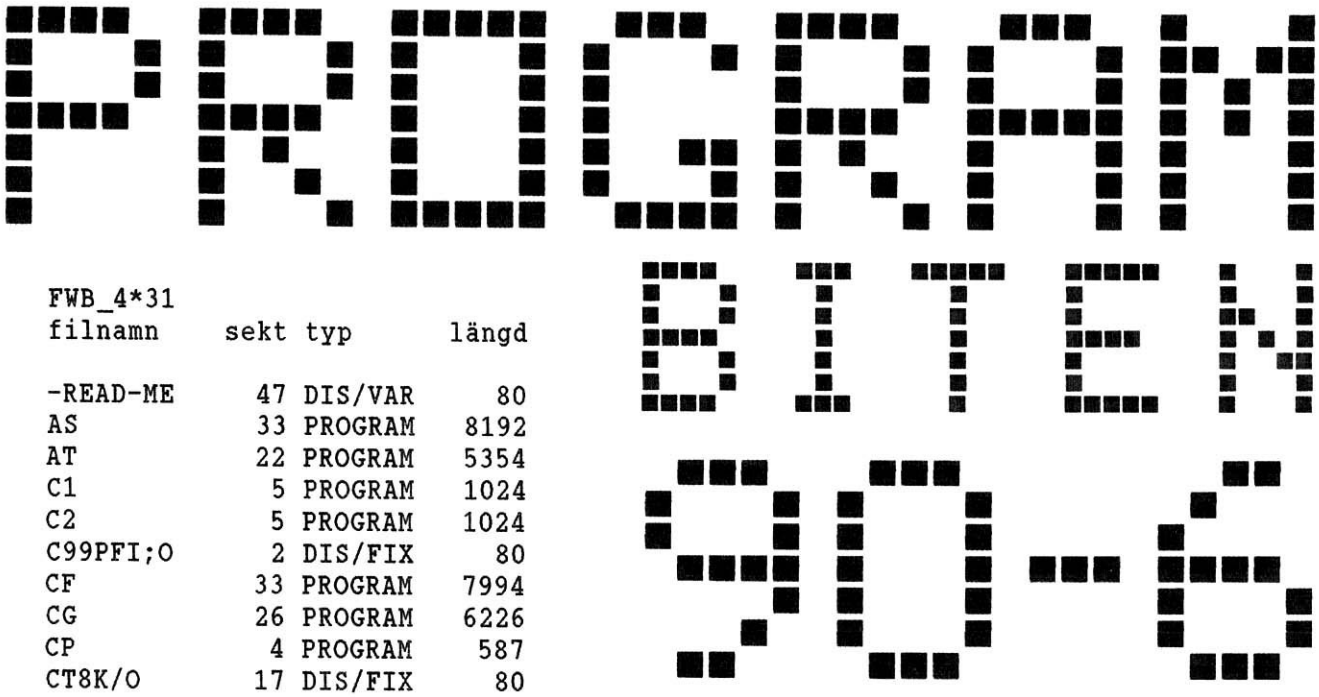


FWB_4*31

filnamn	sekt	typ	längd
-READ-ME	47	DIS/VAR	80
AS	33	PROGRAM	8192
AT	22	PROGRAM	5354
C1	5	PROGRAM	1024
C2	5	PROGRAM	1024
C99PFI;O	2	DIS/FIX	80
CF	33	PROGRAM	7994
CG	26	PROGRAM	6226
CP	4	PROGRAM	587
CT8K/O	17	DIS/FIX	80
D1	4	PROGRAM	542
DR	33	PROGRAM	8192
DS	31	PROGRAM	7652
EA	9	PROGRAM	1860
ED	33	PROGRAM	8192
EE	18	PROGRAM	4328
FO	33	PROGRAM	8192
FP	16	PROGRAM	3620
FSAVE	7	DIS/FIX	80
FW	33	PROGRAM	8152
FWDOC/DR40	75	DIS/VAR	80
FWDOC/DR41	64	DIS/VAR	80
FWDOC/EASM	37	DIS/VAR	80
FWDOC/LOAD	51	DIS/VAR	80
FWDOC/REPT	58	DIS/VAR	80
FWDOC/TIWR	32	DIS/VAR	80
FWDOC/UTIL	49	DIS/VAR	80
LDFW	11	DIS/FIX	80
LH	16	PROGRAM	3836
LL	10	PROGRAM	2064
LOAD	31	PROGRAM	7661
QD	12	PROGRAM	2622
QF	11	PROGRAM	2544
SCRIPT	4	DIS/VAR	80
SL	10	PROGRAM	2130
SYSCON	6	PROGRAM	1214
UL	4	PROGRAM	542
XB4THLD	2	PROGRAM	203

FWB_80-COL

filnamn	sekt	typ	längd
DR80	40	PROGRAM	9984
DR81	32	PROGRAM	7918
ED80	33	PROGRAM	8192
ED81	21	PROGRAM	5080
FWDOC/DR80	119	DIS/VAR	80
FWDOC/DR81	89	DIS/VAR	80
FWDOC/EDAV	47	DIS/VAR	80

ISSN 0281-1146

FWB OCT/30-NOV/04

Funnelweb finns nu i version 4.31 NOV/04/90 (se vidare sid 29). Om du har 80-kolumnsversionen av Disk Review från OCT/30/90 så kan du ändra genom att använda programmets egen "file search" efter >9C04 i DS (DR81). Detta ord finns på två ställen nära varandra. Ändra båda dessa till >8C04. Ändringen gör att omdefinierade färger i Graphic6-moden ställs in riktigt. Följande filer har ändrats i november-versionen: DR80/DR81, ED/EE, FW, LOAD samt några DOC-filer. Den reviderade 40-kolumnseditorn ED/EE klarar av att byta till en annan skiva i Show Directory så att man inte behöver återvända till editorn. FW och LOAD är ändrade så att inte datorn ska låsa sig när man hoppar runt bland program och menyer.

ÅRSMÖTE 1991

Boka preliminärt årsmöte den 16 mars 1991. Vi återkommer senare med tidpunkt och plats. Reservation för att dagen kan ändras. Tag kontakt med valberedningen eller styrelsen om du kan tänka dig att vara med i styrelsen eller utföra något specifikt uppdrag inom föreningen. Det finns behov av nya friska krafter om föreningen ska kunna förnyas och leva vidare.

FRACTALS 2.1 9938

Ett nytt program för fraktaler enligt Mandelbrot finns för videoprocessorn 9938 som finns i Myarc Geneve, Mechatronic 80-kolumn eller DIJIT AVPC. Du kan beställa den för DM 10 + kostnad för skivor (4 st 180 kB) och porto genom:

Roland Meier
Nik.-Schwarzkopfstr. 10
D-6074 RÖDERMARK 1
Tyskland

Manusstopp 15 jan PB 91-1
15 mar PB 91-2
15 maj PB 91-3

Redaktör: Jan Alexandersson
Utmanarredaktör: Anders Persson
TI-74 redaktör: Lars Herold Andersen
Programbankir: Börje Häll

Föreningens adress:
Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 9 NB
S-121 46 JOHANNESHOV
Sverige

Postgiro 19 83 00-6
Medlemsavgiften för 1990 är 120:-

Datainspektionens licensnummer:
82100488

Annonser, insatta av enskild medlem (ej företag), som gäller försäljning av moduler eller andra tillbehör i enstaka exemplar är gratis.

Övriga annonser kostar 200 kr för hel sida. För lösblad (kopieras av annonsören) som skickas med tidningen gäller 200 kr per blad. Föreningen förbehåller sig rätten att avböja annonser som ej hör ihop med föreningens verksamhet eller ej på ett seriöst sätt gäller försäljning av originalexemplar av program.

Föreningens tillbehörsförsäljning:
Följande tillbehör finns att köpa genom att motsvarande belopp insätts på postgiro 19 83 00-6 (porto ingår)

Användartips med Mini Memory	20:-
Nittinian T-tröja	40:-
99er mag. 12/82, 1-5,7-9/83(st)	40:-
Nittinian årgång 1983	50:-
Programbiten årgång 84-89(styck)	50:-
TI-Forth manual	100:-
Hel diskett ur programbanken(st)	30:-

Enstaka program 5:- st + startkostnad 15 kr per skiva eller kassett (1 program=20kr, 3 program=30 kr). Se listor i PB89-3 och PB90-4.

Jan Alexandersson, Springarvägen 5, 3 tr, 142 61 TRÅNGSUND (08-7710569)

DSK.SKIVA.FILNAMN

av Jan Alexandersson

Det är möjligt att anropa en fil med DSK.SKIVNAMN.FILNAMN istället för DSK1.FILNAMN. Programmet kommer då att börja att söka på DSK1 efter en skiva med namnet SKIVNAMN. Om denna skiva inte hittas på DSK1 så fortsätter sökningen på DSK2 o.s.v. tills skivan hittas varefter filen FILNAMN sökes på denna skiva. Jag tycker att denna sökning från skivenhet till skivenhet känns mycket långdragen så den enda vettiga användningen är för skivor som skall stoppas i DSK1. Detta är mycket användbart om du endast har en skivenhet, hårddisk med DSK-emulering eller RAM-disk på CRU >1000.

Om du har ett TI-kontrollkort (på CRU >1100) så kommer sökningen från Basic att börja med en eventuell RAM-disk CRU >1000 och sedan att ske i nummerordning innan felmeddelande ges: DSK1 - DSK2 - DSK3 - I/O - ERROR 57. Någon RAM-disk DSK9 på CRU >1200 eller högre kan ej nås.

Jag har själv Myarc HFDC (DSK5-8) på CRU >1000 och TI-kontrollkort (DSK1-3) på CRU >1100 vilket medför att sökningen blir: WDS1 - DSK5 - DSK6 - DSK7 - DSK8 - DSK1 - DSK2 - DSK3 - I/O ERROR 57. RAM-disk på höga CRU-adresser kan ej nås eftersom TI-kortet hindrar detta. Multiplan som lagras på hårddisk i subdirectory WDS1.DSK.TIMP kommer således att ladda snabbt och fint.

Om jag tar bort TI-kortet och endast har Myarc HFDC på CRU >1100 tillsammans med Horizon RAM-disk (DSK9) på höga CRU-adresser kommer sökningen att bli: WDS1 - DSK1 - DSK2 - DSK3 - DSK4 - DSK9 - I/O ERROR 50. Höga CRU kan således nås.

Om samma sökning görs från Extended Basic så kommer märkligt nog sökningen att upprepas två gånger innan felmeddelandet ges i alla tre ovanstående fall. Med endast TI-kort fås som exempel DSK1 - DSK2 - DSK3 - DSK1 - DSK2 - DSK3 - I/O ERROR 07. Myarc-kortet ger felmeddelandet I/O ERROR 00. Kanske beror detta på min DIJIT AVPC. Hör av dig och berätta hur en vanlig 99/4A fungerar. ■

CRU-ADRESSER

av Jan Alexandersson

Nästan alla kort i expansionsboxen använder samma 8 kbytes-adresser på CPU-adress >4000 - >5FFF. För att detta ska fungera så får endast ett kort åt gången vara inkopplat till dessa adresser. Detta ordnas genom den för TMS 9900 unika CRU-adressbussen. Det är möjligt att adressera 16 olika kort genom att aktivera CRU-adress >1000, >1100, >1200 o.s.v. upp till >1F00. Du får aldrig ställa in två kort så att de svarar på samma CRU-adress. Det kan då finnas risk att de förstörs när de samtidigt är aktiverade. Här följer en lista över de kort jag känner till och möjliga CRU-adresser (HEX):

Myarc HFDC (16 olika)	1000-1F00
GRAM-karte (16 olika)	1000-1F00
Horizon RAM-disk (8 olika)	1000-1700
Corcomp RAM-disk (2 olika)	1000+1400
Mechatronic 80-kolumns	1000
Myarc RAM-disk	1000
TI diskkontrollkort	1100
Corcomp diskkontrollkort	1100
Myarc FDC	1100
RS232/PIO nr1	1300
Mechatronic 128kB+Printer	1400
DIJIT AVPC 80-kolumns	1400
RS232/PIO nr2	1500
TI Thermal Printer	1800
Mechatronic EPROM-er	1900
Triple Tec	1D00
Foundation RAM-disk	1E00
P-code för Pascal	1F00

Tony Lewis har i sin Interface Standard & Design Guide (se PB 90-1) föreslagit följande val av CRU-adresser för nykonstruerade kort:

1000	mass storage
1100	disk controller
1200	math coprocessor
1300	RS232-1
1400	internal modem
1500	RS232-2
1600	prototype low
1700	attached computer
1800	MIDI/music
1900	programmer
1A00	speech/DSP
1B00	utility card
1C00	video
1D00	real time clock
1E00	prototype high
1F00	P-code

■

MCGOVERNS XB-SKOLA, DEL 7

av Tony McGovern, Australien

Vi fortsätter nu XB-skolan med detaljerade metoder att krympa programmets längd. Som jag påpekat tidigare, är det inte ett ämne som jag är tveklöst positiv till, eftersom denna serie har ägnats åt bättre XB-programmering, då de flesta åtgärder du kan göra för att krympa Basic-program gör dem mindre läsliga för vanliga människor, med hänsyn till rimliga programmeringskunskaper. Det andra skälet till min tveksamhet är att denna typ av diskussion tenderar att urarta till en lista med osammanhängande saker, ytterligare en uppsättning "Tips", när jag istället vill att denna XB-skola skall vara en enkel systematisk titt på maskinen och dess språk.

Hur som helst låt oss börja med mindre saker och arbeta oss uppåt till större skala. Sist tittade vi på det utrymme som upptogs av enkla variabler. Det mest närliggande är att använda korta variabelnamn. Jag rekommenderar inte detta förrän sent i utvecklingen eftersom det är ett så enkelt och uppenbart sätt att käna bytes att du lika gärna kan behålla hjälpen av beskrivande variabelnamn tills du är helt desperat efter bytes. Total desperation har inte inträffat förrän du har gått igenom flera omgångar med att spara bytes. Det kortaste variabelnamnet har endast ett bokstavstecken, men TI Basic tillåter officiellt även "@" (shift-2) och "_" (fctn-U) som variabelnamn. Det kräver ett ganska långt SUB-program innan du behöver fler än 26 enkla numeriska variabler men det kan inträffa. På min egen konsol finns det även 3 andra enkla tecken som kan användas som variabelnamn (Ä fctn-R, Ö fctn-Z och Å fctn-T ö.a.). Prova om de även finns på din maskin. Det tråkiga är att detta aldrig har dokumenterats.

Det finns ett annat sätt att använda variabelnamn för att korta ned ett program. Från senaste XB-skolan kommer du ihåg att en numerisk konstant med med ett tecken behandlas som en sträng och behöver 3 bytes, medan en enkel bokstavsvariabel endast tar 1 byte. Om ett visst numeriskt värde uppträder ofta i ett SUB-program, 0 eller 1 är vanliga exempel, så kan det vara värt overheaden, 14 bytes plus tilldelningskommandot, för en ny variabel med detta värde när du kan spara 2 bytes vid upprepade tillfällen. En ofta använd numerisk konstant, som kan förekomma i CHAR eller SPRITE tilldelningar, kräver fler bytes varje gång. Det gäller att göra en noggrann hushållning och bytes-räkning i varje SUB-program. När du en gång har börjat att följa denna linje bör du tänka på ytterligare möjligheter till inbesparingar -- om du har definierat S=7 och F=5 kan du sedan spara bytes genom att skriva S*F istället för 35. Om du kan återanvända ett redan definierat variabelnamn så kan investeringen betala sig ännu snabbare, men detta kräver att du noggrannt följer programflödet. Gå tillbaka till exemplet en KEY/JOYST-rutin tidigare i XB-skolan och titta efter om du kan reducera antalet använda variabler.

Utbyte av tal mot variabler används i andra språk. I TI-FORTH behandlas talen 0,1,2,3 inte som ett tal utan som ett definierat ord i språket.

Det finns ytterligare ett listigt sätt att mata in tecken som kan korta ned programmet. Detta gäller inmatning av grafiska tecken med ASCII-koder över 127 tillhörande de högre färggrupperna (13-14) i XB, genom att skriva strängar med DISPLAY AT istället för CALL HCHAR och VCHAR (detta får ej finnas i program som skall publiceras i PB, eftersom dessa tecken ej kan skrivas ut med skrivaren på något begripligt sätt ö.a.). Tecken inom detta intervall (128-143) kan matas in i strängar i programrader med hjälp av CTRL-tangenten, istället för CHR$-funktionen. Det tenderar att göra programmet obegripligt eftersom alla dessa visas som blanktecken på skärmen. De visas med sitt definierade utseende om raden editeras efter programmets körning. Dessa koder

används även som XB-symboler och kan således endast användas inom strängar. Jag måste tillägga att jag är helt överens med TI:s konstruktörer att inte tillåta förkortad (direkt symbol) inmatning av reserverade ord i Basic (som finns i Sinclair Spectrum ö.a.). Om du vill ha detta så bör du återvända till Sinclair eller Commodore, och du tror då sannolikt inte heller på relokerbara objektsfiler.

Användning av fältvariabler (vektorer/matriser) för att representera en mindre uppsättning av tal kräver noggrann planering. Vinsten av mindre overhead i variabeltabellen och förenklad parameteröverföring till underprogram måste vägas mot de extra bytes som behövs vid varje användning i programmet. Låt programmets logik vara din första vägvisare.

Denna idé att använda färre bytes för att representera tal leder oss till den allmänna frågan om att packa data. En bytes kan bära 256 olika värden, och en tredjedel till en fjärdedel av dessa kan matas in från tangentbordet på ett enkelt sätt. Det är överdrivet att använda 8 bytes flyttal för att representera ett fåtal värden, eller t.o.m. enbart logiska (boolska) variabler vilka endast behöver en bit. Vissa språk packar boolska variabler som bitar i ett eller flera ord. Enbitarsbussen CRU i TMS-9900 tillhandahåller ett utmärkt verktyg för att lagra och testa enstaka bit, men som i många andra avseenden gör inte 99/4a-hårdvaran rättvisa åt sin CPU. Den nyare TMS-9995 har faktiskt ett litet inbyggt CRU-minne för detta ändamål.

Möjligheterna till datakomprimering är begränsade i XB p.g.a. språkets struktur (det har endast teckensträngar, flyttal och fältvariabler av dessa som datatyper) och det inskränkta långsamma sätt som det har förverkligats på via GROM- och VDP-minne. Varje metod att koda eller krympa kräver beräkningar för packning och uppackning. På maskinkodsnivå är valet mellan minnesutymme och snabbhet olika mot vad som gäller Basic, i synnerhet TI-99 Basic, eftersom Basic är mycket långsammare. Enligt min egen erfarenhet är användning av strängvariabler för att krympa data i aktiva delar av ett program nästan alltid dömt att misslyckas p.g.a. långsam stränghantering och uppehåll för 'garbage collection'. Komprimering av data kan dock vara användbart vid tilldelning av ursprungliga grafiktecken och för musikdata. Det finns alltid ett begränsat antal noter, volym och varaktighet som används i ett kort musikstycke, och eftersom varje not tar sin tid att spela och detta handhas av maskinen som interrupt, kan denna tid användas för de beräkningar som behövs för att plocka fram data för nästa not.

Låt oss titta på exemplet med den grafiska skärmen. Antag att vid skapandet av en spelskärm, endera av två tecken, kan ha samma mönster i två olika färggrupper, skall skrivas till 20 platser på olika delar av skärmen. Det enklaste sättet är en hel uppsättning CALL HCHAR - med antagandet att bilden inte är lämplig att skapas med DISPLAY AT - och detta är vad du brukar hitta i många program (liksom långa listor av CALL SOUND). Vad som är fullständigt oförlåtligt är att finna inkompetenta tidskrifter och kommersiella program med otillräcklig kodning som tvingar användaren till obekväma saker som CALL FILES(1) eller (2) när detta hade kunnat undvikas (Basic- och XB-program för publicering i PB bör ej vara längre än 11838 bytes ö.a.).

```
1000 CALL HCHAR(23,12,105)
1010 CALL HCHAR etc etc
```

Detta tar mer än 600 bytes. Hur kan det kortas ned? Ett sätt, som är något av en återvändsgränd i detta exempel, är att använda flera kommandon per rad. Detta skulle korta ned med ungefär 30 bytes, och vara obetydligt snabbare. Den verkliga förbättringen är att ta bort upprepningen av CALL HCHAR - kom ihåg att CALL är billigt men HCHAR är dyrt - genom att använda en slinga med DATA-satser.

```
1000 FOR I=1 TO 20 :: READ A
,B,C :: CALL HCHAR(A,B,C)::
NEXT I
1010 DATA 23,12,105, etc etc
```

Nu har alla utom en av dessa CALL HCHAR försvunnit. Kostnaden för detta är loop- och DATA-overhead samt den ökade möjligheten för skrivfel eftersom DATA-satserna har skiljts från sitt sammanhang. I detta skede kan du känna dig mycket nöjd med dig själv, men sedan upptäcker du att för att få plats med ytterligare saker i programmet så behöver du mer utrymme. Nu är det dags att seriöst fundera på datakomprimering. Ett kolumnnummer för HCHAR kan endast ha värdena 1 till 32 och raderna 1 till 24. Ett av dessa värden kan uttryckas med 1 byte. Antag att du således använder 1 byte för varje rad- eller kolumnvärde. Att uttrycka dessa bytes som DATA på ett ekonomiskt sätt är nästa problem - det åtgår några bytes av overhead för varje data i en DATA-lista, och DATA-listor med många korta data är ökända för att orsaka ett felmeddelande "line too long". Så låt oss packa dem i en ensam sträng och använda SEG$ för att packa upp dem, med ASC för att omvandla ett ASCII-tecken tillbaka till värden för HCHAR. Ett mindre problem är att tecken 1 till 32 inte kan matas in direkt i XB, så börja istället med tecken "A" och subtrahera 64. Det motsatta problemet kan uppstå med strängar för teckenvärden om de övre grafikseten används. Använd då istället lägre värden och addera ändringen. Så nu kan koden se ut så här

```
1000 READ A$,B$,C$ :: FOR I=
1 TO 20 :: CALL HCHAR(ASC(SE
G$(A$,I,1))-64,ASC(SEG$(B$,I
1))-64,ASC(SEG$(C$,I,1)+32):
: NEXT I
1010 DATA "W... ","L... ","I.
.. "
```

Du kan gå ännu längre och packa data i en enda sträng och modifiera SEG$-funktionen med hänsyn till detta, men det är kanske inte värt detta. Kom ihåg att problemet består i att bara skriva två olika tecken och räkna ut hur du kan krympa ytterligare för detta begränsade fall. Detta exempel bygger på en av de metoder som användes för att krympa TEX-BOUNCE till konsolens minnesutrymme. Ett extremt exempel på datakomprimering uppstår när data är tillräckligt regelbundet för att genereras med en formel eller procedur. Detta är något som måste undersökas i varje enskilt fall.

Användningen av loopar som i ovanstående exempel är tillämpbart i andra fall, speciellt när det gäller CHAR-definitioner. XB tillåter multiple-argument i CHAR, COLOR, SPRITE och liknande underprogram. Detta är bättre och snabbare än att använda individuella anrop av CALL för varje post i listan. Det verkliga problemet uppstår när du försöker använda en loop för att krympa programmet ytterligare. Kritiska delar av programmet kan bli oacceptabelt långsamma så att du tvingas att använda kompakt och långsam kod i vissa delar av programmet och längre men snabbare former i övrigt. I förbigående ska jag påminna dig om att nollställa vid uthopp från underprogram, när det gäller alla strängvariabler som inte måste ha kvar sitt värde till nästa CALL. Detta gäller i synnerhet strängvariabler som används för READ, INPUT, PRINT o.s.v. som kan ha långa strängar. Kom ihåg att det är programmets längd under körningen som har betydelse.

IX. Ytterligare buggar i XB

Jag har tittat på några buggar i tidigare delar av XB-skolan där de passade in på ett naturligt sätt, i huvudsak i ACCEPT AT. Denna gång har vi ytterligare två vackra exempel. Den första av dessa avslöjades i TI*MES våren 85 av John Bingham. För att se den i funktion, knappa in följande lilla program

```
100 I=1 :: IF I=1 THEN J=1 :
: GOSUB 200 ELSE K=1 :: J=2
110 PRINT K;J :: STOP
200 RETURN
```

Innan du kör detta, försök gissa vad det ska skriva ut! Kör det sedan och se vad du får. Byt sedan ordning på kommandona mellan THEN och ELSE och kör det igen. Nu kommer det att fungera som du hade väntat dig, K=0 och J=1. Om din XB gör rätt även i första fallet, så är den annorlunda än den som jag har. Närvaron av GOSUB alldeles före ELSE tycks ha lurat XB-mekanismen som håller reda

på ELSE, och programmet har gått vidare och inte brytt sig om ELSE och kommandon efter detta och kört efterföljande kommandon som den borde ha låtit bli helt och hållet när den fortsätter till nästa rad. Försök att byta ut GOSUB mot ett CALL till ett underprogram. XB kommer då att fungera precis som väntat. Detta är ytterligare ett skäl till att använda underprogram istället för GOSUB (jämför PB 90-4 sid 24 Swedlow XB och PB 90-5 sid 3 HTA punkt 15 ö.a.).

XB-manualen radar upp ett antal förbud när det gäller vad som får finnas i IF..THEN..ELSE.. kommandot, men nämner inte denna lilla skönhet. Det tycks, trots varningarna, som om FOR..NEXT loopar kan komma efter sista ELSE utan problem, men denna användning kan inte rekommenderas eftersom man ej kan vara säker på att detta gäller samtliga XB-moduler.

Jag måste medge att när jag läste denna nyhet blev jag något orolig, eftersom jag har skrivit långa och ordentligt avlusade program med några invecklade IF..THEN..ELSE, och jag har aldrig råkat ut för detta problem. Hur kan det komma sig? Det första rådet i XB-skolan har betydelse, och jag använder mycket få GOSUB och väldigt många underprogram om jag inte är fullständigt desperat efter mera bytes. Detta gällde i åtskilliga fall när det gäller TEX-BOUNCE och det viktigaste underprogrammet, ett av 12 i programmet, som självt innehåller 12 subrutiner skrivna för att spara bytes. Noggrann genomgång av koden för TEX-BOUNCE visade att inget av dessa GOSUB var skrivna så att de släppte lös denna elaka bug. När man tittar tillbaka på detta, kan man undra om du har slängt delar av koden som aldrig fungerade tillfredsställande, p.g.a. helt felaktiga skäl.

Den andra bugen har inte utforskats fullt ut. Den visade sig i CALL LINK från XB till en assemblerrutin som även överförde en strängvariabel som skall användas av denna rutin. Det inträffade i COLIST-programmet, den fullständiga versionen av SIMPLIST-programmet som finns i tidigare avsnitt av XB-skolan, symptomet var att maskinen kraschade fullständigt när den skrev ut en rad av sin egen listning, efter det att den redan hade skrivit ut flera hundra rader, av vilka många var ganska lika den som orsakade problemet. Inte ett fel som fångas in och rapporteras av XB, utan ett fullständigt paralyserande grepp. Det visade sig efter spårande med TRACE att det låg i själva raden som som bearbetades för utskrift, och som berodde på att LINK-namnet befanns sig på en viss position i textraden som överfördes med STRREF. Problemet tycks bero på att CALL LINK utökar sitt sökande av LINK-namnet till ställen den inte borde, men mer utredande behövs.

En tråkig upptäckt gjordes under letandet av buggen. Jag disassemblerade de XB-maskinkodsrutiner som laddas av CALL INIT för att se om problemet låg i STRREF. Den goda nyheten är att koden verkar vara OK, men den dåliga nyheten är att STRREF läser strängar från VDP på samma långsamma sätt som konsolen gör, 1 byte åt gången, och återställer VDP-adresserna varje gång. Detta är det besvärande långsamma sätt som GPL gör det på eftersom konsolen knappt har något CPU-RAM, men det är knappast nödvändigt i STRREF som endast kan användas när det finns expansionsminne närvarande.

Ytterligare en bug har rapporterats i USA i några äldre moduler, jag misstänker att det gäller modeller före V110 som har sålts i Australien. Detta uppträder när länknamnet kommer som en strängvariabels, som i CALL LINK("A$"). Om 'garbage collection' utföres i VDP-RAM av XB-interpretatorn mellan tilldelningen av A$ och användningen av det i CALL LINK, kommer denna rutin att tappa kontakten med A$. Jag har inte själv råkat ut för denna bug, men jag ska försöka locka fram den i den modul jag har. Detta påminner mig om det underliga förhållandet att XB har möjligheten av DELETE av en fil som strängvariabel medan detta inte är möjligt med RUN av en fil som stränvariabel, vilket gett upphov till en hel uppsättning av program som kringgår detta problem (RUN F$ går ej men RUN "DSK1.FILNAMN" fungerar ö.a.).

X. Några sista små saker

Som du säkert vet, kan inte alla Basic-program köras i XB, och vissa misslyckas fullständigt, vanligen p.g.a. att färre färggrupper kan användas än i konsol-Basic. Antag att du vill ha ett program som kan köras med både Basic och XB, och bestämma vilken interpretator som har kontrollen utan att krascha programmet. Om det kan köras med båda så bör du även kunna editera det med båda! Några förslag är att använda PI som är ett reserverat ord i XB där det ger det vanliga matematiska värdet. Detta fungerar dåligt vid editering. Ett bättre sätt är att titta på det första värde som ges av RND. Det ursprungliga startvärdet har ett entydigt värde som är olika i de två fallen. Använd bara inte RANDOMIZE först (se även PB 85-2 sid 3 programrad 240-250 ö.a.).

Om du skriver ett sådant program måste du även ta hänsyn till hur du använder kolon i PRINT-satser för att generera frammatning av flera rader. XB kommer att uppfatta två kolon i rad som som symbolen för separation mellan två satser. Det är nödvändigt att använda mellanslag mellan två kolon i PRINT-satser så att dessa inte ska uppfattas som separation mellan två satser som i

```
200 PRINT "A": : :"B"
```

(det är inte alls nödvändigt att använda semikolon som Tigercub antyder ö.a.).

XI. Återblick

Denna XB-skola skrevs ursprungligen ungefär ett år efter det att TI övergav 99/4a. Den har till och med översatts till svenska sedan dess. Nu är det mer än 5 och på det 6:e året sedan den svarta fredagen i slutet av 83 och den gamla 99/4a lever fortfarande vidare, och den industriella styrkan hos det expanderade systemet har tillåtit många typer av nyutvecklingar, RAM-diskar på megabytes om så önskas, ett hårddiskkontrollkort, och det allra bästa ett 80-kolumnskort med användning av Yamahas vidareutveckling av TI:s ursprungliga videoprocessor. Maskinens möjligheter är på inga sätt fullständigt utnyttjade och det kvarstår en fascinerande dator att arbeta med, och även den ursprungliga grund-konsolen framstår som en levande och prisvärd introduktion till datorernas värd för en nybörjare. ■

HARD DISK BACKUP

by Garry Christensen, Australia

It is important to back up your hard drive on a regular basis but I am sure you are aware of the time that it takes. This programme is not a replacement for a proper back-up but may suffice for the more regular ones.

If you are like me, you will not have used half of the hard disk. The other half is just sitting there (spinning actually) blank.

The BACKUP programme will copy, sector for sector, the contents of the disk onto the second, unused half. If necessary you can recover the sectors and write them onto to first half.

This programme is distributed under the concepts of fairware. Monetary donations are not necessary however letters from users or donations of software will be appreciated. Of course, if you feel the desire to throw your wealth at me I could put it to productive use.

Garry J Christensen
17 Centaur St
Redcliffe Qld 4020
Australia

(HFDC-ägare som önskar en kopia kan kontakta Jan Alexandersson) ■

KÖPES

Kabel mellan TI-99/4A och expansionsbox med vidhängande kort och datorkontakt till höger om 99:an.

TI-Artist, GRAPH-X eller annat grafikprogram till TI-99/4A köpes.

Kenth Arvestrand, Flädervägen 11, 533 72 LUNDSBRUNN. Tel. 0511-57211.■

ASSEMBLER TUTORIAL - 2

by Tony McGovern, Australia

JUST ABOUT EVERYWHERE

This is the second instalment of the assembly tutorial on writing code to execute at addresses other than those where it was originally loaded and/or linked. We now look at how to write code that inherently does not care where it is placed. This is known as position independent code, or sometimes as PC (program counter) relative code in processors that have this as an explicit addressing mode.

The essential requirement for such code is that it cannot contain any references to absolute addresses within the code block itself, because by the very requirements it is not to be reliably found at any given address block. The only absolute address references allowable are to fixed system addresses or addresses elsewhere in the program that are established at load time. As we saw last time pure register operations or immediate addressing are fine because they are workspace pointer relative. So here is a simple piece of code that will execute anywhere on the left, and on the right a version that will not.

```
* Position independent

START  EQU  $
       LI   R0,>100
START1 MOV  *R1+,*R2+
       DECT R0
       JGT  START1
       RT
```

```
* Not position independent

START  EQU  $
       MOV  @LEN,R0
START1 MOV  *R1+,*R2+
       DECT R0
       JGT  START1
       RT
LEN    DATA >100
```

The code on the left contains no explicit addresses in the block of code itself. It presumes, as does the other side, that the contents of R1 and R2 have already been set up. The LI instruction reads a data word immediately following. Here it is obviously intended as count data. In other circumstances (say if R1 were loaded with a pointer to data in this relocatable code block) such data words cannot be a pointer in the position independent block which is fixed by either the assembler or the loader. On the right hand side however the code is NOT position independent because the label LEN written in to the first instruction by the assembler and/or loader does not shift with the code. Data location LEN would have to live at fixed address somewhere and not part of the block for the code to be position independent.

Full word addresses local to the position independent block have to be adjusted on the fly. The essential requirement is that the code block must be able to determine its current address to have any chance of adjusting full word address references. Here we come to one of the most amazing omissions from the 9900 instruction set. It already has a STWP instruction so that a program can save the current Workspace Pointer and so find where its register set is located, and a STST for preserving the current processor Status Register contents of status bits and interrupt mask. That makes 2 of 3 of the registers that define the processor state. Then why on earth did TI not go the whole hog and have a "STPC" instruction to give the programmer complete control? It is in fact possible to work around this difficulty by a programming trick. Remember that BL, Branch and Link, loads R11 with the address of the following instruction when it takes the branch. So do a BL to a dummy subroutine when the position independent block is entered and preserve the contents of R11. Remember that this BL itself must be a position independent call.

```
*  Method 1 to fake up "STPC"
PIENTR EQU  $
       BL   @RTADR
PCBASE EQU  $
       MOV  R11,R9

*  RTADR is address of an RT
*  R9 now contains PCBASE
```

```
*  Method 2 for "STPC" (better)

PIENTR EQU  $
       LI   R9,>045B
       BL   R9
PCBASE EQU  $
       ..

*  Also here in R9
```

Either of these does the job. This first presumes a reliable fixed or relocated address is known for an RT instruction. This may not always be available, or the position independent routine itself is all the code there is. In this, the usual case, the second method must be used. Here we construct an RT instruction in a register and do a BL to the register. Writers of books on structured coding would shake their heads in horror at the thought of dynamically creating a code sequence in the registers and executing it there. Then you bring out the "X" instruction !! In fact it is a perfectly valid technique and a nifty solution to problems in bank-switching of memory. Actually the machine code produced by high level language compilers such as C or Pascal does far more incomprehensible things than this. Use of a high level compiled language is well recognized as a good way to make disassembly of binary files very difficult to follow. Assemblers for most microprocessors would balk at an instruction like this, but remember that in the 9900 family register addressing is just a short form of memory addressing relative to the workspace pointer. You could do a version with BLWP too, but it is not worth the extra trouble.

So how to use this ? Indexed addressing mode combined with careful setting up of all addresses as offsets from the PCBASE pointer is the solution.

```
*  Demo example for Position Independent code

START  EQU  $
       MOV  R11,R10          Save return
       LI   R9,>045B         Load R9 with RT instruction
       BL   R9               Branch and Link to it
PCBASE MOV  R11,R9           Use R9 to hold offset base
       BL   @SUBR1X(R9)      Call with index from R9
       ;;
       B    *R10             Return

SUBR1  EQU  $
       MOV  @FLAG1X(R9),@FLAG2X(R9) Indexed to make P.I.
       RT

FLAG1  DATA >0               Data/flag item
FLAG2  DATA >0               Data/flag item

*  Data, subroutine pointers as offsets from PCBASE
*  Must follow code to avoid Illegal Forward References

SUBR1X EQU  SUBR1-PCBASE
FLAG1X EQU  FLAG1-PCBASE
FLAG2X EQU  FLAG2-PCBASE
```

Use of the index register R9 containing PCBASE now converts the offsets with respect to PCBASE within the position independent code block to actual memory addresses at the time of execution of the code, and makes these references independent of the actual position of the code block, as this base reference is obtained on the fly each time the code is executed. The price paid is that one register is tied up as the relocation register, and that indexed addressing is no longer available on the dynamically relocatable addresses.

Very few complete programs on the TI-99/4a are written to be totally position independent. One such example is the UL User List utility program in the Funnelweb system. For reasons of flexibility within the system, this in later issues was rewritten in this fashion so that the same program could be used as a utility with no conflict with the central menu UL entry. If you SAVE a strictly position independent program as an E/A program file then all it takes to have it run at some other address is to change the load address in the file header. Do that on a normal E/A program file and it will usually crash.

If you have the TI technical manual look up their DSR specifications. There it specifies a partial version of this sort of code. Obviously TI did not intend the PAD always to be at >8300, as for instance in the 9995 processor the fast memory block is on chip at a fixed address of >F000. Whatever the PAD address it was intended that the GPL workspace is at PAD+>E0 so offsets into the PAD could be derived at a STWP when in the GPL workspace, followed by subtraction of >E0 to get the dynamic relocation index register. This restriction is responsible for some of the contortionist code found in TI DSRs, but is no longer necessary for new DSRs as in Horizon style RAMdisks as all TI-99/4a's out there use >8300.

A lesser form of position independence peculiar to the 9900 family is that necessary to write subroutines which can be executed from any workspace. We might call this "Workspace Independent Code". The usual problem which prevents this is the use of absolute addresses for the least significant bytes of workspace registers. The E/A utilities contain some well known offenders. Usually a judicious use of SWPB or STWP instructions will resolve even the worst problems at the cost of only a word of code or so.

Code fragment from the E/A module utilities

```
**  Write VDP address
*
R2LB   EQU  UTILWS+5
*
WVDPAD MOV  *R13,R2          Get VDP address
       MOVB @R2LB,@VDPWA     Write the low byte of address
       SOC  R1,R2            Properly adjust VDP write bit
       MOVB R2,@VDPWA        Write high byte of address
       MOV  @2(R13),R1       Get CPU RAM address
       MOV  @4(R13),R2       Get byte count
       RT                    Return to calling routine
```

R1 is either cleared or loaded with >4000 before entry to set write or read. The problem here is that the LSB of the VDP address as specified in R2 is accessed at the absolute address of this byte. This makes the routine usable only from the UTILWS workspace. A better form of code that takes no more space is to use

```
MOVB @1(R13),@VDPWA
```

This is entirely in the spirit of the rest of the routine and in line with TI's recommendations in all their manuals for accessing register data from BLWP routines. I just don't know how such an ugly line of code got past any internal reviews of code at TI, unless the intention

was to prevent outside use of this routine as its address is not made externally available by the standard mechanisms. While they were at it TI also broke another one of their system rules that you see in DSR or console ROM code - finding the VDP addresses from R13 of the GPL workspace. Again one of these rules that the orphan status of the 99/4a has rendered of no concern any more (except to the designers of the Geneve).

That about wraps up this brief introduction to some of the more advanced subtleties of 9900 programming. I hope it will prove useful as guidance for aspiring programmers out there. ■

HFDC DSK1 FILE EMULATION SELECTOR

by Garry Christensen, Australia

For those who have the hard disk controller there is now a programme available that will make using it a lot easier. The controller has a method of redirecting all file access from "DSK1" to the hard disk by using a DSK1 emulate file. One file on the hard disk is active and it is to that file that all DSK1 access is sent.

Many programmes require that the disk be in drive 1 to load. These can be saved to the hard disk but before they can be used, the file needs to be made the active one. Till now the only programme that will do this is MDM5. This new programme will make that process much easier.

When it is run, the menu will be displayed. Select from 1 to 9 for the file that you wish to become active. Selecting the 0 option will clear the pointer so that the DSK1 file emulation is disabled.

(HFDC-ägare som önskar en kopia kan kontakta Jan Alexandersson) ■

MULTIPLAN SYLK FILES - 2

by Bill Harms, USA

More on Basic programs that create Symbolic Link (SYLK) files to load data into Microsoft's Multiplan.

The first article showed how to create a Multiple (!) SYLK file for one cell. It only described a few of the many SYLK symbols. See pages 205-208 of the Multiplan Manual for a complete list.

The most significant change in this month's program is that it can create a file with many cells -- a couple of columns of data.

After using this program, you could load the file into Multiplan and save it as a Normal file. Then you could eXternal Copy it or parts of it into another spreadsheet that has formulas and other data. You might load a month of information into a sheet that has many months of data plus formulas for calculating Year-to-Date, Average, Units per Time Period, etc.

As I mentioned last month, the Basic program must write a Display Fixed 128 file. You must change the File Descriptor Record (FDR) to make it look like the file is Internal Fixed 128. Otherwise Multiplan cannot read it. The only trick is to find the FDR!

This is a very sad deal, but it is very easy to change it with a good sector editor such as Miller Graphic's ADVANCED DIAGNOSTICS. Change byte 12 of the FDR for your SYLK file from Hex 0002 to Hex 0202. The FDR's are in Sectors 2 thru 32.

Another way is to use Barry Traver's fine program RAW (Read And Write) which is written in Assembly language. It will make the change right out of your Extended Basic program.

I said that the maximum width of a Multiplan cell (to view) is 27 characters. WRONG! It is 32, but the cell can hold 127 characters and you can format cells to be "Continuous". This allows TEXT to dis-

play/print right on over as far as it goes.

Each SYLK record (remember - files are made up of records) can include cell content (text, numeric value or formula), row and/or column numbers and many other symbols for you to describe windows, sheet bounds, formatting of sheet and individual cells, sheet links and more. In fact, you can include just about everything except Multiplan commands such as COPY and DELETE.

Your cell content data can be split into more than one record. You just keep on creating 128 character records of SYLK symbols and data until you run out of information that you want put into the SYLK file. Then you fill the last record in the SYLK file with nulls <CHR$(0)> so that it is 128 characters long.

Last month's program had 27 nulls which made it a 73 byte record. I couldn't get it to load using a larger number of nulls (like 123). R. Mitchell's program as published in the May, June and August, 1985 SUPER 99 MONTHLY had the key.

You continue building a string (numerics are included via STR$(xx)), using the & to concatenate each new symbol or data item onto the string as you go UNTIL the LENgth of the string exceeds 128. Then you print the first 128 characters of the string into your disk SYLK file. Right after printing you move any characters in your string beyond 128 to a temporary variable.

Then you move the remaining characters back into your main string and continue building until it has more than 128 characters again and you do another print to disk of another SYLK record.

There are two types of SYLK symbols: Record Type Descriptors (RTD) of 1 or 2 characters and Field Type Descriptors (FTD) which are preceded by a ";". These need to be surrounded by double quote marks (not mentioned in the Multiplan manual). CR/LF's are used to separate SYLK record types (careful - a Record Type or RTD is a SYLK symbol while a record is part of a file - records contain record types).

Let's look at a sample: "C;K". This is a RTD C which means that it is a data point. The FTD ;K means that the value of the data point follows. Once you set a row or column number with the RTD C and FTD of say ;Y (for row), all the remaining data points (;K or ;E) are put at that row. You only need give another "C;X" or "C;Y" when changing the row (;Y) or column (;X) of the data point.

One of the fastest ways to learn the correct formatting of SYLK records is to enter some stuff into Multiplan and save the sheet in Symbolic format. Then you can look at the file with a sector editor like ADVANCED DIAGNOSTICS.

The following program creates a SYLK file of 2 columns with 4 rows of values and a formula for the SUM of the 2nd column. The 2nd column is NAMEd per your choice.

NAMEing is a Multiplan technique. It is quite helpful since a name of a cell or a range of cells can be used in a formula (to wit) SALES-COST (SALES minus COST). You would have NAMEd some cells SALES and some others COST, so the cell with this formula yields the profit. It could be named PROFIT.

A normal formula might look like R4C6-R9C6. For a bunch of stuff you might use SUM(R1:40C3)+SUM(R90C4:5). Relative references (relative to the cell that has the formula) as in my program look like SUM(R[-5]C:R[-2]C).

The whole sheet looks like this when loaded into Multiplan:

```
column >> 1          2
          1 18       15
          2 1        75
       r  3 33       199
       o  4 400      77
       w  5          ------
          6          366
```

The value in R6C2 (366) is from the formula SUM(R[-5]C:R[-2]C).

Using Craig Miller's ADVANCED DIAGNOSTICS we can see the whole sheet!

```
I D ; P M P * * F ; W 1   2
 6 * * F ; D G 2 G 8 * * B
; Y 7 ; X 3 * * N N ; N W H
H ; E R 1 : 6 C 2 * * C ; Y
6 ; X 2 ; E S U M ( R [ - 5
] C : R [ - 2 ] C ) * * C ;
X 1 C ; Y 1 ; K 1 8 * * C ;
Y 2 ; K 1 * * C ; Y 3 ; K 3
3 * * C ; Y 4 ; K 4 0 0 * *
C ; X 2 C ; Y 1 ; K 1 5 * *
C ; Y 2 ; K 7 5 * * C ; Y 3
; K 1 9 9 * * C ; Y 4 ; K 7
7 * * C ; Y 5 ; K " - - - -
- - " * * W ; N 1 ; A 1   1
* * E * * * * * * * * * * *
```

The 128 Character SYLK record ends after the fourth character in the 15th line. This is also the end of the SYLK file.

R. Mitchell's program reads a Display Fixed 80 file and writes the data into a SYLK file that Multiplan can read. It has one word processor line on each row. You could change it to read Display Variable 80 or any other type of file.

Multiplan has a 255 row and 64 column limit. I have found that about 23 columns and 40 rows of formulas, data and labels (text) is about all that the 4A's RAM can hold.

The program I wrote, which is based on R. Mitchell's, follows. These comments may help you understand it.

Line 300 adds sheet identification, windows, format and bounds.

Line 310 adds the name you inputted in line 280 and the range for that name (R1:6C2).

Line 320 adds the formula for R6C2.

Line 400 adds the cell content as numerics (here taken from the array AMT).

Lines 370, 450 and 470 work together to ensure that each record is 128 characters long.

Lines 510 thru 540 write the SYLK records to disk.

Now you can load your data into Multiplan to suit your needs. Instead of DATA statements you can use a file INPUT statement instead of the READ statement in line 210.

```
100 ! SYLK * 11/85
110 DISPLAY AT(1,5)ERASE ALL:"A SMALL
SPREADSHEET"
120 ! data input section ********
130 OPTION BASE 1
140 DIM AMT(4,2)! array for input data
 *****
150 DATA 18,15
160 DATA 1,75
170 DATA 33,199
180 DATA 400,77
190 FOR A=1 TO 4 ! rows
200 FOR B=1 TO 2 ! columns
210 READ AMT(A,B)
220 NEXT B :: NEXT A
230 ! end of input routine ***********
240 !
250 DISPLAY AT(6,1):"ENTER DESIRED SYL
K FILE NAME   DSK"
260 ACCEPT AT(7,4)BEEP SIZE(-15):FILE2
$ :: IF FILE2$="" THEN 260
270 OPEN #2:FILE2$,DISPLAY ,FIXED 128,
OUTPUT
280 DISPLAY AT(9,1):"Enter DESIRED nam
e for 2nd  column. " :: ACCEPT AT(10,1
0)SIZE(10)VALIDATE(UALPHA):NAME$ :: IF
 NAME$="" THEN 280
290 R$=CHR$(13)&CHR$(10)
300 T$="ID;PMP"&R$&"F;W1 2 6"&R$&"F;DG
2G8"&R$&"B;Y"&STR$(A+2)&";X"&STR$(B)&R
$
310 T$=T$&"NN;N"&NAME$&";ER1:6C2"&R$
320 T$=T$&"C;Y6;X2"&";ESUM(R[-5]C:R[-2
]C)"&R$
330 !
340 FOR COL=1 TO 2
350 T$=T$&"C;X"&STR$(COL)
360 FOR ROW=1 TO A-1
370 IF LEN(T$)>128 THEN CALL WRITE(T$,
T1$):: T$=T1$
380 T$=T$&"C"
390 T$=T$&";Y"&STR$(ROW)
400 T$=T$&";K"&STR$(AMT(ROW,COL))&R$
410 NEXT ROW
420 NEXT COL
430 !
440 T$=T$&"C;Y"&STR$(ROW)&";K"&CHR$(34
)&"------"&CHR$(34)&R$
450 IF LEN(T$)>128 THEN CALL WRITE(T$,
T1$):: T$=T1$
460 T$=T$&"W;N1;A1 1"&R$&"E"&R$
470 IF LEN(T$)>128 THEN CALL WRITE(T$,
T1$):: T$=T1$                          ■
```

PERISCOPE – SPEL FÖR XB

av J.R.Dew, USA

```
100 REM ************
110 REM * UP SCOPE *
120 REM ************
130 REM BY J.R.DEW
140 REM 99'ER 82-11 XB
170 DEF WT(X)=X*100+100*INT(
RND*6)
180 GOSUB 1570 :: DISPLAY AT
(8,1):"BEST SCORE FILE:":" C
-CASSETTE":" D-DISKETTE":" N
-NONE":" I-INITIALIZE" :: MA
SK$="CDNI" :: GOSUB 1560
190 SFTYPE=K2 :: ON K2 GOTO
200,210,240,230
200 OPEN #1:"CS1",FIXED,INPU
T ,INTERNAL :: GOTO 220
210 OPEN #1:"DSK1.FISHFILE",
INPUT ,INTERNAL
220 FOR X=1 TO 5 :: INPUT #1
:B(X),BEST$(X):: NEXT X :: C
LOSE #1 :: GOTO 240
230 FOR X=1 TO 5 :: B(X)=-1
:: NEXT X :: DISPLAY AT(11,2
):"":"":"CHOOSE C OR D" :: M
ASK$="CD" :: GOSUB 1560 :: S
FTYPE=K2
240 TURN=0
250 RANDOMIZE :: GOSUB 1570
:: X$=RPT$("0",16)
260 DISPLAY AT(22,1):"INSTRU
CTIONS (Y/N)" :: MASK$="YN"
:: GOSUB 1560
270 MASK$="TDP" :: IF K2=2 T
HEN 460
280 DISPLAY AT(1,2)ERASE ALL
:"YOU HAVE 3 COMMANDS:":" P-
PERISCOPE:BRING A SHIP":"
INTO VIEW ON YOUR":"   PERIS
COPE":" T-FIRES TORPEDO"
330 DISPLAY AT(6,2):"D-DIVE.
 THIS SHOULD BE USED":"   ON
LY WHEN UNDER ATTACK.": :" W
HEN THE SCREEN TURNS RED":"
YOU ARE UNDER ATTACK. YOU"
370 DISPLAY AT(11,2):"MUST E
ITHER SINK YOUR":" ATTACKER
WITH TORPEDO FIRE":" OR DIVE
 FOR SAFTEY-BUT":" WATCH OUT
 FOR DEPTH CHARGES"
410 DISPLAY AT(16,2):"THE OB
JECT OF THE GAME IS":" TO SI
NK THE MAXIMUM TONAGE":" OF
ENEMY SHIPS WITH YOUR":" SUP
PLY OF TORPEDOES."
450 DISPLAY AT(24,2):"PRESS
ENTER TO PLAY" :: ACCEPT AT(
24,26):E$ :: CALL CLEAR
460 CALL CHAR(136,X$&"3F7F3F
000000000000000000000"&"40E
0FCFEFC0000000000")
470 CALL SPRITE(#6,136,3,1,1
00,2,0):: CALL MAGNIFY(4)
480 IF B(1)THEN GOSUB 1580
490 DISPLAY AT(24,2):"PRESS
ENTER TO PLAY"
500 CALL SONAR :: GOSUB 1520
 :: CALL KEY(3,K,S):: IF K<>
13 THEN 500
510 CALL CLEAR :: CALL DELSP
RITE(#6):: Y$="FFFFFFFFFFFF
FFF" :: CALL COLOR(13,9,1,12
,4,1,9,6,4)
520 CALL CHAR(128,Y$,120,Y$)
:: CALL SUBMERGE
530 CALL CLEAR :: DISPLAY AT
(2,7):"SUBMARINE NAME" :: AC
CEPT AT(3,12)SIZE(12)BEEP:A$
540 CALL CHAR(116,X$&"004100
1108050215"&X$&"000400102040
08050")
550 CALL CLEAR
560 SITED,PSTAT=0 :: T=16 ::
 A=INT(RND*10):: GOTO 640
570 D=INT(RND*10)
580 IF D<6 THEN 640
590 IF D<8 THEN Q=2 :: GOSUB
 1530 :: GOTO 610
600 Q=1 :: IF D=8 THEN GOSUB
 1540 ELSE GOSUB 1550
610 GOSUB 1330 :: GOSUB 1410
620 DISPLAY AT(14,13):"ATTAC
KING"
630 CALL SCREEN(10):: CALL S
ONAR :: CALL SHIP :: GOTO 65
0
640 CALL SCREEN(15)
650 GOSUB 1330 :: CALL SONAR
 :: GOSUB 1440
660 DISPLAY AT(14,1):""
670 IF T=0 THEN 1160 ELSE CA
LL KEY(3,K,S):: IF S THEN K2
=POS(MASK$,CHR$(K),1):: IF K
2=0 THEN 680 ELSE ON K2 GOTO
 900,1020,720
680 IF SITED=0 THEN 670 ELSE
 CALL POSITION(#9,Y,X):: IF
X>96 THEN 670 ELSE CALL DELS
PRITE(#9)
690 IF D>5 AND Q>0 THEN 1150
700 DISPLAY AT(13,1):" " ::
DISPLAY AT(14,1):" " :: DISP
LAY AT(5,22):" " :: DISPLAY
```

```
AT(6,22):" "
710 CALL FIREDISP(0):: SITED
,F,D,Q=0 :: A=INT(RND*(TURN+
10)):: GOTO 570
720 REM PERISCOPE
730 IF SITED THEN 670 ELSE D
,SITED=1
740 IF PSTAT=1 THEN PSTAT=2
:: CALL SURFACE
750 IF A>8 THEN 820
760 IF A>3 THEN 770 ELSE R$=
"FREIGHTER" :: GOSUB 1480 ::
 W=WT(65)
770 IF A<7 THEN 780 ELSE R$=
"TANKER" :: GOSUB 1490 :: W=
WT(92)
780 IF A<4 OR A>5 THEN 790 E
LSE R$="TRANSPORT" :: GOSUB
1480 :: W=WT(100)
790 IF A<>6 THEN 800 ELSE R$
="AMMUNITION SHIP" :: GOSUB
1480 :: W=WT(90)
800 IF A=6 THEN Q=1 ELSE Q=2
810 GOTO 890
820 E=INT(RND*10):: D=9
830 IF E THEN 840 ELSE R$="B
ATTLESHIP" :: GOSUB 1500 ::
W=WT(330):: Q=6 :: GOTO 890
840 IF E<>1 THEN 850 ELSE R$
="AIRCRAFT CARRIER" :: GOSUB
 1510 :: W=WT(250):: Q=4 ::
GOTO 890
850 IF E<>2 THEN 860 ELSE R$
="HEAVY CRUISER" :: GOSUB 14
50 :: W=WT(99):: Q=3 :: GOTO
 890
860 IF E<>3 THEN 870 ELSE R$
="LIGHT CRUISER" :: GOSUB 14
50 :: W=WT(9):: Q=3 :: GOTO
890
870 IF E<4 OR E>5 THEN 880 E
LSE R$="DESTROYER" :: GOSUB
1450 :: W=WT(21):: Q=2 :: GO
TO 890
880 R$="DESTROYER ESCORT" ::
 GOSUB 1460 :: W=1350 :: Q=1
 :: GOTO 890
890 CALL SHIP :: GOSUB 1410
:: GOTO 670 !DRAW SHIP&DESCR
IBE
900 REM TORPEDO
910 IF Q<1 THEN CALL SONAR :
: GOTO 670
920 CALL FIREDISP(F+1):: F=F
+1 :: CALL SOUND(600,110,5,-
7,0)
930 T=T-1 :: DISPLAY AT(4,5)
SIZE(3):T
940 IF RND>.275 THEN Q=Q-1
950 IF Q>0 AND A>8 THEN CALL
 SCREEN(10)
960 TURN=TURN+.2 :: IF Q>0 T
HEN 670
970 O=O+W :: L=L+1 :: CALL P
OSITION(#9,Y,X)
980 CALL SPRITE(#9,116,7,Y,X
,0,0)
990 DISPLAY AT(14,13):"  SUN
K" :: DISPLAY AT(24,14):O
1000 TURN=TURN+.8 :: CALL SC
REEN(12)
1010 F,D,Q=0 :: CALL SONAR :
: CALL FIREDISP(0):: CALL DE
LSPRITE(#9):: A=INT(RND*10):
: GOTO 640
1020 REM DIVE
1030 PSTAT,F=0 :: CALL DELSP
RITE(#9):: CALL SCREEN(4)::
CALL CLEAR
1040 CALL SPRITE(#1,136,13,1
,128,2,0):: CALL SONAR
1050 IF Q=0 THEN 1080
1060 CALL CHAR(104,RPT$("0",
15)&"3C"&RPT$("0",30)&"C3")
1070 FOR X=1 TO INT(RND*1900
):: NEXT X :: CALL SPRITE(#2
,104,2,1,128,1+INT(RND*7),0)
1080 CALL POSITION(#1,Y,X)::
 IF Y>1 THEN 1120
1090 CALL POSITION(#2,Y,X)::
 IF Y>192 THEN 1120
1100 CALL DISTANCE(#1,#2,X)
1110 IF X<25 THEN 1130 ELSE
CALL SONAR :: GOTO 1080
1120 CALL SCREEN(12):: CALL
DELSPRITE(#1,#2):: PSTAT,SIT
ED,D,Q=0 :: CALL SUBMERGE ::
 A=INT(RND*(TURN+10)):: GOTO
 640
1130 CALL DELSPRITE(ALL):: C
ALL SOUND(2500,-7,0)
1140 DISPLAY ERASE ALL:"THE
USS ";A$:"HAS BEEN SUNK BY":
"DEPTH CHARGES" :: GOTO 1180
1150 DISPLAY ERASE ALL:"THE
USS ";A$:"HAS BEEN SUNK BY G
UNFIRE" :: GOTO 1180
1160 IF D>5 AND Q>0 THEN 115
0
1170 DISPLAY ERASE ALL:"OUT
OF TORPEDOES":"END OF MISSIO
N"
1180 PRINT "YOU SUNK";L;" SH
IPS":"        ";O;" TONS" ::
 CALL DELSPRITE(ALL)
1190 IF SFTYPE=3 THEN 1250
1200 FOR X=1 TO 5 :: IF O<=B
(X)THEN 1240
1210 FOR Y=5 TO X STEP -1 ::
 B(Y)=B(Y-1):: BEST$(Y)=BEST
```

```
$(Y-1):: NEXT Y :: B(X)=O ::
 BEST$(X)=A$ :: GOSUB 1580
1220 IF SFTYPE=2 THEN OPEN #
1:"DSK1.FISHFILE",OUTPUT,INT
ERNAL ELSE OPEN #1:"CS1",FIX
ED,OUTPUT,INTERNAL
1230 FOR X=1 TO 5 :: PRINT #
1:B(X),BEST$(X):: NEXT X ::
CLOSE #1 :: GOTO 1250
1240 NEXT X
1250 L,F,O=0 :: Q=300
1260 CALL SOUND(Q*2,131,0)
1270 CALL SOUND(Q,165,0)
1280 CALL SOUND(Q,196,0)
1290 CALL SOUND(Q*1.5,220,0)
1300 CALL SOUND(.5*Q,165,0)
1310 CALL SOUND(2*Q,220,0)
1320 CALL SOUND(4000,3000,30
):: GOTO 240
1330 REM DRAW PERISCOPE ETC
1340 IF PSTAT THEN 1400 ELSE
 PSTAT=1
1350 DISPLAY AT(1,14-LEN(A$)
/2-2):"USS ";A$
1360 CALL HCHAR(2,12,128,10)
:: CALL HCHAR(11,12,128,10)
1370 CALL VCHAR(3,12,128,8):
: CALL VCHAR(3,21,128,8)
1380 DISPLAY AT(3,3)SIZE(5):
"TORPS" :: DISPLAY AT(4,5)SI
ZE(3):T
1390 FOR X=3 TO 6 :: CALL HC
HAR(X,13,93+X,8):: CALL HCHA
R(X+4,13,120,8):: NEXT X
1400 DISPLAY AT(21,1):O :: R
ETURN
1410 SITED=1 :: DISPLAY AT(1
3,4):"ENEMY"
1420 DISPLAY AT(13,13):R$ ::
 DISPLAY AT(5,23):W
1430 DISPLAY AT(6,24):"TONS"
 :: RETURN
1440 DISPLAY AT(20,2):"P=PER
ISCOPE":" T=TORPEDO":" D=DIV
E" :: DISPLAY AT(16,2):"ORDE
RS COMMANDER?" :: RETURN
1450 CALL CHAR(132,X$&"00000
00000147F3F"&X$&"0000000060F
4FEFC"):: RETURN ! CRUISER &
 DESTROYER
1460 CALL CHAR(132,X$&"00000
000000A3F1F"&X$&"00000000006
0FCF8"):: RETURN ! ESCORT
1470 CALL CHAR(132,X$&"00000
0000000000F"&X$&"00000000000
060F0"):: RETURN ! TORPEDO B
OAT
1480 CALL CHAR(132,X$&"00000
000007F3F1F"&X$&"00000050F8F
EFCFE"):: RETURN ! FREIGHTER
 ETC
1490 CALL CHAR(132,X$&"00000
0000000FF7F"&X$&"00000000287
CFFFE"):: RETURN ! TANKER
1500 CALL CHAR(132,X$&"00000
0000065FF75"&X$&"00000040C0E
6FFFE"):: RETURN ! BATTLESHI
P
1510 CALL CHAR(132,X$&"00100
0000800FF7F00000000000040000
00000000038FFFE"):: RETURN !
 CARRIER
1520 CALL SOUND(1000,1000,30
):: RETURN
1530 R$="DESTROYER" :: W=210
0 :: GOSUB 1450 :: RETURN
1540 R$="DESTROYER ESCORT" :
: W=1400 :: GOSUB 1460 :: RE
TURN
1550 R$="TORPEDO BOAT" :: W=
75 :: GOSUB 1470 :: RETURN
1560 CALL KEY(3,K,S):: IF S<
1 THEN 1560 ELSE K2=POS(MASK
$,CHR$(K),1):: IF K2=0 THEN
1560 ELSE RETURN
1570 CALL SCREEN(4):: DISPLA
Y AT(4,9)ERASE ALL:"UP SCOPE
!" :: RETURN
1580 DISPLAY AT(6,9):"BEST G
AMES" :: FOR X=1 TO 5
1590 DISPLAY AT(X+6,2):"USS
" :: DISPLAY AT(X+6,6):BEST$
(X)
1600 DISPLAY AT(X+6,17):B(X)
;" TONS" :: NEXT X :: RETURN
1610 SUB SONAR
1620 CALL SOUND(100,440,0)::
 FOR X=3 TO 1 STEP -1 :: CAL
L SOUND(X,440,14):: NEXT X :
: SUBEND
1630 SUB SHIP :: IF RND>.6 T
HEN RS=-2 ELSE RS=-1
1640 CALL SPRITE(#9,132,13,1
7,129,0,RS):: SUBEND
1650 SUB FIREDISP(A):: IF A=
0 THEN 1660 ELSE X$="FIRED"
:: Y$=STR$(A):: GOTO 1670
1660 X$,Y$=""
1670 DISPLAY AT(6,3)SIZE(5):
X$ :: DISPLAY AT(8,5)SIZE(2)
:Y$ :: SUBEND
1680 SUB SURFACE
1690 FOR Y=96 TO 99 :: FOR X
=1 TO 16 :: CALL CHAR(Y,RPT$
("F",X)):: NEXT X :: NEXT Y
:: SUBEND
1700 SUB SUBMERGE :: FOR X=9
6 TO 99 :: CALL CHAR(X,""):
 NEXT X :: SUBEND
```

■

TIPS FROM TIGERCUB #59

TIPS FROM THE TIGERCUB

#59

Tigercub Software
156 Collingwood Ave.
Columbus OH 43213

I am still offering over 120 original and unique entertainment, educational and utility programs at just $1.00 each, or on collection disks at $5.00 per disk.

The contents of the first 52 issues of this newsletter are available as ready-to-run programs on 5 Disks REDUCED TO $5 each!

My three Nuts & Bolts Disks, ARE NOW REDUCED TO JUST $10 EACH. Each contains over 100 subprograms for you to merge into your own programs to do all kinds of wonderful things.

My catalog is available for $1, deductable from your first order (specify TIGERCUB catalog).

****

TI-PD LIBRARY

I have selected public domain programs, by category, to fill NOW OVER 300!! disks, as full as possible if I had enough programs of the category, with all the Basic-only programs converted to XBasic, with an E/A loader provided for assembly programs if possible, instructions added and any obvious bugs corrected, and with an autoloader by full program name on each disk. These are available as a copying service for just $1.50 postpaid in U.S. and Canada. No fairware will be offered without the author's permission. Send SASE for list or $1, refundable for NEW 11-page catalog listing all titles and authors. Be sure to specify TI-PD catalog.

The first few disks that I distributed of my Tips #58 had a poor version of my Datawriter program. Those which are titled TIPS #58.1 have the correct version. If you received the bad one, please ask me for another.

In Tips #42 I published this algorithm and asked why I could not get C to equal 9 or 12 or 99.

```
100 CALL CLEAR
110 DISPLAY AT(18,1):"A? " :
: ACCEPT AT(18,4):A :: DISPL
AY AT(20,1):"B? " :: ACCEPT
AT(20,4):B
120 IF A=2 THEN IF B=2 THEN
C=4 ELSE IF A=2 THEN IF B=3
THEN C=6 ELSE IF A=3 THEN IF
 B=3 THEN C=9 ELSE IF A=3 TH
EN IF B=4 THEN C=12 ELSE C=9
9
130 DISPLAY AT(22,1):"C=";C
:: GOTO 110
```

A couple of programmers wrote to tell me the error I was making. This way it will work -

```
120 IF A=2 THEN IF B=2 THEN
C=4 ELSE IF B=3 THEN C=6 ELS
E C=99 ELSE IF A=3 THEN IF B
=3 THEN C=9 ELSE IF B=4 THEN
 C=12 ELSE C=99 ELSE C=99
```

If an IF is not true, the program execution jumps to the first ELSE which is not already paired with an IF. If there is no such unpaired ELSE, execution jumps to the next program line. In the correct example, IF B=2 is paired with ELSE IF B=3, and IF B=3 is paired with the first ELSE C=99, so if A=2 is not true then execution jumps to ELSE IF A=3. The second IF B=3 is paired with ELSE IF B=4 and IF B=4 is

paired with the second ELSE C=99, so if A=3 is not true, execution jumps to the final ELSE C=99.

Here's a bit of a new idea in a spelling program. If your word has the correct number of letters but some are wrong, it shows you the wrong ones.

```
100 DISPLAY AT(3,11)ERASE AL
L:"SPELLIT" !by Jim Peterson
110 DATA HIPPOPOTAMUS,CRITIQ
UE,KHAKI,IRIDESCENT,ARCHAIC,
PNEUMONIA
120 !add as many DATA statem
ents as you want
130 FOR CH=97 TO 122 :: CALL
 CHARPAT(CH-32,CH$):: CALL C
HAR(CH,CH$):: NEXT CH :: CAL
L COLOR(9,8,2,10,8,2,11,8,2,
12,8,2)
140 DATA END
150 READ M$ :: T=100 :: IF M
$="END" THEN CALL CLEAR :: S
TOP
160 GOSUB 230 :: ACCEPT AT(1
2,1)SIZE(-28)BEEP:Q$
170 IF Q$=M$ THEN CALL SOUND
(100,392,5):: CALL SOUND(200
,523,5):: DISPLAY AT(12,1):"
" :: GOTO 150
180 FOR J=1 TO LEN(Q$):: IF
SEG$(Q$,J,1)=SEG$(M$,J,1)THE
N 210
190 DISPLAY AT(12,J):CHR$(AS
C(SEG$(Q$,J,1))+32);
200 T=T+50 :: IF LEN(Q$)=LEN
(M$)THEN GOSUB 230 :: GOTO 2
10 ELSE DISPLAY AT(12,J+1):"
 " :: J=LEN(Q$):: GOTO 160
210 NEXT J
220 T=T+50 :: GOTO 160
230 DISPLAY AT(10,1):M$ :: F
OR D=1 TO T :: NEXT D :: DIS
PLAY AT(10,1):"" :: RETURN
```

When I'm not writing programs, I like to write songs but the only way I can get anyone to listen to my music is to program it. Lucie Dorais of the Ottawa UG made this neat routine to program waltz music, and I plugged one of my songs into it.

```
100 CALL CLEAR :: CALL SCREE
N(2):: FOR X=4 TO 12 :: CALL
 COLOR(X,16,2):: NEXT X
110 CALL CHARPAT(39,A$,44,B$
,38,C$):: CALL CHAR(61,A$,96
,B$,64,C$)
120 DISPLAY AT(2,7):"SEDUCTI
ON WALTZ":"":"Words @ music
  Program by":"by Jim Peters
on Lucie Dorais"
130 FOR D=1 TO 700 :: NEXT D
140 CALL CLEAR :: CALL START
(95,23,95,199)
150 LB=123 :: LC=131 :: LD=1
47 :: LE=165 :: LF=175 :: LG
=196 :: LA=220 :: B=494 :: C
=523 :: D=587 :: E=659 :: F=
698 :: G=784 :: A=880
160 HB=988 :: HC=1047 :: RV=
5 :: CV=9
170 DIM M$(16):: FOR J=1 TO
16 :: READ M$(J):: NEXT J
180 GOTO 190 :: A$,TIME,X,Y
:: CALL DELSPRITE :: !@P-
190 FOR TIME=1 TO 4
200 Y=Y+1 :: DISPLAY AT(1,1)
:M$(Y):: CALL BAR(RV,CV,-RV,
-CV,G,C,C,LC,LE,LG)
210 CALL BAR(0,CV,0,-CV,C,D,
D,LC,LE,LG)
220 CALL BAR(-RV,CV,RV,-CV,C
,E,E,LC,LE,LG)
230 CALL BAR(-RV,0,RV,0,E,E,
E,LC,LE,LG)
240 Y=Y+1 :: DISPLAY AT(1,1)
:M$(Y):: CALL BAR(-RV,-CV,RV
,CV,G,C,C,LC,LE,LG)
250 CALL BAR(0,-CV,0,CV,C,D,
D,LC,LE,LG)
260 CALL BAR(RV,-CV,-RV,CV,C
,E,E,LC,LE,LG)
270 CALL BAR(RV,0,-RV,0,E,E,
E,LC,LE,LG)
280 Y=Y+1 :: DISPLAY AT(1,1)
:M$(Y):: CALL BAR(-RV,CV,RV,
-CV,G,A,A,LF,LA,LC)
290 CALL BAR(0,CV,0,-CV,A,F,
G,LF,LA,LC)
300 CALL BAR(RV,CV,-RV,-CV,A
,G,A,LF,LC,LA)
310 CALL BAR(RV,0,-RV,0,G,E,
E,LC,LE,LG)
320 Y=Y+1 :: DISPLAY AT(1,1)
:M$(Y):: CALL BAR(RV,-CV,-RV
,CV,G,G,D,LG,LB,LD)
330 CALL BAR(0,-CV,0,CV,F,E,
B,LG,LB,LD)
340 CALL BAR(-RV,-CV,RV,CV,D
,C,C,LC,LE,LG)
350 CALL BAR(-RV,0,RV,0,C,C,
C,LC,LE,LG)
360 CALL PATTERN(#1,132,#2,1
28):: CALL MOTION(#1,0,0,#2,
```

```
0,0)
370 NEXT TIME
380 CALL BAR(RV,0,RV,0,G,G,D
,LG,LB,LD)
390 CALL BAR(RV,0,RV,0,F,E,H
C,LG,LB,LD)
400 CALL BAR(RV,0,RV,0,HB,HC
,HC,LC,LE,LG)
410 CALL DELSPRITE(ALL)
420 CALL BAR(RV,0,RV,0,HC,HC
,HC,LC,LE,LG)
430 DISPLAY AT(24,11):"AGAIN
? Y" :: ACCEPT AT(24,18)SIZE
(-1)VALIDATE("YN"):A$
440 IF A$="N" THEN END ELSE
DISPLAY AT(24,11):"" :: Y=0
:: CALL START(95,23,95,199):
: GOTO 190
450 !@P+
460 DATA Dance with me by th
e light  of the moon,And swa
y to the throbbing  guitars
,The love that we feel is a
 love that is real
470 DATA And it=s witnessed
by only  the stars,Is it wro
ng to love and be  loved?,Is
 it wrong to kiss and     ca
ress?,Is it wrong to rest in
 the  arms`
480 DATA Of the one you love
 the     best?,What=s the di
fference what  others may sa
y?,Does it matter who is to
   blame?
490 DATA When the wind of sp
ring=s inyour hair,And the m
oon makes your     hearts be
at the same
500 DATA "Come with me` let
us run`   let us go",In the
darkness and no one  will se
e,Come with me to a place th
atI know
510 DATA On this wild night
of love  stay with me
520 !@P+
530 SUB START(R1,C1,R2,C2)
540 CALL CHAR(128,"070F0A0B0
B0307070F0F1F1F3F3F3F021C1C0
8FC1CFC1C1C9C88C8C8E8E8E818"
)
550 CALL CHAR(132,"3838103F3
83F38383911131317171718E0F05
0D0D0C0E0E0F0F0F8F8FCFC40")
560 CALL CHAR(136,"030301030
303030307070F0F1F1F1F0280800
08080808080C0C0E0E0F0F0F080"
)
570 CALL MAGNIFY(4):: CALL S
PRITE(#1,128,8,R1,C1,#2,132,
10,R2,C2):: SUBEND
580 SUB BAR(R1,C1,R2,C2,T1,T
2,T3,B1,B2,B3)
590 P2=128-4*(P1=128):: P1=1
28-4*(P2=128)
600 CALL PATTERN(#1,P1,#2,P2
):: CALL MOTION(#1,R1,C1,#2,
R2,C2):: FOR T=1 TO 4 :: CAL
L SOUND(-999,T1,5,T1*1.01,5,
B1,10):: NEXT T
610 FOR T=1 TO 6 :: CALL SOU
ND(-999,T2,5,T2*1.01,5,B2,10
):: NEXT T :: CALL PATTERN(#
1,136,#2,136)
620 FOR T=1 TO 4 :: CALL SOU
ND(-999,T3,5,T3*1.01,5,B3,10
):: NEXT T :: P1=P2 :: SUBEN
D
```

If you're watching your diet -

```
100 DISPLAY AT(1,1)ERASE ALL
:"NUTRITION LABEL INTERPRETE
R":"     by Jim Peterson":""
:"  To help you understand t
hemandatory FDA nutrition
  labels on food packages."
110 DISPLAY AT(8,1):"Calorie
s per serving?" :: ACCEPT AT
(8,23)VALIDATE(NUMERIC)BEEP:
C
120 DISPLAY AT(10,1):"Grams
of fat?" :: ACCEPT AT(10,15)
VALIDATE(NUMERIC)BEEP:F
130 DISPLAY AT(12,1):"Grams
of sucrose & other":"sugars?
" :: ACCEPT AT(13,9)VALIDATE
(NUMERIC)BEEP:S
140 DISPLAY AT(15,1):"Grams
of starch and other":"carboh
ydrates?" :: ACCEPT AT(16,16
)VALIDATE(NUMERIC)BEEP:K
150 F=INT(F*9/C*100+.5):: S=
INT(S*4/C*100+.5):: K=INT(K*
4/C*100+.5)
160 DISPLAY AT(18,1):STR$(F)
&"% of calories from fat":ST
R$(S)&"% of calories from su
gars":STR$(K)&"% of calories
 from":"starches and carbohy
drates"
170 DISPLAY AT(22,1):STR$(10
0-F-S-K)&"% of calories from
 dietary fiber."
180 GOTO 110
```

I came across this beauty of a routine in the Summer 1989 newsletter of the Lehigh 99'er Computer Group.

Assemble it with only the R option, as TRACK/O.

```
*****************************
*    BOOT TRACKING          *
*   Adrian Robinson         *
*      May 1989             *
* CALL LINK("TRACK",A$)     *
*****************************

       DEF  TRACK
STRASG EQU  >2010
VMBR   EQU  >202C
MYWS   BSS  32
H30    BYTE >30    >3011
       BYTE 11
DSK    BSS  11
       EVEN
TRACK  LWPI MYWS
       LI   R0,>3FF5
       LI   R1,DSK
       LI   R2,11
       BLWP @VMBR
       AB   @H30,@DSK
       CLR  R0
       LI   R1,1
       LI   R2,DSK-1
       BLWP @STRASG
       LWPI >83E0
       CLR  @>837C
       B    @>70
       END
```

When you load a program, the DOS saves its drive number at >3FF5 and the filename in the next 10 bytes. By recovering this drive number, you can write your program to open and read files from whatever drive the program may be loaded from.

You can load this routine by a CALL LOAD, but that must be preceded by a CALL INIT which will wipe out the record in >3FF5, so you will need to start the program with this line -

```
100 CALL INIT :: DISPLAY AT(
12,1)ERASE ALL:"DRIVE NO.?"
:: ACCEPT AT(12,12):D$ :: CA
LL LOAD("DSK"&D$&".TRACK/O")
:: CALL LINK("TRACK",A$)
```

Then, when you want to open a file,

```
OPEN #1:"DSK"&SEG$(A$,1,1)&"
."&(the filename)
```

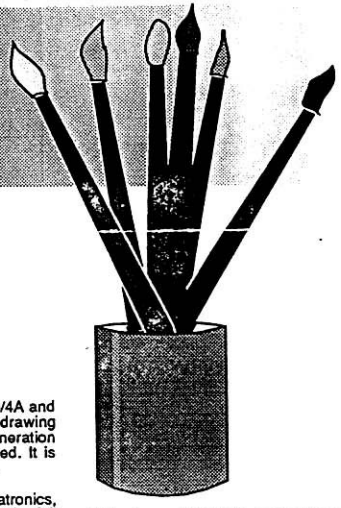
However, if you imbed this routine in your program with Todd Kaplan's invaluable ALSAVE routine, you can omit the LOAD routine!

And to fill the column -

```
100 CALL CLEAR :: FOR J=1 TO
 12 :: CALL COLOR(J,16,2)::
NEXT J
110 CALL SCREEN(2):: DISPLAY
 AT(3,5):"SNOWFALL ON GANYME
DE":;:;:" THE SNOWFLAKES ON
THE THIRD":;:;:"MOON OF JUPI
TER ARE LARGE"
120 DISPLAY AT(12,1):"AND IN
 MANY COLORS, BUT LIKE":;:;:
"THOSE OF EARTH THEY ARE":;:
;:"ALWAYS SYMMETRICAL AND NO
"
130 DISPLAY AT(21,1):"TWO AR
E EVER THE SAME."
140 C=3 :: Y=1 :: CALL MAGNI
FY(3):: RANDOMIZE :: FOR CH=
40 TO 120 STEP 4
150 FOR J=1 TO 8 :: FOR K=1
TO 8 :: Z=INT(12*RND+2):: X=
INT(Z*RND+1):: X$=X$&STR$(AB
S(X=Z)):: Y$=STR$(ABS(X=Z))&
Y$ :: NEXT K
160 CALL BIN_HEX(X$,H$):: A$
=A$&H$ :: B$=H$&B$
170 Y$=SEG$(Y$,2,7)&"0" :: C
ALL BIN_HEX(Y$,H$):: C$=C$&H
$ :: D$=H$&D$ :: X$,Y$="" ::
 NEXT J
180 CALL CHAR(CH,A$&SEG$(B$,
3,14)&"00"&C$&SEG$(D$,3,14))
:: CALL CLEAR :: CALL SPRITE
(#Y,CH,C,150,150,15*RND-15*R
ND,15*RND-15*RND)
190 A$,B$,C$,D$="" :: C=C+1+
(C=16)*14 :: Y=Y+1 :: NEXT C
H
200 GOTO 200
210 SUB BIN_HEX(B$,H$):: HX$
="0123456789ABCDEF" :: BN$="
0000X0001X0010X0011X0100X010
1X0110X0111X1000X1001X1010X1
011X1100X1101X1110X1111"
220 L=LEN(B$):: IF L/4<>INT(
L/4)THEN B$="0"&B$ :: GOTO 2
20
230 FOR J=L-3 TO 1 STEP -4 :
: X$=SEG$(B$,J,4)
240 X=(POS(BN$,X$,1)-1)/5 ::
 T$=SEG$(HX$,X+1,1)&T$ :: NE
XT J :: H$=T$ :: T$="" :: SU
BEND
```

■

Yet Another Paint Program



Y.A.P.P., by Alexander Hulpke, is the most advanced drawing system for the TI-99/4A and the Myarc Geneve 9640. Combining all the popular painting features of other drawing programs, along with the spectacular graphics capabilities provided by the newgeneration of graphics hardware, *Y.A.P.P.* may be the last paint program you will ever need. It is certainly the type of programthe new generation of hardware was designed to run.

Y.A.P.P. is designed to function on a TI-99/4A with an 80-column device (by Mechatronics, Dijit or Asgard), or on a Myarc Geneve 9640. It is the first program designed to take full advantage of the features offered by the 9938 Video processor used in those devices. Furthermore, it is a full-featured drawing program that includes:

- Support for 4 different drawing modes including 256x212 dots with 256 colors, 256x424 with 256 colors, 512x212 with 16 colors, and 512x424 with 16 colors. Unlike a standard TI-99/4A, each dot can be any of the available colors.
- An icon driven interface that works with the *Asgard Mouse*, the *Myarc Mouse*, a *Mechatronics/Dijit* mouse, or a joystick.
- Extensive drawing commands including an airbrush tool, different drawing brushes, lines, boxes, frames, filling, circles/ellipses, etc.
- Built-in support for moving and copying parts of a picture.
- A fast zoom drawing mode (192K video RAM required for some graphics modes).
- Built-in support for *TI-Artist* compatible fonts - type on the screen with ease in your favorite fonts.
- Support for creating and using color clip-art. An "undo" function for erasing mistakes.
- The saving and loading of pictures, including support for *My-Art* picture format, and a built-in mouse/joystick driven disk cataloger.
- Extensive support for 10 different logic functions that work with almost all commands for use in creating special effects.
- A complete printout utility for Epson or compatible printers for printing color pictures in gray-scale on most dot-matrix printers.
- Complete, and superior, support for GIF format pictures.

All features of Y.A.P.P. are included in one convenient to use program that can be run equally well from a TI-99/4A or a Geneve. Y.A.P.P. includes a large collection of original example pictures and fonts, and a manual/tutorial with illustrations. A German language manual is also available on request.

Requires either: (1) a TI-99/4A with an 80-column card, 32K, a disk system, and either an Asgard Mouse, 80-column device mouse or a joystick; or (2) a Myarc Geneve 9640 with a disk system, an Asgard or a Myarc Mouse.

Suggested retail:
$29.95

U.S. add $2.50 S&H
Canada add $3.00 S&H
Airmail add $5.00 S&H

Asgard Software • P.O. Box 10306 • Rockville, MD 20849

Page Pro 99

Page Pro 99 is the remarkable page-making program for the TI-99/4A and Myarc Geneve 9640 computers that is unlike any other such program available. Unlike any page-making program for the 99/4A, or most for the Geneve, the page you see on the screen is exactly as it will appear when printed out.

Page Pro 99 is fast (written in assembly), easy-to-use (with extensive documentation and on-screen help), and powerful (a built-in TI-Writer editor and dozens of features unique to this program). It is compatible with standard TI-Artist fonts and pictures, as well as a growing library of *Page Pro 99* specific artwork (unlike other page programs *Page Pro* can use pictures as large as the page - all others are limited to small pictures). Page Pro will also work on any system from a TI-99/4A with a single disk drive to a Geneve with hard disks. It includes 3 thorough manuals with tutorials. It is very dependable and well-tested (as it should be after over 2 years of development). It is the best supported of any page-making program with dozens of packages of fonts, pictures, pre-made templates for all uses (labels, greeting cards, envelopes, business forms, etc.), borders and utilities. And finally, it produces beautiful pages.

Because it is "what-you-see-is-what-you-get", it is easy-to-use. Because it includes numerous examples, a good collection of starter fonts and pictures, and in-depth tutorials, you can use it straight out of the box. Because it is by Asgard Software, it has the support that it, and you deserve. Who can ask for more?


Page Pro Times

Vol.2 Winter '98

PAGE PRO 99 RELEASED!

ROCKVILLE, MD- Asgard Software recently released its long-awaited update to "Page Pro 99" today to a thunderous standing ovation as members of the press viewed the many new features and capabilities of Version 1.6 of this popular program.

Hundreds of viewers clamoured to be among the first to receive the update to this package. After handing out updates, Asgard surprised participants by breaking out party favors and champagne for a celebration.

PROGRAM FEATURES

- 28 Pictures/Page
- Large font & small font on each page
- 2 Linestyles/page
- Print in single, double or "hi-res"
- Type up/down/left or right
- Import/Export TI-Writer text
- Save/Load Page
- Use TI-Artist, other artwork
- Catalog disks
- Clip area and save as picture
- Much more!

ONLY $24.95
U.S. Add $0.75 S&H
Can. Add $1.25 S&H
Airmail Add $4.00 S&H

After the party, that lasted into the wee hours, and at it's peak featured Asgard President Chris Bobbitt serenading his wife on his saxophone, accompanied by several members of the press using empty bottles as percussion instruments, the whole affair was broken up by local citizens and police outraged at not being invited. Crowds threw floppy disks at the celebrating throng. No arrests were reported but reportedly Mr. Bobbitt was offered a recording contract.

Asgard Software
Box 10306
Rockville, MD 20849

- Paste up to 28 pictures of any size on a page at once
- Type in one large font and one small font per page
- Two different linestyles can be used at once
- Print out pages in single, double sharp "hi-res" mode
- Type in any direction - left, right, up or down
- Catalog a disk at any time while loading files
- Import/Export TI-Writer text - even columns of text
- Clip portions of the page and save as picture - reusable graphic elements!
- Includes utilities for converting fonts as well as creating two column text
- Use TI-Writer keys for editing text, place pictures wherever the cursor is
- Useful for forms, letterheads, greeting cards, business cards, labels, signs, certificates, envelopes, etc.
- Requires 32K, Disk, and Epson or compatible printer

Asgard Software
P.O. Box 10306
Rockville, MD 20849
(703)255-3085

Only $24.95

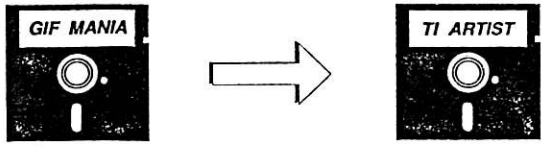
U.S. Add $0.75 S&H
Canadians Add $1.25
Europe/Asia/Australia
please add $4.00 S&H

The hottest utility of the year has arrived!!!
GIF Mania is a "must have" for all graphics enthusiasts.

GIF MANIA

Imagine having the world's largest collection of clip-art and scanned images at your fingertips. Sounds like a dream come true, right? Perhaps. But what if dreams did come true... if they did, and they sometimes do, this dream would be called GIF Mania!

That's right, now all 99ers have the world's largest collection of artwork at their fingertips. For the first time ever, using GIF Mania, industry standard GIF files can be viewed on an ordinary TI-99/4a. In addition, GIF Mania can convert any GIF file into a standard TI Artist file. When in TI Artist format you can do virtually anything can be with the image -- it can be altered, printed, and even be used to create animated movie sequences!



But wait a minute!!! If you don't already know what a GIF image is, you may be scratching your head about now. No need to worry... we'll let you in on this little secret of ours. GIF, which stands for Graphics Interchange Format, is a universal graphics format that was originally developed by CompuServe Information Services. CompuServe (which by the way is the largest provider of on-line information services in the United States) developed this format so that users of all different computers could exchange graphics files regardless of what computer platform (IBM, Apple, Commodore, Atari, or TI) they were using. Well, needless to say, the GIF format took hold and is now a world-wide graphics standard. Over one hundred thousand GIF images exist throughout the world today, many of which are available to you free through on-line services such as CompuServe, Genie and Delphi. In addition, many user group libraries already have GIF files in them.

GIF Mania is not only a true "wonder" program (this is yet another thing the experts said couldn't be done), it's easy-to-use as well. Fully menu driven, GIF Mania puts you in control of the world's largest collection of artwork. **Only $14.95**

A small collection of GIF files is included free with GIF Mania. GIF Mania will operate on the Geneve in GPL mode with the same color limitations of the TI-99/4a.

Starfleet Technical Drawings III

Starfleet III has officially been launched!!! This incredible collection of all new precision drawings features the spacecraft that have appeared in original Star Trek television series, the syndicated Star Trek: The Next Generation television series, and the five Star Trek major motion pictures from Paramount. Drawings included in Starfleet III are: civilian starships from various world members of the Federation (including Earth, Vulcan, Alpha Centauri and Andorian), starships from the Federation adversary worlds (including the Romulans, The Gorn Empire, Orion Pirates and the Tholian Assembly), the latest two versions of the Starship Enterprise as seen on a recent episode of Star Trek: The Next Generation, (1701-B and 1701-C), a chart comparing all five USS Enterprises, numerous recognition charts of Federation, Romulan and Klingon starships shown in scale to each other, and miscellaneous Federation and Klingon starships. In addition to the drawings, each disk of the Starfleet disks includes a small command file that allows the drawings to be viewed in an automated slide show format using Display Master (sold separately). Plus, two extraordinary battle scenes are also included. ***$12.95 for all 4 disks!***

Special Offer!!!
Order all three Starfleet Technical Drawings Volumes for only $27.95!!!

TEXAMENTS

Office: (516)475-3480 53 Center Street, Patchogue, New York 11772 BBS: (516)475-6463

Please add $3.00 for domestic first class (and Canadian) delivery, $8.50 for foreign insured air mail delivery. Orders are usually shipped within a 48 hour period.

Spell It! Updated

Asgard Software is pleased to announce the availability of Version 1.1 of *Spell It!*, by Jim Reiss.

Spell It! is a fast, modern spelling checker for the TI-99/4A and Myarc Geneve 9640 that has received rave reviews from users and reviewers alike for its ease of use, speed, and many features and capabilities.

The most recent version features the following additions and changes over the previous version:

- The capacity for unique words has been expanded from 1500 to 1900 on a standard TI-99/4A. *Spell It!* only checks each word in a text file once - if it's corrected the program will change all instances of that word. Most documents have many words in them but few unique words.
- The program now fully supports alternate CHARA1 files - excellent for use with foreign languages.
- The program now displays a count of the total number of words in a particular text file, as well as the number of unique words.
- *Spell It!* will allow you to correct one word into two or three words. After all corrections have been made to the list of misspelled words Spell It! generates the program creates a copy of the text file checked with all corrections made.
- The "View Context" function, which is used to display a few lines of text in the text file around the misspelled word, now displays TI-Writer control characters (such as the line feed and carriage return symbols).
- The program will now check a text file typically 10-15% faster then before - it has been the fastest spelling checker for the TI-99/4A and the Myarc Geneve 9640 since it was released - up to 15 times faster then all others.
- Compatibility with the HFDC on the TI-99/4A and the Myarc Geneve 9640 is now assured. The program is now more compatible with the Horizon RAM-disk.

Spell It! is compatible with all major word processing programs for the TI-99/4A and the Myarc Geneve 9640. In addition to the features listed above the program will also allow you to lookup unknown words in the dictionary.

Spell It! is available in three versions: a DS/DD disk version with a suggested retail price of $19.95, a SS/SD disk version that retails for $24.95, and the HFDC version, which features over 250,000 words, and retails for $34.95.

If you already own *Spell It!* you can upgrade to the most recent version by returning your original program disk, and a check for $5.00. This price includes shipping and handling. If you'd like to order *Spell It!*, send a check with for the amount listed above of the version desired, plus regular shipping & handling.

Spell It! requires a TI-99/4A with 32K, a disk drive system, and either Extended BASIC, Editor/Assembler or TI-Writer; or a Myarc Geneve 9640 with a disk system.

Send orders to:

Asgard Software
P.O. Box 10306
Rockville, MD 20849

Shipping:
U.S. add $2.00/order
Canada add $2.50/order
Airmail add $4.00/order

TI BASE
Version 3.0

More Features • More Power • More Flexibility

The best just got better... again. TI Base Version 3.0. With its massive file handling capabilities, extensive command programming language, and unmatched information processing facilities, TI Base is clearly the most powerful and flexible database system available for the TI-99/4a.

Overwhelming File Handling

TI Base supports up to five active databases. Each database can consist of 16129 records, with 17 fields per record, and 255 characters per field. Summed up, that's almost 70 megabytes of information! And using the generic conversion facility included, you can convert your present data files, from another database or TI Writer, to be used with TI Base. Now that's power... with almost no limitations!

Extensive Command Language

TI Base employs a database "engine" that is controlled by a procedural command language similar to the one used by Ashton-Tate in dBASE. The language consists of 50 different commands that allow you to access your databases on-the-fly, and create powerful program command files for automatic and complex data processing. You can even produce your own database applications!

Unsurpassed Features and Support

No other database system offers you more features, power, or flexibility, in one single package, than TI Base. In addition, TI Base is the most widely used and supported database system available. Here's a short list of some of the outstanding features found in TI Base:

- Databases can be created, deleted, restructured (without losing data), and appended to one another. Records can be added, edited, deleted, sorted, and searched for in a variety of ways.
- Free interchange of data; numerical, character, date, and local variables may be freely interchanged.
- Complete mathematical functions; arithmetic, logical, trigonometric, and Boolean functions. Numerical fields may be independently summed and averaged.
- Formatted display and printing capabilities; character manipulations, screen scrolling, color changing, and more.
- Structured command language; over 50 different commands, local variable creation, ability to nest cmd files.
- Eight level nested sort capability; sort records on multiple fields. (TI Base is also fully compatible with TI Sort.)
- Database-driven Report Generator permits reports to be created, saved and later recalled when needed. Report headers, footers and bodies may be defined. Local variables may be used from within a report.
- Disk management functions; catalog and format disks, copy and delete files from within TI Base.
- 40 column file editor to create and edit your own command (program) files from within TI Base.
- Global processing of records using simple commands or complex command (program) files.
- Command files may be executed from any device, or installed and run from high-speed VDP memory.
- System setup; allows the definition of disk locations, printer configuration, data stamping, and other misc. functions.

Not only is TI Base powerful, but it is affordable as well. For only $24.95 (plus shipping) you get the TI Base system and tutor disks, keyboard overly, quick reference card, and a comprehensive instruction guide. TI Base requires a disk system, 32K memory expansion, and either an Extended Basic, Editor/Assembler, or Mini Memory cartridge to operate. TI Base has been tested (but is not guaranteed) to be compatible with the Geneve 9640 (in GPL mode), all Myarc and CorComp peripheral expansion cards, and the New Horizon's RAMdisk.

Upgrade to Version 3.0

Previous owners of TI Base may upgrade to Version 3.0 for only $14.95 (plus shipping). Anyone who purchased TI Base after March 31, 1990 is entitled to receive a free Version 3.0 upgrade. *When ordering your upgrade, please include both of your original TI Base disks (the system and tutor disks) along with your upgrade fee. If you are entitled to receive a free upgrade, please include a copy of your dated sales receipt (free upgrades will not be shipped without a valid sales receipt). Please add the correct shipping charges to all upgrade orders (this includes "free" upgrades), otherwise they will be returned freight collect. All upgrade orders must be placed by mail.*

TEXAMENTS

Office: (516)475-3480 53 Center Street, Patchogue, NY 11772 BBS: (516)475-6463

Please add $3.00 for domestic first class (and Canadian) delivery, $8.50 for foreign insured air mail delivery. Orders are usually shipped within a 48 hour period. C.O.D. orders are accepted and must be placed by phone. Sorry, no credit card orders accepted.

GuideLines
For TI Artist Plus!

Banner Borders, Templates, Fonts, Instances and More!

This is the one you asked for. GuideLines. Two disks loaded with new and original graphics artwork for TI Artist PLUS!. Fonts, instances and slides, the standard components of any good TI Artist companion, are all included. What makes GuideLines truly special are the unique banner borders and templates found in this outstanding package.

Although not all too difficult, creating banners with TI Artist PLUS! can be a tedious trial and error process. With banner borders anyone, a novice or expert, can create elegant banners without failure (and all that wasted paper). The 12 different graphic banner styles included with GuideLines will have you producing beautiful banners in no time.

Templates, templates, and more templates. Seven in all. There are templates to assist you in creating fancy calendars, disk sleeves and jackets, standard mailing labels, banner borders, banners and full page drawings. Templates make your work easier and eliminate the guesswork involved in producing these items.

And lets not forget those "standard components"! Included with GuideLines are 15 fonts (with the exception of one, all of which contain upper and lower case letters, numbers, and full punctuation), 17 small instances and two full sets of slides.

GuideLines Complete Only $9.95

TI Artist Plus!

Only $24.95! Existing TI Artist owners can upgrade to TI Artist PLUS! for only $14.95! Just send in your original disk and front page of your manual.

More than just an ordinary drawing package, TI Artist PLUS! is a complete drawing system that consists of six dynamic graphics development modules. With these modules virtually anyone can create, edit, transform, scale, print and present the most dazzling of graphics. And with its innovative point-and-shoot menu system, TI Artist PLUS! is extremely user friendly.

TI Artist PLUS! is also the most compatible program available. It works with almost any printer, including a few color printers. Its backwards compatibie with all of the existing artwork available for the original TI Artist. And its compatible with the Geneve 9640 (in GPL mode), most RAMdisks, and the Myarc HFDC. (Please contact us regarding specific product compatibility).

TEXAMENTS
53 Center Street, Patchogue, New York 11772

Please add $2.50 for domestic first class (and Canadian) delivery, $8.00 for foreign insured air mail delivery. Orders are usually shipped within a 48 hour period. C.O.D. orders are accepted and must be placed by phone. Sorry, no credit card orders accepted.

FROM THE EDITOR

Just when you thought it was all over, here we go again with Supplement #3 of Home Publishing On The 99/4A. I had a number of reasons for wanting to give this series one more shot.

First, and foremost, some parts of TI-Artist Plus needed further documentation over and above what came with the program. If that need hadn't been there, I don't think I would have even thought about this project. TIA+ is one heck of a program, and about as good as we will ever get for 4A graphics. However, if you aren't familiar with the first Artist, you may miss a lot of the possibilities of the new one. Even if you're already a graphics freak, this extra bit of information won't hurt.

Another driving force was the new program by Bob Coffey that is on the included disk of graphics utilities. I had been beating on Bob a long time to finish the Artist Sampler. Now that it's done, I admit to having selfish interest in the program, (*I* wanted a picture catalog that wouldn't take much space) but you folks deserve it too.

Then, you will notice a second name on the cover of this manual, that of Paul Scheidemantle. The name shouldn't be new to you, Paul is about the best all around graphics man this community has. I wanted him to work on something with me and this seemed like the best possible choice. Paul and I split TIA+ in half with him taking the most creative parts. I know that he will come up with more ideas than I ever could to fire your own creative thinking. Paul has also supplied programming to the utilities disk, and I'm sure you will find a lot of uses for his bright ideas.

A fair piece of the manual is taken up with TIA+, but toward the back you will also find some information on other recent graphic programs. PagePro for instance, the newest of the desk-top publishers, is opening up new worlds to the TIer. Although you may already be familiar with it, you may not be aware of the new additions to version 1.5 that we discuss here.

It just never stops coming folks. I doubt that the 4A will live forever, nothing does, but I think we have only arrived at middle age. We have a long, long way to go. -HTB-

SWEDLOW EXTENDED BASIC

```
X   X   BBBB      # 17 to 18
 X X    B   B
  X     BBBB      By
 X X    B   B     Jim
X   X   BBBB      Swedlow
```

(This article originally appeared in the User Group of Orange County, California ROM)

SPEEDING UP XB

I have been looking for ways to speed up Extended Basic programs and have found some interesting things.

Most of the information in this article is based on a FOR NEXT loop. To compare, for example, <PRINT A> with <PRINT A;>, I ran a program like this:

```
10 INPUT A$
20 FOR A=1 TO 1000
30 PRINT A
40 NEXT A
```

Line 10 is <INPUT A$> to make sure that the pre-scan time did not skew the results and to give me a marker to start my stopwatch.

I ran the program three times and then averaged the run times. Next I changed line 30 to:

```
30 PRINT A;
```

Again, I ran the program three times and averaged the run times. Then I did the same thing with DISPLAY AT. The results were:

Line 30	Run Time
PRINT A	105.5 seconds
PRINT A;	57.6 seconds
DISPLAY AT(5,5):A	66.3 Seconds

The loop with no line 30 took about 10.6 seconds. Therefore, PRINT A takes a little less than a tenth of a second to execute.

What follows are some things that I tried and the results. Some conclusions challenge advice I have read and some is new.

REMarks AND ! TAILS

Most of the material I have read suggests that removing REM/! lines and ! tails will improve execution time. It will, but not by very much.

A plain 1 TO 1000 loop took 10.6 seconds. Adding either a REM or ! line INSIDE the loop increased execution time to 11.2 seconds. The same applied to a ! tail.

Removing these will only slightly speed up your program.

INSIDE ARRAYS

I wondered if it took longer to access one member of an array versus another. I DIMentioned an array at 200 and then looked at different members.

I expected to find some relationship between the number and the time (faster to access the beginning or end). What I found was that the run time depended on the size of the number inside the parenthesis:

Array Members		Run Time
B(1)	to B(9)	15.3 seconds
B(10)	to B(99)	15.6 seconds
B(100)	to B(200)	16.0 seconds

Apparently the more digits a number has, the longer it takes XB to read and digest it. Further, it takes about the same amount of time to access any member of an array.

DEFinitions ARE GLACIAL

I knew from experience that DEFinitions were time consuming but I was surprised to find out just how slow they are. I ran a loop with a calculation done thru a DEF and then without the DEF. The DEF loop was 55 seconds slower than the non-DEF loop.

It takes a long time (in computer terms) for your 4A to find a DEF (NOT counting the actual time to

execute the calculation).

Avoid DEFinitions!

ORDER OF VARIABLES

One of the things your TI does during pre-scan is to build a variable table. This is a table of all of the variables in the program and the memory location of each variable's current value.

Each time your program uses a variable, it first searches the table for the variable and then goes to the memory location to find the value.

Our TI's list variables in reverse order. The first variable it finds in a program is at the end of the table and the last variable found is at the beginning.

I created a program with 26 variables and then used one of them inside the loop. Times were:

Variable used	Run Time
First in program	19.0 seconds
Last in program	12.4 seconds

Your TI finds your variables in either the order of use or whatever order you placed them in the pre-scan list. These results suggest that if you change the order so that infrequently used variables appear first and those that are used often appear last, your execution time should decrease.

To test this, I took a program that had the variables listed in alphabetical order. I rearranged them to reverse order of use. The results were impressive:

Variable Order	Run Time
Alphabetical	14.3 minutes
Reverse use	12.5 minutes

This step reduced run time 1.8 minutes or 12% percent. NOT BAD!

LINE NUMBERS

Your TI also uses a line number table. This table is similarly in reverse order (first line last, last line first, etc). When you use a line number in your program (GOTO, etc), your 4A must search the line number table to find the memory location of the line contents. Then it reads the line instructions, crunches them and executes.

I constructed a program that looks something like this:

```
10 INPUT A$
20 FOR I=1 TO 1000
30 ??????
40 NEXT I
50 RETURN
-----------------------
Lines 60 thru 2050
- 200 lines -
are all REM lines
<60 REM>, <70 REM>, etc
-----------------------
2060 RETURN
2070 SUB A :: SUBEND
```

Run times depended on line 30:

Line 30	Run Time
GOSUB 50	26.9 seconds
GOSUB 2060	12.6 seconds
CALL A	15.6 seconds

GOSUB 50 took longer than GOSUB 2060 as the computer had to wade thru 202 lines to find line 50 in the line number table versus one line to find line 2060.

Clearly, in long programs, put your frequently used subroutines and groups of code at the end of your program.

SUMMARY

For other ideas read the September, 1984 issue of Millers Graphics' "The Smart Programmer" and a November, 1983 article in "99'er Home Computer Magazine" by John Dow, "Squeezing the most out of TI Basic".

Experiment. You will find other time savers. If you run some of the programs in this article you will probably get slightly different times because you may be more accurate

with your stopwatch and your programs may differ slightly from mine.

I hope that these ideas will help you write and refine your XB programs.

WHAT IS A NIBBLE, ANYWAY?

This month I am going to try and explain all of the various number words we run across. With luck, after you finish reading this, you will have some understanding of bit, byte, nibble, word, hex, binary and where -31952 really is in memory. With luck.

Computers really think in binary. In this numbering system there are two numbers, 0 and 1 (or, if you are a computer, off and on). While this works for your 4A, binary is cumbersome for humans. For example, in binary 41,576 is 1010001100011100.

Hex, or hexadecimal, has sixteen numbers from zero to F. Here are the first sixteen numbers in binary, decimal and hex:

DECIMAL	HEX	BINARY
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

The next number would be 16 or >10 or b10000 (> means hex and b means binary).

One binary number is a bit. Four bits is a nibble. So, 10 or A or 1010 takes four bits or a nibble to express.

A byte is eight bits or two nibbles. With a bit you can count from zero to one. A nibble gets you from zero to fifteen. The range of byte is:

Base	Low	High
Binary	0	11111111
Hex	0	FF
Decimal	0	255

You have probably noticed the numbers 16 and 255 when using your TI. ASCII character run from 0 to 255. There are sixteen colors (1 to 16, really 0 to 15). A string can be up to 255 characters long. And on and on.

Before tackling the next thing, a word, let's see if we can decode something. Let's take b10100 or >14. To convert either number to decimal, we need a method:

```
>14 is >10 plus >4
>10 is 16 and >4 is 4
16 plus 4 is 20
Hence, >14 is 20

b10100 is b10000 plus b100
b10000 is 16 and b100 is 4
16 plus 4 is 20
b10100 is 20
```

Further than that I cannot go in this space.

A word is sixteen bits or four nibbles or two bytes. The range of a word is:

Base	Low	High
Binary	0	1111111111111111
Hex	0	FFFF
Decimal	0	65,535

But there are no negative numbers. Since we need them, we use something called twos compliment (which is way beyond the scope of this column and this writer). I can tell you, however, the impact;

Hex range	Decimal Range
0-7FFF	0 to 32,767
8000-FFFF	-32,768 to -1

Remember that >8000 is the next number after >7FFF.

Some examples:

```
7FFF is  32,767
8000 is -32,768
FFFF is -1
   0 is  0
```

Confused? So was I until I worked with it for a while. These conversion rules may help:

>>Any number less than or equal to 32,767 requires no conversion.

>>Subtract 65536 from any number over 32,767.

>>Add 65536 to any number less than zero.

This conversion process can be expressed in basic:

```
AD=AD+65536*(AD>32767)
```

If AD is the address, this returns the same number if AD is less than or equal to 32767. If AD is greater than 32767, the test returns true (-1) and a negative 65536 is added to AD. Try it on your computer.

Bottom line time. Suppose you see CALL PEEK(-31952,A,B). Where is -31952? Well, since it is less than zero, we add 65536 and get 33584 or >8330. Now you know!

FANS

I have heard good things about a PEB fan from Paul Johnson's company STATO, Inc. The fan is $15.50 plus $2.50 shipping (or $18) and is supposed to be super quiet.

Warning: it is a bit of a job to take your PEB apart to replace the fan. I have done it (to fix a broken on-off switch) and if you are good with a screwdriver, it can be done. Just do it very slowly and carefully.

Enjoy! ■

FUNNELWEB 4.31 NOV/04/90

by Tony McGovern, Australia

The 40-col DiskReview has been enhanced to match most capabilities of its 80-col big brother, including disk manager and sector editor, and supports single drive file copy.

DM-1000 files MG/MH and Disk-Patch file DP have been dropped as DiskReview fills all essential functions of these. The program files from Vn 4.30 remain compatible with Vn 4.31 if continued use is desired.

Improved support is provided for multi-lingual use of the Editors, as the character file names may be edited in the main program as needed. The 40-col Editor SD now allows selection of another drive directly by number.

A separate Quick Directory file QF now allows file marking from within the Assembler or Formatter.

The 80-col DiskReview now loads Myart RLE image files automatically in <V>iew mode, as well as having improved sector editor and disk manager functions. ■

DANSKT RIKSMÖTE I GRAVENSHOVED

av Kent Edgardh

Jag och några stycken till här i Sverige hade börjat fundera på vad man gör hos våra utländska vänner. Vi har ju läst kommersiella tidskrifter, föreningstidningar som vi prenumererat på från bl.a. USA, Canada, Storbritannien, Tyskland och även Danmark, som tycks ha upphört då ingen tidning anlänt. Därmed skulle man kunna tro att klubbverksamheten i Danmark skulle ha dött ut. Vi har ju själva sett ett kraft-

igt medlemsras hos Programbiten nu några år i följd. Det skulle ju då inte vara helt otroligt att några föreningar skulle somna in för gott. Men så är inte fallet i Danmark, med andra ord "min bortgång och död vill jag härmed förklara är starkt överdriven".

De nästan 6 danska föreningarna är små men kanske desto mer livaktiga. De är TI-Syd i stort sätt begränsat av ett geografiskt område på Jylland söder om en linje Esbjerg - Fredericia med 23 medlemmar. Formand (=ordförande) Leif Pedersen, möten en gång per månad företrädesvis hemma hos formanden. Næstved på Själlands sydkust cirka 6 medlemmar. Möten hålles 2 gånger per månad. Ålborg vid Limfjorden på norra Jylland cirka 5 medlemmar. Århus på Jyllands östkust med cirka 7 medlemmar. Odense på Fyn med cirka 2 medlemmar. København som är litet speciell med sina 4 medlemmar. Där lär aktiviteten vara så låg att de är med i TI-Syd. TI-Syd tar ut en kontingent (=medlemsavgift) för TI 99 på 100 DKK per kvartal dvs 400 DKK per år och 150 DKK per kvartal dvs 600 DKK per år för Geneve från medlemmarna. Kontingenten finansierar programinköp, enkel info samt porto och kuvert m.m.

Första veckoslutet i juli träffas danskarna på Ålborgs kommuns internatskola för ungdomar i Gravenshoved. Med veckoslut menas fredag - lördag - söndag. Eleverna har sommarlov nu och därför kan deras hem lånas. Man meddelar Leif Pedersen på telefon 00946-74579311 hem eller 74566400 på skolan att man vill komma samt var när och hur. Ett begränsat antal sängplatser med vandrarhemsstandard finns. 24 stycken i stora dubbelrum. Man tar alltså med sig sänglinne. Madrass, täcke och kudde lånar man. Kan man ta med sig husvagn/husbil, vilket någon också gjorde är det bara bra. Camping är även det ett övernattningssätt. Området är mycket stort och ligger avskilt från annan bebyggelse. Hotell kan vara ett alternativ för den som vill kosta på sig, men privatrum är extremt dyrt. Mycket dyrare än hotell med frukost.

Gravenshoved 1658 kanske någon känner igen från historieboken, den gamla historien med kungarnas historia. Jag tog tåget tur och retur där Lillebælt är som smalast men på 1600-talet gick tåget i retur här via Brandsø, mest för att skoja med han som satt i Christiansborg på Själland.

Uppackning av datorer sker vart efter man anländer från fredag middag. Hoppackning och städning för att återställa allt som det var före ankomsten beroende på hur lång hemfärden är sker under söndagseftermiddagen. Det var minst 10 datorer i samlingslokalen när jag kom varav 5 var Geneve. Det lär finnas totalt 15 Geneve i Danmark. Ni må tro att den ensamme medlemmen med sin Geneve i Sverige sken upp som en sol. Hans veckoslut blev bara hälften så långt som oss andra svenskars, trots att vi var där samtidigt hela tiden.

En Danmarksresa utan kro-besök är väl knappast en riktig Danmarksresa och då speciellt tillsammans med danskar. Nu kanske ni tror att en gåsmarsch med bilar mitt ute på det öppna landskapet skulle sett lustigt ut, när en till antalet gott och väl hel skolklass skall förflytta sig till närmaste kro(g). Men så blev det inte. Gåsmamman stuvar in sina ungar i skolans buss och kör iväg. Skulle det visa sig vid framkomsten att en unge fattas så kör mamman tillbaka och letar. Maten är dubbelt så god men hälften så dyr om nu ingen hade förstått det.

Förr om årena har här kommit TI-folk från både Tyskland och Holland, men de lyste med sin frånvaro vilket även solen gjorde nästan hela tiden. Datorväder med andra ord. Då gick man 100 meter ner till stranden med jämna mellanrum eller glömde bort det här med datorer helt och hållet.

Nu kanske ni mina vänner undrar vad vi pratade om, visade för varandra, skrev på printrarna och körde på datorerna. Men det hade jag tänkt någon annan skulle få berätta för vi var 6 stycken från Sverige som var där. Jag tror inte det var någon som glömt att preliminärboka 1991.

(Bilder från mötet i Gravenshoved finns på följande sidor)

Bild 1 Fullt snack om datorerna på skandinaviska.

Bild 2 Sven Lundgren, Staffanstorp blev sugen framåt småtimmarna på kvällen, --
eller finns det diskdrivar här?

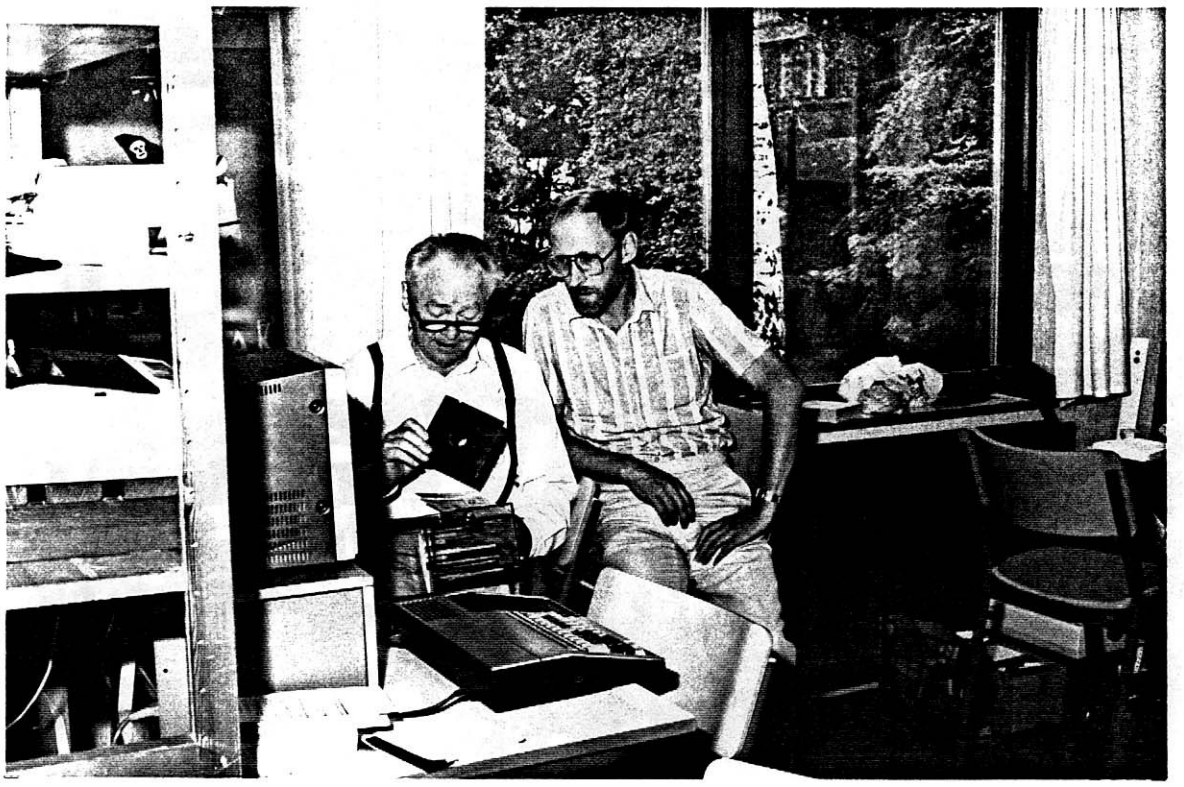


Bild 3 Jens Övnböl, Ålborg demonstrerar för Jörgen Alstrup Hansen, Köbenhavn.

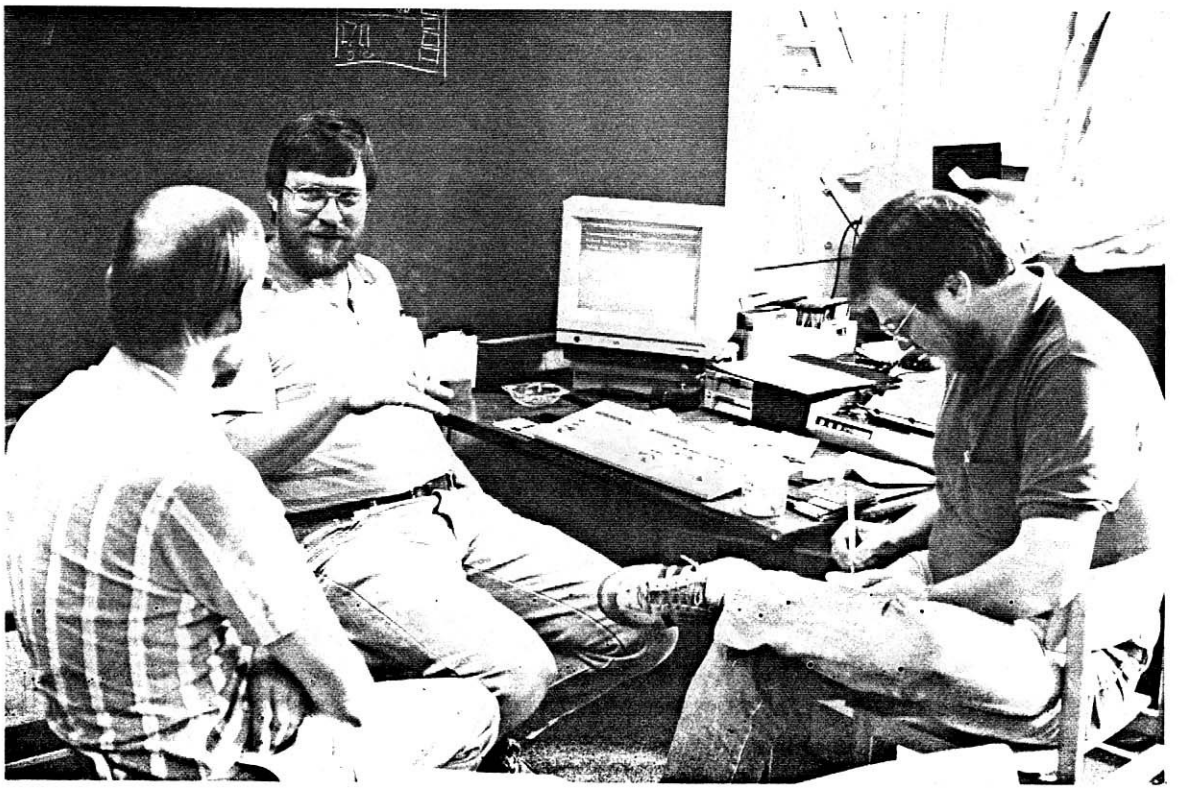


Bild 4 Jörgen Alstrup Hansen, Leif Petersen och Hans Pedersen, Fredericia vid Geneven.