For additional documentation for this package, refer to materials that may be included with your package. If no such materials are included, documentation may be purchased separately.

Mass-Transfer

Mass-Transfer provides more sophisticated terminal emulation, including XMODEM and YMODEM file transfers, with BBSs and on-line information services. This package is fairware, and a donation should be sent to:

Stuart Olson
6625 W. Coolidge Str.
Phoenix, AZ 85033

For additional documentation for this package, refer to materials that may be included with your package. If no such materials are included, documentation may be purchased separately.

Remind-Me!

Finally, Remind Me! is a scheduling program by John Johnson that can be used to manage a schedule or calendar. This easy to use program allows you to enter notes for any day of any year, save those notes to a disk file, recall them later, and print them.

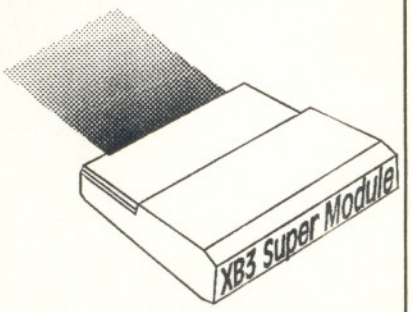
For additional documentation for this package, refer to materials that may be included with your package. If no such materials are included, documentation may be purchased separately.

Copyright © 1993 - Asgard Software
All Rights Reserved

Extended BASIC 3

Cartridge Manual

For the Extended BASIC 3 Super Module



A•S•G•A•R•D S•O•F•T•W•A•R•E

Extended BASIC 3 Super Module

Cartridge Documentation

Introduction

This documentation covers the *Extended BASIC 3 Super Module* - one of the ways that *Extended BASIC 3* is distributed for the TI-99/4A.

This cartridge provides all of the basic programs used by TI users in one, easy-to-use cartridge - with everything accessible from 1 or 2 menus. As such, the *Extended BASIC 3 Super Module* is not just an enhanced version of Extended BASIC - it is an entire productivity environment.

This module requires the following:

At least 32K RAM

A disk system with at least one (1) SS/SD disk drive and a Speech Synthesizer is recommended.

Using the Cartridge

To use the cartridge do the following:

1. With the computer turned off, insert the cartridge into the module port located on the front of the computer
2. Turn on your peripherals - Peripheral Expansion box, etc.
3. Turn on the computer. The "XB-3 Super Module" title screen will appear, and if the Speech Synthesizer™ is plugged in, the computer will say "Ready to Start"
4. Press any key to advance to the Main Menu

The Main Menu

After advancing to the main menu the following options are displayed:

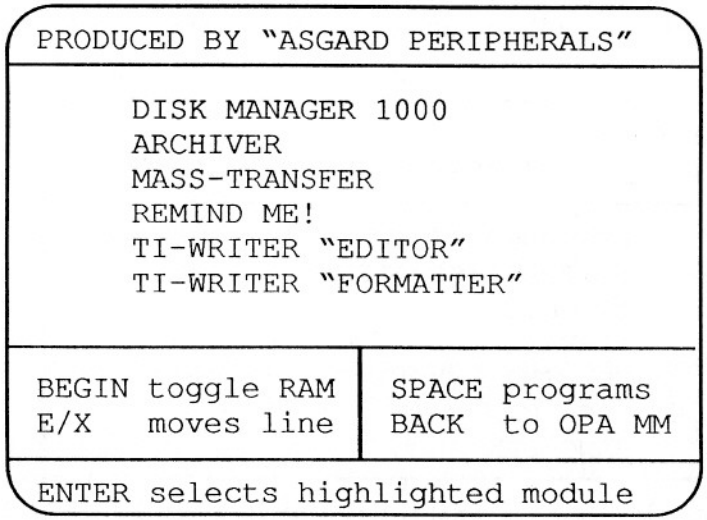
```
XB3/CART
PRODUCED BY "ASGARD PERIPHERALS"

      EXPANDED BASIC
      EDITOR/ASSEMBLER
      TI-WRITER
      TI-BASIC
      TERMINAL EMULATOR II
      HANG-MAN GAME
      HANG-MAN DEMO

BEGIN toggle RAM  | SPACE programs
E/X   moves line  | BACK  to OPA MM

ENTER selects highlighted module
```

If the Speech Synthesizer™ is installed, the computer will also say "module" - meaning you are at the module menu.

To select a module, move the highlight bar up and down with the Up or Down Arrows keys; <E>, <X>, <FCTN><E> or <FCTN><X>. When the module you wish to load is highlighted, press <ENTER>.

Pressing BEGIN key (<FCTN><5>) does nothing but cause the computer to say "module".

Press the <SPACE BAR> causes the following menu to be displayed:



If the Speech Synthesizer™ is inserted, the computer will say "Program" - meaning that this is the "Program Menu". Again, to select a program, highlight it with the highlight bar and press <ENTER>. To return to the "Module Menu" press the <SPACE BAR> again.

Pressing BACK (<FCTN><9>) at either the "Module" or "Program" menus results in the OPA Memory Manager being loaded. This is a special program used to manage the contents of the cartridge.

This program serves as a simple disk manager and program loader. To use it first highlight the device you wish to catalog and press <ENTER>. The catalog of that device will be displayed, and you'll have the opportunity to select a program to run from the list by highlighting it and pressing <ENTER>. Or, you can press BACK (<FCTN><9>) and it will take you back to the device listing. If you select a program to run it will load and run it.

To return to the Module and Program menus, you can do one of two things: either press QUIT (<FCTN><=>) and start over, or press the <SPACE BAR> to move to the file listing and highlight the option "XB3/CART PROGRAM MENU" - press <ENTER>

The Modules

The following is a description of the cartridges in the *Extended BASIC 3 Super Module:*

Extended BASIC 3

Please refer to the *Extended BASIC 3 Supplemental Manual* or the *Extended BASIC 3 Manual* , depending on the version you purchased. The *Supplemental Manual* accompanies cartridges sold with the TI Extended BASIC manual as the main reference. The full *Manual* accompanies cartridges sold with the replacement Extended BASIC reference written specifically for *Extended BASIC 3*.

Editor/Assembler

This is the 1993 version of Editor/Assembler by Winfried Winkler. While it functions much like the standard TI Editor/Assembler, there are a number of differences. Please refer to the appropriate documentation that may accompany your package for supplemental information.

To load the Editor, select 1. The program will prompt you for the number of the "disk drive" where the editor will be loading from. Since the editor is actually located in the cartridge, it really isn't on a disk drive. However, this version of Editor/Assembler will load the cartridge saved version of the Editor if you answer <9>.

To load the Assembler, select 2. Again, you'll be asked for the disk drive number. Again, press <9>.

To select Loaders, select 3. Another menu will then be displayed. Option #1 is used to load a Dis/Fix-80 assembly language file. Option #2 will start a program of that type previously loaded. Option #3 will run a Program-image assembly program. Option #4 reinitializes the cartridge, and Option #5 displays information about any assembly programs that have been loaded. Press <FCTN><9> to return to the E/A menu.

The last option on this menu is #4 - to catalog a disk. Simply enter the number of the disk drive to catalog and press <ENTER>. A catalog will be displayed - and you can press <ENTER> to return to the Editor/Assembler menu. Press BACK (<FCTN><9>) to restart the cartridge.

TI-Writer 5.0

This is an updated version of TI-Writer - in fact, TI-Writer version 5.0 by Art Green. This program is fairware. Any fairware donation should be sent to:

R.A. Green
1032 Chantenary Dr.
Gloucester, Ontario
Canada K1C 2K9

To use this package, refer to the documentation that may be included with your cartridge.

TI-BASIC

This is the standard TI-BASIC with a few enhancements. Refer to the supplemental documentation on Editor/Assembler for more information.

Terminal Emulator II

This is an updated version of the Terminal Emulator II cartridge from TI. It provides simple terminal emulation, as well as the utilities to allow text-to-speech in *Extended BASIC* 3.

When first loaded the cartridge displays a title screen. Press <ENTER> to proceed.

Next, you'll be asked the following:

The Baud Rate of your modem (300 or 1200 baud) - press <1> or <2> and <ENTER>

The Parity (1=Even, 2=Odd and 3=None) - press <1>, <2> or <3> and <ENTER>

The Duplex (1=Full, 2=Half) - press <1> or <2> and <ENTER>

The RS232 port your modem is attached to - <1> or <2> and press <ENTER>

The width of the display screen in columns.

After setting these parameters, you will be taken to a terminal emulation screen, where you can enter commands to dial a modem and connect to a BBS or On-line information service such as Compuserve, GEnie or Delphi (all of which have TI-99/4A Bulletin Boards).

For more information, refer to the included documentation.

Hang-Man Game

Please refer to the documentation within the program for details.

The Programs

DM-1000 6.1

DM-1000 is a general purpose Disk Manager - along the lines of Disk Manager 2. This is a fairware program, and donations should be directed to:

Ottawa TI User Group
3489 Paul Anka Dr.
Ottawa, Ontario
Canada K1V 9K6

For additional documentation for this package, refer to materials that may be included with your package. If no such materials are included, documentation may be purchased separately.

Archiver 3.03

Archiver 3.03 is a program that allows you to combine up to an entire disk worth of files into a single file that takes up less space. It will also allow you to take these "archived" files and reverse the process.

This utility is fairware, and a donation should be sent to:

Barry Boone
P.O. Box 1233
Sand Springs, OK 74061

Disclaimer

Asgard Software provides no warranty, implicit or otherwise, that the programs constituting this package will be free from error, or meet the needs or expectations of the user. Asgard Software provides no warranty beyond that covering the physical components consisting of the program media, which may be returned for a free replacement at any time within 90 days of purchase if defective. After 90 days, this product may be returned for service or replacement (at the option of Asgard Software) for the cost of return postage. This product is warranted in this manner for its lifetime.

Asgard Software reserves the right to refuse to service or replace any product that has been damaged by accident, neglect, unreasonable use, improper service or any other cause not arising out of the quality or defects in materials or craftsmanship. Products damaged in this manner may be replaced at the cost of $20.00 by returning the original cartridge.

Asgard Software is not liable for any damage that may be incurred by the user to the user or the users computer as the result of the use or misuse of this program or its component parts.

Send all defective or damaged software to:

Asgard Software
1423 Flagship Dr.
Woodbridge, VA 22192

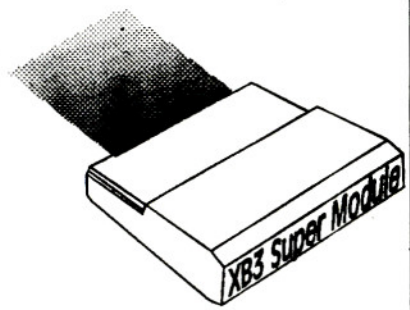
Programs: Copyright © 1993 Winfried Winkler
Documentation: Copyright © 1993 Asgard Software
Hardware: Copyright © 1993 O.P.A.

ALL RIGHTS RESERVED

Extended BASIC 3

Supplemental Manual

For the Extended BASIC 3 Super Module



A•S•G•A•R•D S•O•F•T•W•A•R•E

Extended BASIC 3
Supplemental Manual

INTRODUCTION

This manual assumes that you have previous experience with another version of Extended BASIC for the TI-99/4A Home Computer, and that you own or have access to a complete TI Extended BASIC manual. This manual is an addendum to that manual, and describes the differences between Extended BASIC 3 and TI Extended BASIC. For a full Extended BASIC 3 manual, please review any accompanying literature, or contact Asgard Software.

Additionally, please refer to the "Packing List" specific to your module for an inventory of the specific documentation included with your package.

REQUIREMENTS

Extended BASIC 3 requires:

A TI-99/4A console
32K memory expansion
A cassette recorder

Other peripherals that can be utilized include:

Extended memory cards (AMS, AEMS, etc.)
One or more floppy disk drives
A printer

Because Extended BASIC 3 is distributed in a number of different cartridge formats, please refer to the manual sub-titled "Using Your Cartridge" included with the specific Extended BASIC 3 cartridge you purchased.

CHANGES IN SPECIAL KEY FUNCTIONS

Extended BASIC 3 uses keys differently than TI Extended BASIC. The following is a list of changes (errata) to the description of these key presses as found in Chapter 1 of the TI Extended BASIC manual - in the section "Special Key Functions".

1. All <CTRL> keys are active and "a macro" when pressed. This text may be changed using the INTERNAL command described below with the right version of Extended BASIC 3 - otherwise they will always result in the predefined value being inserted at the command line, while editing a program line, or at a running program's INPUT function.

2. <FCTN><0> is a toggle to make the <CTRL> keys active or inactive. When the <CTRL> keys are inactive, the cursor turns white. When inactive, pressing a <CTRL> key results in the corresponding control character being inserted at the cursor's location - at the command line, while typing/editing a program line, or at a running program's INPUT function.

   The default is <CTRL> keys being active.

   Note: As the INPUT and ACCEPT statements use the same subroutine, pressing <FCTN><0> from within a running program The default for this mode is "inactive" if a program is running.

3. Pressing TAB <FCTN><7) results in the cursor being placed one space behind current end of the line for ap pending input when in the program line Edit mode.

4. Pressing BEGIN <FCTN><5> results in the cursor being placed at the start of the line when in the program line Edit mode.

5. Pressing PROCEED <FCTN><6> or BACK <FCTN><9> in the Edit mode results in the cursor moving up or down a line, if the line being edited is not in the first or last row already.

CHANGES IN COMMANDS

A number of changes have been made in basic command line functions. Please note that these changes are an errata to the "Commands" section of Chapter 2 of the TI Extended BASIC manual.

SAVE

The original "SAVE" command has been modified as follows:

1. The ",MERGE" option is removed — use the OUTPUT command instead (seeabove).

2. All programs saved using the ",PROTECTED" option are changed in a way that any attempt to LIST them will cause all versions of Extended BASIC to crash. The programs can be run, but not edited/listed or saved in MERGE format. This allows you to truly protect your programs from modification or viewing. Please remember to save a copy of the program WITHOUT the protected option for your own use and later modification - because PROTECTED programs will also be unchangeable to you as well.

SIZE

This command now shows the amount of memory available in the low memory area (useful to determine the amount of space available for relocatable assembly routines.)

NEW COMMAND LINE FUNCTIONS

In addition to the changes listed previously, the following is a list of new command line function that serves as an addendum to the same section.

#"Devicename."

Produces a catalog for that device, if it is compatible to TI's specifications. This routine has been tested with the TI, BWG, Myarc, Atronic and Corcomp floppy control lers, the Horizon RAM Disk, and the Myarc HFDC (only without the "DSK1" emulation).

Example:

```
> #"DSK3."
> #"WDS1.PROGRAMS.XB."
```

APPEND

Results in all CTRL-Characters (ASCII 128 and above) being redefined in "inverse Video". This helps when tracking down control characters in listings.

ERASE [Startline] [- Endline]

Erases all all the program lines in the range specified

Example:

```
> ERASE 5-55
```

OUTPUT "Device.Filename" [Startline] [- Endline]

This command saves only the lines specified using "MERGE" format. If no Start and Endlines are given, the entire program is saved.

Example:

```
> OUTPUT "DSK1.MERGEFILE" 5-100
```

PERMANENT ON

The Default mode for Extended BASIC 3 - this results in the TI character set being replaced with Extended BASIC's own character set - which features a true lower-case, and uppercase letters shifted up by 1 pixel. The character data being used can be found at GRAM address @>BCDE, and may be modified by the MOVE statement (see below).

PERMANENT OFF

Activates the standard TI upper/lowercase character set. Use for full compatibility to TI Extended BASIC.

PERMANENT UALPHA

De-activates all lowercase letters permanently. All characters beyond ASCII 95 stay the way they have been redefined by a running program. I.E.: If a program is interrupted using BREAK <FCTN><4>, only the uppercase letters are redefined, all others still look the way they were defined by the interrupted program.

USING

This command lists all the "CALL" statement included in the internal table, if a running program was interrupted. You can use this to get a list of calls for inserting your own Pre-scan commands.

VARIABLE

This command lists all user-DEFined functions and variables on the screen, if a running program was interrupted. This is also useful when inserting Pre-scan commands, and also location DEF function and variable name conflicts, locating the number of array dimensions declared, etc.

CHANGES IN ASSIGNMENT AND INPUT STATEMENTS

The following is an addition to the description for a current Extended BASIC command:

ACCEPT AT...

The "VALIDATE" Option now supports an additional entry: LALPHA This option works the same the same as UALPHA, except for restricting input to lower case letters instead.

Example:

```
10 ACCEPT AT(1,1) VALIDATE(LALPHA)
BEEP:NAME$
```

CHANGES IN EXTENDED BASIC FUNCTIONS

The following TI Extended BASIC functions operate differently when used in Extended BASIC 3:

ASC(character-string)

This function now is able to handle empty strings: ASC("")= -1 As "-1" can never be a valid ASCII code. Now no additional test is needed in an "if asc(x$) then ..." loop to prevent an error occuring if x$ is empty (for example: if browsing through text files).

Example:

```
10 INPUT F$
20 PRINT ASC(F$)
```

WARNING:

This may result in errors occuring "in another place" if the value of ASC(), being -1, is used by another function. These errors may cause confusion if one does overlook the ASC() function being the (possible) cause ...

CLOSE #ALL

This will close all open files at once.

Example:

```
10 CLOSE #ALL
```

DEF

All user-DEFined functions now may be used outside a running program. This allows you to interrupt a running program one to test the value produced by a defined variable/function at the command

WARNING:

The DEF statement itself still may be used only inside a program

VAL(> "Hexadecimal-string")

This function (Note the additional " > ") converts a hexa decimal string of up to four characters length into the corresponding numerical value.

Example:

```
> PRINT VAL(>"FFFC");VAL(>"11")
> -4 33
```


CHANGES IN PROGRAM CONTROL STATEMENTS

The following routines, used to call subroutines, have been modified:

CALL GOSUB (line-number)

Functions the same as in TI Extended BASIC, but "line-number" can be a non-zero variable.

Example:

```
10 X=100
20 CALL GOSUB(X)
30 STOP
100 PRINT "HELLO"
110 RETURN
```

CALL GOTO (line-number)

Functions the same as in TI Extended BASIC, but "line-number" can be a non-zero variable.

Example:

```
10 X=100
20 CALL GOTO(X)
100 PRINT "HELLO"
110 GOTO 10
```

NEW FUNCTIONS

The following is an addendum to the list of new functions available for use in your Extended BASIC 3 programs:

CALL BYE

Now legal within a running program.

Example:

```
10 CALL BYE
```

DATE$, TIME$

These two are new "predefined strings" like the already known number "PI". Extended BASIC 3 constantly updates these functions with the current date and time, as derived from the following clock devices:

Corcomp TripleTech Card
BWG DiskController with Clock
Austrian HardwareClock Card (with compatible DSR)

If no clock card is detected, the content of these strings is set to a warning message.

Example:

```
10 REM: TI XB
20 OPEN#1:"CLOCK",INPUT
30 INPUT#1:TIME$,DATE$
40 CLOSE#1
50 PRINT TIME$,DATE$
```

```
10 REM: XB III
20 PRINT TIME$,DATE$
```

HEX$(Number,Length)

This function supplies a hexadecimal number of LENGTH digits, corresponding to the NUMBER given (as decimal number, variable or numeric array field) valid ranges for the "Number" parameter are from -32768 to +32767, and for the "Length" parameter of from 1 to 4.

Example:

```
> PRINT HEX$(31,3);" ";HEX$(-1,4)
> 01F FFFF
```

CALL NEW

Now legal within running program.

Example:

```
10 CALL NEW
```


CHANGES IN SUBROUTINE CALLS

The following is a list of changes to the way some subroutine CALLs found in TI Extended BASIC function in Extended BASIC 3.

CALL CHAR(character_code,"hex_definition")

The string definition used the strings used may now be longer than 64 characters (you can define more than 4 characters at a time).

Example:

```
10 CALL CHAR(64,"FFFFFFFFFFF
FFFFF00000000000000EEEEEEE
EEEEEEEEE1111111111111111DDD
DDDDDDDDDDDD222222222222222
2")
```

CALL COLOR(character_set,foreground,background)

You can use ALL as an option to define the color of all character sets at once.

Example:

```
10 CALL COLOR(ALL,2,15)
```

CALL INIT

Now loads a SHORTER version of the known subroutines starting at Low-memory location >2464 (not >24F4 as in TI Extended BASIC). All workspaces and BLWP-vector pointers, as well as the four well known VDP "BL" routines (at >23CA , >23E6, >2406 and >241A) remain at the same addresses for maximum compatibility.

WARNING:

The ROM Routines (for example CIF) have been moved. Use XMLLNK to call these routines - direct addressing is not allowed.

CALL PEEK(...)

Editor/Assembler like syntax is now supported (I.E. empty string ("") denotes a new address).

CALL VERSION(X)

The value of X returned will be 150 if in command-mode (not in running program).

CALL LOAD("access-name" [,address,byte1 [, ...], file-field, ...])

This subroutine call, used to load and link assembly routines, is the same speed as the CALL LOAD command in TI Extended BASIC. However, it now supports the compressed object code format with references, and the END/START auto-start feature of the Editor/Assembler loader.

The following REFerences are pre-defined and available for use in your assembly routines:

STRREF	STRASG	NUMREF
NUMASG	VSBR	VSBW
VMBR	VMBW	VWTR
ERR	KSCAN	XMLLNK(see below)
CFI	CIF	CSN
CNS	FADD	SADD
FSUB	SSUB	FMUL
SMUL	FDIV	SDIV
FCOMP	SCOMP	VPUSH
VPOP	ASSGNV	SCROLL
PAD	FAC	ARG
GPLWS	VDPWD	VDPWA
VDPRD	VDPSTA	GRMRD
GRMRA	GRMWD	GRMWA
SOUND	SPCHRD	SPCHWT
VSWR	VSWW	

Please note that all REFerenced symbols above are explained (andlisted) in the Editor/Assembler Manual. Look there for detailed data. The only exceptions are VSWR and VSWW. These are both "WORD" (not

BYTE) derivatives of VSBR and VSBW, but are called by "BL" (not BLWP) and use R3/R4 (not R0).

Additional notes:

NOTE 1: All REFerences are FIRST searched for in the Low-Memory DEF Table. If not present there, the pre-defined values are used. Thisallows for any symbol listed above to be replaced by user-defined ones (by loading the appropriate symbol and program), and incidently assures compatibility with existing object code libraries.

NOTE 2: CALL LOAD now automatically issues a CALL INIT if it was not done in the program.

NOTE 3: TI Extended BASIC's "XMLLNK" routine differs from the routines of SAME NAME used in the Editor/Assembler and Mini-Memory cartridges. Therefore, if you are loading E/A or MiniMem subroutines - first load their XMLLNK routine (compatible one from disk).

Included on the disk accompanying this package you will find special Extended BASIC 3 versions of the not yet included DSRLNK and GPLLNK routines, too.

Special WARNING:

Some programs utilize the pointers to the FIRST and LAST FREE LOW MEMORY ADDRESS (move the assembler's loading space to high-memory or such). Because REFerences are now supported, in the worst case Extended BASIC 3 will have to search through 56K of memory for REFs corresponding to the DEFs just loaded (If the Last/First pointers are set, non-standard or bad)

It will seem that the computer has "hung up", but it's just doing this very time-consuming search.

NEW SUBROUTINE CALLS

In addition to the changes above, the following is a list of new subroutine calls added to Extended BASIC 3:

CALL ALL(character-code)

Fills the entire screen with the ASCII code specified.

Example:

```
10 CALL ALL(65)
```

CALL ALLSET

Like CALL CHARSET, but includes lowercase TI characters. See the section above on PERMANENT OFF

Example:

```
10 CALL ALLSET
```

CALL ALOCK(value)

The number "value" is set to one if the Alpha-Lock is depressed, and zero if it is not.

Example:

```
10 CALL ALOCK(N)
20 PRINT N
```

CALL BEEP

Issues the standard "accept-beep" sound

CALL CHAR ALL

Like the CALL CHARSET command, but includes lowercase Extended BASIC 3 characters. See the section above on PERMANENT ON.

Example:

```
10 CALL CHAR ALL
```


CALL CHIMES

Issues the "bell sound" found in many other programs.

Example:

```
10 CALL CHIMES
```

CALL CLRS

Like CALL CLEAR, but clears only the part of the screen usually used for text (columns 3-28). Used for keeping a border.

CALL FIND(string-1,string-array2(),index)

Sets "index" to the first occurance of string "string-1" within the one dimensional string array "string-array2()". This functions automatically detects the current OPTION BASE setting.

Example:

```
10 DIM B$(100)
20 CALL FIND("STRING",B$(),C)
30 PRINT "Array element cont
aining STRING: ";C
```

CALL GPEEK(address,value)

Same as PEEK, but for use with GRAM/GROM.

CALL GPOKE(address,value)

Same as CALL LOAD with bytes, for use with GRAM.

CALL HONK

Issues the standard "error-honk" sound.

Example:

```
10 CALL HONK
```

CALL KEYS(acceptable-characters,key-pressed)

Waits for a key listed in the string "acceptable characters" to be pressed and then the position of the key within that string in the variable "key pressed".

Example:

```
10 OK$="ABCDEFGHIJKLMNOPQRST
UVWXYZ"
20 CALL KEYS(OK$,K)
30 PRINT K
```

CALL MLOAD(filename [,address])

Loads a program file (memory-image) saved with MSAVE either to the address the MSAVEd file was saved from, or to the address specified by the optional "address" parameter.

WARNING: There must be enough free VDP RAM space to load the file.

Example:

```
10 CALL MSAVE("DSK1.SCREEN",0,768)
20 CALL INIT
30 CALL MLOAD("DSK1.SCREEN")
```

CALL MOTION GO/STOP

Simultaneously starts or stops all the sprites on the screen. This is useful to allow you to set all sprites motions prior to starting them - so that no motion occures until desired.

Example:

```
10 CALL MOTION STOP
20 CALL SPRITE(#1,65,2,100,100,-
10,-10)
30 CALL SPRITE(#2,66,2,100,100,
10,10)
40 CALL MOTION GO
```


CALL MOVE(type-move,source,destination,number-bytes)

This function moves "number-bytes" number of bytes from "source-address" to "destination-address", depending on the type of move "type-move". The parameters possible are as follows:

Type Move	Description
1	from VDP RAM to VDP RAM
2	from VDP RAM to CPU RAM
3	from CPU RAM to VDP RAM
4	from CPU RAM to CPU RAM
5	from GROM/GRAM to CPU RAM
6	from GROM/GRAM TO VDP RAM
7	from CPU RAM to GROM/GRAM
8	from VDP RAM to GROM/GRAM
9	from GROM/GRAM to GROM/GRAM

Example:

```
10 CALL MOVE(1,0,4096,768)
```

WARNING:

There is no test for overlapping source and destination addresses.

CALL MSAVE(filename,starting-address,number-bytes)

Saves "number-bytes" starting at address "starting-address" into filename "filename". If the "starting-address" is greater than >2000 (decimal 8192), CPU RAM is saved. If the "starting-address" is less than >2000 (decimal 8192), VDP RAM is saved.

A number of useful addresses:

>000 to >2FF (0-767)	- Contents of the screen
>300 to >36F (768-879)	- Sprite positions
>3F0 to >77F (1008-1919)	- Character definitions
>780 to >7EF (1920-2031)	- Sprite motion datas
>370 to >3EF (880-1007)	- "Internal" XB data areas
>81F to ??? (2079-???)	- NEVER CHANGE THESE
>2000 to >3FFF (8192-16483)	- All relocatable routines

Example:

```
10 CALL MSAVE("DSK1.SCREEN",0,768)
```

CALL PRNTPAT(number-char,character-data-string)

Similar to CALL CHARPAT, but the resulting "character-data-string" contains the character data for character "number-character" appropriate for printing on an Epson or compatible printer.

Example:

```
10 CALL PRNTPAT(65,A$)
20 PRINT A$
```

Please note:

You will need to supply codes specific to your printer to print the results of this SUBROUTINE.

CALL QUIT OFF/ON

Used to turn ON and OFF the QUIT (<FCTN><=>) key.

Example:

```
10 CALL QUIT OFF
20 PRINT "YOU CAN'T QUIT"
30 CALL KEY(0,K,S) :: IF S=0 THEN 30
40 CALL QUIT ON
50 PRINT "NOW YOU CAN"
```

CALL RESTORE(line-number)

Same as the RESTORE statement, but for use with numerical variables. Note that the RESEQUENCE command can cause trouble with this

CALL RND(variable)

Sets "variable" to a random number in the range of 0 to 99.99 (uses only 2 places after the decimal point). This function is very fast, but not as "random" as numbers generated with the RND function.

Example:

```
10 CALL RND(X)
```


CALL SCREEN OFF/ON

This function is useful when placing characters on a complicated screen. You can use it to turn on and off the screen from within your program. When the screen is "off", only the background color is displayed.

Example:

```
10 CALL SCREEN OFF
20 CALL CLEAR
30 CALL HCHAR(10,20,65,10)
40 CALL VCHAR(3,3,66,20)
50 CALL SCREEN ON
```

CALL VPEEK(address,value)

This statement is the same as the CALL PEEK command, except it is for use with VDP RAM.

Example:

```
10 CALL VPEEK(0,A)
20 PRINT A
```

CALL VPOKE(address,value)

This statement is the same as the CALL LOAD command, except it is for use with VDP RAM.

Example:

```
10 A=10
20 CALL VPOKE(0,A)
```

CALL WAIT(time)

This statement will cause the computer to pause for the specified period of time, or until a key is pressed. The variable "time" can be any value between -16383 to +16383 - in 1/50's of a second. If the variable is negative, the statement causes a "beep" sound to be made first before the pause begins.

Example:

```
10 CALL WAIT(500)
```

Appendix 1: Current Known Problems

While this version of Extended BASIC 3 is to the best of our knowledge bug free, but, because some parts of the language are used in only very strange conditions, and since every possible situation has not been tested in the two-years that this product has been in testing, we cannot guarrantee that all errors have been removed. However, any errors discovered will be removed if reported in writing to Asgard Software.

Appendix 2: Compatibility with TI Extended BASIC

Because compatibility with prior versions of Extended BASIC was a very high priority, the vast majority of programs written for TI Extended BASIC should function fine. However, any major revision in the language is bound to introduce changes in what constitutes acceptable parameters. The differences are as follows:

1. The number of "reserved words" (see the TI Extended BASIC manual) has been increased. However, this version is able to run even those programs containing the new additional "reserved words" used as variable names, as long as all lines which contain one of those variables are NOT edited. The new reserved words in Extended BASIC 3 are:

   DATE$, HEX$, LALPHA, LWRC$, OFF, TIME$, UPRC$

2. Some range/type-checks are now more restrictive (compared to TI Extended BASIC). Among those are:

   CALL LOAD with values bigger than 255
   (supposed to take BYTES as arguments anyway - "WORD" values no longer supported)
   EOF with file numbers being zero or bigger than 255 (see above)

3. In order to gain speed, Extended BASIC 3 now uses (whenever possible) several ROM-routines already included in TI Extended BASIC for type-checking and type-conversion. These routines are written in assembler and thus faster than the interpreted GPL code replaced bythem. As a result, different error messages will be generated by these routines. For example, EOF(0) now issues "bad value" instead of "file error"

4. Some programs, especially games, depend on the speed of TI Extended BASIC, and may not run properly. For example, if in a game a CALL COINC loop is executedonly a certain number of times to check for a coincidence, in Extended BASIC 3 the sprite may have no chance to "hit" the target before the loop is finished because the loop is executed faster, but the speed of the sprite remains the same. The sprite would be several pixels from the "target" when the end of the coincidence loop is reached.

5. Finally, some programs use undocumented tricks for speedup that are no longer possible. The most common example of this would be the CALL PEEK used to generate a RANDOM number between 0 and 99 found in a number of Extended BASIC programs. That special CALL PEEK now generaters numbers between 0 and 36 or so. The number of such incompatibilities is unknown at this time - but usually if a program doesn't work properly, it is because of one of them.

Appendix 3: TI BASIC Compatibility and Graphics

All character graphics oriented subroutines in Extended BASIC 3 (CALL VCHAR, CALL HCHAR, CALL CHAR, etc.) will now accept character codes from 143 to 159 - a range previously available only to TI BASIC programs. This will allow you to load TI BASIC programs and run them as they normally run in TI BASIC, without "bad value" errors, and a lot faster.

However, characters above 143 share the same memory area in the VDP RAM as the table used by Extended BASIC 3 to store sprite motion data. However, because TI BASIC does not use sprites, and normal TI Extended BASIC programs do not use characters above 143 - programs written for either will not try to use the same area for two different things (sprite motion data and the extra character sets).

However, programs written for Extended BASIC 3 can do both - use characters up to 159 and moving sprites at the same time. In other words, that part of memory can be used for two things at the same time. The consequence, however, is that moving sprites will destroy character definitions above 143 - and assigning character definitions to character above 143 may cause previously defined sprites to move off in unusual directions.
However, this also presents an opportunity for the programmer that likes to

experiment. It is possible to now assign sprite motion with the faster CALL CHAR statement - as well as extract and store the current sprite motion setting(s) with the CALL CHARPAT statement.

Appendix 4: Speed

While Extended BASIC 3 is on the average faster than TI Extended BASIC, there is no way to determine the difference in speed for a "typical program". Why? Because while some functions execute much faster (as much as 2-3 times), others are only barely faster and a few are even a little bit slower. Therefore, the speed a program increases will depend on the type of functions used by the program.

Many of the speed enhancements found in Extended BASIC 3 come from the more efficient use of memory. An example of this is the process of loading a program from disk. To load a program, TI Extended BASIC:

1. Kills everything in VRAM no longer needed
2. Tries to load the program file at the first free VRAM address
3. Move to the end of VRAM and converts pointers according to total VRAM available
4. Checks for the existance of the memory expansion
5. Copies the program to the start of memory expansion
6. Moves to the end of high-memory and convert pointers according to memory available

OR:

7. If the first loading attempt failed, it tries to load the program as data file (Int/Fix-254 format) at the start of memory expansion
8. And moves to the end of highmemory, convert pointers, etc.

Extended BASIC 3 was optimized to access the memory expansion directly, so the above procedure is reduced from 6 steps to 3 steps.
Other areas which are noticably faster due to better memory use include:

1. String functions
2. The "garbage collection" routine that retrieves memory for use and discards unneeded code and data as necessary
3. Loading, saving and deleting lines of code at the command line

Other functions are faster because they were re-written from scratch (including all floating point math functions and random number generation), or because the GPL and ROM-based subroutines that they use have themselves been optimized (most sprite routines and graphic CALLs).

Documentation:

Copyright 1993 - Asgard Software &
Winfried Winkler

Programs:

Copyright 1993 - Winfried Winkler,
O.P.A. Other copyrights
where applicable

All Rights Reserved