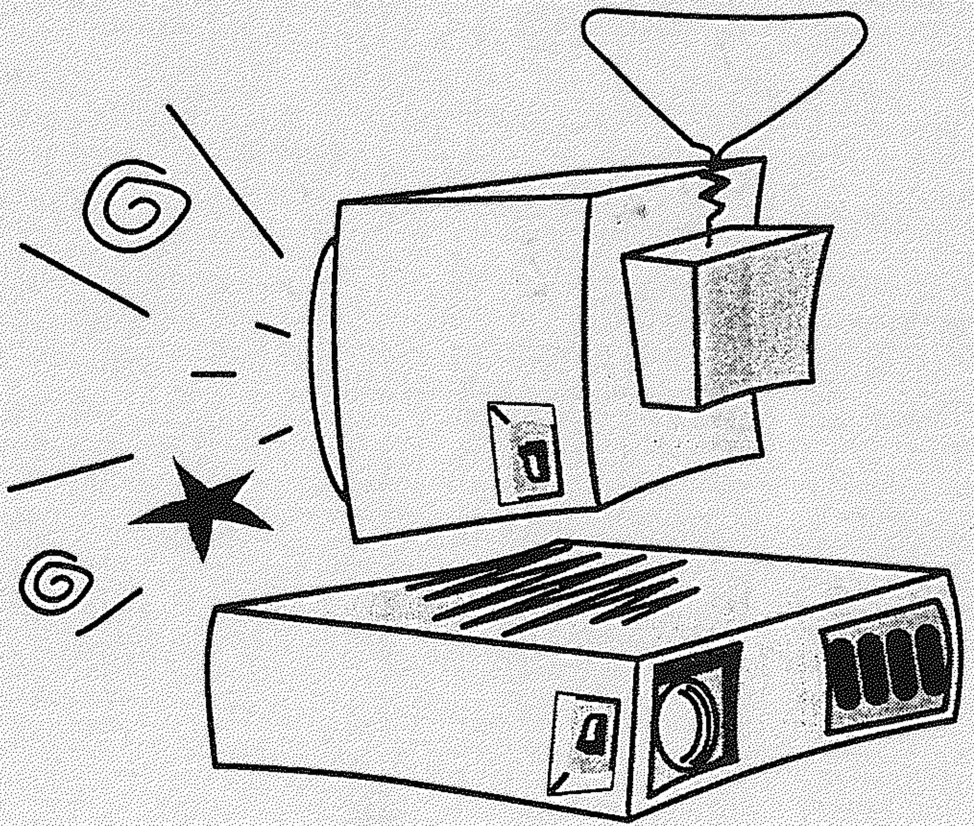
TISHUG

NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 14, Number 4 May, 1995

PRINT POST Approved - PP244099/00016



Sydney, New South Wales, Australia $3

TIsHUG (Australia) Ltd.
A.C.N. 003 374 383

TIsHUG News Digest

All correspondence to:
C/o 3 Storey St.
Ryde 2112 Australia

The Board

Co-ordinator
Dick Warburton (02) 918 8132
Secretary
Thomas Marshall (02) 671 7535
Treasurer
Cyril Bohlsen (02) 639 5847
Directors
Percy Harrison (02) 808 3181
Loren West (047) 21 3739

Sub-committees

News Digest Editor
Loren West (047) 21 3739
BBS Sysop
Ross Mudie (02) 456 2122
BBS telephone number (02) 456 4606
Merchandising
Percy Harrison (02) 808 3181
Software Library
Larry Saunders (02) 644 7377
Technical Co-ordinator
Geoff Trott (042) 29 6629

Regional Group Contacts

Central Coast
Russell Welham (043) 92 4000
Glebe
Mike Slattery (02) 692 8162
Hunter Valley
Geoff Phillips (049) 42 8176
Illawarra
Geoff Trott (042) 29 6629
Liverpool
Larry Saunders (02) 644 7377
Sutherland
Peter Young (02) 528 8775

Membership and Subscriptions

Annual Family Dues	$35.00
Associate membership	$10.00
Overseas Airmail Dues	A$65.00
Oversens Surface Dues	A$50.00

TIsHUG Sydney Meeting

The April Meeting will start at 2.0 pm on the 6th May 1995 at Meadowbank Primary School, Thistle Street, Meadowbank.

Project Group starts at 10.30 am

Printed by
Kwik Kopy West Ryde

TIsHUG News Digest ISSN 0819-1984

INDEX

IBM INDEX

PUZZLE

This months list of words is based around the subject of "WAR AIRPLAINES"

```
I J F A S U H P R D W H D N F H B C S K
S D W E G H G I J T A E R H F Y D O O T
M E H C O R A Y G J W M C A A J L X A G
B K X M U S C H V O F L I K T A A C X R
P O G M R H A K I Y H T O J F A D Y X O
Q D H O A T L T P U O L H F A L G T C N
J A C I S M I U T F E K U J I K N G G X
N T K U J H X M Q V R B E W E Z A Y W L
E A H M E S S E R S C H M I T T I N M Q
B H W L T G M X X E C P Z Y V P C R P E
Z I A S T A K M R J P G L A D I A T O R
O O E C K S B B C X U H T B D K H L Z Z
P R S E I R C S K P O F I A T U D G Z N
T H E X P R W N S P I T F I R E L U G T
W X L Z C N R W A G O I A B H E T F X U
X Q H A E O M U U W H T R M K J O P L G
Q P L T Z F S N H C Z K V N G K E D N T
P G E R Y L G S C O I E I T K V T N Z W
A O B Y D Y X A I E V E C E K K D Z Z T
R R L E R G N U Q N B X R A F F X D Q W
```

Find these hidden words

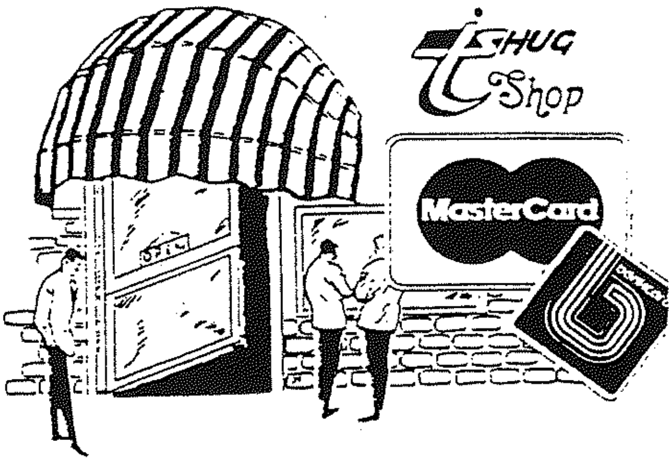
HOW TO PLAY

In this puzzle there are (20) words somewhere horizontally, vertically, diagonally even backwards.

GOOD LUCK

BUFFALO	CHAIKA	CORSAIR
DEWOITINE	FIAT	FOKKER
GLADIATOR	GRUMMAN	HEINKEL
HURRICANE	MACCHI	
MESSERSCHMITT	METEOR	MUSTANG
NAKAJIMA	RATA	SPITFIRE
WILDCAT	YAKOLEV	ZERO

This puzzle was compiled using Ashley Lynn's programe "Word Puzzle" which can be ordered through TIsHUG.



with Percy Harrison.

Once again we had a very good roll-up at our April meeting and happily renewal subscription fees started to flow in as April and May are the two months in the year that most subscriptions come due. At the time of writing this we are still down about 15 members but hopefully these will trickle in before the posting date of this magazine as we connot afford to send members copies once they become unfinancial.

Ashley Lynn, our member from Coonabarabran, has sent me a disk containing an IBM compatible car racing game called "Chequered Flag" which was written by two Year 2 students at Coonabarabran High School, Paul Geerts and Jay Stone. The game is being made available through the club on a 3.5 disk at our usual club price of $3.50 but will have to be ordered from me as supplies will not be stocked. If you want to purchase a copy, please phone me or see me at the next meeting. Many thanks to Paul and Jay for this program.

Had a note from John Scott, one of our regular meeting attendees, who advises that he has not been able to attend our meetings in recent months because he needs a knee reconstruction and also carpel tunnel operations on both wrists which he hopes will be done sometime in March. It will take 6 to 8 weeks to fully recover from the operations so we hope to see John back at our club meetings around about May or June. Our best wishes to you, John, and may your surgery be a total success.

We may be able to source a number of obsolete 268 computers complete with a small Hard Drive, a floppy drive and 1 or 2 MB RAM for around $400.00 each but we require a minimun order quantity of 20 units. If anyone is interested in purchasing a unit, please advise me and if we can get 20 orders we will go ahead and make the purchase.

Also, we still have available 1 or 2 Conner 20MB Hard Drives at $25.00 each. These units are not new but are in good working order and would be ideal for using as a drive on which to have your operating system installed thereby leaving your other drive for superstoring your other programs. If you want one of these, please contact me as soon as possible as they are in very short supply.

There has been a complaint from two of our members claiming that there is nothing of interest at our monthly meetings for dedicated TI Users. This I find very very hard to accept as we endeavour to ensure that first preference is given to looking after our long-standing TI members by ensuring that there is something of interest being displayed on the 2 or 3 TI systems that we set up each month. I might point out that we have, on many occasions, asked our members what they would like us to demonstrate at our meeting and, invariably, there is little or no response. We are not mind readers. If any member wants something specific to be demonstrated or would like a tutorial run on any TI program, then they should speak up and it will be attended to. If you cannot offer anything constructive, then please do not criticise your committee. Remember, their time and effort is voluntary and that is what helps to make this club successful.

Another point worth mentioning is that the income from IBM compatible PC users is now far in excess of that received from the sales of TI hardware and software and without this extra income your club would soon find itself in a position whereby we would not be able to meet our running cost, so take heed and do not bite the hand that feeds you. A goodly percentage of our income from PC users comes from non-members of our club and it is from this source that considerable financial support is derived. So let each of us make a concerted effort to work together to maintain the success of our club.

While on the subject of TI clubs, I note in the last issue of the TI Ottawa Group magazine that they folded up in March. In the final issue of their User's Group Magazine they produced a list of the TI Newsletters which are still current around the world with their brief comments about each. Ours was rated as being "still full of original material, the best", so I think that we can be proud of both our club and its magazine.

That's all for this month except to remind you that we should all make a determined effort to co-operate with each other and ensure our club maintains its number one position in the TI community.

Bye for now.

TIsHUG SOFTWARE FILE

April 1995

Disk#1:G114
Used= 277 Free= 81

Tourament Solitaire

This disk contains 7 card games. In most games the ACTIVE keys are.

ENTER = Draw card
FCTN 1 = Auto stack
FCTN 7 = Back to main menu
FCTN 9 = Next game
SPACE BAR = Use or Mark card.

GOLF

Try to get all the cards onto the turn up card at the top of screen, e.g. if it is a 3 you can put a 4 or 2 on it. If you put a 4 on it you can put a 3 or 5 on it. The card can only be moved from the piles if it will fit on the up turn pile. If you can not go, them turn up another card by pressing ENTER.

Ace you can only go put a 2. King you can not put another card on, (you must draw another card.)

PYRAMID

The object is to use all the 52 cards, cards can only be moved if at the base of the Pyramid or faced up on the stack. Ace is 1, Two is 2, King is 13, Queen is 12, etc. Cards only can be used if it or with one more card add to 13. e.g. 9 and 4 = 13, King = 13, Ace and Queen = 13.

KLONDIKE

Every one should know the rules to this classic game.

CANFIELD

Face down on the left turn up 3 cards at a time. The pile face up on the left can be used one card at a time. The card/cards (top of screen) are the building cards. e.g. 10 of Hearts goes the 9 of Hearts.

Four cards face up can be built on if it will go.

CALCULATION

Ace of Hearts goes up by One card at a time.
Two of Hearts goes up by Two card at a time.
Three of Hearts goes up by Three card at a time.
Four of Hearts goes up by Four card at a time.

From the base pack you turn up one card at a time and you must use it. If it will not fit on the Ace, 2,3,4. It must be place on one of the four vacant stacks below. The object is to get all 52 cards onto the top four stacks.

CORNERS

Refer canfield game.

CALC	18	Prog	CALU	18	Prog
CANFIELD	37	Prog	CORNERS	30	Prog
GOLF	12	Prog	HISCORES	6	Prog
KLONDIKE2	33	Prog	LOAD	24	Prog
PYRAMID	22	Prog	ROS	33	Prog
SCOREBOAR	3	I 30	SCOREBOARD	3	I 30

Disk#1:P115
Used= 358 Free= 0

Page Pro Headliner Fonts #2

COMPACT_HF	43	I 13	DCASUAL_HF	43	I 13
KOLOSS_HF	61	I 13	RAM1_I	6	d 80
SPRSTAR_HF	77	I 13	SPTLT1_HF	66	I 13
U/ROMAN_HF	62	I 13			

Disk#1:U115
Used= 353 Free= 5

Pre Editor is set for 40 columns with no extra memory, but using the config program on disk it can be change to 80 columns and 4K or 8K extra memory.

PrEditor Quick-Reference

Window Functions

FCTN 4 - Roll down
FCTN 5 - Next Screen
FCTN 6 - Roll up
FCTN 7 - Tab
FCTN S - Cursor left
FCTN D - Cursor down
ENTER - Next line
CTRL 5 - Next Screen
CTRL H - Beginning of Line
CTRL A - Append
CTRL F - Fordward a word
CTRL G - Go to line

Editing Functions

FCTN 1 - Delete Character
FCTN 2 - Insert Character
FCTN 3 - Delete Line
FCTN 8 - Insert Line
CTRL 2 - Split Line
CTRL 4 - "as-it" mode
FCTN K - Delete to End

Block Functions

FCTN , - Begin Block
FCTN . - End Block
CTRL 8 - Copy Block
CTRL 9 - Move Block
CTRL 0 - Delete Block

File Functions

CTRL 6 - Search
CTRL 7 - Search and replace
CTRL = - Purge Buffer
CTRL L - Load File
CTRL S - Save File

Miscellaneous Functions

FCTN 9 - Escape
FCTN 0 - Show Memory
FCTN Q - Quit
CTRL C - Catalog Disk
CTRL V - View File

Page Pro Poster Maker

This program loads with the Editor Assembler that is on this diskette.

(1) Pick Poster Maker.
(2) Pick 3 HI Res Printer
(3) Type DSK1.POSTER and then press ENTER

The 4 squares at top are.
1st Standed page or picture.
2nd 2 X 2 Blowup
3rd 4 X 4 Blowup
4th 8 X 8 Blowup

CAR	19	I 13	CHARA1	9	Prog
ED/AS	33	Prog	HOUSE	43	I 13
LOAD	5	Prog	PENCIL	18	I 13
POSTER	64	D 80	PRCONFIG	33	Prog
PRCONFIH	27	Prog	PREDITOR	33	Prog
PREDITOS	14	Prog	READ	6	d 80
ROOT	28	Prog	SHOWFLYER	21	Prog

Disk#1:P117
Used= 356 Free= 2

Page Pro Headliner Fonts #1

ALIGRPH_HF	46	I 13	BRICK_HF	64	I 13
CAROSL1_HF	66	I 13	CASLON_HF	50	I 13
RALGH1_HF	65	I 13	RALGH2_HF	65	I 13

TiSHUG SOFTWARE FILE
By Larry Saunders
May 1995

Diskname U118
Used= 138 Free= 220

Beyond Video Chess. The Video Chess Module must have the interupp wire cut.

The easy way to run it is to turn of the RAM Menu, plug in the VCM, call up the Menu, Select 3, type in DSK1.CHESSLR, press ENTER.

The harder way plug in the Extended Basic Module and run the load program. Them unplug the EBM and plug in the VCM.

CHESSBAS	37	Prog	CHESSLR	42	D 80
CHESSPF	24	Prog	LOAD	35	Prog

Diskname P119
Used= 358 Free= 0

Page Headline Fonts and Headliner maker.

3D_HF	80	I 13	ANTIQUE_HF	75	I 13
BKMN1_HF	76	I 13	BRDWY_HF	61	I 13
HL/FILE1	2	d 80	HL/PIC-01	4	I 13
LOAD	11	*Prog	PPFS-V100	16	*Prog
PPHM-V100	33	*Prog			

Diskname P119
Used= 358 Free= 0

3D_HF	80	I 13	ANTIQUE_HF	75	I 13
BKMN1_HF	76	I 13	BRDWY_HF	61	I 13
HL/FILE1	2	d 80	HL/PIC-01	4	I 13
LOAD	11	*Prog	PPFS-V100	16	*Prog
PPHM-V100	33	*Prog			

Diskname G120
Used= 332 Free= 26

Game disk. The Waterworks, instructions are with the game program.

LOAD	3	Prog	LOADWW	9	D 80
UTIL1	4	Prog	WWHELP	135	D 28
WWPF0	2	Prog	WWPF1	2	Prog
WWPF2	2	Prog	WWPF3	2	Prog
WWPF4	2	Prog	WWPF5	2	Prog
WWPF6	2	Prog	WWPHS	36	Prog
WWPRG1	33	Prog	WWPRG2	49	Prog
WWPRG3	49	Prog			

Diskname U121
Used= 343 Free= 15

EZ-Keys Plus

CHARA1	5	Prog	CHARDEF	16	Prog
EZKUTILS	34	Prog	EZOBJECT	87	D 80
FATFONT	5	Prog	HMEZKEYS	28	Prog
HMKEYS	28	Prog	HMUTILS	32	Prog
LOAD	30	Prog	LOAD*	30	Prog
MENUDATA	5	d163	PRINT	36	Prog

END OF ARTICLE

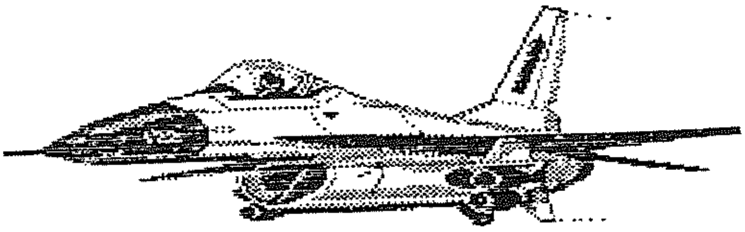
TREASURER'S REPORT

by Cyril Bohlsen

```
Income for previous month ....... $ 1560.00
Expenditure for previous month .. $  331.43
Profit for previous month ....... $ 1228.57
Membership accounted for $1025.00 of Income.
Shop sales ............. $ 535.50 of Income.
```

The expenditure was made up of the following

```
Administration cost .............. $  46.20
Printing and posting TND ......... $ 285.23
```



CRU TEST

CRU TEST Program, Source files.

This file contains the source files for the program CRUTESTXB in the TEXPAC downloadable programs for June 1988. The four source files have been placed in one file for the BBS. The original files were named:

```
"CRUTST/SRC"  main program
"CRUTST/TXT"  screen text
"CRUTST/VDP"  vdp data
"CRUTST/UTL"  utilities
```

Each file is named at the commencement.

```
*---------------------------------------
* File 1, CRUTST/SRC , main program.


*******************
* 9901 CRU TESTER *
*******************
* E.P. REBEL      *
* V1.0 08-05-86   *
* V2.0 13-07-86   *
*******************

       TITL '9901 CRU TESTER V2.0'
       IDT  'CRU-TEST'

       DEF  CRUTST
       DEF  SLAST,SLOAD,SFIRST
       REF  VDPWA,VDPWD,SCAN

SFIRST
SLOAD
CRUTST LWPI WORKSP               use own workspace
       CLR  @>83C4               no user interrupts
                                 allowed
       MOV  @LOAD,@>FFFC
       MOV  @LOAD+2,@>FFFE
       LI   R0,>0500
       MOVB R0,@>8374
       BL   @VDPIN               vdp init.

       SETO R7                   vdp interrupt on
       LI   R8,>E890             time out counter

STRTLP LI   R3,KTABLE
       LI   R5,>0800

MAINLP
```

```
       LI   R4,8            8 lines to print
       LI   R0,7+15        first screen position
       MOV  *R3,R1
       LI   R2,21           length
SCRNLP BLWP @VMBW
       AI   R0,40           next line on screen
       A    R2,R1           next text
       DEC  R4
       JNE  SCRNLP          print next line

DISPLP LI   R12,>0024       keyboard select
       LDCR R5,4            four bits to write

       CLR  R12             examine bits 0-10
       LI   R4,11           11 bits to test
       LI   R0,4+37        first screen position
PRTLP1 TB   0
       JNE  LOW
       LI   R1,ONE
       JMP  PRINT1
LOW    LI   R1,ZERO
PRINT1 BLWP @VSBW

       INCT R12             next cru bit
       AI   R0,40           next screen position
       DEC  R4
       JNE  PRTLP1

       LI   R12,>0024       examine bits 18-25
       LI   R4,8            8 bits to test
PRTLP2 TB   0
       JEQ  HIGH
       LI   R1,ZERO
       JMP  PRINT2
HIGH   LI   R1,ONE
PRINT2 BLWP @VSBW

       INCT R12             next cru bit
       AI   R0,40           next screen position
       DEC  R4
       JNE  PRTLP2

       TB   1               examine bit 27
       JEQ  EQUAL
       LI   R1,ZERO
       JMP  PRINT3
EQUAL  LI   R1,ONE
PRINT3 BLWP @VSBW


       LI   R6,>2000        equal bit
       CLR  R4
       LIMI 2
       LIMI 0
       BLWP @KSCAN
       MOVB @STATUS,R4      get key value
       COC  R6,R4           new key
```

```
       JNE  DISPLP          display values again

       MOV  R8,@>83D6       time out counter

       MOVB @ASCII,R4       get key
       CI   R4,CTRLK        <ctrl>k pressed?
       JNE  TEST2

       INCT R3              next screen text
       AI   R5,>0100        next keyboard selection
       CI   R5,>1100        done all?
       JNE  MAINLP
       JMP  STRTLP

TEST2  CI   R4,CTRLA        <ctrl>a pressed?
       JEQ  AUDTOG
       CI   R4,CTRLM        <ctrl>m pressed?
       JEQ  MAGTOG
       CI   R4,CTRLV        <ctrl>v pressed?
       JEQ  VDPTOG
       CI   R4,CTRL1        <ctrl>1 pressed?
       JEQ  CT1TOG
       CI   R4,CTRL2        <ctrl>2 pressed?
       JEQ  CT2TOG
       CI   R4,FCTN7        <fctn>7 pressed?
       JEQ  HELP
       B    @DISPLP

AUDTOG LI   R12,>0030       audio gate
       JMP  TOGGLE

MAGTOG LI   R12,>0032       magnetic tape output
       JMP  TOGGLE

CT1TOG LI   R12,>002C       cassette control 1
       JMP  TOGGLE

CT2TOG LI   R12,>002E       cassette control 2

TOGGLE TB   0
       JNE  SET
       SBZ  0
       B    @MAINLP
SET    SBO  0
       B    @MAINLP

VDPTOG LI   R12,>0004       vdp interrupt
       MOV  R7,R7           on?
       JNE  VDPOFF
       SBO  0
       SETO R7
       LI   R1,ON
       JMP  VDPPRT
VDPOFF SBZ  0
       CLR  R7
       LI   R1,OFF
VDPPRT LI   R0,6+30
```

```
       LI   R2,2
       BLWP @VMBW
       B    @MAINLP

HELP   LI   R0,>0203
       BLWP @VWTR
HELPLP LIMI 2
       LIMI 0
       BLWP @KSCAN
       MOVB @STATUS,R4
       COC  R6,R4
       JNE  HELPLP
       LI   R0,>0200
       BLWP @VWTR
       B    @MAINLP

       UNL
       COPY "DSK2.CRUTST/TXT"  screen text
       COPY "DSK2.CRUTST/VDP"  vdp data
       COPY "DSK2.CRUTST/UTL"  utilities
       LIST

LOAD   DATA WORKSP,CRUTST

SLAST  EQU  $

       DORG >8300
WORKSP BSS  >0020
UTILWS BSS  >0020

GPLWS  EQU  >83E0
RESET  EQU  >0000
ASCII  EQU  >8375
STATUS EQU  >837C
CTRLK  EQU  >8B00
FCTN7  EQU  >0100
CTRL1  EQU  >B100
CTRL2  EQU  >B200
CTRLA  EQU  >8100
CTRLM  EQU  >8D00
CTRLV  EQU  >9600

       END
```

```
* File 2, CRUTST/TXT , screen text.


KTABLE DATA KMODE0
       DATA KMODE1
       DATA KMODE2
       DATA KMODE3
       DATA KMODE4
       DATA KMODE5
       DATA JMODE6
       DATA JMODE7
       DATA ALPHA
```

```
SCREEN TEXT '        * 9901 CRU TESTER V2.0 *        '
       TEXT '                                        '
       TEXT ' Addr.  # Typ Function                V '
       TEXT ' ===== == === ==================== = '
       TEXT ' >0000  0 CON Control                   '
       TEXT ' >0002  1 INT External interrupt        '
       TEXT ' >0004  2 INT VDP interrupt on          '
       TEXT ' >0006  3 INP                           '
       TEXT ' >0008  4 INP                           '
       TEXT ' >000A  5 INP                           '
       TEXT ' >000C  6 INP                           '
       TEXT ' >000E  7 INP                           '
       TEXT ' >0010  8 INP                           '
       TEXT ' >0012  9 INP                           '
       TEXT ' >0014 10 INP                           '
       TEXT ' >0024 18 OUT Keyboard select bit 2     '
       TEXT ' >0026 19 OUT Keyboard select bit 1     '
       TEXT ' >0028 20 OUT Keyboard select bit 0     '
       TEXT ' >002A 21 OUT Alpha lock select bit     '
       TEXT ' >002C 22 OUT Cassette control 1        '
       TEXT ' >002E 23 OUT Cassette control 2        '
       TEXT ' >0030 24 OUT Audio gate control        '
       TEXT ' >0032 25 OUT Magnetic tape output      '
       TEXT ' >0036 27 INP Magnetic tape input       '

KMODE0 TEXT 'Keyboard <=> line    '
       TEXT 'Keyboard <SPACE> line'
       TEXT 'Keyboard <ENTER> line'
       TEXT '                     '
       TEXT 'Keyboard <FCTN> line '
       TEXT 'Keyboard <SHIFT> line'
       TEXT 'Keyboard <CTRL> line '
       TEXT '                     '

KMODE1 TEXT 'Keyboard <.> line    '
       TEXT 'Keyboard <L> line    '
       TEXT 'Keyboard <O> line    '
       TEXT 'Keyboard <9> line    '
       TEXT 'Keyboard <2> line    '
       TEXT 'Keyboard <S> line    '
       TEXT 'Keyboard <W> line    '
       TEXT 'Keyboard <X> line    '

KMODE2 TEXT 'Keyboard <,> line    '
       TEXT 'Keyboard <K> line    '
       TEXT 'Keyboard <I> line    '
       TEXT 'Keyboard <8> line    '
       TEXT 'Keyboard <3> line    '
       TEXT 'Keyboard <D> line    '
       TEXT 'Keyboard <E> line    '
       TEXT 'Keyboard <C> line    '

KMODE3 TEXT 'Keyboard <M> line    '
       TEXT 'Keyboard <J> line    '
       TEXT 'Keyboard <U> line    '
       TEXT 'Keyboard <7> line    '
       TEXT 'Keyboard <4> line    '
       TEXT 'Keyboard <F> line    '
       TEXT 'Keyboard <R> line    '
       TEXT 'Keyboard <V> line    '

KMODE4 TEXT 'Keyboard <N> line    '
       TEXT 'Keyboard <H> line    '
       TEXT 'Keyboard <Y> line    '
       TEXT 'Keyboard <6> line    '
       TEXT 'Keyboard <5> line    '
       TEXT 'Keyboard <G> line    '
       TEXT 'Keyboard <T> line    '
       TEXT 'Keyboard <B> line    '

KMODE5 TEXT 'Keyboard </> line    '
       TEXT 'Keyboard <;> line    '
       TEXT 'Keyboard <P> line    '
       TEXT 'Keyboard <O> line    '
       TEXT 'Keyboard <1> line    '
       TEXT 'Keyboard <A> line    '
       TEXT 'Keyboard <Q> line    '
       TEXT 'Keyboard <Z> line    '

JMODE6 TEXT 'Joyst. 1 <FIRE> line '
       TEXT 'Joyst. 1 <LEFT> line '
       TEXT 'Joyst. 1 <RIGHT> line'
       TEXT 'Joyst. 1 <DOWN> line '
       TEXT 'Joyst. 1 <UP> line   '
       TEXT '                     '
       TEXT '                     '
       TEXT '                     '

JMODE7 TEXT 'Joyst. 2 <FIRE> line '
       TEXT 'Joyst. 2 <LEFT> line '
       TEXT 'Joyst. 2 <RIGHT> line'
       TEXT 'Joyst. 2 <DOWN> line '
       TEXT 'Joyst. 2 <UP> line   '
       TEXT '                     '
       TEXT '                     '
       TEXT '                     '

ALPHA  TEXT '                     '
       TEXT '                     '
       TEXT '                     '
       TEXT '                     '
       TEXT '<ALPHA LOCK> line    '
       TEXT '                     '
       TEXT '                     '
       TEXT '                     '

ZERO   EQU  '0 '
ONE    EQU  '1 '

HLPSCR TEXT '       * 9901 CRU TESTER V2.0 *         '
       TEXT '                                       '
       TEXT ' Key:    Function:                     '
       TEXT ' ======= ============================  '
       TEXT ' <CTRL>1 Toggle cassette control 1     '
       TEXT ' <CTRL>2 Toggle cassette control 2     '
       TEXT ' <CTRL>A Toggle audio gate control     '
       TEXT ' <CTRL>K Change keyboard select lines  '
       TEXT ' <CTRL>M Toggle magnetic tape output   '
       TEXT ' <CTRL>V Toggle vdp interrupt on/off   '
       TEXT '                                       '
       TEXT ' Text:   Meaning:                      '
       TEXT ' ======= ============================  '
       TEXT ' CON     Control bit 9901 (bit 0)      '
       TEXT ' INT     Used as interrupt input       '
       TEXT ' INP     Used as input port            '
       TEXT ' OUT     Used as output port           '
       TEXT '                                       '
       TEXT ' Bit nr: Purpose:                      '
       TEXT ' ======= ==================#
==========  '
       TEXT '   1- 6  Dedicated interrupt inputs    '
       TEXT '   7-15  Programmable interrupts       '
       TEXT '  16-22  Dedicated I/O ports           '
       TEXT '  23-31  Programmable I/O ports        '

ON     TEXT 'n '
OFF    TEXT 'ff'

*---------------------------------------
* File 3, CRUTST/VDP , vdp data.


*********
* VDPIN *
*********
*
* vdp initialisation
*
VDPIN
       LI   R0,VDPDAT
       LI   R1,12               8 vdp registers/screen
                                start address
REGLP  MOVB *R0+,@VDPWA         write vdp register values
       DEC  R1
       JNE  REGLP
       MOVB @VDPDAT+2,@>83D4    for kscan
       LI   R1,>0300            96 character definitions
PATLP  MOVB *R0+,@VDPWD         set pattern definitions
       DEC  R1
       JNE  PATLP               next definition byte

       CLR  R0                  screen address
       LI   R1,SCREEN
       LI   R2,24
       BLWP @VMBW

       LI   R0,>0C00            help screen
       LI   R1,HLPSCR
       BLWP @VMBW

       RT
```

```
*
* vdp data
*
VDPDAT DATA >0080            no bitmap/external video
       DATA >F081            textmode
       DATA >0082            screen start: V0000
       DATA >0184            pattern table start: V0800
       DATA >F587            screen colors: white on
                             blue
       DATA >0049            pattern table start
                             address

* character data *
       DATA >0000,>0000,>0000,>0000
       DATA >0000,>1010,>1010,>0010 !
       DATA >0028,>2828,>0000,>0000 "
       DATA >0028,>287C,>287C,>2828 #
       DATA >0038,>5450,>3814,>5438 $
       DATA >0060,>6408,>1020,>4C0C %
       DATA >0020,>5050,>2054,>4834 &
       DATA >0008,>0810,>0000,>0000 '
       DATA >0008,>1020,>2020,>1008 (
       DATA >0020,>1008,>0808,>1020 )
       DATA >0000,>2810,>7C10,>2800 *
       DATA >0000,>1010,>7C10,>1000 +
       DATA >0000,>0000,>0030,>1020 ,
       DATA >0000,>0000,>7C00,>0000 -
       DATA >0000,>0000,>0000,>3030 .
       DATA >0000,>0408,>1020,>4000 /
       DATA >0038,>4444,>4444,>4438 0
       DATA >0010,>3010,>1010,>1038 1
       DATA >0038,>4404,>0810,>207C 2
       DATA >0038,>4404,>1804,>4438 3
       DATA >0008,>1828,>487C,>0808 4
       DATA >007C,>4078,>0404,>4438 5
       DATA >0018,>2040,>7844,>4438 6
       DATA >007C,>0408,>1020,>2020 7
       DATA >0038,>4444,>3844,>4438 8
       DATA >0038,>4444,>3C04,>0830 9
       DATA >0000,>3030,>0030,>3000 :
       DATA >0000,>3030,>0030,>1020 ;
       DATA >0008,>1020,>4020,>1008 <
       DATA >0000,>007C,>007C,>0000 =
       DATA >0020,>1008,>0408,>1020 >
       DATA >0038,>4404,>0810,>0010 ?
       DATA >0038,>445C,>545C,>4038 @
       DATA >0038,>4444,>7C44,>4444 A
       DATA >0078,>2424,>3824,>2478 B
       DATA >0038,>4440,>4040,>4438 C
       DATA >0078,>2424,>2424,>2478 D
       DATA >007C,>4040,>7840,>407C E
       DATA >007C,>4040,>7840,>4040 F
       DATA >003C,>4040,>5C44,>4438 G
       DATA >0044,>4444,>7C44,>4444 H
       DATA >0038,>1010,>1010,>1038 I
       DATA >0004,>0404,>0404,>4438 J
       DATA >0044,>4850,>6050,>4844 K
       DATA >0040,>4040,>4040,>407C L
       DATA >0044,>6C54,>5444,>4444 M
       DATA >0044,>6464,>544C,>4C44 N
       DATA >007C,>4444,>4444,>447C O
       DATA >0078,>4444,>7840,>4040 P
       DATA >0038,>4444,>4454,>4834 Q
       DATA >0078,>4444,>7850,>4844 R
       DATA >0038,>4440,>3804,>4438 S
       DATA >007C,>1010,>1010,>1010 T
       DATA >0044,>4444,>4444,>4438 U
       DATA >0044,>4444,>2828,>1010 V
       DATA >0044,>4444,>5454,>5428 W
       DATA >0044,>4428,>1028,>4444 X
       DATA >0044,>4428,>1010,>1010 Y
       DATA >007C,>0408,>1020,>407C Z
       DATA >0038,>2020,>2020,>2038 [
       DATA >0000,>4020,>1008,>0400 \
       DATA >0038,>0808,>0808,>0838 ]
       DATA >0000,>1028,>4400,>0000
       DATA >0000,>0000,>0000,>007C _
       DATA >0000,>2010,>0800,>0000  `
       DATA >0000,>0030,>0878,>483C  a
       DATA >0020,>2038,>2424,>2438  b
       DATA >0000,>0018,>2420,>2418  c
       DATA >0004,>041C,>2424,>241C  d
       DATA >0000,>0018,>243C,>201C  e
       DATA >0008,>1410,>3810,>1010  f
       DATA >0000,>001C,>241C,>0438  g
       DATA >0020,>2038,>2424,>2424  h
       DATA >0010,>0030,>1010,>1038  i
       DATA >0008,>0008,>0808,>2810  j
       DATA >0020,>2024,>2830,>2824  k
       DATA >0030,>1010,>1010,>1038  l
       DATA >0000,>0078,>5454,>5454  m
       DATA >0000,>0038,>2424,>2424  n
       DATA >0000,>0018,>2424,>2418  o
       DATA >0000,>0038,>2438,>2020  p
       DATA >0000,>001C,>241C,>0404  q
       DATA >0000,>0028,>3420,>2020  r
       DATA >0000,>001C,>2018,>0438  s
       DATA >0010,>1038,>1010,>140C  t
       DATA >0000,>0024,>2424,>241C  u
       DATA >0000,>0044,>4444,>2810  v
       DATA >0000,>0044,>5454,>5428  w
       DATA >0000,>0044,>2810,>2844  x
       DATA >0000,>0044,>2810,>1010  y
       DATA >0000,>003C,>0408,>103C  z
       DATA >0018,>2020,>4020,>2018  {
       DATA >0010,>1010,>0010,>1010  |
       DATA >0030,>0808,>0408,>0830  }
       DATA >0000,>2054,>0800,>0000  ~
       DATA >0000,>0000,>0000,>0000  █

*--------------------------------------
* File 4, CRUTST/UTL , utilities.
```

```
************************
* ASSEMBLER UTILITIES *
************************

KSCAN  DATA UTILWS,KSCAEN
VSBW   DATA UTILWS,VSBWEN
VMBW   DATA UTILWS,VMBWEN
VWTR   DATA UTILWS,VWTREN

*********
* KSCAN *
*********
KSCAEN LWPI GPLWS
       MOV  R11,@UTILWS+>16
       BL   @SCAN
       LWPI UTILWS
       MOV  R11,@GPLWS+>16
       RTWP

********
* VSBW *
********
VSBWEN BL   @VWRITE
       MOVB @>0002(R13),@VDPWD
       RTWP

********
* VMBW *
********
VMBWEN BL   @VWRITE
VWTMOR MOVB *R1+,@VDPWD
       DEC  R2
       JNE  VWTMOR
       RTWP

*
* LOAD VDP ADDRESS TO WRITE
*
VWRITE MOV  *R13,R2
       MOVB @UTILWS+5,@VDPWA
       ORI  R2,>4000
       MOVB R2,@VDPWA
       MOV  @>0002(R13),R1
       MOV  @>0004(R13),R2
       RT

********
* VWTR *
********
VWTREN MOV  *R13,R1
       MOVB @1(R13),@VDPWA
       ORI  R1,>8000
       MOVB R1,@VDPWA
       RTWP
```

END OF ARTICLE

STARTING A DATABASE FROM SCRATCH Pt 1.

by Chris Buttner of TIsHUG, July 1987.

There are many programs which allow us to organise a mailing list, create a data base and so on. Some are very powerful, some commercial, some shareware, but have you ever tried designing one yourself? It's not such a hard thing to do provided you know clearly what you expect of the system and set about designing it in a systematic and logical manner.

What I plan to do is start from scratch and I hope you will join me on this learning experience. It will be spread over a number of issues of the magazine so don't be put off thinking it will all be too much and beyond your comprehension.

In the commercial world of that other (IBM) machine, there are software packages galore which extol their virtues in sorts, data file size and so on and compete with one another in the race to have more and more "go gear" installed and menu selectable by the operator. I am often prompted to ask the question "why". They are terrific routines but will I ever use them. As this project develops, don't for one moment think it is meant to be the definitive answer to anything and everything. By the time we get to the end however, you should be able to design a program to suit YOUR particular needs, and what's more, have it operate efficiently.

Our starting point is really the think tank stage. There is no paper - just thoughts, a glimmer of hope and finally the light at the end of the tunnel, which in this case is the desire to have "something" which will allow me to retrieve names quickly from a file. You may decide on something else but always bear in mind how you expect to use the program to retrieve your data. I'm not interested in sorting the list of names; just finding them and their associated data; regularly and quickly. It doesn't matter to me whether the correct record is numbered 5 or 955 - the important thing is the name.

On paper, this conceptual idea looks something like this:-

```
         DATA BASE (FILE)
                |
                |
             SEARCH
                |
                |
```

DISPLAY RESULT

This is the real meat in the package. Whatever I do from this point onward must be directed at achieving that end.

Stage 2 of the idea starts to take shape like so:-

```
            DATA BASE (FILE)
                  |
         -------------------
         |   |     |   |
         |   |     |   |
       ADD UPDATE DELETE SEARCH
                           |
                           |
                   DISPLAY RESULT
```

I now have a framework within which I can design the various modules of the program necessary to get it to all hang together. These various sections in turn can be broken down into smaller tasks. The smaller the task the easier it to to write the program and the more feasible the project becomes. Notice that at this stage I haven't attempted to write a single line of program. The extra time spent now will save me countless hours of heartbreak and frustration later.

In the programs which follow some conventions have been observed:-

(1) all are written in XBasic;
(2) NO attempt has been made to fully utilise the multiple statement capability of XBasic;
(3) many of the program lines have been kept short in the interests of clarity;
(4) comments (REMarks) are liberally made to assist your understanding;
(5) the style is not necessarily the most elegant or efficient;
(6) your are free to modify/doctor any or all of the program/s as you see fit. In fact you are encouraged to do so if it will (a) aid your understanding or (b) make the program work better for YOU!
(7) the programs and sub-programs will be available from the club in MERGE format so you can easily incorporate them into this and any subsequent programs you are inspired to write.

As a general rule there are three aspects to any database. They are (1) Structure, (2) Records, and finally (3) Fields. The field is the smallest working part of the database. Various fields make up each a Record. Structure is the way the fields and records are organised in the database file.

STRUCTURE

Since the main program will use another file to store all the data it is necessary to define some of the parameters. Firstly, the data file will be a DIS/FIX type to allow manipulation of the data with simple basic programs.

The next step is deciding on the method of retrieving the information. The common methods of doing this are (1) checking every record until a match is found (long and tedious); (2) implementing a shell type search if the database is in sorted order; (3) using pointers to indicate the desired record and (4) hash coding. For this exercise, I plan to follow hash coding techniques.

If you are new to hash coding, it is a procedure which structures (builds) a list (records). The address in the list is derived from a~~n~~ mathematical equation. In layman's terms, this means I will have to decide which part of each individual record will be the "key". (If you have seen the Navarone Database Management System you should be familiar with the expression "Key Field"). Once I have done this, everything else becomes automated because the program will decide the record number (a job to which it is well suited and something which I don't want to worry about). For my application, the record number is immaterial; what counts is that I can quickly find the record if it exists.

For best results with hash coded lists, the actual number of records should be somewhere between 50% and 60% of the total list capacity. If I anticipate 60 records, my file should be capable of holding about 120 records. This may seem to be something of a waste but in fact it is a compromise between record retrieval speed and file size. If the file is 70% full the search time for my record will increase by about 50% and at 90% capacity if will have increased 400%.

Additionally, the list will work better if the maximum number of records is a prime number which yields a remainder of 3 when divided by 4. A small program will calculate this number for us.

RECORD and FIELDS

The final consideration relating to the file is the number of individual parts or fields and their respective size. For this example we will use the following fields and sizes:-

SURNAME	16
FIRSTNAME	12
STREET	21
TOWN	18
POSTCODE	4
TELEPHONE	7
AREA CODE	3

giving a total record size of 81(bytes) for 7 fields.

The first program creates the database file. Some of the things you need to keep in mind are:-

(a) try to keep your field names as short as possible: for example use SNAME for Surname;

(b) record 0 will store the total number of records in the file and also the parameters for each record;

(c) 3 bytes are reserved at the start of record 0 to record the total number of records;

(d) each record is limited to a maximum of 21 lines;

(e) if your field titles exceed (record size -3) bytes, you will be forced to start defining your records again;

(f) take care your "design work" all fits within the parameters of the TI Disk System. A single sided, single density drive will leave you roughly 350 sectors. Each sector contains 256 bytes. Divide 256 by the number of bytes in each record. This will give you the number of records per sector. Multiply this by 175 (because the number of actual records is roughly half the maximum) and you will have some indication of the number of records you will get on a SS/SD disk.

(g) you are allowed a maximum of only 6 characters to name your file. The program will add on the suffix "_BDF" so you will always be able to tell your database files from other files on the same disk.

(h) the moral of all this is PLAN before you start committing anything to paper (or in this case the program).

One of these fields has to be the key and I have chosen the surname. What happens if there are two people with the same surname? The algorithm we develop for coding the field will take care of this and ensure we finish with two distinct records rather than overwriting one with the other.

One further aspect must be settled before we put pen to paper and create the database program and that is how we will interface with it. The options are almost limitless and range from menu selections through to primitive prompts. I propose having a command line where commands (or more precisely nmemonics) will be entered along with a parameter. It should now be obvious this is a very specific program: not one designed to do a multitude of tasks for various users but one which will fulfil the specific need I have defined. (With modifications, you will customise it to suit your specific applications.)

My working screen will look something like this:-

```
 ___________________________________________
|                                           |
| (Command Area)                            |
| ----------------------------------------- |
|                                           |
| (Data Display Area)                       |
|                                           |
|                                           |
 ___________________________________________
```

Commands are entered above the line and normally this area is reserved for operator input.

The Data Display area is where information is displayed by the program and where the entry/updating of data takes place. Think of the data area as a blackboard if you want to simplify the explanation.

Almost immediately you should now start to envisage some of the smaller subtasks which must be designed. The most obvious one is is "wipe" to clear the blackboard. If you think of the tasks in real terms you fill find it easy to define and then write the subprograms to carry out those tasks. Another task which can be included is a help menu to increase user friendliness.

Getting down to the "nitty-gritty", the line dividing commands from data will be at row 3. Commands will appear on row 1. With three rows remaining "untouched" that leaves 21 rows to be wiped (24-3). This can easily be done by successively wiping 3 sections each of 7 rows. One way to "wipe" is to write the space character successively. Here is an example without program line numbers:

```
A$=" " (A$ takes the meaning of the space.)
A$=RPT$(A$,196) (A$ is redefined to now mean
                 196 space characters.)
```

By writing the newly defined A$ three times starting at locations 7 rows appart I can wipe the entire data area. This is easilly done in a for next loop. Because I will use this task repeatedly, it is ideally suited to being written as a subprogram. When I need it, I call the subprogram - simple!

By resequencing the actual program to start at a high line number, I can save it as a MERGE file (after testing to make sure it does what I want) and at the end, merge it back in to my base program.

This process of breaking the program down into smaller parts is repeated. The partial listing which I have for you is definitely not elegant and I deliberately have not attempted to make full use of Extended Basic's multiple statement lines. You should however start to get a feel for the way the program is developing and posibly how you can modify it to suit your needs.

A progran to create a data file is in the downloadable software area of the BBS with the name CREATEDBF .

Editors note! This file is not on the BBS, I would try looking in the software libary or ask the Sysop.

END OF ARTICLE

EDITORS COMMENTS

The April meeting had a busy schedule going, I had the 80 column card and Funnelweb up and running, WOW what a programe, I think IBM's only have memory size and speed, conpared to our old trusty TI. (All of the editoring for the magazine is done on the TI.)

We also had the CD ROM up and going for the IBM users.

Easter has passed us once again and I hope everyone enjoyed it. For those of us who have younger children, like myself, I would like to point out the fun and enjoyment to be had with children. Going to the Easter show is always a big highlight in any childs life, no matter how old, being dragged from one ride to the next hoping to be able to keep up the fast pace. Well anyway we survived that day, then the eggs, I didnt think I would ever get to not like chocolate eggs, well almost (have the chocolate eggs lost there rich flavour or is it just me).

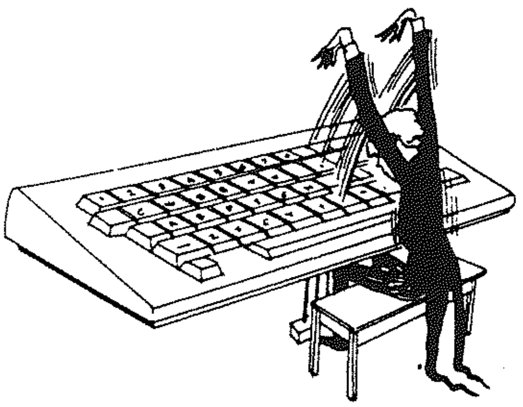
The next TIsHUG meeting will be at Meadowbank Public School, Saturday 6th May 1995. Bring your unfinished projects that you need help with, software and hardware.

The doors will be open from about 10.30am, for the project side of the meeting.

I hope to be able to bring the 80 column card and the 80 column Funnelweb along to the next meeting for those who would like a fiddle.

I believe Ross may be bringing his true to life TI99/4A train set.

See you all there. (ED)



ARRAYS

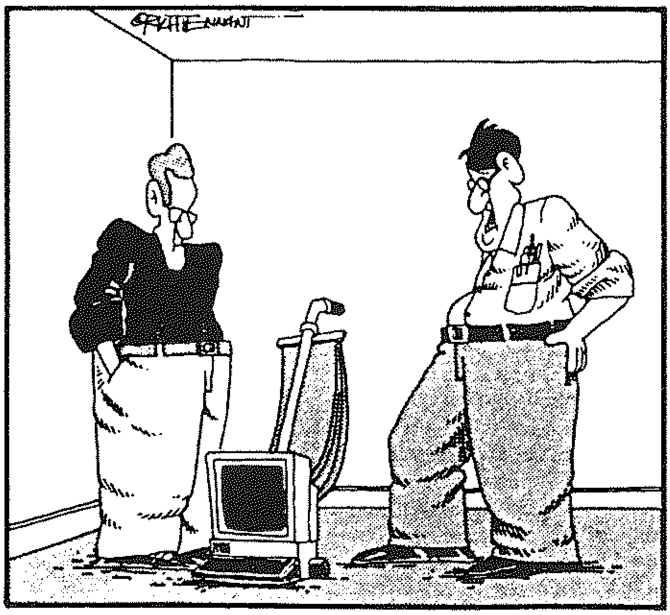
```
******************************************
*                                        *
* Written on November 5 1985 by J. Peter *
* Hoddie for use by Barry Traver         *
* in reading articles in his             *
* TRAVelER Diskazine.                    *
*                                        *
* Called by the following code:          *
* 10 DIM A$(500)                         *
* 20 CALL LINK("ARRAY",A$(),"DSKx.file") *
*                                        *
* Use 6 to page up, 4 to page down, E to *
* line up, X to line down, C to change   *
* screen colors and Q to quit.           *
*                                        *
* Modified on April 13, 1986             *
* NOTE: You may now also use the S key   *
* to toggle between single and double    *
* spacing.  The routine will start in    *
* single space but will remember what it *
* was in last between successive calls.  *
*                                        *
******************************************
*                                        *
* This code is Copyright (C) 1986 by     *
* by J. Peter Hoddie.                    *
* Permission for its free use is hereby  *
* granted and encouraged, provided that  *
* (1) credit be given to the author in   *
* any program in which it is used, and   *
* (2) that it not be used commercially   *
* without express permission of the      *
* author, viz., J. Peter Hoddie.         *
*                                        *
******************************************

       DEF  ARRAY

QNTR   EQU  >8356
KSCAN  EQU  >201C
STRASG EQU  >2010
STRREF EQU  >2014
VSBR   EQU  >2028
VMBR   EQU  >202C
VSBW   EQU  >2020
VMBW   EQU  >2024
VWTR   EQU  >2030

LINE1  TEXT '6:PAGE UP 4:PAGE DOWN E:UP X:DOWN Q:QUIT'
LINE2  TEXT 'TEXT UTILITY BY J. PETER HODDIE: LOADING'
LINE3  TEXT 'LOADING OTHER PAGES . . .               '
```

```
COLORS BYTE >F5,>F1,>12,>17,>FF
       EVEN

COLORP DATA >0000

*************************

* CALL LINK("ARRAY",M$(),"DSK1.HELP")

PAB    EQU  >0820         * WHY NOT USE THE CRUNCH
                          BUFFER AREA?
PABDAT EQU  >0840

DBLSPC DATA 0             * DOUBLE SPACE, 0=FALSE, 0<>
                          TRUE

VDPSAV BSS  256
POINT  BSS  2
POINT1 BSS  2
ARRBUF BSS  84
ARRCNT BSS  2
ARRART BSS  2
MYREGS BSS  32

ADATA  DATA >0014,PABDAT,>5050,>0000,>0000
       BSS  15
       EVEN

ARRAY
       MOV  R11,@ARRART
       LWPI MYREGS

       LI   R0,>300       *
       LI   R1,VDPSAV     * SAVE THE PART OF VDP WE'RE
                            GONNA DESTROY
       LI   R2,256        *
       BLWP @VMBR         *
 *

       LI   R0,>01F0
       BLWP @VWTR
       SWPB R0
       MOVB R0,@>83D4

       LI   R0,>07F5      * THESE LINES OMITTED FOR SHOW
                           /4080 WHICH
       BLWP @VWTR         * HAS SEPARATE CALL
                            LINK("TXTCOL",F,B)

       BL   @CLS

       LI   R2,LINE2
       BL   @DSPLY

       LI   R3,>0F00      *
       CLR  R0            *
       LI   R1,2          * GET THE FILE NAME
       LI   R2,ADATA+9    * TO READ FROM
```

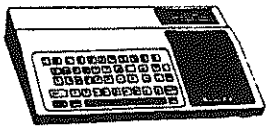
```
       MOVB R3,*R2        *
       BLWP @STRREF       *

       LI   R0,PAB
       LI   R1,ADATA      SET UP PAB IN VDP RAM
       LI   R2,>20
       BLWP @VMBW

       LI   R7,PAB+9
       MOV  R7,@QNTR
       BLWP @DSRLNK       OPEN FILE
       DATA 8

       LI   R0,PAB        *
       LI   R1,>0200      * CHANGE OP-CODE TO READ
       BLWP @VSBW         *

       CLR  @ARRCNT       * INITIALIZE ARRAY POINTER

ARRAY1 INC  @ARRCNT       * INCREMENT COUNTER

       MOV  @ARRCNT,R0
       CI   R0,24
       JNE  ARRAY2
       LI   R0,1
       MOV  R0,@POINT
       BL   @SHOW
       LI   R2,LINE3
       BL   @DSPLY

ARRAY2 LI   R7,PAB+9      *
       MOV  R7,@QNTR      * READ NEXT
       BLWP @DSRLNK       * RECORD
       DATA 8             *

       LI   R0,PAB+1      * ADDRESS OF STATUS BYTE
       BLWP @VSBR         * GET STATUS BYTE
       ANDI R1,>E000      * ISOLATE ERROR BITS
       JNE  ARREX         * IF AN ERROR CLOSE FILE AND
                            GET OUT

       LI   R0,PABDAT     * ARRAY BUFFER
       LI   R1,ARRBUF+1   * ADDRESS OF STRING
       LI   R2,80         * GET 80 BYTES  . . . CAN'T BE
                            LONGER ANYWAY
       BLWP @VMBR         * GET THE STRING

       LI   R0,ARRBUF+1   *
       LI   R1,40         * TAKE CARE OF
       LI   R3,>6000      * BASIC OFFSET
BASOFF MOVB *R0,R2        *
       AB   R3,R2
       MOVB R2,*R0
       INC  R0            *
       DEC  R1            *
       JNE  BASOFF        *

       LI   R0,PAB+5      * ADDRESS OF STRING LENGTH
       BLWP @VSBR         * READ THE BYTE
       MOVB R1,@ARRBUF    * PUT IN STRING LENGTH

       MOV  @ARRCNT,R0    * GET ARRAY ELEMENT NUMBER
       LI   R1,1          * FIRST ELEMENT IN LINK LIST
       LI   R2,ARRBUF     * ADDRESS OF BUFFER
       BLWP @STRASG       * ASSIGN THE STRING
       JMP  ARRAY1        * GET THE NEXT PIECE

ARREX  LI   R2,LINE1
       BL   @DSPLY

       LI   R0,1
       MOV  R0,@POINT

       BL   @SHOW

       LI   R0,>0300      * SET KEY UNIT
       MOVB R0,@>8374

WAIT   BLWP @KSCAN        * GET KEY PRESS
       LI   R3,>2000
       MOV  @>837C,R4
       COC  R3,R4
       JNE  WAIT
       CLR  R3
       MOVB @>8375,R3     * GET KEY THAT WAS PRESSED
       CI   R3,>FF00      * WAS NO KEY PRESSED?
       JEQ  WAIT

       CI   R3,>5800      * CHECK FOR UP
       JNE  CHECK1
       INC  @POINT
       B    @SHOW
CHECK1 CI   R3,>4500      * CHECK FOR DOWN
       JNE  CHECK2
       DEC  @POINT
       B    @SHOW
CHECK2 CI   R3,>5100      * CHECK FOR QUIT
       JEQ  QUIT
CHECK3 CI   R3,>3400      * CHECK FOR UP PAGE
       JNE  CHECK4
       MOV  @POINT,R0
       MOV  @DBLSPC,R15
       JEQ  CHECKA
       AI   R0,11
       JMP  CHECKB
CHECKA AI   R0,23
CHECKB MOV  R0,@POINT
       B    @SHOW
CHECK4 CI   R3,>3600
       JNE  CHECK5
       MOV  @POINT,R0
       MOV  @DBLSPC,R15
       JEQ  CHECKC
       AI   R0,-11
       JMP  CHECKD
CHECKC AI   R0,-23
```

```
CHECKD MOV  R0,@POINT
       B    @SHOW
CHECK5 CI   R3,>4300
       JNE  CHECK6
       INC  @COLORP
       MOV  @COLORP,R2
       LI   R0,>0007
       MOVB @COLORS(R2),R0
       CI   R0,>FF07
       JNE  COLOR1
       CLR  R2
       MOV  R2,@COLORP
       MOVB @COLORS(R2),R0
COLOR1 SWPB R0
       BLWP @VWTR
       B    @WAIT
CHECK6 CI   R3,>5300      * IS IT S? TOGGLE SPACING.
       JNE  CHECK7
       MOV  @DBLSPC,R15
       JEQ  DBLSP1
       CLR  @DBLSPC
       JMP  DBLSP2
DBLSP1 SETO @DBLSPC
DBLSP2 BL   @CLS
       BL   @SHOW
CHECK7 B    @WAIT

QUIT   LI   R0,>01E0
       BLWP @VWTR
       SWPB R0
       MOVB R0,@>83D4

       LI   R0,>300       *
       LI   R1,VDPSAV     * RESTORE THE VDP MEMORY WE
                            DESTROYED
       LI   R2,256        *
       BLWP @VMBW         *

       LI   R0,767
       LI   R1,>8000
QUIT1  BLWP @VSBW
       DEC  R0
       CI   R0,-1
       JNE  QUIT1

       LWPI >83E0
       MOV  @ARRART,R11
       RT

************************

DSPLY  CLR  R0
       LI   R3,40
CLR2   MOVB *R2,R1
       AI   R1,>6000
       BLWP @VSBW
       INC  R0
       INC  R2
       DEC  R3
       JNE  CLR2
       RT

*************************

SHOWRT BSS  2

SHOW   MOV  R11,@SHOWRT

SHOW0  MOV  @POINT,R0     *
       CI   R0,1          *
       JGT  SHOW01        * DO RANGE CHECKING
       LI   R0,1          *
       JMP  SHOW02        *
SHOW01 MOV  @ARRCNT,R1    *
       MOV  @DBLSPC,R15
       JEQ  SHOWA1
       AI   R1,-11
       JMP  SHOWB1
SHOWA1 AI   R1,-22        *
SHOWB1 C    R0,R1         *
       JLT  SHOW02        *
       MOV  R1,R0         *
SHOW02 MOV  R0,@POINT     *

SHOW1
       CLR  @POINT1

SHOW2  LI   R0,>5000
       MOVB R0,@ARRBUF

       LI   R1,ARRBUF+1   *
       LI   R2,>8000      *
       LI   R3,40         *  CLEAR OUT OUTPUT FIELD WITH
                             SPACES
SHOW3  MOVB R2,*R1        *
       INC  R1            *
       DEC  R3            *
       JNE SHOW3          *

       MOV  @POINT,R0
       A    @POINT1,R0
       LI   R2,ARRBUF
       LI   R1,1
       BLWP @STRREF

       MOV  @POINT1,R0    \
*                         |  GET THE VDP ADDRESS TO
                             WRITE TO
       INC  R0            |
       MOV  @DBLSPC,R15   |
       JEQ  SHOWA         |
       LI   R2,80         |
       JMP  SHOWB         |
SHOWA  LI   R2,40         |
SHOWB  MPY  R2,R0         |
       MOV  R1,R0         /

       LI   R1,ARRBUF+1
       LI   R2,40
       BLWP @VMBW        * PUT LINE ON SCREEN

       INC  @POINT1
       MOV  @POINT1,R3
       MOV  @DBLSPC,R15
       JEQ  SHOW2A
       CI   R3,12
       JNE  SHOW2
       JMP  SHOWEX
SHOW2A CI   R3,23
       JNE  SHOW2

SHOWEX MOV  @SHOWRT,R11
       RT

*************************

CLS    LI   R0,999
       LI   R1,>8000
CLR1   BLWP @VSBW
       DEC  R0
       CI   R0,39
       JNE  CLR1
       RT

       COPY "DSK1.DSRLNK/S"

       END
```

END OF ARTICLE



"IT STARTED OUT AS A KIT, AND WHILE I WAS WAITING FOR PARTS, THEY MERGED WITH A VACUUM CLEANER COMPANY."



LEARN TO KNOW YOUR TI

LESSON 26

with Percy Harrison

As promised last month, we will take a look at Snipping Strings by demonstrating the use of the functions SEG$, LEN and POS.

These functions, together with the concatenation operation "&", allow complete freedom to cut up strings and glue them back in any order.

The SEG$() function is similar to the MID$() of some other dialects of BASIC. As such, it can do the job of RIGHT$() and LEFT$() too.

The LEN function just returns the number of characters in its string argument.

TI BASIC uses the "&" for concatenation of strings. Other BASIC dialects often use "+". The latter is more confusing, because it is also used in arithmetic.

The POS statement is a nice feature of TI BASIC. It allows you to search for one string inside of another string. It reports the position of the first character of the first occurrence of the string. The search starts at a given "starting number".

Now let's get started with the lesson.

LESSON 26 SNIPPING STRINGS: SEG$, LEN, POS

GLUING STRINGS

You already know how to glue strings together:

```
Example:     55 A$="CON" & "CAT" & "EN" & "ATION"
             60 PRINT A$
```

The real name for "gluing" is "concatenation".

Concatenation means "make a chain". Maybe we should call them "chains" instead of "strings".

Let's cut a piece off a string. Enter and run:

```
10 REM >>> SCISSORS <<<
20 CALL CLEAR
30 N$="123456789"
35 Q$=SEG$(N$,3,4)
40 PRINT Q$, N$
```

The SEG$ function snips out a piece of the string. The snipped off piece can be put into a box or printed or whatever.

Here is what line 35 does:

Gets the string from box N$.
Counts over three numbers and starts saving numbers into box Q$.
Saves 4 numbers

Rule: The SEG$() function needs three things inside the () signs.

The string you want to snip.
The number of the last character before snipping is to start.
The number of characters that you want snipped out.

MORE SNIPPING AND GLUING

The peices of string you snip off can be glued back together in a different order.

```
Run:     10 REM ::: SCISSORS AND GLUE :::
         20 CALL CLEAR
         30 N$="123456789ABCDEF"
         35 FOR I=1 TO 13
         40 L$=SEG$(N$,I,3)
         42 M$=SEG$(N$,14-I,3)
         45 Q$=M$ & L$
         50 PRINT Q$
         60 NEXT I
```

HOW LONG IS A STRING?

```
Run:     10 REM ::: LONG ROPE :::
         20 CALL CLEAR
         30 PRINT "GIVE ME A STRING: "
         31 INPUT N$
         40 L=LEN(N$)
         45 CALL CLEAR
         50 PRINT "THE STRING: ";N$
         55 PRINT
         56 PRINT "IS";L;"CHARACTERS LONG."
```

The function LEN() tells the number of characters in the string. It counts everything in the string, even spaces.

LOOK MA, NO SPACES

ENTER:

```
10 REM <<< NO SPACES >>>
20 PRINT
21 PRINT
30 PRINT"GIVE ME A LONG SENTENCE"
31 PRINT
35 INPUT S$
40 L=LEN(S$)
45 T=""
49 REM --------LOOK AT EACH CHARACTER
50 FOR I=1 TO L
60 L$=SEG$(S$,I,1)
70 IF L$=" " THEN 90
71 REM --------SKIP SPACES
72 T$=T$ & L$
90 NEXT I
92 PRINT
94 PRINT "HERE IT IS WITH NO SPACES:"
96 PRINT
98 PRINT T$
```

Line 60 snips justone letter at a time out of the middle of the string.

LOOKING FOR A WORD IN A SENTENCE

The POS() function tells where one (short) string is located in another (long) string.

"POS" is short for "position".

Run:

```
10 REM WORM
15 A$="CAT RAT DOG HORSE MOUSE BIRD WORM
AARDVARK TURTLE FISH CATERPILLER"
20 Z=POS(A$,"WORM",1)
30 B$=SEG$(A$,Z,4)
40 PRINT B$, Z
```

Line 15 A long string is put into box A$.
Line 20 The POS() function looks for "WORM" in A$.
Line 30 The "WORM" is snipped out of the A$.
Line 40 And PRINTed.

POS() is a function. It "returns a value". It works like this:

POS(long string, short string, start at number)

The short string is supposed to be somewhere inside the long string.

You start looking at the "start number". You usually will start at the beginning of the "long string", so the "start number" will usually be 1.

Then the computer counts characters starting from the left until it gets to the first character of the "short string". It "returns" to the expression with the number at which the short string started.

Assignment 26

1. Write a secret cipher making program. You give it a sentence and it finds how long it is. Then it switches the first letter with the second, third with the fourth, etc. Example:

   THIS IS A DRAGON becomes:

   HTSII S ARDGANO

2. Write a question answering program. You give it a question starting with a verb and it reverses verb and noun to answer the question. Example:

   ARE YOU A TURKEY?

   YOU ARE A TURKEY.

3. Write a PIG LATIN program. It asks for a word. Then it takes all the letters up to the first vowel and puts them on the back of the word, followed by AY. If the word starts with a vowel, it only adds LAY. Examples:

   BOX becomes OXBAY

   APPLE becomes APPLELAY

As the assignment question last month was only to encourage you to practice EDITing and did not require a published answer that's all I have for this month. Next month we will practice Switching Numbers With Strings.

Bye for now.

END OF ARTICLE

TI BITS * Number 13

by Jim Swedlow

XMODEM

You may have heard of a transfer protocol called XMODEM and wondered what it is. If you use FAST-TERM or 4A TALK, you probably use it. The following should give you some idea of how it works.

When you communicate with another computer on phone lines thru modems, your data must travel thru the same voice phone lines that we use every day. Some connections are better than others. Most have noticeable static.

Your brain, a computer whose power has never been equaled, can usually distinguish the 'data' (voice) from the 'noise' (static). It is almost impossible for your computer to make this judgement.

In the early days of data transfer, data was simply sent and the receiving computer had to do as good a job as it could to distinguish between data and noise. In a text, or DV80 file, this was not a major problem. If one character was bad you could easily find the problem and edit it.

With a memory image or Program file, however, one bad byte could render the entire file useless. Although editing is possible, it is very tricky.

In August 1977, Ward Christensen developed an error detection method he called MODEM2. It was also dubbed "Christensen" protocol or XMODEM.

It was very simple. Data is sent in blocks of 128 bytes. XMODEM adds up the values of all the characters in each block and compares that number with a total that is sent by the sending computer. If they don't agree, the receiving computer sends a code to the sending computer and the block is transmitted again.

In 1982, Ward Christensen and Chuck Forsberg released an enhancement called Cyclic Redundancy Checking (CRC). CRC does sequential division on each character in the block resulting is a significant improvement in error detection.

Both protocols continue to be called XMODEM. Although others have been developed, XMODEM is used by all major systems, including Compuserve. (Source: an article in FOGLIGHT)

TI WRITER TIP

Find String (FS) is a powerful tool for finding something in a document. Just hit FCTN 9 and enter FS. Your TI Writer then will prompt: *FIND enter /string/ :*

You enter your string and use the slash as limiters. If you want to find the word "John", you would enter /John/. If you wanted to find John only when it is used as the last word in a sentence, you would enter /John./.

Should the "John" you find not be the one you wanted, you would go back to command mode and enter FS again. You will find /John./ still there. You just press enter and the search resumes.

Lets say, however, that now you want to find the word "Mo". But /John./ is on your screen. You could delete /John./. You could type in Mo but then you would have this: */Mo/n./*

Need you worry about the text after the second slash? No. Your TI Writer only searches for the information between the first and second slash. It ignores everything after the second slash.

You will have a problem with that if you use Replace String, but that is another story.

TRICK QUESTION OF THE MONTH

If a plane crashed on the border between the U.S. and Canada, who would bury the survivors? Answer next time.

Answer to the last trick question: How many birthdays does the average man have? One -- you celebrate it many times but you are born once.

THE PAPERLESS OFFICE

One of the things that futurists often project is the paperless office. Everything would be done on computers so paper would virtually dusappear.

Not necessarily so. According to an article in a recent issue of 'The Office', the demand for paper has been increasing at the rate of 5% to 8% a year. Growth is expected to continue at that rate.

Cited reasons include the continuing shift from a production to a service economy and the fact that computers generate reams of paper. Also noted were the need to generate hard copies for filing and the proliferation of photocopy machines.

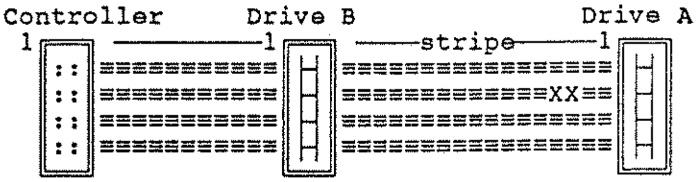
Enjoy.

DISK DRIVES- Cables and Connectors

I have been asked for information on disk drive connections and cables in PCs on a number of occasions. The following information has been copied from a bulletin board and reprinted here for members use, with appreciative thanks to Eoin VK2WCR.

FLOPPY DRIVES

FLOPPY cable with twist (control & data, 34 pin)



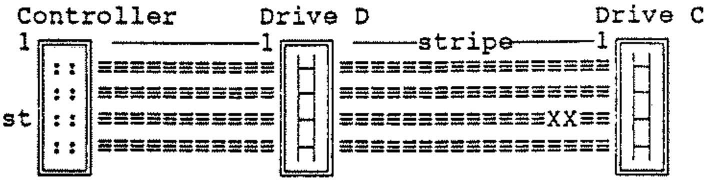
Pins 10-16 are twisted before the connector. (7 wires)

The connector on a floppy drive consists of 34 conductors. Both control and data use this same cable. Most cables have a twist that interchanges pins 10 through 16 at the end of the cable (on drive 1). Most floppy connectors have a "key" between pins 4 & 6, and 3 & 5, to prevent the cable from being reversed. At the other end, the dual row connector that attaches to the controller card will usually have a set of ridges that coincide with cutouts in the controller card's connector. Note that old style (XT) floppy-only controllers used a card-edge connector just like that of the drive. 3.5" drives usually have a connector similar to that of the controller.

HI/LO DENSITY	>	2	1	GND
N/C		4	3	
N/C		6	5	
INDEX	<	8	7	
MOTOR ENAB. A	>	10	9	
DRIVE SEL. B	>	12	11	
DRIVE SEL. A	>	14	13	
MOTOR ENAB. B	>	16	15	
DIRECTION SEL.	>	18	17	
HEAD STEP	>	20	19	
WRITE DATA	>	22	21	
WRITE GATE	>	24	23	
TRACK 00	<	26	25	
WRITE PROTECT	<	28	27	
READ DATA	<	30	29	
HEAD SELECT	>	32	31	
DISK CHANGE	<	34	33	GND

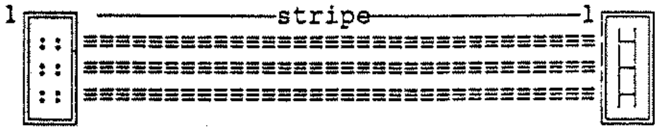
> Input (At the
< Output Drive Conn.)

ST412 & ESDI Hard Drive cable with twist (control)



Pins 25-29 are twisted before the connector. (5 wires)

ST412 & ESDI Hard Drive (data, 20 pin)



(no twists) Each drive has it's own data cable.

ST506/412 HARD DRIVE (MFM & RLL)

This standard drive system uses two cables; a 34 conductor control cable, and a 20 conductor data cable. The control cable contains a twist of the conductors going to the farthest drive, which is drive "C" on most systems. This twist consists of conductors 25 through 29. As with the floppy cable, the ST506/412 cables normally have a key to prevent reversal, and the controller end has a pin-type connector, while the drive end has a card-edge type connector.

HEAD SEL. 8	2	1	GND
HEAD SEL. 4	4	3	
WRITE GATE	6	5	
SEEK COMPLETE	8	7	
TRACK 0	10	9	
WRITE FAULT	12	11	
HEAD SEL. 1	14	13	
RESERVED	16	15	
HEAD SEL. 2	18	17	
INDEX	20	19	
READY	22	21	
STEP	24	23	
DRIVE SEL. 1	26	25	
DRIVE SEL. 2	28	27	
DRIVE SEL. 3	30	29	
DRIVE SEL. 4	32	31	
DIRECTION IN	34	33	GND

DRIVE SEL'D	1	2	GND
RESERVED	3	4	
	5	6	
	7	8	GND
RESERVED	9	10	RESERVED
GND	11	12	GND
* WRITE DATA+	13	14	* WRITE DATA-
GND	15	16	GND
* READ DATA+	17	18	* READ DATA-
GND	19	20	GND

*(MFM or RLL)

Though control signals go through a single 34 conductor cable, data flows through seperate 20 conductor cables for each drive (C,D).

ESDI HARD DRIVES

Though ESDI and ST506/412 drives share similar looking cables, even to the point of having a twist, the actual data and control signals are very different. One should never mix components from these two drive types. While the ST506/412 interface utilizes a standard pulse code to transmit data between the drive and controller, ESDI uses a pulse code that does not require the level to return to zero between pulses. This format is refered to as NRZ, or Non Return to Zero. By utilizing NRZ, the clock that data is transfered by can be increased, thereby increasing the throughput to and from the ESDI drive.

HEAD SEL. 3	2	1	GND
HEAD SEL. 2	4	3	
WRITE GATE	6	5	
CONFIG/STAT DATA	8	7	
TRANSFER ACK.	10	9	
ATTENTION	12	11	
HEAD SEL. 0	14	13	
SECT/ADD.MK. FOUND	16	15	
HEAD SEL. 1	18	17	
INDEX	20	19	
READY	22	21	
TRANS.REQUEST	24	23	
DRIVE SEL. 1	26	25	
DRIVE SEL. 2	28	27	
DRIVE SEL. 3	30	29	
READ GATE	32	31	
COMMAND DATA	34	33	GND

DRIVE SEL'D	1	2	SECT/ADD.MK. FOUND
SEEK COMPLETE	3	4	ADDRESS MARK ENABLE
RESV'D FOR STEP MODE	5	6	GND
WRITE CLOCK+	7	8	WRITE CLOCK-
CARTRIDGE CHANGED	9	10	READ REF. CLOCK+
READ REF. CLOCK-	11	12	GND
NRZ WRITE DATA+	13	14	NRZ WRITE DATA-
GND	15	16	GND
NRZ READ DATA+	17	18	NRZ READ DATA-
GND	19	20	GND

NOTE: Pin #1 on any drive cable SHOULD be indicated by a colored stripe. If you should find the stripe by connector pin 34 (or 20), inspect the whole cable VERY throughly!

DRIVE SELECT JUMPERS For both Floppy and Hard drives, when the 34 pin cable has a twist, the device number should be set to the second position. Drives numbered 0-3, set to 1, those numbered 1-4, set to 2. When cables without a twist are used, Floppy "A", and (or) Hard drive "C" should be set to 1, & the second Floppy and (or) Hard drive should be set to 2.

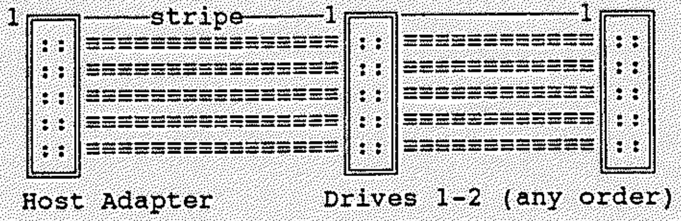
TERMINATORS: When using more than one drive on a cable (ie; 2FDs or 2HDs), the terminating resistor pack should be left on the drive furthest from the controller, and removed from the drive closest to the controller.

NOTE: On SCSI drives, the Host Adapter also has resistors. These are needed to terminate both ends of the bus. Since the SCSI bus can have up to 7 devices attached to it, only the Host Adapter and the device farthest from it will retain the resistors. All devices in between should have theirs removed.

IDE (AT) HARD DRIVES

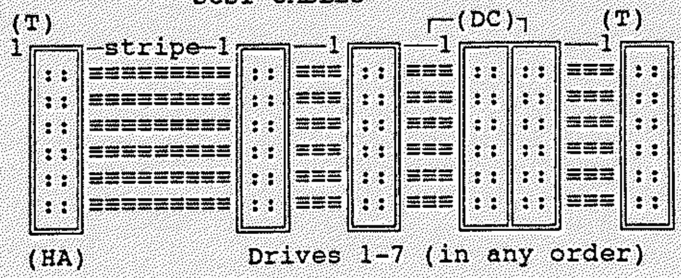
IDE, or Integrated Drive Electronics is a more recent drive interface to gain popularity. Often, the control circuitry is built into the mother-board, eliminating the requirement for a seperate Host Adapter. There are 3 types of IDE interfaces...those for the 8-bit XT bus, and those for the 16-bit AT bus (detailed here), with later boards now available for use with VL(32 bit) bus systems. The cable for IDE contains 40 conductors and has no twists. Like an SCSI cable, the IDE cable uses a Dual-row Pin connector for both ends. A single cable may be used to connect two drives, or two cables may be Daisy-Chained. Most IDE Host Adapters will support two hard drives. The first drive should be jumpered as the Master drive, and the second as the Slave drive. Plug-in IDE Host Adapters are often called Paddle-Boards, and may contain a floppy controller, and serial & parallel ports.

The IDE Host Adapter connector may be on a plug-in Paddle-Board or may be integrated on the Motherboard.



SCSI CABLES

On an SCSI cable, the terminating resistors (T) remain at the END devices on the cable, even when 2 cables are "Daisy-Chained" (DC). Also, the external connector may be used, requiring the removal of the Host Adapter's internal Term. resistors.



SCSI HISTORY

SCSI has it's roots in the mainframe world, but it's first implementation in the PC world came soon after the first PC. Shugart Associates devised an inter- face that they designated the SASI, or "Shugart Associates Standard Interface" They proposed that SASI be adopted by ANSI for small computers, but durring the work required for ratification, they discovered the process would take too much effort, an that the IPI groups were already well into their effort. (which had many features the same as SASI) A decision was made to take features of both interfaces, and put forth a new specification for a new interface, SCSI was born, and ratified in 1986 by ANSI. Since then, many have said that the original spec. was not tight enough, and that it allowed Manufacturers to make drives that met the ANSI spec., but would not talk to each other. More recently, the ANSI SCSI committee has proposed newer, tighter, more extended specs., for SCSI-2, and now SCSI-3.

REGIONAL GROUP REPORTS

Meeting Summary For MAY

Central Coast	13/05/95 Saratoga
Glebe	11/05/95 Glebe
Hunter Valley	14/05 21/05/95
Illawarra	09/05/95 Keiraville
Liverpool	12/05/95 Yagoona West
Sutherland	19/05/95 Jannali

===================================

CENTRAL COAST Regional Group

Regular meetings are normally held on the second Saturday of each month, 6.30pm at the home of John Goulton, 34 Mimosa Ave., Saratoga, (043) 69 3990. Contact Russell Welham (043)92 4000.

===================================

GLEBE Regional Group

Regular meetings are normally on the Thursday evening following the first Saturday of the month, at 8pm at 43 Boyce Street, Glebe. Contact Mike Slattery, (02) 692 8162.

===================================

HUNTER VALLEY Regional Group

The Meetings are usually held on the second or third Sunday of each month at members homes starting at 3pm. Check the location with Geoff Phillips by leaving a message on (049) 428 617. Please note that the previous phone number (049) 428 176 is now used exclusively by the ZZAP BBS which also has TI support. Geoff.

===================================

ILLAWARRA Regional Group

Regular meetings are normally held on the first Tuesday of each month after the TIsHUG Sydney meeting at 7.30pm, at the home of Geoff Trott, 20 Robsons Road, Keiraville. A variety of investigations take place at our meetings, including Word Processing, Spreadsheets and hardware repairs. Contact Geoff Trott on (042) 29 6629 for more information.

===================================

* LIVERPOOL Regional Group *

Regular meeting date is the Friday folling the TIshug Sydney meeting at 7.30 pm. Contact Larry Saunders (02) 644-7377 (home). After 10.30 PM or at work (02)602 3312 Liquorland Liverpool West for more information.

*** ALL WELCOME ***

12th MAY 1995
My Place : 34 Colechin st. Yagoona West

9th JUNE 1995	My Place
7th JULY 1995	My Place

Bye for now Larry.
Liverpool Regional Co-Ordinator

===================================

SUTHERLAND Regional Group

Regular meetings are held on the third Friday of each month at the home of Peter Young, 51 Jannali Avenue, Jannali at 7.30pm. Peter Young.

===================================

TIsHUG in Sydney

Monthly meetings start promptly at 2pm on the first Saturday of the month. They are held at the MEADOWBANK PRIMARY SCHOOL, on the corner of Thistle Street and Belmore Street, Meadowbank. Cars can enter from Gale Street and park in the school grounds. Regular items include news from the directors, the publications library, the shop, and demonstrations of monthly software.

MAY MEETING - 6th MAY

JUNE MEETING - 3rd JUNE

****

The cut-off dates for submitting articles to the Editor for the TND via the BBS or otherwise are:

JUNE - 13th MAY

These dates are all Saturdays and there is no guarantee that they will make the magazine unless they are uploaded by 6:00 pm, at the latest. Longer articles should be to hand well before the above dates to ensure there is time to edit them.

****
****

The next TISHUG meeting will be at Meadowbank Public School, Saturday 6th May 1995. Bring your unfinished projects that you need help with, software and hardware. The idea is to allow other members to help you to finish the project.

The TI99/4A train set will also be there, come along and have a drive of the train set. More information in file TRAIN in news menu.

Remember, next meeting:

6th May, 10.30am, Thistle St Meadowbank.