, COORS, EVER_CLEAR and WHAT_ELSE'S_AROUND.
The many versions of the FIFTH language reflect the sophistication and financial status of the user. Commands in the ELITE dialect include: VSOP, LAFITE and WAITERS_RECOMMENDATION. The GUTTER dialect commands include; THUNDERBIRD, RIPPLE and HOUSE-RED. The BEER dialect is a particular favorite of frustrated FIFTH programmers who end up using this language.

LAIDBACK
This language was developed at the Marin County Center for T'ai Chi Mellowness & Computer Programming (now defunct) as an alternative to the more intense atmosphere in nearby Silicon Valley. The center was ideal for programmers who liked to soak in hot-tubs while they worked. Unfortunately, few programmers could could survive there because the center outlawed Pizza and Coca-Cola in favor of Tofu and Perrier.
Many mourn the demise of LAIDBACK because of its reputation as a gentle and non-threatening language since all of its error messages are in lower case. For example, LAIDBACK responded to syntax errors with the message:
"I hate to bother you, but I just can't relate to that. Could you possibly find time to try it again?"

LITHP
This otherwize unremarkable language is distinguished by the absence of an "S" in its character set. Programmers and users must substitute "TH". LITHP is said to be useful in prothething liththth. This language was developed in San Francisco.

REAGAN
This language was developed in California, but now is widely used in Washington D.C. It is the current subset to the international bureaucratic language known as DOUBLESPEAK. Commands include DEFENSE_ENHANCEMENT, STOCKMAN, CAP_WEINBERGER, CHEVIOT, MALCOLM_BALDRIDGE, CHOP_WOOD, LAXALT and SCENARIO. BURFORD and WATT have been removed from the commands while there is now a current effort to add MEESE.
The operating system used is NEW_RIGHT and designated memory is THE_RANCH. The compile SCENARIO is a compile with NEWS, followed by a link with BONIO resulting in a SNOOZE. BUGS (program bugs) are removed with a SPENCER command. A REAGAN program commences with LANDSLIDE and terminates with SENILITY.

RENE
Named after the famous French philosopher and mathematician Rene DesCaters. RENE is a language used for artificial intelligence. The language is being developed at the Chicago Center of Machine Politics and Programming under a grant from the Jane Byrne Victory Fund. A spokesman described the language as "Just as great as dis (sic) great city of ours."
The center is very pleased with progress to date. They say they have almost succeeded in getting a VAX to think. However sources inside the organization say that each time the machine fails to think it ceases to exist.

SATRE
Named for the late existential philosopher. SATRE is extremely unstructured. SATRE Statements have no purpose: they just are Thus SATRE programs are left to define their own functions and SATRE programmers tend to be boring and depressing and are no fun at parties.

SIMPLE
SIMPLE is an acronym for Sheer Idiot's Monopurpose Programming Linguistic Environment. This language, developed at the Hanover College for Technical Misfits, was designed to make impossible writing code with errors. The statements are therefore confined to BEGIN, END and STOP. No matter how you arrange the statements you can't make a syntax error.

SLOBOL
SLOBOL is best known for the speed or lack of, of the compiler. Although many compilers allow you to take a coffee break while they compile, the SLOBOL compiler lets you to travel to Columbia to pick the coffee. Forty three programmers are known to have died of boredom sitting at their terminals waiting for a SLOBOL program to compile.

VALGOL
From its modest beginnings in Southern California's San Fernando Valley, VALGOL is enjoying a dramatic surge of popularity across the industry. VALGOL commands include: REALLY, LIKE, WELL and Y'KNOW. Variables are assigned with the =LIKE and =TOTALLY operators. Other operators include the California Boleans: AX & NOWAY. Repetitions of code are handled in the FOR - SURE loops. Here is a sample program:

```
      LIKE, Y'KNOW (I MEAN) START
  IF PIZZA     =LIKE BITCHEN ONE
  GUY          =LIKE TUBULAR ONE
  VALLEY GIRL =LIKE TOTAL*MAX
                    TOTURE)*2

  THEN
      FOR I =LIKE 1 TO OH*MAYBE 100
          DOUGH  - CUTTY*$2)
          BARF(I) =TOTALLY GROSS(OUT)
      SURE

      LIKE BAG THIS PROGRAM
      REALLY
      LIKE TOTALLY (Y'KNOW)
      IM*SURE
      GOTO THE MALL
```

VALGOL is characterized by its unfriendly error messages. For example when the user makes a syntax error, the interpreter displays the message:
GAG ME WITH A SPOON!!

```
10 REM  ******************
20 REM  *                *
30 REM  *  B U D G E T   *
40 REM  *                *
50 REM  ******************
60 REM
100 DATA UTILITIES,GAS,ELECT
RICITY,WATER,SEWER,CABLE
110 DATA EXP. MAINTENANCE,HO
ME ONE,,AUTO INS.,MORTGAGE,L
IFE INS.
120 DATA SUPPLIES,TOOLS,PAIN
T,HARDWARE,YARD,AUTO
130 DATA REPAIRS,MISC.,PAINTI
NG,OUTDOOR,PLUMBING,ELECTRICA
L
140 DATA NECESSITIES,FOOD,CL
OTHING,FURNITURE,ENTERTAINME
NT,MEDICAL
150 DATA MISC.,NEWSPAPERS,HO
BBIES,SAVINGS,EDUCATION,VACA
TION
160 DATA MONTH..,ANNUAL," TO
TAL"," "," SEL
ECT #,CATEGORY,SUB-CATEGORY,
" PRESS ENTER"
170 DATA JANUARY,FEBRUARY,MA
RCH,APRIL,MAY,JUNE,JULY,AUG
UST,SEPTEMBER,OCTOBER,NOVEMBE
R,DECEMBER
180 DIM F$(5,5),F(12,6,5),M$
(12),B(5,5)
190 FOR I=0 TO 5
200 FOR A=0 TO 5
210 READ F$(I,A)
220 NEXT A
230 NEXT I
240 READ A$,B$,D$,Z$,S$,C$,U
$,E$
250 FOR I=1 TO 12
260 READ M$(I)
270 NEXT I
280 M=1
290 GOTO 1110
300 CALL CLEAR
310 GOSUB 2320
320 GOSUB 2370
330 X=1
340 K=8
350 N$=A$&" "&M$(M)
360 GOSUB 1050
370 X=3
380 K=3
390 N$=S$&A$&"(1-12):"
400 GOSUB 1050
410 R=1
420 \=47
430 K=26
440 GOSUB 2120
450 IF N$="" THEN 510
460 M=VAL(N$)
470 X=1
480 K=16
490 N$=M$(M)
500 GOSUB 1050
510 CALL HCHAR(3,3,32,24)
520 X=20
530 K=6
540 N$=S$&C$&":"
550 GOSUB 1050
560 K=25
570 \=48
580 X=20
590 GOSUB 2120
600 C=VAL(N$)
610 IF (C<0)+(C>5)THEN 560
620 CALL CLEAR
630 PRINT TAB(10);F$(C,0): :
:
640 FOR I=1 TO 5
650 PRINT :I;" ";F$(C,I)
660 NEXT I
670 PRINT : : : : : : : :
680 GOSUB 2370
690 R=2
700 X=20
710 K=6
720 N$=U$&":"
730 GOSUB 1050
740 X=20
750 K=25
760 GOSUB 2120
770 X=22
780 K=4
790  =VAL(N$)
800 CALL HCHAR(X,K,32,33-K)
810 N$=F$(C, )&D$&": "
820 GOSUB 1050
830 K=25
840 \=46
850 GOSUB 2120
860 P=VAL(N$)
870 F(M,C, )=F(M,C, )+P
880 F(M,C,0)=F(M,C,0)+P
890 F(0,C,0)=F(0,C,0)+P
900 F(0,C, )=F(0,C, )+P
910 F(M,6,0)=F(M,6,0)+P
920 W=W+P
930 X=2*  +5
940 K=22
950 N$=STR$(F(M,C, ))
960 CALL HCHAR(X,K,32,33-K)
970 GOSUB 1050
980 X=17
990 K=4
1000 N$=F$(C,0)&D$&":  "&STR
$(F(M,C,D))
1010 GOSUB 1050
1020 CALL HCHAR(22,4,32,29)
1030 CALL HCHAR(23,25,32)
1040 ON R GOTO 560,740
1050 CALL HCHAR(X,K,32,24)
1060 FOR J=1 TO LEN(N$)
1070 CALL HCHAR(X,J+K,ASC(SE
G$(N$,J,1)),1)
1080 NEXT J
1090 RETURN
1100 IF R=2 THEN 300
1110 CALL CLEAR
1120 PRINT "    B U D G E T"
: :" 1. LOAD  BUDGET":" 2. I
NPUT BUDGET :" 3. SAVE BUDGE
T"
1130 PRINT :"     EXPENCES":
 :" 4. LOAD FILE DATA":" 5
. INPUT NEW DATA":" 6. SAVE
 FILE DATA"
1140 PRINT :"      DISPLAY D
ATA": :"  7. MONTHLY  ":D$;
"S":"  8. ANNUAL  ":D$;"S"
:"  9. CATEGORY ":D$;"S": :
"  0. QUIT": :
1150 INPUT "          CHOICE "
:X
1160 IF (X<0)+(X>9)THEN 1110
1170 CALL CLEAR
1180 ON X+1 GOTO 1190,2780,2
890,1660,300,1920,1220,
1410,1440
1190 INPUT "FINISHED ? (Y/N)
":N$
1200 IF N$<>"Y" THEN 1110
1210 END
1220 INPUT " SELECT MONTH (1
-12): ":M
1230 PRINT : :TAB(4);A$,M$(M
);
1240 A=0
1250 FOR C=0 TO 5
1260 PRINT :C;;F$(C,0);TAB(1
0);"$";F(M,C,0);TAB(20);"$";
INT(B(C,0)*25/3)*.01*-(M<>0)
-(M=0)*(B(C,0))
1270 A=A+F(M,C,0)
1280 NEXT C
1290 PRINT :TAB(3);D$;":";TA
B(10);"$";A;TAB(20);"$";(M=0
)*-T-(M<>0)*V;  ;U$;
1300 INPUT "   ":C
1310 D=0
1320 PRINT :F$(C,0);" ";M$(M
);
1330 FOR I=1 TO 5
1340 @=INT(B(C,I)*25/3)*.01*
-(M<>0)-B(C,I)*(M=0)
1350 PRINT :I;;F$(C,I);TAB(1
0);"$";F(M,C,I);TAB(20);"$";
@
1360 D=D+@
1370 NEXT I
1380 PRINT :TAB(3);D$;":";TA
B(10);"$";F(M,C,0);TAB(20);"
$";(M<>0)*-D-(M=0)*B(C,0);  :
E$;
1390 INPUT " ":N$
1400 GOTO 1110
1410 M=0
1420 PRINT : :TAB(4);B$,D$;
:
1430 GOTO 1240
1440 PRINT TAB(5);C$;D$;  :
1450 GOSUB 2320
1460 INPUT "  SELECT CATEGORY:
 ":C
1470 CALL CLEAR
1480 PRINT :TAB(5);F$(C,0):
:
1490 FOR J=1 TO 12
1500 PRINT M$(J);TAB(10);"$"
;F(J,C,0);TAB(20);"$";INT(B(
C,0)*25/3)*.01
1510 NEXT J
1520 PRINT :D$;TAB(10);"$";F
(0,C,0);TAB(20);"$";B(C,0):
:U$;" ";F$(C,0): :
1530 FOR I=1 TO 5
1540 PRINT I;"   ";F$(C,I)
1550 NEXT I
1560 INPUT " ENTER TO RETURN
   ";N$
1570 IF N$="" THEN 1110
1580 A=VAL(N$)
1590 U=0
1600 CALL CLEAR
1610 PRINT :TAB(5);F$(C,A):
:
1620 FOR J=1 TO 12
1630 PRINT M$(J);TAB(10);"$"
;F(J,C,A);TAB(20);"$";INT(B(
C,A)*25/3)*.01
1640 NEXT J
1650 PRINT :D$;TAB(10);"$";F
(0,C,A);TAB(20);"$";B(C,A):
: :E$; : :
1660 INPUT " ":N$
1670 GOTO 1110
1680 OPEN #1:"DSK1.XDATA",IN
TERNAL,INPUT ,FIXED 192
1690 INPUT #1:[,],
1700 \=2
1710 FOR A=0 TO 5
1720 FOR J=[ TO ]
1730 FOR I=1 TO 5
1740 \=\+1
1750 IF \<21 THEN 1790
1760 \=0
1770 INPUT #1:F(J,A,I)
1780 GOTO 1800
1790 INPUT #1:F(J,A,I),
1800 F(J,A,0)=F(J,A,0)+F(J,A
,I)
1810 F(0,A,I)=F(0,A,I)+F(J,A
,I)
1820 NEXT I
1830 F(0,A,0)=F(0,A,0)+F(J,A
,0)
1840 F(J,6,0)=F(J,6,0)+F(J,A
,0)
1850 NEXT J
1860 W=W+F(0,A,0)
1870 NEXT A
1880 CLOSE #1
1890 PRINT " FILE DATES": :"
 FROM   ";M$(C); :" TO     "
;M$(J); : : : :E$;
1900 INPUT " ":N$
1910 GOTO 1110
1920 PRINT " STARTING WITH";
A$; :" (1-12) ?";
1930 INPUT "  ":[
1940 PRINT : :" ENDING WITH"
;A$; :" (1-12) ?";
1950 INPUT "  ":]
1960 OPEN #1:"DSK1.XDATA",IN
TERNAL,OUTPUT,FIXED 192
1970 \=2
1980 PRINT #1:[,],
1990 FOR J=[ TO ]
2000 FOR A=0 TO 5
2010 FOR I=1 TO 5
2020 \=\+1
2030 IF \<21 THEN 2070
2040 \=0
2050 PRINT #1:F(J,I,A)
2060 GOTO 2080
2070 PRINT #1:F(J,I,A),
2080 NEXT I
2090 NEXT A
2100 NEXT J
2110 GOTO 2990
2120 Z=K
2130 N$=""
2140 CALL SCRN(-200,1397,0)
2150 CALL KEY(4,G,H)
2160 IF H<1 THEN 2150
2170 IF G=13 THEN 2250
2180 IF G=140 THEN 1100
2190 IF G=136 THEN 2270
2200 IF (G<\)+(G>57)THEN 215
0
2210 CALL HCHAR(X,Z,G)
2220 N$=N$&CHR$(G)
2230 Z=Z+1+(Z=32)
2240 GOTO 2150
2250 CALL HCHAR(X,K,32,33-K)
```

```
2260 RETURN
2270 IF Z=K THEN 2150
2280 CALL HCHAR(X,Z-1,32)
2290 N$=SEG$(N$,1,LEN(N$)-1)
2300 Z=Z-1
2310 GOTO 2150
2320 FOR I=0 TO 5
2330 PRINT #1:I;TAB(5);F$(I,0)
2340 NEXT I
2350 PRINT #1:::::::::
2360 RETURN
2370 N$="(FNCT) PROCEED WHEN
 FINISHED"
2380 X=24
2390 K=2
2400 GOSUB 1060
2410 RETURN
2420 PRINT TAB(7);B$;" BUDGE
T"; :
2430 GOSUB 2320
2440 GOSUB 2370
2450 FOR I=0 TO 5
2460 X=19
2470 N$=F$(I,0)
2480 K=4
2490 GOSUB 1050
2500 FOR A=1 TO 5
2510 X=21
2520 K=4
2530 N$=F$(I,A)&SEG$(Z$,1,15
-LEN(F$(I,A)))&STR$(B(I,A))
2540 GOSUB 1050
2550 X=22
2560 Z=20
2570 GOSUB 2130
2580 IF N$="" THEN 2630
2590 D=B(I,A)
2600 B(I,A)=VAL(N$)
2610 B(I,0)=B(I,0)+B(I,A)-D
2620 T=T+B(I,A)-D
2630 CALL HCHAR(21,1,32,64)
2640 NEXT A
2650 N$="$"&STR$(B(I,0))
2660 K=20
2670 X=6+2*I
2680 GOSUB 1050
2690 V=INT(T*25/3)*.01
2700 NEXT I
2710 K=4
2720 N$=O$&"          $"&STR$
(T)
2730 X=19
2740 K=5
2750 GOSUB 1050
2760 CALL KEY(0,G,H)
2770 IF H THEN 1110 ELSE 276
0
2780 OPEN #1:"DSK1.BDATA",IN
TERNAL,INPUT ,FIXED 192
2790 FOR J=0 TO 3 STEP 3
2800 FOR A=J TO J+2
2810 FOR I=0 TO 5
2820 INPUT #1:B(A,I),
2830 NEXT I
2840 NEXT A
2850 INPUT #1:T
2860 NEXT J
2870 V=INT(T*25/3)*.01
2880 GOTO 2980
2890 OPEN #1:"DSK1.BDATA",IN
TERNAL,OUTPUT,FIXED 192
2900 FOR J=0 TO 3 STEP 3
2910 FOR A=J TO J+2
2920 FOR I=0 TO 5
2930 PRINT #1:B(A,I),
2940 NEXT I
2950 NEXT A
2960 PRINT #1:T
2970 NEXT J
2980 CLOSE #1
2990 GOTO 1110
```

INTERRUPT ME !

How to use (and abuse) your Computer's interrupt capabilities

-By Jim Hess

Chapter 1

Whether you program in assembly language on a TI99/4A computer or any other, you probably have come across a few reference to "Interrupt Handling". If you react to the unknown the way I do, you probably skipped it, because you didn't need to know what it was all about. Once I had some of the basics of assembly language down I came back to it. I had just seen an interrupt driven clock timer program and decided it was time to see what made it tick. (sorry, that was irresistable) In this first chapter, we'll see what interrupts are all about; why are they necessary, when do they come in handy, and a beginning on the Do's and Dont's of programming with interrupts.

*The beast we all call "Computer" is a narrow minded individual. It does what you tell it and does not pay attention to anything else. From the time you turn the power on it is performing a task In the case of the TI99/4A, a 'power-up' routine is run, checking all the closets and corners to see what peripherals are connected then displaying the color bar screen, waiting for you to press a key. The problem with single-mindedness is that there are always a couple of things you would like to be able to do at any point in a program. For instance, you would like to be able to QUIT at any time. But if your program is busy adding numbers or printing text or displaying graphics, it doesn't care about your desire to QUIT. It's busy. The solution is to have a timer stop the program on a regular schedule, and check to see if there are alternative tasks to perform, such as paying attention to the goof trying to QUIT. The TI99/4A and most modern computers have that capability built into the operating system. Normally it is used to implement internal functions, as system reset or peripheral data transfers (disk or cassette). You don't want your program running before your disk drive is finished transferring data, for instance. But in the case of our TI99/4A there is a provision made for the user to add his/her own interrupt. There is an address located in the onboard CPU RAM which is checked 60 times per second. If there is a memory address 'poked' into here, the program will branch to that address and perform what ever task is placed there. So, you could have a keyboard scan routine there to check for a 'QUIT' command, or a screen color or character change command, or any program your lil' heart desires. There is a very SERIOUS problem with branching to your interrupt program. See if you can figure it out. I'll start Chapter 2 with that subject. *** -JH ****

TIPS FROM THE TIGERCUB

#28

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

NUTS & BOLTS DISK No. 2 is now ready, and I think it's better than the first one. It contains 100 utility subprograms in merge format, including many new character fonts and screen display routines as well as 2-dimensional array sorts, variable line numbers in GOSUB, GOTO and RESTORE, on-screen editing and much, much more. The price is $19.95 postpaid, or you can order both Nuts & Bolts disks for $37 ppd.

And I have put together 18 different collection disks each containing 5 or 6 of my catalog programs for just $12 postpaid. The programs on each disk are all of the same category, and I have filled up the rest of the disk with public domain programs of the same category, as a bonus.
I want to make it very plain that I am NOT - repeat, NOT - selling public domain programs! My own programs on these disks are offered at a great discount and the public domain programs are just thrown in for free! Together with this issue of the Tips I am mailing to each user's group a copy of my catalog #6 with an added page describing these new offerings, and a rebate offer to user's groups.

My catalog will be sent to individuals for $1, which is deductable from your first order. If you already have my catalog #6, the added page will be sent to you free on request.

My full disk collections will now be available to bona-fide retailers at standard wholesale prices. Inquiries on your letterhead are invited.

And so, on to old business. Yes, I know that RESequencing a program does not resequence references to line numbers in REMs. I just forgot! In line 270 of the Menu Loader in Tips #27, the reference should be to lines 280 and 290, of course.
While programming the file reader in that menu loader, I ran into a peculiarity of the TI-99/4A that surprised most of the expert programmers whom I called for help. When you "read blind" you must read everything as a string, because attempting to read a string as numeric will crash the program. This is no problem with DISPLAY files - but when I tried it with INTERNAL files, I got the strangest garbage! My solution (not quite foolproof) was to identify a record as numeric if it was 8 bytes long and contained an ASCII out of printable range, and then RESTORE the file, read back to that point and re-read it as numeric. Not very efficient!
The following routine will save a numeric input in an internal file, read it back out as a string, show you the way it was saved, and then attempt to translate it back to numeric. It works for positive and negative integers or non-integers of not less than -99, but not for less than that.

```
100 INPUT X :: OPEN #1:"DSK1
.TEST",INTERNAL,OUTPUT :: PR
INT #1:X :: CLOSE #1
110 OPEN #1:"DSK1.TEST",INTE
RNAL,INPUT :: INPUT #1:A$ ::
 PRINT A$ :: CLOSE #1
120 FOR J=1 TO 8 :: PRINT AS
C(SEG$(A$,J,1));:: NEXT J
130 FOR J=1 TO 8 :: A(J)=ASC
(SEG$(A$,J,1)):: NEXT J
140 X=A(1)-63 :: IF X<73 THE
N 150
142 X=192-A(1):: N$="-" :: F
OR J=2 TO X+1 :: N$=N$&STR$(
256-A(J)):: NEXT J :: GOTO 1
60
150 FOR J=2 TO X+1 :: N$=N$&
STR$(A(J)):: NEXT J
160 IF A(J)<>0 THEN N$=N$&".
"&STR$(A(J))
170 J=J+1 :: IF A(J)<>0 THEN
 N$=N$&STR$(A(J)):: GOTO 170
180 N=VAL(N$):: N$="" :: PRI
NT N :: GOTO 100
```

So, here is another Tigercub Challenge! Can you fix it? Let's HEAR from you this time!

Another problem that I ran into was in recovering from an I/O error. When ON ERROR is used to prevent crashing on such an error, the file is "ajar" - you can't close it and you can't open it. My solution was to simply RUN the program again - and this will show you how the pre-scan speeds that up. Since then, I have learned of three other ways. The method described in the Sydney (Australia) newsletter is a bit complicated, but Irwin Hott gave me a simple solution - just increment the file number! Works fine if you don't increment it into the number of another open file on the disk. Chuck Grimes gave me an even better way - open and close anything else, even "PIO"! Example -

```
100 ON ERROR 110 :: OPEN #1:
"DSK1.TEST",OUTPUT :: PRINT
"CONTINUE PROGRAM" :: END
110 OPEN #1:"PIO" :: CLOSE #
1 :: PRINT "I/O ERROR":"CHEC
K DISK AND DRIVE":"THEN PRES
S ANY KEY" :: ON ERROR STOP
120 CALL KEY(0,K,S):: IF S=0
 THEN 120 ELSE 100
```

There is a reason for that ON ERROR STOP, and it's why I don't use ON ERROR if I can avoid it. When an error occurs, the program goes to the line number specified by the last open ON ERROR statement, takes whatever action is directed by that line, and RETURNs as directed. If the error was not one that you expected to happpen, the results can be very confusing!
For that reason, when you set out to modify a program, the first thing you should do is delete, temporarily, all the ON ERROR statements. The next thing you should do, if the program has a routine to turn off the pre-scan, is to disable that. Otherwise, you will be driven crazy by invalid SYNTAX ERROR messages and other strange happenings.
The third thing you should do is to make a list of all the lines that a GOTO or GOSUB goes to, so you don't delete or change them. And here is a program to do just that for you -

```
100 !GO-SEARCH by Jim Peters
on searches a MERGE format f
ile, finds all line numbers
containing a jump, sorts int
o "to" line number sequence,
110 !prints "to" line number
, statement (GO, GOTO or GOS
UB) and "from" line number
120 DIM C(200):: A=1 :: GO$(
1)="GO" :: GO$(2)="GOTO" ::
GO$(3)="GOSUB"
130 INPUT "FILENAME? DSK1.":
F$
140 OPEN #1:"DSK1."&F$,INPUT
,VARIABLE 163 :: OPEN #2:"P
IO"
150 LINPUT #1:A$
160 IF POS(A$,CHR$(133),1)=0
 AND POS(A$,CHR$(134),1)=0 A
ND POS(A$,CHR$(135),1)=0 THE
N 210
170 LN=ASC(SEG$(A$,1,1))*256
+ASC(SEG$(A$,2,1)):: T=133 :
: P=1
180 G$=CHR$(T):: X=POS(A$,G$
```

```
,P):: IF X=0 THEN 200 :: LRE
F=ASC(SEG$(A$,X+2,1))*256+AS
C(SEG$(A$,X+3,1))!:: PRINT #
2:LN;GO$(T-132);LREF :: P=X+
1 :: GOTO 180
190 C$=STR$(LREF)&"."&STR$(L
N)&STR$(T-132):: C(A)=VAL(C$
):: A=A+1 :: P=X+1 :: GOTO 1
80
200 IF G$=CHR$(135)THEN 210
:: T=T+1 :: P=1 :: GOTO 180
210 IF EOF(1)THEN CLOSE #1 :
: GOTO 220 :: ELSE 150
220 A=A-1 :: CALL LONGSHELLN
(A,C())
230 FOR J=1 TO A :: A$=STR$(
C(J)):: X=POS(A$,".",I):: Y=
VAL(SEG$(A$,LEN(A$),1)):: A$
=SEG$(A$,1,LEN(A$)-1)
240 PRINT #2:SEG$(A$,1,X-1);
TAB(7);GO$(Y);" FROM ";TAB(2
1);SEG$(A$,X+1,LEN(A$)):: NE
XT J
250 SUB LONGSHELLN(N,NN())
260 D=N
270 D=INT(D/3)+1 :: FOR I=1
TO N-D :: IF NN(I)<=NN(I+D)T
HEN 300 :: T=NN(I+D):: J=I
280 NN(J+D)=NN(J):: J=J-D ::
 IF J<1 THEN 290 :: IF T<NN(
J)THEN 280
290 NN(J+D)=T
300 NEXT I
310 IF D>1 THEN 270
320 SUBEND
```

According to the User's Reference Guide that came with your computer, if you open a file without specifying INPUT, OUTPUT, UPDATE or APPEND, the computer will assume the UPDATE mode as the default and "UPDATE files may be both read and written. The usual processing is to read a record, change it in some way, and then write the altered record back out on the file." This is a very dangerous bit of misinformation! It is true only if you are using RELATIVE files with the REC clause. In any other case, the first record you write to the file will become the record FOLLOWING the last record you read, and it will also become the LAST record in the file - any records beyond that point will be lost!

The moral of the story - get in the habit of NEVER opening a file without specifying the mode. The only way to update a sequential file is to read it ALL into an array, update it, and then write it back to the file.

I reviewed hundreds of programs, in my PD library of about 2600, in order to select some of the best to fill up the collection disks. Often they needed only a few minor changes to greatly improve them.

One frequent flaw was in interpreting the status of CALL KEY. The User's Reference Guide says that a status variable of -1 means that "the same key was pressed during the performance of CALL KEY as was pressed during the previous performance." This is misleading. It actually means that the same key is STILL BEING pressed. Try this -

```
100 DISPLAY AT(12,1)ERASE AL
L:"TYPE YOUR NAME" :: R=12 :
: C=3
110 CALL KEY(0,K,S):: IF S=0
 THEN 110 :: DISPLAY AT(R,C)
:CHR$(K):: C=C+1 :: GOTO 110
```

Difficult to type without unwanted repetition of letters? Now try changing the S=0 to S<1 !

IF S<1 (if S is less than 1) means that if no key is pressed (S=0) or if the same key is still being held down (S=-1) then CALL KEY again.

Another frequent flaw is

```
INPUT "WANT TO PLAY AGAIN? "
:Q$ :: IF Q$<>"Y" THEN END
```

- or, more professionally programmed, IF SEG$(Q$,1,1)<
>"Y" THEN....,

which will accept either "Y" or "YES" as a reply. The problem is still that this question is often asked at the end of a joystick game, for which the Alpha Lock will be unlocked - and a response of a lower case "y" then terminates the program! One solution is to precede the INPUT with a dummy CALL KEY(3,K,S), which will cause any subsequent upper case CALL KEY, INPUT, LINPUT or ACCEPT AT response to be read as lower case until you turn it off with CALL KEY(5,K,S).

Here's one that does nothing except look pretty.

```
100 DISPLAY AT(3,8)ERASE ALL
:"COLORSQUARES" :: DISPLAY A
T(8,1):"Select option 1, 2 o
r 3" ! by Jim Peterson, Tige
rcub Software
110 CALL KEY(0,K,ST):: IF ST
=0 OR K<49 OR K>51 THEN 110
:: ON K-48 GOTO 150,120,130
120 FOR CH=38 TO 142 STEP 8
:: CALL CHAR(CH,RPT$("A55A",
4)):: NEXT CH :: GOTO 150
130 FOR CH=38 TO 142 STEP 8
:: FOR L=1 TO 4 :: RANDOMIZE
 :: X$=SEG$("0018243C425A667
E8199A5BDC3DBE7FF",INT(16*RN
D+1)*2-1,2)
140 B$=B$&X$ :: C$=X$&C$ ::
NEXT L :: CALL CHAR(CH,B$&C$
):: B$,C$=NUL$ :: NEXT CH
150 CALL CLEAR :: RANDOMIZE
:: FOR SET=0-(K>49)TO 14 ::
CALL COLOR(SET,SET+2+(K>49),
SET+2):: NEXT SET
160 Y=INT(4*RND+3):: R=INT(1
2*RND+1):: R2=25-R-Y :: C=IN
T(7*RND+7):: C2=32-C-Y :: IF
 K=49 THEN X=INT(14*RND+1)*8
+22 ELSE X=INT(13*RND+1)*8+3
0
170 FOR T=R TO R+Y :: CALL H
CHAR(T,C,X,Y):: CALL HCHAR(T
,C2,X,Y):: NEXT T
180 FOR T=R2 TO R2+Y :: CALL
 HCHAR(T,C,X,Y):: CALL HCHAR
(T,C2,X,Y):: NEXT T :: GOTO
160
```

The asterisk on the Gemini printer looks rather like a bug squashed side- ways, and it was confusing some folks in the condensed print of my newsletter, so I improved it with this -

```
150 PRINT #2:CHR$(27);CHR$(4
2);CHR$(1);CHR$(42);CHR$(0);
CHR$(8);CHR$(34);CHR$(8);CHR
$(0);CHR$(62);CHR$(0);CHR$(8
);CHR$(34);CHR$(8);
```

And at the same time I improved the slashed zero -

```
140 PRINT #2:CHR$(27);CHR$(4
2);CHR$(1);CHR$(48);CHR$(0);
CHR$(64);CHR$(30):CHR$(96);C
HR$(17);CHR$(72);CHR$(5);CHR
$(66);CHR$(61);CHR$(0);
```

```
90 !THIS WON'T WORK, WILL IT
?
100 DISPLAY AT(9999,9999)ERA
SE ALL:SEG$("CAN'T DO THAT!"
,1,3)&SEG$("CAN'T DO THAT!",
6,8)
```

If the Tigercub Math Puzzle in Tips #27 was a bit too tough, these changes will add a couple of easier levels.

```
105 DISPLAY AT(6,1):"Level 1
, 2, 3 or 4?" :: ACCEPT AT(6
,21)VALIDATE("1234"):L$ :: L
=VAL(L$)
106 IF L<3 THEN M$="Insert +
, -, or * (multiply)" ELSE M
$="Insert +, -, * (multiply)
 or / (divide)"
110 DISPLAY AT(5,1):M$;" bet
ween the digits":" to equal
the total": :"Type Q to give
 up"
120 ! **DELETED LINE **
130 DISPLAY AT(12,1):" " ::
T,X=INT(9*RND+1):: M$=STR$(X
):: Z$=M$&" "
140 FOR J=1 TO 4 :: Y(J)=INT
(9*RND+1):: @=3+ABS(L>2):: Z
=INT(@*RND+1):: ON Z GOSUB 2
40,250,260,270 :: Z$=Z$&STR$
(Y(J))&" " :: NEXT J
150 IF L/2<>INT(L/2)AND T<>I
NT(T)THEN 130 :: Z$=Z$&"="&S
TR$(T)
```

MEMORY FULL

Jim Peterson

Arcade Action Software
4122 N. Glenway
Wauwatosa, WI 53222

WALL STREET ANALYST

$$$$$$$$$$$$$$$$$$$$

HOW-TO-USE....package documentation

Detailed instructions for program use displayed on screen or to printer

$$----$----$----$----$----$----$----$----$$

$TREND$.....technical trend analysis

Sets up and analyses index trendlines moving averages and pace line Includes trend files for DJ INDUSTRIALS, S&P 500, NYSE INDEX & Weekly A/D

$$----$----$----$----$----$----$----$----$$

$PORTFO$.....portfolio management

Keeps up-to-date analysis of individual securities plus the complete portfolio, records, results and returns

$$----$----$----$----$----$----$----$----$$

$ANAL1$.....fundamental security analysis

Corporate balance-sheet statistical comparison to industry key business ratios, shows liquidity, efficiency & profitability Includes files for 50 industries

$$----$----$----$----$----$----$----$----$$

$ANAL2$.....income & statistics analysis

Determines yield, PE ratio, volativity, risk, price correlation & deviation, Alpha, Beta, coefficient of determination, price and earnings projections, value analysis, warrant, option and debenture evaluations

$$----$----$----$----$----$----$----$----$$

$MARKET$....comprehensive market evaluation

Market trend prediction based on current fundamental, technical, speculative, economic and monetary data

$$----$----$----$----$----$----$----$----$$

$GRAPH$.....bitmap trend graphing

Graphs to screen or printer 156 data, 3 years of weekly or 7 months of daily prices of securities or indexes. Incudes graph files for all major indexes.

$$$

Written in TI BASIC
Complete package on tape or disk$

$ 50.00

DATE 11 , 22 , 85

DJ-IND
INDEX AVERAGE 1464.33

MOVING AVERAGE IS
13 WEEKS 1335
26 WEEKS 1290.85
39 WEEKS 1257.39
52 WEEKS 1226.26

TREND FACE IS
13 WEEKS 9.65
26 WEEKS 6.28
39 WEEKS 5.06
52 WEEKS 4.47

39 WEEK SPREAD 206.95

SHORT TERM INDEX.. 96

LONG TERM INDEX.. 99

GULF&WESTERN 6%'88
1 BOND(S) $ 815

CRANE 7% '94
10 BOND(S) $ 6237.5

YOUNGSTOWN 9.875%'91
10 BOND(S) $ 7900

YOUNGSTOWN 4.5%'90
20 BOND(S) $ 12700

INVESTMENT 27849.57
7 29 85 27652.5

DIVIDEND/INTEREST TOTAL $ 19561.25

PORTFOLIO RETURN TO DATE 8.52 %

CAPITAL GAINS TOTAL $ -197.07

ANNUAL CAPITAL GAIN RATE -.14 %

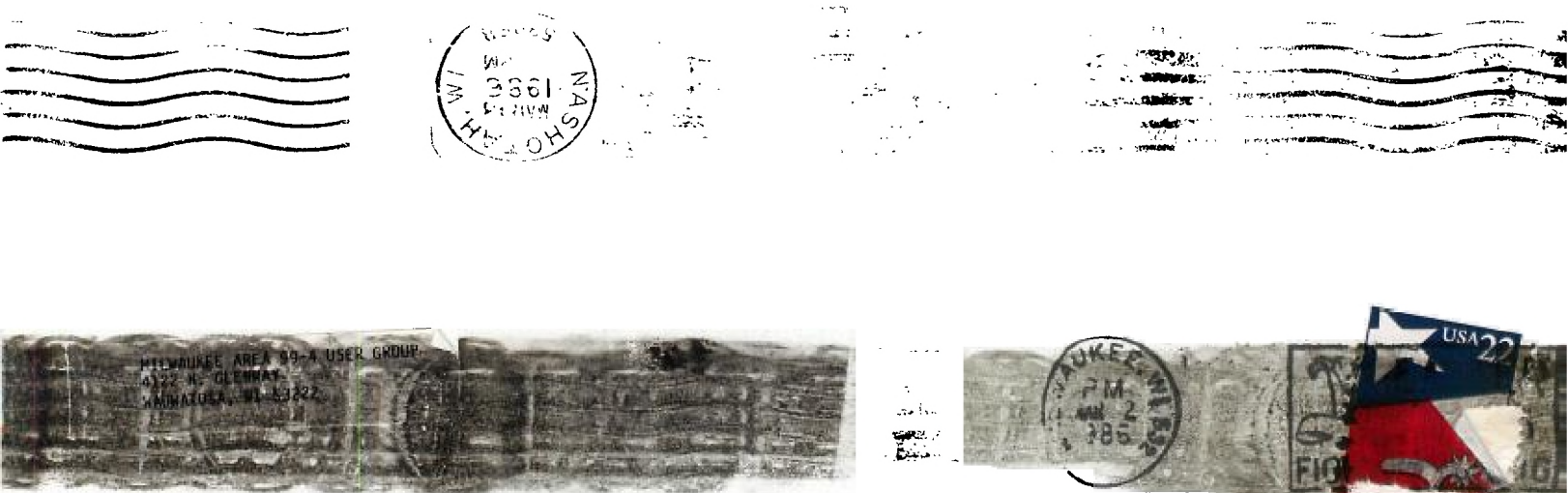
TOTAL GAIN/RETURN RATE 8.39 %

CURRENT PORTFOLIO RETURN 9.57 %

CRANE 7% '94 (CR)

DEBENTURE - - 10
DATE..... 10 27 85
PURCHASE DATE 9 27 77
TOTAL COST $ 8057.97
CURRENT VALUE $ 6237.5
BOND(S) COST $ 825.797
BOND(S) VALUE $ 623.75
ANNUAL INTEREST $ 70
TOTAL RECIEVED $ 4900
INTEREST RETURN 7.25 %
CAPITAL GAIN RATE -3.56 %
CURRENT RETURN 11.22 %

MATURITY, 8.67 YEARS
YIELD TO MATURITY 16.9 %



Edmonton 9906

P.O. Box 11983

Edmonton Alberta

Canada T5J-3L1