X C O M

Extended Basic Compiler Version 1.0

User's Guide

Addendum for XCOM Version 1.1

XCOM version 1.1 offers several enhancements over its predecessor which are described in the following paragraphs. Unless specifically stated, the functional description of the software as described in the version 1.0 User's Guide remains unchanged.

Version 1.1 now offers compatibility with the Editor/Assembler module. Loading of the compiler and compiled routines remains essentially the same. However, since the Editor/Assembler module uses a portion of low memory and does not offer supplemental RAM, the size of compiled programs that can be loaded and executed is somewhat smaller. The maximum compiled image size is about 16,300 bytes for the Editor/Assembler compared to about 24,500 for Mini Memory. XCOM now displays the number of bytes of generated code upon completion. This not only allows the programmer to determine if a particular program will load under Editor/Assembler, but will also allow experimentation with compiler options to determine their effect on program size.

Compiled code generated under either Editor/Assembler or Mini Memory may be loaded and executed with either module with the restriction discussed above.

On completion, compiled programs can be easily restarted by pressing ENTER at the prompt. Exiting to the title screen is accomplished by pressing QUIT.

The RUN statement is now fully supported. This means that other compiled programs may be loaded and run from your program. The string argument of the RUN statement is not limited to constants as it is in Extended Basic. RUN may also be used to load and execute standalone assembly language routines, but return of control to the compiled program is not possible.

A bug in the TAB function has been corrected.

Compilation may now be aborted by pressing CLEAR. This closes all open files and deletes the work file.

The VERSION subprogram now return a value of 110.

PURCHASE AGREEMENT

DKM Enterprises does not warrant that this program or the accompanying user's guide will be free from error or meet the specialized requirements of the user. The user assumes sole responsibility for any actions or decisions rendered on the basis of information obtained through the use of this program.

This software is copyrighted. Reproduction for the purpose of distributing to others is strictly prohibited. This software is copy protected and attempts to modify this copy protection may render the program nonfunctional.

Program disks which are or become defective and judged by DKM Enterprises not to be the result of deliberate tampering with the intent to defeat copy protection will be replaced free of charge with return of the original program disk.

Future releases of this software will be made available to legitimate users at a discount purchase price of $10 with the return of the original program disk.

XCOM -- Extended Basic Compiler

User's Guide

==========

Note: TI Basic and TI Extended Basic are registered trademarks of Texas Instruments Incorporated.

==========

Functional description

XCOM is a full-featured compiler for TI Basic and TI Extended Basic programs. It transforms programs saved in MERGE format to machine language code which may then be executed at many times the speed of the original program. XCOM supports nearly all Extended Basic statements. Several additional capabilities have been added to enhance the versatility and power of the Extended Basic language. Full integer arithmetic and true lower case character set are among these enhancements.

Minimum system requirements

XCOM requires Mini Memory module (to load and execute compiled programs), 32K memory expansion, and one disk drive.

Preparing programs for compilation

Before any program can be compiled, it must first be saved in MERGE file format. This is easily accomplished in Extended Basic by using the MERGE option on the SAVE command. For example,

```
SAVE DSK1.XYZ,MERGE
```

Since TI Basic has no such option for the SAVE command, a different approach has to be taken if you do not have the TI Extended Basic module to save your programs. An assembly language program has been provided on your XCOM disk to create MERGE format files. To use this load the merge utility in TI Basic as follows (assuming the XCOM disk is in drive #1):

```
CALL LOAD("DSK1.MERGE")
```

To save your TI Basic program in MERGE format, OLD your program into memory and call the merge utility. For example:

```
OLD DSK1.XYZ
CALL LINK("MERGE","DSK1.XYZMRG")
```

This loads a TI Basic program called XYZ and saves it on disk as a MERGE format file called XYZMRG. Note that the merge utility uses the screen RAM as a buffer area and that nonsensical characters will appear in the top few lines of the screen. This is a normal part of the operation of the merge utility.

Operation

XCOM is loaded from disk using option 1 of the Mini Memory menu. Assuming the XCOM disk is in drive 1, enter "DSK1.XCOM" for the filename. Once XCOM is loaded, the disk may be removed from the drive. The compiler will prompt for a source filename. This is the Basic program stored in MERGE format (see above discussion). The compiler then prompts for an object filename. This is the file that will contain the compiled code for later execution. Finally XCOM prompts for special options. Available options are discussed in the next section. XCOM will then proceed to compile the program, generating any error messages as it does.

Note: XCOM opens a work file on DSK1 which it deletes on completion. Therefore, a write-enabled disk must reside in this drive during compilation.

Compilation options

The following options are available. To invoke any option, enter its corresponding letter when XCOM prompts for options.

F — Forces all numeric constants to be stored as floating point. By default, all numeric constants within the range of +/- 32767 and not containing a decimal point or exponent are stored as integer.

I — Sets the default datatype for variables to integer. By default, any variable name that does not contain a datatype suffix ($ or @), is assumed to be floating point unless this option is active.

L — Suppresses generation of line number code. This results in a program that uses less memory and executes somewhat faster, but no line numbers will be available for error reporting or trapping.

Running compiled programs

Once compiled, programs may be executed as with any machine language program by loading the program under option 1 of the Mini Memory menu. However, before loading a program, a subroutine program must first be loaded from the XCOM disk. This program contains support routines for the XCOM Basic programs. To do this, load the program LOAD under option 1 of the Mini Memory menu. Once loaded, any number of compiled programs may be loaded and executed without the need to reload the subroutine program. However, anytime a program other than a compiled Basic program is loaded, the subroutine program must be reloaded prior to executing any additional compiled programs.

Once a compiled program is loaded, execution begins immediately. To terminate a program, enter FCTN-4 (CLEAR) just as you would in Extended Basic. Entering FCTN-= (QUIT) also exits the program, but if device I/O is in progress, data may be lost.

When a program terminates either by controlled exit, by entering FCTN-4, or following an untrapped error, a message is displayed and the program will pause until FCTN-= is entered to return to the title screen. To rerun the program, select option 2 of the Mini Memory menu and enter a blank line. Programs may be rerun in this fashion repeatedly without the need to reload.

Integer arithmetic

One of the shortcomings of TI Extended Basic was the lack of support for integer variables. Integer variables use less memory (2 bytes compared to 8 bytes for a floating-point variable) and provide much faster program execution under most circumstances. XCOM provides full integer arithmetic support. Because of this, some differences exist in the evaluation of arithmetic expressions which deserve mention.

Integer variables may be declared in one of three ways. By adding an "@" character to the end of a variable name (e.g., X@), XCOM identifies the variable as an integer. This also holds true for arrays and functions. A second method exists for declaring scalar variables (i.e., simple variables, not arrays or functions) as integer. This is the INT statement. When the compiler encounters this statement, no code is generated. Instead, all variables in the list that follows are declared as integer. Thus, the statement:

```
INT X,Y,ZZZ
```

declares the variables X,Y, and ZZZ as integer. The INT statement must proceed any references to the variables in the list that follows.

Note: Because the "@" character has special meaning in XCOM Basic, it can no longer be imbedded in a variable name. Thus, a name such as X@YZ is not a valid variable name.

Finally, variables without datatype suffixes ($ and @) may be globally declared as integer by using the I option described above. Note that when this option is active, a variable with no datatype suffix, though integer, is treated as a different variable from one with the same name but with an explicit integer datatype. Thus, variable XYZ and XYZ@ are both integer variables when the I option is active, but are not the same variable.

Unless the F option is specified, XCOM interprets any constants containing no decimal point or exponent and within the range of +/- 32767 (the maximum signed values that can be stored in 2 bytes), as integer. This has implications in the evaluation of arithmetic expressions. Thus, the constants

```
2,-32767,528
```

are all stored as integer, while the constants

```
2.,1.6,3E5,32769
```

are all stored as floating point.

XCOM evaluates arithmetic expressions using the same operator precedences as Extended Basic. Arithmetic operations between two integers are performed in integer arithmetic, generating an integer result. This is true in the intermediate stages of the evaluation of more complex expressions as well. Thus,

```
PRINT I@/2
```

where I@ has a value of 3, produces the result of 1. Arithmetic operations involving at least one floating point number are performed in floating point arithmetic. Thus,

```
PRINT I@/2.
```

produces a result of 1.5 when I@=3.

Note that while the expression

```
1/2
```

generates a result of 0.5 in Extended Basic, the same expression in compiled Basic produces a result of 0. Care must be taken to identify these instances as they will likely have an impact on program execution. Appending a decimal point to a constant will force the compiler to store it as a floating-point number.

Description of statements

It is assumed that the reader has a working knowledge of TI Extended Basic. The purpose of the following discussion is to point out differences between TI Extended Basic and XCOM Basic. Only those statements with differences in syntax or operation are listed below. All other statements function exactly as they do in Extended Basic.

ACCEPT statement:

Functions the same with certain enhancements:

SIZE can be up to a length of 255 characters. FCTN-6 (PROC'D) moves the cursor to the end of the input area. FCTN-5 (BEGIN) moves the cursor to the beginning of the input area. FCTN-E (up-arrow) and FCTN-X (down-arrow) no longer function the same as ENTER. They now move the cursor in the corresponding direction within the input area. FCTN-2 (INS) now toggles in and out of insert mode. The status of the insert mode is reflected in the shape of the cursor (full cursor= overwrite, half cursor=insert).

BREAK statement:

Not supported.

CALL statement:

Fully supported. Integer and floating-point arguments may be used interchangeably with the single exception of arrays passed by reference. In this instance, the datatype of the array in the CALL must exactly match that defined in the corresponding SUB statement.

CHARPAT subprogram:

Functions the same, except that since XCOM Basic provides true lower case characters, the corresponding codes returned for these will differ from Extended Basic.

DEF statement:

Same as Extended Basic except that up to 7 arguments may be specified.

DISPLAY statement:

Same except that SIZE may be up to 255. As with Extended Basic, XCOM Basic clears the display area if the SIZE specification is a positive number. However, XCOM Basic does this before displaying the print list, while Extended Basic does this after displaying the print list. This may create minor differences in the format of the displayed text.

ERR subprogram:

Functions the same as Extended Basic. Although XCOM Basic provides more detailed error messages for I/O errors, the error codes returned remain the same.

FILES subprogram:

Functions similar to Extended Basic. Sets the maximum number of disk files which can be open at once (range, 1-9). Can be called anytime during program execution, but should never be called while any disk files are currently open.

INIT subprogram:

Assembly language support is not provided.

INPUT statement:

See ACCEPT for description of cursor control enhancements. When reading numeric data from a file in INTERNAL format, integer and floating point datatypes can be freely mixed. XCOM Basic automatically makes the necessary conversions. See comments under PRINT statement for additional information on INTERNAL format files.

INT statement:

As a statement, provides a means to declare variables as integer types. See above discussion for details. As a function, INT is the same as in Extended Basic.

LINK subprogram:

Not supported.

LOAD subprogram:

Not supported for loading assembly language programs. Functions identically to POKE subprogram described below.

ON ERROR statement:

Enables trapping of all warnings and nonfatal errors. XCOM Basic treats all warnings and errors the same. Thus, overflow and input warnings can now be trapped.

ON WARNING statement:

Same as ON ERROR statement.

PEEKG and PEEKV subprograms:

Provide read access to GPL and VDP memory, respectively. Syntax identical to PEEK subprogram.

POKE, POKEG, and POKEV subprograms:

Provide write access to CPU, GPL, and VDP memory, respectively.

PRINT statement:

Integer constants or variables printed to a file in INTERNAL format take only 3 bytes of space (a length byte and 2 bytes for the integer). Floating point constants or variables take 9 bytes of space just as in Extended Basic.

RUN statement:

No support provided for RUNning external programs. Other formats of RUN statement fully supported.

SAY subprogram:

Similar to Extended Basic except that more complete access to the built-in vocabulary is available.

SOUND subprogram:

Same as Extended Basic except that the maximum value for the frequency is 32767, not 44733.

SPGET subprogram:

Returns an error if speech synthesizer is not attached or if word is not in built-in vocabulary. No longer returns code for "UHOH" when word is not found.

SUB statement:

Unlike Extended Basic, names of built-in subprograms are considered reserved. Thus, you cannot redefine built-in subprograms in XCOM Basic. While not truly recursive, XCOM Basic does not prevent a subprogram from calling itself. This may be useful in certain applications. XCOM Basic does not prevent branching into or out of a subprogram, but doing so may have unpredictable effects and is not recommended. Furthermore, variables within a subprogram are distinct from those elsewhere in the program even if they share the same name. Thus, if a subprogram branches to code outside its bounds, no variables from the subprogram will be accessible to that code. This differs from Extended Basic. For example,

```
10 CALL XYZ
20 STOP
30 PRINT I::RETURN
40 SUB XYZ::I=5::GOSUB 30::SUBEND
```

prints a result of 5 in Extended Basic, but prints 0 in XCOM Basic. Again, this practice is not recommended.

TRACE, UNBREAK, and UNTRACE statements:

Not supported.

VERSION subprogram:

Returns value of 100 under version 1.0.