Additions to Xtal BASIC

4.12 + 4.20

Dave Coomber,
G8UYZ.

1985.

APPENDIX F2

COMMAND/FUNCTION EXTENSION.

In 1979, Xtal introduced a capability which is probably unique to this type of BASIC. It allows the creation of an auxiliary reserved word table. This means that machine-code routines can be written and added to the interpreter as if they were commands and functions already built into the language. A good knowledge of machine-code programming is needed to take real advantage of this facility, and users who have not yet experienced machine code are advised to get studying! The ability to create what is, in effect, a personalised BASIC conforming to your own requirements is an extremely powerful tool indeed.

1. PROGRAM STORAGE

Before describing the method of adding auxiliary reserved words, it would be helpful to consider the way in which a program is stored within the text area. Many users will already know that Xtal BASIC does not actually use a line as typed but instead shortens each reserved word into a unique one or two-byte 'token'. This speeds up program execution, and also saves storage space. In addition, a null byte is appended to each line, so that we have a delimiter between each line of text (i.e. each numbered line). The line number is stored as a two-byte quantity (hexadecimal), and an additional two byte number is stored, which gives the offset to the start of the next line in the program text.
To illustrate this point, consider the following line of program text, stored in memory:

```
300 FOR I=0 to 9: PRINT SQR(I): NEXT:END
```

A normal text editor would store this line in memory in the form of ASCII codes thus:

```
3  0  0     F  O  R      I  =  0     T  O     9  :
33 30 30 20 46 4F 52 20  49 3D 30 20 54 4F 20 39 3A 20

P  R  I  N  T     S  Q  R  (  I  )  :
50 52 49 4E 54 20 53 51 52 28 49 29 3A 20

N  E  X  T  :  E  N  D   CR
4E 45 58 54 3A 45 4E 44 0D
```

This would be abbreviated by the Interpreter, into the following form:

```
 300 FOR I = 0 TO 9 : PRINT
1B 00 2C 01 8F 20 49 7E 30 20 72 39 3A 20 A2 20

SQR( I ) : NEXT: END
D9 28 49 29 3A 20 9B 3A 8E 00
```

Here, the first two bytes give the offset to the next line (this is &001B, as you will find if you count, starting from 0 to the first byte of the offset). The next pair gives the line number (&012C= 300). Finally, you will note that the spaces are significant, and remain in the text. They make little difference to the operating speed of the Xtal BASIC programs, and allow the user to lay out programs in the way that suits him/her. Removing them does, of course, save space, but this should not be done at the expense of readability unless absolutely necessary. Note that even '=' is treated as a reserved word, although it has only one character anyway. This is so that execution will be faster when scanning for relational operators (including ' ' and ','). The above format still applies if the line is the last in the program, since we always indicate the end of the program text by means of a null pair, i.e. the last THREE bytes of a Xtal BASIC program are 00. The pointer TXTTOP always points one ABOVE the last byte.

Since variable names and constants use ASCII codes from 00 to &5F (lower case variable names are internally converted to upper case immediately after entry), we may use codes &60 to &FF to represent our reserved words, and TATUNG/Xtal BASIC 4.2 actually uses codes &6F to &E9 in the standard reserved word table, and &80 to &98 in the auxiliary reserved word table.

Within REM and DATA statements and between double quotes, however, this compression does not occur, so that all ASCII codes, including lower case letters and graphic characters, may be included in these cases.

LIST expands the reserved word codes (but not within quotes, REM or DATA statements) into the actual words used, so that the user is not normally aware that all of this is going on.

2. RESERVED-WORD CONSTRUCTION

The reserved word table appears within the interpreter as a long string of reserved words held together, and separation is achieved by setting the top bit of the first byte in each word. In TATUNG/Xtal BASIC 4.2 the start of the table look like this:

```
            S  P  C  (  S  T  E  P
           7B D3 50 43 28 D3 54 45 50
Token:        6F          70

            T  A  B  (  T  H  E  N  ..... etc.
           D4 41 42 28 D4 48 45 4E .....
Token:     71          72
```

The first byte of this table is the total number of reserved words allocated (&7B, or 123 in this case), in case a corrupted program should happen to contain a non-existent token.

This is, in fact, not the best place to look at the table, since all of these words are special, in that they are not commands/functions in their own right, but appear only in certain statements. If we look a bit further through the table, we pass through the arithmetic and relational operators, and finally arrive at the commands:

```
            A  U  T  O  C  H  A  I  N
           C1 55 54 4F C3 48 41 49 4E
Token:     80          81

            C  L  E  A  R  C  L  O  S  E
           C3 4C 45 41 52 C3 4C 4F 53 45
Token:     82          83
```

Associated with each command or function is an address, where the routine for executing it may be found. All of these addresses build into an address table which, at RUN time, is indexed according to the token supplied.

a. The reserved word table is NOT used (or needed) at RUN time, only the address table.

b. All commands/functions may be accessed at RUN time at the same speed, so that the order in which they appear within the tables is immaterial.

3. THE AUXILIARY TABLES

This much is done in a similar way on many BASICs -- the point about Xtal BASIC is that it has TWO reserved word tables, one of which is empty, and may be expanded by the user. All user-defined reserved words are stored as TWO-byte tokens, the first one always being &FF, to distinguish them from the inbuilt reserved words. These words are stored in an AUXILIARY reserved word table, with their addresses being stored in an auxiliary address table. Both of these occupy no space within the interpreter, and so the user must create extra space in memory for the tables, in addition to that needed for the actual routines themselves.

Earlier versions of Xtal BASIC used a fixed area of RAM for holding the tables, but TATUNG/Xtal BASIC 4.2 uses the two pointers AUXCMD and AUXADR in the scratch-pad area, which may be set up by means of the PTR command. Hence the user may make his/her tables as big or as small as desired, the only requirement being that the last byte of the auxiliary reserved word table MUST be an 80H code, and MUST have the correct total of reserved words given at its start.

4. COMMANDS AND FUNCTIONS

There is an important distinction to be borne in mind when creating commands or functions and each will be checked by BASIC for correct syntax when being used. If the reserved word is to be used as a function, the word MUST end with a '(' (ASCII &28) to indicate that an argument is to follow.

In a command routine, the HL register pair is treated as the text pointer and, on entry, holds the memory address of the first non-space point to the statement separator (':') or the end-of-line byte 00. A simple RET instruction may be used to get back to BASIC. No other registers need to be preserved.

In a function routine, on the other hand, the text pointer has already been PUSHed onto the stack, and should be POPped and incremented to find the value of the argument. The routine almost always has a special end, since a closing parenthesis ')' MUST follow the argument expression.

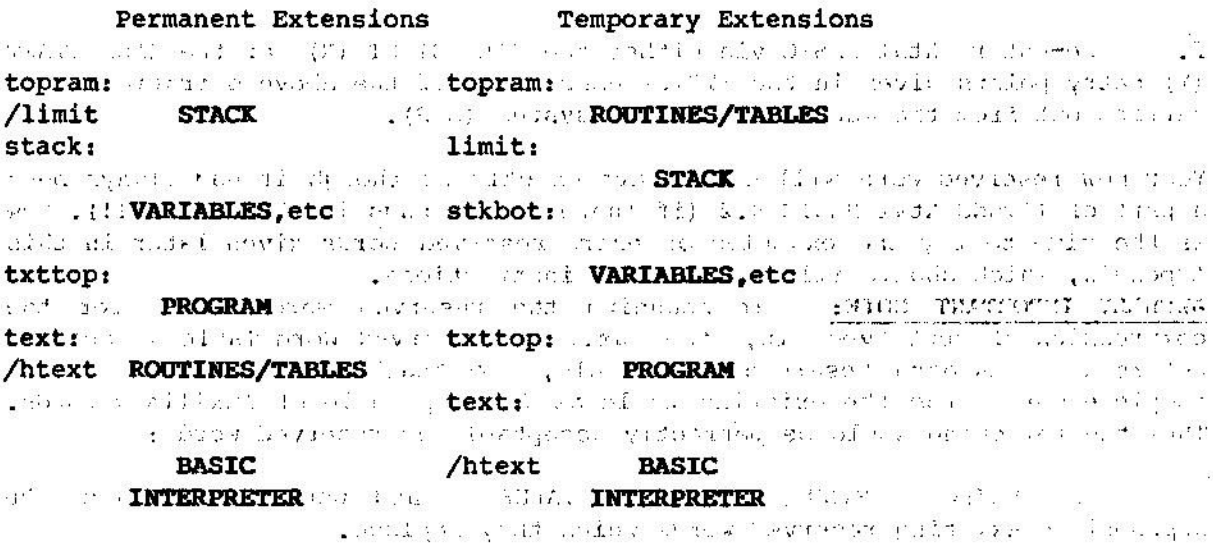
NOTE: If an auxiliary reserved word has been defined and used in a program, but has subsequently been cleared from the tables (or the tables themselves have been re-initialised), the program will still be LISTable, but all references to the word will display as a decimal number preceded by a question-mark (e.g. ?64).

5. HOW TO ENTER EXTRA RESERVED WORDS

Without any further ado, let us now give the step-by-step method for adding extra words to Tatung Xtal BASIC .

a. Decide whether the new words are to become a permanent part of TATUNG/Xtal BASIC 4.2, or are just to be added on temporarily. For example, you may have some 'Tool-Kit' type commands which may be required to assist with development of a BASIC program. You may then wish to drop those routines later, so that the space may be utilised by the program developed. To do all of this, use the CLEAR command to set aside a machine-code area at the top of the memory space, put your routine(s) and tables in there, either storing them in a .OBJ file POKE/DOKEing them from a BASIC program. Temporary extensions may be removed by executing a 'Cold start' to Xtal BASIC. It is quite in order for a BASIC program to define its own reserved words, which it will use itself later on within the program, and then to remove these extra words on completion.

If, on the other hand, you wish to make a permanent addition to the system, this may be done by moving up the HTEXT pointer (using a PTR 0, I command), so that the routine(s) and tables may be placed in the area created. They then become a natural extension to the interpreter, which may subsequently be saved to disc or tape (as your operating system allows). In this case, it is advisable to make the auxiliary tables larger than required, so that additional extensions may then use the same tables.
The simplified memory maps below illustrate the two methods:



For the remainder of the discussion, it is assumed that only one command or function is being entered, although clearly the same instructions apply to the addition of several words at once.

b. Having found our free area, write the machine code routine for performing the command/function within this area. This may be POKEd in from Xtal BASIC, or entered from within the machine-code monitor (MOS), of your machine.

c. The name of the routine, its reserved word, must now be written into the Auxiliary Reserved Word Table (pointed to by AUXCMD) as a set of ASCII codes, the first letter having its top bit set, as shown in section 2 of this Appendix. Do not forget to set up or modify the first byte of the table for the number of reserved words in the table, otherwise the command/function will return an error when later invoked !. The address of the table held in AUXCMD may be entered from within BASIC by means of a PTR 3, I command, if desired.

d. The appropriate address in the Auxiliary Address Table (which is pointed to by AUXADR) is then set up for the start of the newly entered machine-code routine so that, when the command or function is invoked, this routine will be executed. The address of this table may be set up in AUXADR by means of a PTR 8, I command, if desired.

NOTE: This also applies to c., above. Do NOT use the PTR command to set up the Auxiliary Tables when making permanent extensions, because the next 'Cold Start' will simply remove them! For permanent extentions, set the pointers in the 'default scratch-pad' (which is copied to the scratch-pad area whenever a 'Cold Start' is executed). The necessary addresses are given at the end of this Appendix.

e. If permanent extensions have been made, save a new copy of TATUNG/Xtal BASIC 4 onto disc before running it up, not forgetting to include the area added to the end of the Interpreter!

f. Re-enter Xtal BASIC via either the COLD START (X) or the WARM START (Y) entry points given in the SYSTEM HANDBOOK, if the above operations were carried out from the machine operating system (MOS).

Your new reserved word will now behave exactly as though it had always been a part of TATUNG/Xtal BASIC 4.2 (if there are no bugs in the routine!!). Now is the time to try the examples of extra reserved words given later in this Appendix, which should illustrate these instructions.

AWFULLY IMPORTANT NOTE: In scanning the reserved word tables for the compression of text typed in, the Auxiliary Reserved Word Table is scanned before the Standard Reserved Word Table, so that it is possible to use complete words from the existing table as part or whole of Auxiliary words. Thus the following would be perfectly acceptable as reserved words:

PRINTUSING SINH DELAY VALUE and would not affect the appropriate existing reserved words which they replace.

However, if READ was included in the Auxiliary tables, it would assume priority over the existing word READ, with rather interesting results! In particular, a .XBS file containing READ statements would continue to execute the existing READ statement, but any lines to that program would correspond to the new READ command, if READ was typed into any of those lines. This option should therefore be used with some care.

USEFUL SUBROUTINES

Note: All of the addresses given within this appendix are specified in Hexadecimal. This appendix has been provided for assiting the generation of extra reserved words in an efficient manner. It is not complete, but we think that the most useful routines are all present in this list!

Error Messages

Not much more need be said about this than has already been covered in ??, except to give the address of the routine ERROR, which actually handles errors, and may be found at 06CF. The only register which matters here is E, which contains the error number, as defined in ??. It is not necessary to CALL this routine, just jump to it!

User-Function Termination Routines

```
0284  FNBIT:   ;RETURN BIT VALUE IN CARRY FLAG.
0287  FNENDB:  ;RETURN BYTE VALUE IN A.
028A  FNENDI:  ;RETURN INTEGER VALUE IN AB
                (high byte in A).
028A  FNENDP:  ;RETURN NUMERIC VALUE IN FPA
                (Floating-point).
0290  FNEND:   ;RETURN EXPR. VALUE IN FPA
                (may be a string)
```

These routines should NOT be CALLed, but used to terminate your function routine. The routines all assume that the text pointer is on stack, so that the registers may contain anything on entry to these routines (except, of course, for the ones returning results!). Also, see section ? of this appendix for the use of STREND, the usual way of returning string results.

General-Purpose Text Scanning Routines

As explained in Chapter ?, Xtal BASIC uses the HL register pair as the pointer to the current position in the program text. The following routines make use of this:

RDLN 2697 Reads in a line of text from the keyboard or current input device to the BUFFER, pointed to by BUFPTR. This makes use of the editing facilities described in CHAPTER ?, according to the IOM setting currently in use. On entry, if 'Line Edit' mode is in force, the character contained in A is printed as a prompt at the start of the line.
On exit, the carry flag is set if the line has been abandoned by ESC, but is reset if CR has been used to complete the line. In this case, the line in the buffer is terminated with a 00 byte, and HL is left pointing to one byte before the start of the buffer.
Registers affected: A and HL.

PR 2600 Print character in A register, to VDU or current output device. The side-effect of this is that the location PRTCOL is adjusted to give the correct column on the screen/printer, for TABs, etc. In addition, a delay is imposed if the SPEED command has been used to slow down the print rate.
Registers affected: Flags only. In particular, carry is always reset.

PRTNUM 02FB Prints the contents in the HL register pair as an integer in the range 0-65535. All registers may be affected.

PRM 02B8 Prints the message immediately following the sub-routine call, terminated by having the MSB of the last character set. This means that all other characters codes must have ASCII values in the range 0-&7F. Thus, to print "Hello there", we do:

```
CD B9 02 48 65 6C 6C 6F 20 74 68 65 72 E5
   H  e  l  l  o     t  h  e  r
```

On exit, A holds the last character printed, still with its top but set, and the return address is that immediately following the last character in the message, but no other registers are affected.

CPHLDE 0296 Compare HL and DE and return flags set as follows:

Carry -- Set if HL DE, reset if HL =DE.
Zero -- set if HL DE.
Registers affected: A.

LTRCHK 02D9 Places the character contained at (HL) in A, and tests to see if it is a letter in the range A - Z (i.e. a capital letter). Carry is reset if it is a capital letter, and Set if it is any other character. No other register is affected.

LWRTST 02CF Loads character from (HL) into A and, if in the range &60-&7f, converts it to upper-case (in the range &40-&5F). Only A and the flags are affected.

IGBLK 02AA Increments HL, until the first non-space character is found. On return, A contains the character found, and HL points to that character. ZA flag is set if at the end of statement (null or ':' found), and C flag set numeric character found (0-9).

TSTC 0241 Test character at (HL, ensuring that it is the same as that specified immediately after the call. If note, A Syntax Error occurs. This is effectively a four-byte call, e.g. CD xx xx 28 lookks for a '('. A contains the test character, and HL points to the next non-blank character following the tested one. Note that we may also use this routine to test for a reserved word token.

TSTCOM 029C
RPARN 0291 Special cases of TSTC, test from comma ',' and right parenthesis ')' respectively. These onl require 3 bytes instead of four, though!

FNDLN 033E Searches for the line in the program text given by DE, from the start of text. Returns with the following conditions:
Carry and Zero set: Line found, BC points to start of line, HL points to start of following line (or to 0000 if the line found is the last in the text), as described in Chapter ??.
Carry reset, Zero set: .Line not found, but we have found a line with a number larger than that searched for, BC pointing to that line, and HL pointing to the next line (or 0000).
Other registers affected: A will b affected, but DE will remain unchanged.

NXTLN 0341 As for FNDLN above, but this time searches for the line given in DE from the current position in the text, given in HL.

COMPRSS 0878 Routine to take a line of text in the buffer starting at the location given in HL, and terminated by a 00 byte, and which generates the same line in the compressed format given above, in the input buffer (BUFFER). Note that the new line is ALWAYS shorter than the original. In normal use, when entering a line of text into a program, the compressed line overlays the input line, since the pointer to the original text is always in front of that to the compressed text.
In addition, the line numer is not considered here, since HL is pointing at the next non-blank character after the line number (if one has been used). COMPRSS does NOT generate a compressed line number nor the pointer to the next line.

Registers affected: all. HL points to one byte before the start of the buffer on exit, DE points to the last byte plus two in the compressed line, and C holds the number of bytes in the compressed line, plus four to take account of the space needed for the line number and pointer.

Floating-Point Features
a. Representation of floating-point numbers.
A floating-point number in Xtal BASIC 3 is stored in four consecutive bytes. There are four bytes reserved within the scrath-pad, used for floating-point calculations, called the Floating-Point Accumulator (FPA), and a further byte TEMP is used by the f.p routines for storing temporary calculations. A part from these, only the registers and the stack are used for F.p calculations.

The high byte of the FPA is the exponent, which is a signed power of two. Note that the sign bit is 0 if NEGATIVE, 1 if POSITIVE (for a reason which will become apparent later). The lower 3 bytes for a signed mantissa, the top bit of the top byte being the sign (this time 0 is POSITIVE, 1 if NEGATIVE!). The mantissa is a number between 0 and 1, with the binary point coming above the top bit.

if we let e = Exponent byte, and
m = Mantissa bytes, we express any f.p number N as:

$$N = (1 + m) * 2 (e - 1),$$

with the added convention that any number with a zero exponent is taken as 0. Now we see why 1 is used for a positive sign on the exponent -- e=01 must represent 2 (-128), and 0 is clearly smaller than this (not much!). Note that E=80 represents 2 (-1), or 0.5 up to 1 (depending on the value of m). The advantage of using this convention for 0 is that we can initialise variables and arrays simply by filling them with 0's (each element is then zero).

This is still probably as clear as mud (!), so let's have a few examples, to illustrate the system:

Decimal No.	Hex (f.p) representation	Remarks
0	00 00 00 00	zero
1	81 00 00 00	2 0
2	82 00 00 00	2 1
3	82 40 00 00	1.5*2 1
-3	82 C0 00 00	
3.141593	82 49 0F DB	pi
0.6931472	80 31 72 18	Ln(2)
65536	91 00 00 00	2 16

The RANGE over which we can operate is determined by e, and is thus: 2 (-128) N 2 127, which is 2.938736 * 10 (-39) to 1.701412 * 10 38.

The ACCURACY of calculations is determined by the length of m, which in this case represents 1 part in 2 24, or an error of 5.960464 * 10 (-8), which is better than 7 sig. figs. However, to try and account for rounding errors, we allow one guard digit, and so you will note that all numbers are printed to 6 sig. figs (even this does not ALWAYS account for ALL errors, and you will note, for instance, that 3*4 is displayed as 81.0001, and not 81, as it should be! This is mainly due to problems which conversion from binary to decimal, as well as the accuracy of the method used for calculating powers).

b. Floating-point functions and operators.

The address of the single-argument f.p functions are as follows: In each case, the argument is taken from the FPA on entry, and the result returned in it on exit:

LOG	1B87	LN	1B93	EXP	1E4A
SIN	1ECA	COS	1EC4	TAN	1EB0
ATN	1E8A	RND	1F36	ABS	1D60
SGN	1D6C	INT	1DC0	SQR	1DFD

By 'operators' we mean those in which we are dealing with TWO f.p quantities. In general, we do a calculation in the form a = b o a, where a = contents of FPA, b = conetns of top four bytes of stack, and o is the operation performed. On the stack, the top pair of bytes represent the exponent (high byte) and top byte of mantissa. For each operator, there is another entry point (given a suffix '1'), in which b is stored in the BCDE registers. Here, B contains the exponent, C the high byte of the mantissa, and DE the rest of the mantissa." We call the set of four registers used in this way the Floating-Point Register (FPR). The result of any of these operations is, of course, returned in the FPA.

ADD	1A91	ADD1	1AA6	SUB	1AA1	SUB1	1AA3
MULT	1BD2	MULT1	1BD4	DIVIDE	1C24	DIV1	1C26
POWER	1E06	POWER1	1E07	ADDN	1A88	SUBN	1A9D
MOD	1CAB	MOD1	1CAD	MUL10	1CDA	DIV10	1C18

NOTE: POWER is actually calculated as: X Y = EXP (Y * LOG(X)), with the convention that X 0 = 1 for X =0 and 0 Y = 0 for Y 0, and X Y is not defined for X 0 or for X=0 and y 0.

MUL10 and DIV10 respectively multiply and divide the contents of the FPA by 10, leaving the result in the FPA.
ADDN and SUBN are like ADD1 and SUB1, except that HL points to a memory location at which b may be found. You can place a constant here, or even a temporary result, if you wish. Xtal BASIC stores a large table of constants within the Interpreter, and here are some of the more useful ones:

HALFPI	203A	Pi/2	HALF	203E	0.5
TWOPI	2036	Pi*2	QTR	2042	0.25
ONE	1FFC	1	NEGONE	1FF8	-1

c. Other useful F.P. routines.

STKFPA 1D87 Returns with the FPA on the stack, in the form shown above. Destroys the DE registers.
LDFPR 1D94 Copies the FPA to the FPR, leaving HL pointing to TEMP.
STFPR 1DAC Copies the FPR to the FPA, without effecting any registers.
HLTFPA 1D97 Copies the four bytes starting at (HL) into the FPR AND FPA, leaving HL pointing to the byte following the black of four.

The HOLD/CHAIN combination separates the two sections such that the other sub-programs illustrated can be called up in place of the INITIAL ROUTINES section each time. (The flow of the original program would be structured such that the initial routines are executed before the line containing the HOLD/CHAIN commands is encountered).

Thus:-

HOLD line number :CHAIN"SUB1" calls up the 1st sub-program.
HOLD line number :CHAIN"SUB2" calls up the 2nd sub-program.
HOLD line number :CHAIN"SUB3" calls up the 3rd sub-program.

Each time execution would continue from the beginning of the added sub-program which can then access the COMMON ROUTINES section of the original program.

This method saves file space and greatly improves the efficiency of a CHAIN, by speeding up the loading of each program.

NOTE: The line numbers of the **sub-program** must be selected so as to be greater than those of the COMMON ROUTINES section.

The example Mailing List program given in the section on FILE HANDLING (page 285) illustrates the use of this method of SEMI-CHAINING programs.
The **original program** consists of the COMMON ROUTINES section in lines 10 to 890 and the INITIAL ROUTINES section in lines 1000 to 1110.

The HOLD/CHAIN combination is found in line 900 and the flow of the **original program** is structured such that the INITIAL ROUTINES (lines 1000 to 1110) are executed before line 900 is encountered.

There are three **sub-programs** involved with titles "MSUB1", "MSUB2", and "MSUB3". These sub-programs are accessed according to the input in line 870 which places a value in N$ (given by the user response to the question "which?")

HOLD

Syntax: HOLD L1,L2

L1 and L2 indicate the first and last line numbers of a range held 'in view'. If omitted L1 will default to 0 and L2 to 65535.

Purpose: This is a System Command which will 'hold' a range of line numbers 'in view' for manipulation, (like renumber), or execution. The 'non held' section seems to disappear, but is still held in memory.

Command examples:-

```
HOLD 100,199  will hold line 100 to 199 exclusive.
HOLD 100      will hold all lines from 100 onwards.
HOLD 199      will hold all lines up to 199.
```

This is a very useful command which will permit the user to 'move' blocks of program, or add another program held on a disc. This will allow a programmer to write his routines separately, and then 'build up' his final program in order of preference. **NOTE!** Be careful of line numbering, and double check that the section to be added does not include line numbers held 'in view'. (It could be said that "HOLD" means 'reserve')

NOTE: See also MGE (Page ?)

You can only add external programs to the end of the first program.
Type in and save (as 'AA') the following:-

```
 10 REM START OF HOLD DEMO(AA)
 20 REM DATA
 30 REM DATA
 40 REM DATA40
 50 REM DATA50
 60 REM DATA60
 70 REM DATA70
 80 REM
 90 REM END OF FIRST SECTION
100 REM
110 REM
120 REM DATA 120.
130 REM ETC
140 REM AND SO ON
150 REM PROCESS
160 REM
170 END
```

Now suppose we want to move the lines 40,50,60, and 70 to the end of the program.

```
HOLD 40,70
```

now we can manipulate its line numbers:-

```
RENUM 300,5
```

which will assign a new number sequence to it.

```
MGE
```

If you LIST the program, you should get:-

```
LIST
 10 REM START OF HOLD DEMO(AA)
 20 REM DATA
 30 REM DATA
 80 REM
 90 REM END OF FIRST SECTION
100 REM
110 REM
120 REM DATA 120
130 REM ETC
140 REM AND SO ON
150 REM PROCESS
160 REM
```

```
170 END
300 REM DATA40
305 REM DATA50
310 REM DATA60
315 REM DATA70
```

Now move it to the beginning:-

```
RENUM 1000
```

which should give you:-

```
1000 REM START OF HOLD DEMO(AA)
1010 REM DATA
1020 REM DATA
1030 REM
1040 REM END OF FIRST SECTION
1050 REM
1060 REM
1070 REM DATA 120
1080 REM ETC
1090 REM AND SO ON
1100 REM PROCESS
1110 REM
1120 END
1130 REM DATA40
1140 REM DATA50
1150 REM DATA60
1160 REM DATA70

HOLD 1130,1160
RENUM 300,5
MGE
```

will now LIST to:-

```
 300 REM DATA40
 302 REM DATA50
 304 REM DATA60
 306 REM DATA70
1000 REM START OF HOLD DEMO(AA)
1010 REM DATA
1020 REM DATA
1030 REM
1040 REM END OF FIRST SECTION
1050 REM
1060 REM
1070 REM DATA 120
1080 REM ETC
1090 REM AND SO ON
1100 REM PROCESS
1110 REM
1120 END
```

EXAMPLE 2: Adding another program from a disc. (Note: Be careful to double check the line numbers. If you have a printer, LIST both and work out first what numbers you need.)

Type in and save (QQ):-

```
1000 REM START OF ADDED BIT(QQ)
1010 REM
1020 REM
1030 REM
1040 REM END OF ADDED PART
```

Clear the memory (NEW) and load 'AA'

```
RENUMber it to 100,5
HOLD 190
LOAD 'QQ'
MGE
```

should now get a LIST of:-

```
 100 REM START OF HOLD DEMO
 105 REM DATA
 110 REM DATA
 115 REM
 120 REM END OF FIRST SECTION
 125 REM
 130 REM DATA40
 135 REM DATA50
 140 REM DATA60
 145 REM DATA70
 150 REM
 155 REM DATA 120
 160 REM ETC
 165 REM AND SO ON
 170 REM PROCESS
 175 REM
 180 END
1000 REM START OF ADDED BIT(QQ)
1010 REM
1020 REM
1030 REM
1040 REM END OF ADDED PART
```

Related Keywords: MGE LOAD SAVE

Subroutine address for various versions of Xtal Basics.

Sub-rtn.	V3.0	V4.12	V4.2
FNBIT	0285	0287	0284
FNENDB	0288	028A	0287
FNENDI	028B	028D	028A
FNENDP	028E	0290	028D
FNEND	0291	0293	0290
CPHLDE	0297	0299	0296
TSTCOM	029D	029F	029C
IGBLK	02AA	02AC	02AA
PRM	02B9	02BB	02B8
LTRCHK	02DF	02E1	02D9
PRTNUM	02FC	02FE	02FB
FORMPOS	04E1	04DD	04D7
FORMNUM	056B	056C	0569
ERROR	06CF	06D0	06CD
RDLN	2687	2695	2697
PR	25E7	25FE	2600
LWRTST	02D0	02D2	02CF
IGBLK	02AA	02AC	02AA
LOG	1B51	1B80	1B87
SIN	1E94	1EC3	1ECA
ATN	1E54	1E83	1E8A
SGN	1D36	1D65	1D6C
LN	1B5D	1B8C	1B93
COS	1E8E	1EBD	1EC4
RND	1F00	1F2F	1F36
INT	1D8A	1DB9	1DC0
EXP	1E14	1E43	1E4A
TAN	1E7A	1EA9	1EB0
ABS	1D2A	1D59	1D60
SQR	1DC7	1DF6	1DFD
ADD	1A5B	1A8A	1A91
MULT	1B9C	1BCB	1BD4
POWER	1DD0	1DFF	1E06
MOD	1C75	1CA4	1CAB
SUB	1A6B	1A9A	1AA1
DIVIDE	1BEE	1C1D	1C24
ADDN	1A62	1A91	1A88
MUL10	1CA4	1CD3	1CDA
SUB1	1A6D	1A9C	1AA3
DIV1	1BF0	1C1F	1C26
SUBN	1A67	1A96	1A9D
DIV10	1BE2	1C11	1C18
STKFPA	1D51	1D80	1D87
LDFPR	1D5E	1D8D	1D94
STFPR	1D76	1DA5	1DAC
HLTFPA	1D61	1D90	1D97
FPATHL	1D7F	1DAE	1DB5
DETOHL	1D82	1DB1	1DB8
CHKSGN	1D1B	1D4A	1D51
CHGSGN	1D2E	1D5D	1D64
POLY1	1EE0	1F0F	1F16
POLY	1ED1	1F00	1F07
SINTAB	1FF6	2023	2025
LOGTAB	1FA3	1FD0	1FD2

EXPTAB	1FB0	1FDD	1FDF
EXPR	165A	1695	169C
EXNMCK	1641	167C	1683
PARN	1656	1691	169D
FCHNUM	2034	2061	2063
GETNM	161C	1659	1660
TXTNUM	20D4	2103	2105
TXT1	20D7	2106	2108
UEXINT	15E7	161F	1626
INTEXP	15F4	162C	1633
I255	150D	1642	1649
FORMNUM	056B	056C	0569
FORMPOS	04E1	04DD	04DA
NUMCHK	1644	167F	1686
STRCHK	1647	1682	1689
TYPMCH	1649	1684	168B
FCHSTR	1409	1447	144E
LEN1	143E	147C	1483
ASC1	144D	148B	1492
STRSPC	12C5	1301	1308
ASNSTR	1273	12AF	12B6
STREND	12A6	12E2	12E9
FNDVAR	187D	18AC	18B3

```
 10 REM PROGRAM TO MODIFY BASIC.V4.2
 20 REM TO ALLOW ADDITION OF USER RESERVED WORDS
 30 RST:BCOL1:TCOL15,0
 40 PRINT@6,2;"BASIC MODIFICATION PROGRAM"
 50 CLEAR &6100
 60 REN"XBAS.COM"TO"XBAS.OBJ"
 70 PRINT@0,4;"LOADING BASIC"
 80 LOAD "XBAS.OBJ"
 90 REN"XBAS.OBJ"TO"XBAS.COM"
100 PRINT@0,6;"BASIC LOADED"
110 :
120 REM CLEAR NEW RAM AREA
130 FOR I=0 TO 511:POKE &9E00+I,0:NEXT
140 :
150 REM MOVE COMMAND/FUNCTION TABLE TO NEW RAM
160 FOR I=0 TO 111:POKE &9E00+I,PEEK(&9BC5+I):NEXT
170 :
180 REM MOVE ADDRESS TABLE TO OLD WORD TABLE
190 FOR I=0 TO 49:POKE &9BC5+I,PEEK(&9C35+I):NEXT
200 :
210 REM SET UP NEW POINTERS IN HARD SCRATCH
220 DOKE &8B84,&4001:REM NEW START OF BASIC
230 DOKE &8B88,&3E00:REM NEW AUX COMMAND/FUNCTION
240 DOKE &8B92,&3BC5:REM NEW ADDRESS TABLE
250 :
260 REM CLEAR OLD AUXILIARY TABLE
270 FOR I=0 TO 111:POKE &9BF7+I,0:NEXT
280 :
290 PRINT@0,8;"ENTER VER.NO. SUFFIX(1 TO 3 DIGITS) ";:INPUT X$
300 IF LEN(X$)>3 THEN GOTO 290:ELSE POKE &9DB2,ASC(X$)
310 PRINT@0,10;"ENTER FILE NAME OF NEW BASIC ";:INPUT F$
320 IF F$="" THEN F$="NEWXBAS"
330 G$=F$+".OBJ":H$=F$+".COM"
340 PRINT@0,12;"SAVING NEW BASIC"
350 SAVE G$,&6100,&9FFF
360 REN G$ TO H$
370 PRINT@0,14;"NEW BASIC SAVED"
380 BCOL4
390 END
```

BASIC 4.2
BEFORE RUNNING XDASMOD

```
3B90 D4 41 4E D6 41 4C CC 45 46 54 24 CD 49 44 24 D2 TANVALLEFT$MID$R
3BA0 49 47 48 54 24 C5 52 52 C5 52 4C C5 4F 46 C6 4E IGHT$ERRERLEOFFN
3BB0 C9 4E 43 48 CB 42 44 CD 55 4C 24 CE 4F 54 D0 49 INCHKBDMUL$NOTPI
3BC0 D3 49 5A 45 80 19 C4 52 41 57 D4 43 4F 4C C7 43 SIZE..DRAWTCOLGC
3BD0 4F 4C C2 43 4F 4C D3 50 52 49 54 45 CD 41 47 D3 OLBCOLSPRITEMAGS
3BE0 48 41 50 45 CF 52 49 47 49 4E C5 4C 4C 49 50 53 HAPEORIGINELLIPS
3BF0 45 C4 4F 53 D2 41 44 28 C4 45 47 28 D0 4F 4C 59 EDOSRAD(DEG(POLY
3C00 C6 49 4C 4C D6 50 4F 4B 45 D6 50 45 45 4B 28 D6 FILLVPOKEVPEEK(V
3C10 44 4F 4B 45 D6 44 45 45 4B 28 C2 45 45 50 C2 49 DOKEVDEEK(BEEPBI
3C20 4E 24 28 D2 53 54 CB 45 59 C1 44 43 28 C2 54 4E N$(RSTKEYADC(BTN
3C30 28 D0 53 57 80 14 2D AB 2C B0 2C 9C 2C BB 2E AC (PSW..-+,0,.,;.,
3C40 2E 68 2E EE 2C 94 2D FC 3D DA 34 EC 34 7C 2D 2B .h.n,.-|=Z414|-+
3C50 2E 8D 34 BA 34 9F 34 C7 34 75 30 F9 34 F8 2B 9C ..4:4.4G4u0y4x+.
3C60 2F 34 2C 4C 2C FA 35 2B 10 DE 0A 11 0B 8F 29 61 /4,L,z5+.^....)a
3C70 2C 6D 0E 87 29 86 0E 85 0C 98 10 AE 18 48 04 23 ,m..).........H.#
3C80 2A 86 0E 72 0A 76 0D 8D 0B 9F 0B D2 10 2E 0E 99 *..r.v.....R....
3C90 0E 32 0C 57 09 46 28 FA 10 D3 2B 96 0A 0E 0D 9E .2.W.F(z.S+.....
3CA0 06 E6 0B 79 29 79 04 38 04 BD 0B A6 0F BE 0E 86 .f.y)y.8.=.&.>..
3CB0 0E 70 11 FA 2C 52 0E D1 0B EC 0A 97 28 FC 2C 65 .p.z,R.Q.l..(|,e
3CC0 0A 66 1A D7 28 7F 04 6D 06 4A 36 78 36 40 37 1F .f.W(..m.J6x6@7.
3CD0 36 03 37 38 36 45 31 5F 04 27 06 1D 04 9A 05 F8 6.786E1_.'.....x
3CE0 05 29 04 04 06 11 06 BE 30 EA 30 04 31 89 30 6B .).....>0j0.1.0k
3CF0 04 49 06 24 04 C8 05 FF 05 31 04 0B 06 19 06 A9 .I.$.H...1.....)
>
```

AUX.
TAB

AUX.
TAB

XBAS V4.2

AFTER RUNNING

```
T 3B00 3CFF 10
3B00 41 CC 4F 43 4B D2 45 4E D5 4E 4C 4F 43 4B CD 55 ALOCKRENUNLOCKMU
3B10 53 49 43 C3 41 4C 4C C9 4F 4D CE 55 4C 4C D0 54 SICCALLIOMNULLPT
3B20 52 D3 45 50 D3 50 45 45 44 D7 49 44 54 48 DA 4F RSEPSPEEDWIDTHZO
3B30 4E 45 D4 49 24 D4 45 4D 50 4F D6 4F 49 43 45 D0 NETI$TEMPOVOICEP
3B40 53 47 C1 42 53 C1 53 43 C1 54 4E C3 48 52 24 C3 SGABSASCATNCHR$C
3B50 4F 53 C4 45 45 4B C5 56 41 4C C5 58 50 C8 45 58 OSDEEKEVALEXPHEX
3B60 24 C9 4E 50 C9 4E 54 CC 45 4E CC 4E CC 4F 47 D0 $INPINTLENLNLOGP
3B70 45 45 4B D0 4F 49 4E 54 D0 4F 53 D2 4E 44 D3 43 EEKPOINTPOSRNDSC
3B80 52 4E 24 D3 47 4E D3 49 4E D3 51 52 D3 54 52 24 RN$SGNSINSQRSTR$
3B90 D4 41 4E D6 41 4C CC 45 46 54 24 CD 49 44 24 D2 TANVALLEFT$MID$R
3BA0 49 47 48 54 24 C5 52 52 C5 52 4C C5 4F 46 C6 4E IGHT$ERRERLEOFFN
3BB0 C9 4E 43 48 CB 42 44 CD 55 4C 24 CE 4F 54 D0 49 INCHKBDMUL$NOTPI
3BC0 D3 49 5A 45 80 14 2D AB 2C B0 2C 9C 2C BB 2E AC SIZE.-+.,.,:.,  AUX.
3BD0 2E 68 2E EE 2C 94 2D FC 3D DA 34 EC 34 7C 2D 2B .h.n,.-|=Z414|-+   TAB
3BE0 2E 8D 34 BA 34 9F 34 C7 34 75 30 F9 34 F8 2B 9C ..4:4.4G4u0y4x+.
3BF0 2F 34 2C 4C 2C FA 35 00 00 00 00 00 00 00 00 00 /4,L,z5.........
3C00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C60 00 00 00 00 00 00 00 2B 10 DE 0A 11 0B 8F 29 61 .......+.^....)a
3C70 2C 6D 0E 87 29 86 0E 85 0C 98 10 AE 18 48 04 23 ,m..).........H.#
3C80 2A 86 0E 72 0A 76 0D 8D 0B 9F 0B D2 10 2E 0E 99 *..r.v.....R....
3C90 0E 32 0C 57 09 46 28 FA 10 D3 2B 96 0A 0E 0D 9E .2.W.F(z.S+.....
3CA0 06 E6 0B 79 29 79 04 38 04 BD 0B A6 0F BE 0E 86 .f.y)y.8.=.&.>..
3CB0 0E 70 11 FA 2C 52 0E D1 0B EC 0A 97 28 FC 2C 65 .p.z,R.Q.l..(|,e
3CC0 0A 66 1A D7 28 7F 04 6D 06 4A 36 78 36 40 37 1F .f.W(..m.J6x6@7.
3CD0 36 03 37 38 36 45 31 5F 04 27 06 1D 04 9A 05 F8 6.786E1_.'.....x
3CE0 05 29 04 04 06 11 06 BE 30 EA 30 04 31 89 30 6B .).....>0j0.1.0k
3CF0 04 49 06 24 04 C8 05 FF 05 31 04 0B 06 19 06 A9 .I.$.H...1.....)
>
```

xBAS V4.2
AFTER RUNNING

```
] 3E00 3FFF 10
3E00 19 C4 52 41 57 D4 43 4F 4C C7 43 4F 4C C2 43 4F .DRAWTCOLGCOLBCO
3E10 4C D3 50 52 49 54 45 CD 41 47 D3 48 41 50 45 CF LSPRITEMAGSHAPEO
3E20 52 49 47 49 4E C5 4C 4C 49 50 53 45 C4 4F 53 D2 RIGINELLIPSEDOSR
3E30 41 44 28 C4 45 47 28 D0 4F 4C 59 C6 49 4C 4C D6 AD(DEG(POLYFILLV
3E40 50 4F 4B 45 D6 50 45 45 4B 28 D6 44 4F 4B 45 D6 POKEVPEEK(VDOKEV
3E50 44 45 45 4B 28 C2 45 45 50 C2 49 4E 24 28 D2 53 DEEK(BEEPBIN$(RS
3E60 54 CB 45 59 C1 44 43 28 C2 54 4E 28 D0 53 57 80 TKEYADC(BTN(PSW.
3E70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3E80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3E90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EA0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EB0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3ED0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FA0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FB0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
>
```

XBASX V4

```
T 3B00 3CFF 10
3B00 41 CC 4F 43 4B D2 45 4E D5 4E 4C 4F 43 4B CD 55 ALOCKRENUNLOCKMU
3B10 53 49 43 C3 41 4C 4C C9 4F 4D CE 55 4C 4C D0 54 SICCALLIOMNULLPT
3B20 52 D3 45 50 D3 50 45 45 44 D7 49 44 54 48 DA 4F RSEPSPEEDWIDTHZO
3B30 4E 45 D4 49 24 D4 45 4D 50 4F D6 4F 49 43 45 D0 NETI$TEMPOVOICEP
3B40 53 47 C1 42 53 C1 53 43 C1 54 4E C3 48 52 24 C3 SGABSASCATNCHR$C
3B50 4F 53 C4 45 45 48 C5 56 41 4C C5 58 50 C8 45 58 OSDEEKEVALEXPHEX
3B60 24 C9 4E 50 C9 4E 54 CC 45 4E CC 4E CC 4F 47 D0 $INPINTLENLNLOGP
3B70 45 45 4B D0 4F 49 4E 54 D0 4F 53 D2 4E 44 D3 43 EEKPOINTPOSRNDSC
3B80 52 4E 24 D3 47 4E D3 49 4E D3 51 52 D3 54 52 24 RN$SGNSINSQRSTR$
3B90 D4 41 4E D6 41 4C CC 45 46 54 24 CD 49 44 24 D2 TANVALLEFT$MID$R
3BA0 49 47 48 54 24 C5 52 52 C5 52 4C C5 4F 46 C6 4E IGHT$ERRERLEOFFN
3BB0 C9 4E 43 48 CB 42 44 CD 55 4C 24 CE 4F 54 D0 49 INCHKBDMUL$NOTPI
3BC0 D3 49 5A 45 80 14 2D AB 2C B0 2C 9C 2C BB 2E AC SIZE.¦.-+,0,.,;.,
3BD0 2E 68 2E EE 2C 94 2D FC 3D DA 34 EC 34 7C 2D 2B .h.n,.-¦=Z414¦-+
3BE0 2E 8D 34 BA 34 9F 34 C7 34 75 30 F9 34 F8 2B 9C ..4:4.4G4u0y4x+.
3BF0 2F 34 2C 4C 2C FA 35 81 3E 96 3E B9 3E C5 3E D6 /4,L,z5¦.>.>9>E>V
3C00 3E E1 3E 28 3E 45 3F 00 00 00 00 00 00 00 00 00 >a>(?E?.........
3C10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C60 00 00 00 00 00 00 00 2B 10 DE 0A 11 0B 8F 29 61 .......+.^....)a
3C70 2C 6D 0E 87 29 86 0E 85 0C 98 10 AE 18 48 04 23 ,m..)........H.#
3C80 2A 86 0E 72 0A 76 0D 8D 0B 9F 0B D2 10 2E 0E 99 *..r.v.....R....
3C90 0E 32 0C 57 09 46 28 FA 10 D3 2B 96 0A 0E 0D 9E .2.W.F(z.S+.....
3CA0 06 E6 0B 79 29 79 04 38 04 BD 0B A6 0F BE 0E 86 .f.y)y.8.=.&.>..
3CB0 0E 70 11 FA 2C 52 0E D1 0B EC 0A 97 28 FC 2C 65 .p.z,R.Q.l..(¦,e
3CC0 0A 66 1A D7 28 7F 04 6D 06 4A 36 78 36 40 37 1F .f.W(..m.J6x6@7.
3CD0 36 03 37 38 36 45 31 5F 04 27 06 1D 04 9A 05 F8 6.786E1_.'.....x
3CE0 05 29 04 04 06 11 06 BE 30 EA 30 04 31 89 30 6B .)......>0j0.1.0k
3CF0 04 49 06 24 04 C8 05 FF 05 31 04 0B 06 19 06 A9 .I.$.H...1.....)
>
```

AUX. TABLE EXTRA

XBASX V4.2

```
T 3E00 3FFF 10
3E00 21 C4 52 41 57 D4 43 4F 4C C7 43 4F 4C C2 43 4F .DRAWTCOLGCOLBCO
3E10 4C D3 50 52 49 54 45 CD 41 47 D3 48 41 50 45 CF LSPRITEMAGSHAPEO
3E20 52 49 47 49 4E C5 4C 4C 49 50 53 45 C4 4F 53 D2 RIGINELLIPSEDOSR
3E30 41 44 28 C4 45 47 28 D0 4F 4C 59 C6 49 4C 4C D6 AD(DEG(POLYFILLV
3E40 50 4F 4B 45 D6 50 45 45 4B 28 D6 44 4F 4B 45 D6 POKEVPEEK(VDOKEV
3E50 44 45 45 4B 28 C2 45 45 50 C2 49 4E 24 28 D2 53 DEEK(BEEPBIN$(RS
3E60 54 CB 45 59 C1 44 43 28 C2 54 4E 28 D0 53 57 C8 TKEYADC(BTN(PSWH
3E70 4F 4D 45 C1 53 4E 28 C1 43 53 28 C8 53 4E 28 C8 OMEASN(ACS(HSN(H
3E80 43 53 28 C8 54 4E 28 CC 4F 43 28 D5 43 53 24 28 CS(HTN(LOC(UCS$(
3E90 80 3E 1E C3 00 26 CD 07 3F CD 87 1D CD 94 1D CD >.C.&M.?M..M..M
3EA0 D4 1B 21 FC 1F CD 9D 1A CD FD 1D C1 D1 3A 83 01 T.!!.M..M).AQ:..
3EB0 B7 28 0C CD 26 1C C3 8A 1E CD 07 3F CD 99 3E 21 7(.M&.C..M.?M.>!
3EC0 3A 20 C3 9D 1A CD 07 3F CD FB 3E CD A3 1A 21 83 : C..M.?M(>M#.!.
3ED0 01 7E B7 C8 35 C9 CD 07 3F CD FB 3E CD A6 1A 18 .~7H5IM.?M(>M&..
3EE0 ED CD 07 3F CD 1B 3F CD 4A 1E 21 FC 1F E5 CD 98 mM.?M.?MJ.!!.eM.
3EF0 1A CD 13 3F CD 1B 3F E1 C3 9D 1A CD 4A 1E CD 87 .M.?M.?aC..MJ.M.
3F00 1D CD 13 3F C1 D1 C9 E1 E3 23 CD 83 16 11 90 02 .M.?AQIac#M.....
3F10 E3 D5 E9 01 00 81 51 59 C3 26 1C 21 83 01 7E B7 cUi...QYC&.!..~7
3F20 C8 34 C0 1E 06 C3 CD 06 E1 23 CD B3 18 E5 3A 66 H4@..CM.a#M3.e:f
3F30 01 B7 EB 28 0A 23 23 7E 23 66 6F AF 32 66 01 CD .7k(.##~#fo/2f.M
3F40 DA 04 C3 90 02 E1 23 CD 9C 16 CD 91 02 E5 CD 92 Z.C..a#M..M..eM.
3F50 14 2B 2B 2B 7E D5 CD B6 12 E1 47 7E CD CF 02 12 .+++~UM6.aG~MO..
3F60 13 23 10 F7 C3 E9 12 00 00 00 00 00 00 00 00 00 .#.wCi..........
3F70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3F90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FA0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FB0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3FF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
>
```

EXTR
WO

MACH
ROUTI
EXTR

HOME
ASNL
ACSL
HSNL
HCSL
HTNL
LOCL
UCSL

```
DISP XWORDS.MAC
;*********************************************;
;* RESERVED WORD EXTENSIONS TO XTAL BASIC 4.2  *;
;*            J.RANGELEY MARCH 1985            *;
;*             TATUNG (UK) LIMITED             *;
;*********************************************;

        .Z80
        ASEG

;* EQUATES

PR      EQU     2600H
SQR     EQU     1DFDH
DIV1    EQU     1C26H
MULT1   EQU     1BD4H
ATN     EQU     1E8AH
SUBN    EQU     1A9DH
SUB1    EQU     1AA3H
ADD1    EQU     1AA6H
ASC1    EQU     1492H
EXP     EQU     1E4AH
STKFPA  EQU     1D87H
EXNMCK  EQU     1683H
FNEND   EQU     0298H
ERROR   EQU     06CDH
LOFPR   EQU     1D94H
ADDN    EQU     1A98H
FNDVAR  EQU     1883H
FORMPOS EQU     04DAH
EXPR    EQU     169CH
RPARN   EQU     0291H
ASNSTR  EQU     1286H
LWRTST  EQU     02CFH
STREND  EQU     12E9H
;
NTYPE   EQU     0166H
FPA     EQU     0180H
HALFPI  EQU     203AH
ONE     EQU     1FFCH

        ORG     38F7H                   ; START OF USER AUX ADDR TABLE
        DW      HOME
        DW      ASN
        DW      ACS
        DW      HSN
        DW      HCS
        DW      HTN
        DW      LOC
        DW      UCS$

        ORG     3E00H                   ; START OF AUX CMD/FN TABLE
        DB      19H+8                   ; NUMBER OF WORDS

        ORG     3E6FH                   ; START OF USER AUX CMD/FN TABLE
        DC      'H'
        DB      'OME'                   ; HOME CURSOR
        DC      'A'
        DB      'SN('                   ; ARC SIN
        DC      'A'
```

```
        DB      'CS('                   ; ARC COSINE
        DC      'H'
        DB      'SN('                   ; SINH
        DC      'H'
        DB      'CS('                   ; COSH
        DC      'H'
        DB      'TN('                   ; TANH
        DC      'L'
        DB      'DC('                   ; LOCATION OF VARIABLE
        DC      'U'
        DB      'CS$('                  ; LOWER CASE TO UPPER CASE
        DB      80H                     ; END OF TABLE

; MACHINE CODE ROUTINES

HOME:   LD      A,3B                    ; <HOME> CODE IN A
        JP      PR                      ; OUTPUT IT

ASN:    CALL    TFN                     ; ASN(X)
ASN1:   CALL    STKFPA                  ; STACK X
        CALL    LDFPR
        CALL    MULT1                   ; X^2
        LD      HL,ONE
        CALL    SUBN                    ; 1-X^2
        CALL    SQR                     ; SQR(1-X^2)
        POP     BC
        POP     DE                      ; UNSTACK X
        LD      A,(FPA+3)               ; SPECIAL CASE FOR ASN(1)=PI/2
        OR      A
        JR      Z,ACS2
        CALL    DIV1                    ; X/SQR(1-X^2)
        JP      ATN                     ; ATN(X/SQR(1-X^2))

ACS:    CALL    TFN
ACS1:   CALL    ASN1
ACS2:   LD      HL,HALFPI
        JP      SUBN                    ; PI/2-ASN(X)

HSN:    CALL    TFN
HSN1:   CALL    HSN2
        CALL    SUB1                    ; EXP(X)-EXP(-X)
HALVE:  LD      HL,FPA+3                ; DIVIDE-BY-2 BY JUST
        LD      A,(HL)                  ; DECREMENTING EXPONENT
        OR      A
        RET     Z                       ; NOT IF FPA=0
        DEC     (HL)
        RET

HCS:    CALL    TFN
HCS1:   CALL    HSN2
        CALL    ADD1                    ; EXP(X)+EXP(-X)
        JR      HALVE

HTN:    CALL    TFN
HTN1:   CALL    DOUBLE                  ; X^2
        CALL    EXP                     ; EXP(X^2)
        LD      HL,ONE
        PUSH    HL
        CALL    ADDN                    ; 1+EXP(X^2)
        CALL    RECIP                   ; 1/(1+EXP(X^2))
```

```
        CALL    DOUBLE          ; 2/(1+EXP(X^2))
        POP     HL
        JP      SUBN            ; 1-2/(1+EXP(X^2))
HSN2:   CALL    EXP             ; SET EXP(X) AND EXP(-X)
        CALL    STKFPA
        CALL    RECIP           ; DO EXP(-X) AS 1/EXP(X)
        POP     BC
        POP     DE
        RET

TFN:    POP     HL              ; ROUTINE TO EVALUATE THE
        EX      (SP),HL         ; EXPRESSION BETWEEN THE
        INC     HL              ; BRACKETS, FOR USER-DEFINED
        CALL    EXNMCK          ; FUNCTIONS.
        LD      DE,FNEND        ; WILL EVENTUALLY RETURN TO
        EX      (SP),HL         ; FNEND
        PUSH    DE
        JP      (HL)            ; JUMP TO RETURN ADDRESS

RECIP:  LD      BC,8100H        ; CALCULATE RECIPROCAL
        LD      D,C
        LD      E,C             ; FPR=1
        JP      DIV1

DOUBLE: LD      HL,FPA+3        ; DOUBLE FPA BY INCREMENTING
        LD      A,(HL)          ; EXPONENT
        OR      A
        RET     Z               ; NOT IF FPA=0!
        INC     (HL)
        RET     NZ
        LD      E,06
        JP      ERROR           ; OVERFLOW IF EXPONENT=FF

LOC:    POP     HL
        INC     HL
        CALL    FNDVAR
        PUSH    HL
        LD      A,(NTYPE)
        OR      A
        EX      DE,HL
        JR      Z,LOC1
        INC     HL              ; IF STRING, GET ACTUAL STRING
        INC     HL              ; ADDRESS, NOT JUST POINTER
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        XOR     A
        LD      (NTYPE),A
LOC1:   CALL    FORMPOS
        JP      FNEND

UCS$:   POP     HL
        INC     HL
        CALL    EXPR
        CALL    RPARN           ; GET CLOSING BRACKET
        PUSH    HL              ; AND PUSH TEXT POINTER
        CALL    ASC1
        DEC     HL
        DEC     HL
```

```
        DEC     HL
        LD      A,(HL)
        PUSH    DE
        CALL    ASNSTR          ; MAKE NEW STRING
        POP     HL
        LD      B,A
UC1:    LD      A,(HL)
        CALL    LWRTST          ; CONVERT LOWER-CASE LETTER
        LD      (DE),A          ; AND PLACE IN NEW STRING
        INC     DE
        INC     HL
        DJNZ    UC1
        JP      STREND

        END

0:
```

```
                        ;*********************************************;
                        ;* RESERVED WORD EXTENSIONS TO XTAL BASIC 4.2  *;
                        ;*            J.RANGELEY MARCH 1985            *;
                        ;*             TATUNG (UK) LIMITED             *;
                        ;*********************************************;

                              .Z80
0000'                         ASEG

                        ;* EQUATES

2600                    PR      EQU     2600H
1DFD                    SQR     EQU     1DFDH
1C26                    DIV1    EQU     1C26H
1BD4                    MULT1   EQU     1BD4H
1E8A                    ATN     EQU     1E8AH
1A9D                    SUBN    EQU     1A9DH
1AA3                    SUB1    EQU     1AA3H
1AA6                    ADD1    EQU     1AA6H
1492                    ASC1    EQU     1492H
1E4A                    EXP     EQU     1E4AH
1D87                    STKFPA  EQU     1D87H
1683                    EXNMCK  EQU     1683H
0290                    FNEND   EQU     0290H
06CD                    ERROR   EQU     06CDH
1D94                    LDFPR   EQU     1D94H
1A98                    ADDN    EQU     1A98H
18B3                    FNDVAR  EQU     18B3H
04DA                    FORMPOS EQU     04DAH
169C                    EXPR    EQU     169CH
0291                    RPARN   EQU     0291H
12B6                    ASNSTR  EQU     12B6H
02CF                    LWRTST  EQU     02CFH
12E9                    STREND  EQU     12E9H
                        ;
0166                    NTYPE   EQU     0166H
0180                    FPA     EQU     0180H
203A                    HALFPI  EQU     203AH
1FFC                    ONE     EQU     1FFCH

                              ORG     3BF7H             ; START OF USER AUX ADDR TABLE
3BF7    3E91                  DW      HOME
3BF9    3E96                  DW      ASN
3BFB    3EB9                  DW      ACS
3BFD    3EC5                  DW      HSN
3BFF    3ED6                  DW      HCS
3C01    3EE1                  DW      HTN
3C03    3F28                  DW      LOC
3C05    3F45                  DW      UCS$

                              ORG     3E00H             ; START OF AUX CMD/FN TABLE
3E00    21                    DB      19H+8             ; NUMBER OF WORDS

                              ORG     3E6FH             ; START OF USER AUX CMD/FN TABLE
3E6F    C8                    DC      'H'
3E70    4F 4D 45              DB      'OME'             ; HOME CURSOR
```

```
3E73  C1                  DC    'A'
3E74  53 4E 28            DB    'SN('          ; ARC SIN
3E77  C1                  DC    'A'
3E78  43 53 28            DB    'CS('          ; ARC COSINE
3E7B  C8                  DC    'H'
3E7C  53 4E 28            DB    'SN('          ; SINH
3E7F  C8                  DC    'H'
3E80  43 53 28            DB    'CS('          ; COSH
3E83  C8                  DC    'H'
3E84  54 4E 28            DB    'TN('          ; TANH
3E87  CC                  DC    'L'
3E88  4F 43 28            DB    'OC('          ; LOCATION OF VARIABLE
3E8B  D5                  DC    'U'
3E8C  43 53 24 28         DB    'CS$('         ; LOWER CASE TO UPPER CASE
3E90  80                  DB    80H            ; END OF TABLE

                          ; MACHINE CODE ROUTINES

3E91  3E 1E         HOME: LD    A,30           ; <HOME> CODE IN A
3E93  C3 2600             JP    PR             ; OUTPUT IT

3E96  CD 3FB7       ASN:  CALL  TFN            ; ASN(X)
3E99  CD 1D87       ASN1: CALL  STKFPA         ; STACK X
3E9C  CD 1D94             CALL  LDFPR
3E9F  CD 18D4             CALL  MULT1          ; X^2
3EA2  21 1FFC             LD    HL,ONE
3EA5  CD 1A9D             CALL  SUBN           ; 1-X^2
3EA8  CD 1DFD             CALL  SQR            ; SQR(1-X^2)
3EAB  C1                  POP   BC
3EAC  D1                  POP   DE             ; UNSTACK X
3EAD  3A 8183             LD    A,(FPA+3)      ; SPECIAL CASE FOR ASN(1)=PI/2
3EB0  B7                  OR    A
3EB1  28 0C               JR    Z,ACS2
3EB3  CD 1C26             CALL  DIV1           ; X/SQR(1-X^2)
3EB6  C3 1E8A             JP    ATN            ; ATN(X/SQR(1-X^2))

3EB9  CD 3F87       ACS:  CALL  TFN
3EBC  CD 3E99       ACS1: CALL  ASN1
3EBF  21 203A       ACS2: LD    HL,HALFPI
3EC2  C3 1A9D             JP    SUBN           ; PI/2-ASN(X)

3EC5  CD 3F87       HSN:  CALL  TFN
3EC8  CD 3EFB       HSN1: CALL  HSN2
3ECB  CD 1AA3             CALL  SUB1           ; EXP(X)-EXP(-X)
3ECE  21 8183       HALVE: LD   HL,FPA+3       ; DIVIDE-BY-2 BY JUST
3ED1  7E                  LD    A,(HL)         ; DECREMENTING EXPONENT
3ED2  B7                  OR    A
3ED3  C8                  RET   Z              ; NOT IF FPA=0
3ED4  35                  DEC   (HL)
3ED5  C9                  RET

3ED6  CD 3F87       HCS:  CALL  TFN
3ED9  CD 3EFB       HCS1: CALL  HSN2
3EDC  CD 1AA6             CALL  ADD1           ; EXP(X)+EXP(-X)
3EDF  18 ED               JR    HALVE
```

```
3EE1  CD 3F07          HTN:   CALL   TFN
3EE4  CD 3F1B          HTN1:  CALL   DOUBLE          ; X^2
3EE7  CD 1E4A                 CALL   EXP             ; EXP(X^2)
3EEA  21 1FFC                 LD     HL,ONE
3EED  E5                      PUSH   HL
3EEE  CD 1A98                 CALL   ADDN            ; 1+EXP(X^2)
3EF1  CD 3F13                 CALL   RECIP           ; 1/(1+EXP(X^2))
3EF4  CD 3F1B                 CALL   DOUBLE          ; 2/(1+EXP(X^2))
3EF7  E1                      POP    HL
3EF8  C3 1A9D                 JP     SUBN            ; 1-2/(1+EXP(X^2))
3EFB  CD 1E4A          HSN2:  CALL   EXP             ; GET EXP(X) AND EXP(-X)
3EFE  CD 1D87                 CALL   STKFPA
3F01  CD 3F13                 CALL   RECIP           ; DO EXP(-X) AS 1/EXP(X)
3F04  C1                      POP    BC
3F05  D1                      POP    DE
3F06  C9                      RET

3F07  E1               TFN:   POP    HL              ; ROUTINE TO EVALUATE THE
3F08  E3                      EX     (SP),HL         ; EXPRESSION BETWEEN THE
3F09  23                      INC    HL              ; BRACKETS, FOR USER-DEFINED
3F0A  CD 1683                 CALL   EXNMCK          ; FUNCTIONS.
3F0D  11 0298                 LD     DE,FNEND        ; WILL EVENTUALLY RETURN TO
3F10  E3                      EX     (SP),HL         ; FNEND
3F11  D5                      PUSH   DE
3F12  E9                      JP     (HL)            ; JUMP TO RETURN ADDRESS

3F13  01 8100          RECIP: LD     BC,8100H        ; CALCULATE RECIPROCAL
3F16  51                      LD     D,C
3F17  59                      LD     E,C             ; FPR=1
3F18  C3 1C26                 JP     DIV1

3F1B  21 0183          DOUBLE: LD    HL,FPA+3        ; DOUBLE FPA BY INCREMENTING
3F1E  7E                      LD     A,(HL)          ; EXPONENT
3F1F  B7                      OR     A
3F20  C8                      RET    Z               ; NOT IF FPA=0!
3F21  34                      INC    (HL)
3F22  C0                      RET    NZ
3F23  1E 06                   LD     E,06
3F25  C3 06CD                 JP     ERROR           ; OVERFLOW IF EXPONENT=FF

3F28  E1               LOC:   POP    HL
3F29  23                      INC    HL
3F2A  CD 1883                 CALL   FNDVAR
3F2D  E5                      PUSH   HL
3F2E  3A 8166                 LD     A,(NTYPE)
3F31  B7                      OR     A
3F32  E8                      EX     DE,HL
3F33  28 0A                   JR     Z,LOC1
3F35  23                      INC    HL              ; IF STRING, GET ACTUAL STRING
3F36  23                      INC    HL              ; ADDRESS, NOT JUST POINTER
3F37  7E                      LD     A,(HL)
3F38  23                      INC    HL
3F39  66                      LD     H,(HL)
3F3A  6F                      LD     L,A
3F3B  AF                      XOR    A
3F3C  32 8166                 LD     (NTYPE),A
```

```
3F3F  CD 04DA          LOC1:  CALL   FORMPOS
3F42  C3 0290                 JP     FNEND

3F45  E1               UCS$:  POP    HL
3F46  23                      INC    HL
3F47  CD 169C                 CALL   EXPR
3F4A  CD 0291                 CALL   RPARN          ; GET CLOSING BRACKET
3F4D  E5                      PUSH   HL             ; AND PUSH TEXT POINTER
3F4E  CD 1492                 CALL   ASC1
3F51  2B                      DEC    HL
3F52  2B                      DEC    HL
3F53  2B                      DEC    HL
3F54  7E                      LD     A,(HL)
3F55  D5                      PUSH   DE
3F56  CD 12B6                 CALL   ASNSTR         ; MAKE NEW STRING
3F59  E1                      POP    HL
3F5A  47                      LD     B,A
3F5B  7E               UC1:   LD     A,(HL)
3F5C  CD 02CF                 CALL   LWRTST         ; CONVERT LOWER-CASE LETTER
3F5F  12                      LD     (DE),A         ; AND PLACE IN NEW STRING
3F60  13                      INC    DE
3F61  23                      INC    HL
3F62  10 F7                   DJNZ   UC1
3F64  C3 12E9                 JP     STREND

                              END
```

Macros:

Symbols:

3EB9	ACS	3EBC	ACS1	3EBF	ACS2
1AA6	ADD1	1A98	ADDN	1492	ASC1
3E96	ASN	3E99	ASN1	1286	ASNSTR
1EBA	ATN	1C26	DIV1	3F1B	DOUBLE
06CD	ERROR	16B3	EXNMCK	1E4A	EXP
169C	EXPR	18B3	FNDVAR	0290	FNEND
04DA	FORMPOS	0180	FPA	283A	HALFPI
3ECE	HALVE	3ED6	HCS	3ED9	HCS1
3E91	HOME	3EC5	HSN	3EC8	HSN1
3EFB	HSN2	3EE1	HTN	3EE4	HTN1
1D94	LDFPR	3F28	LOC	3F3F	LOC1
02CF	LNRTST	1BD4	MULT1	0166	NTYPE
1FFC	ONE	2680	PR	3F13	RECIP
0291	RPARN	1DFD	SQR	1D87	STKFPA
12E9	STREND	1AA3	SUB1	1A9D	SUBN
3F07	TFN	3F5B	UC1	3F45	UCS#

No Fatal error(s)

0:

```
 10 REM PROGRAM TO MODIFY BASIC V4.12
 20 REM TO ALLOW ADDITION OF USER RESERVED WORDS
 30 RST:BCOL1:TCOL15,0
 40 PRINT@6,2;"BASIC MODIFICATION PROGRAM"
 50 CLEAR &6100
 60 REN"XBAS.COM"TO"XBAS.OBJ"
 70 PRINT@0,4;"LOADING BASIC"
 80 LOAD "XBAS.OBJ"
 90 REN"XBAS.OBJ"TO"XBAS.COM"
100 PRINT@0,6;"BASIC LOADED"
110 :
120 REM CLEAR NEW RAM AREA
130 FOR I=0 TO 511:POKE &9E00+I,0:NEXT
140 :
150 REM MOVE COMMAND/FUNCTION TABLE TO NEW RAM
160 FOR I=0 TO 111:POKE &9E00+I,PEEK(&9BC4+I):NEXT
170 :
180 REM MOVE ADDRESS TABLE TO OLD WORD TABLE
190 FOR I=0 TO 49:POKE &9BC4+I,PEEK(&9C34+I):NEXT
200 :
210 REM SET UP NEW POINTERS IN HARD SCRATCH
220 DOKE &8B8A,&4001:REM NEW START OF BASIC
230 DOKE &8B8E,&3E00:REM NEW AUX COMMAND/FUNCTION
240 DOKE &8B98,&3BC4:REM NEW ADDRESS TABLE
250 :
260 REM CLEAR OLD AUXILIARY TABLE
270 FOR I=0 TO 111:POKE &9BF6+I,0:NEXT
280 :
290 PRINT@0,8;"ENTER VER.NO. SUFFIX(1 TO 3 DIGITS) ";:INPUT X$
300 IF LEN(X$)>3 THEN GOTO 290:ELSE POKE &9DBB,ASC(X$)
310 PRINT@0,10;"ENTER FILE NAME OF NEW BASIC ";:INPUT F$
320 IF F$="" THEN F$="NEWXBAS"
330 G$=F$+".OBJ":H$=F$+".COM"
340 PRINT@0,12;"SAVING NEW BASIC"
350 SAVE G$,&6100,&9FFF
360 REN G$ TO H$
370 PRINT@0,14;"NEW BASIC SAVED"
380 BCOL4
390 END
```

xBAS V 4.12
BEFORE RUNNING xB

```
T 3B90 3CFF 10
3B90 41 4E D6 41 4C CC 45 46 54 24 CD 49 44 24 D2 49 ANVALLEFT$MID$RI
3BA0 47 48 54 24 C5 52 52 C5 52 4C C5 4F 46 C6 4E C9 GHT$ERRERLEOFFNI
3BB0 4E 43 48 CB 42 44 CD 55 4C 24 CE 4F 54 D0 49 D3 NCHKBDMUL$NOTPIS
3BC0 49 5A 45 00 49 C4 52 41 57 D4 43 4F 4C C7 43 4F IZE..DRAWTCOLGCO
3BD0 4C C2 43 4F 4C D3 50 52 49 54 45 CD 41 47 D3 48 LBCOLSPRITEMAGSH
3BE0 41 50 45 CF 52 49 47 49 4E C5 4C 4C 49 50 53 45 APEORIGINELLIPSE
3BF0 C4 4F 53 D2 41 44 28 C4 45 47 28 D0 4F 4C 59 C6 DOSRAD(DEG(POLYF
3C00 49 4C 4C D6 50 4F 4B 45 D6 50 45 45 4B 28 D6 44 ILLVPOKEVPEEK(VD
3C10 4F 4B 45 D6 44 45 45 4B 28 C2 45 45 50 C2 49 4E OKEVDEEK(BEEPBIN
3C20 24 28 D2 53 54 CB 45 59 C1 44 43 28 C2 54 4E 28 $(RSTKEYADC(BTN(
3C30 D0 53 57 00 FD 2C 94 2C 99 2C 85 2C A4 2E 95 2E PSW.},.,.,.,$...
3C40 51 2E D7 2C 7D 2D 00 00 C7 34 D9 34 65 2D 14 2E Q.W,}-..G4Y4e-..
3C50 7A 34 A7 34 8C 34 B4 34 62 30 E6 34 E1 2B 89 2F z4'4.444b0f4a+./
3C60 1D 2C 35 2C F8 35 26 10 D8 0A 0C 0B 8B 29 4A 2C .,5,x5&.X....)J,
3C70 68 0E 85 29 81 0E 80 0C 91 10 A7 18 4B 04 1E 2A h..).....'.K..*
3C80 81 0E 70 0A 71 0D 88 0B 9A 0B CB 10 29 0E 94 0E ..p.q.....K.)...
3C90 2D 0C 55 09 43 28 F3 10 D9 2B 94 0A 09 0D A1 06 -.U.C(s.Y+....!.
3CA0 E1 0B 7F 29 7C 04 3B 04 B8 0B A1 0F B9 0E 81 0E a..)|.;.8.!.9...
3CB0 69 11 E3 2C 4D 0E CC 0B E7 0A 94 28 E5 2C 63 0A i.c,M.L.g..(e,c.
3CC0 5F 1A D4 28 82 04 70 06 48 36 74 36 49 37 1D 36 _.T(..p.H6t6I7.6
3CD0 0C 37 36 36 32 31 62 04 2A 06 20 04 9D 05 FB 05 .76621b.*. ...{.
3CE0 2C 04 07 06 14 06 AB 30 D7 30 F1 30 76 30 6E 04 ,.....+0W0q0v0n.
3CF0 4C 06 27 04 CB 05 02 06 34 04 0E 06 1C 06 96 30 L.'.K...4......0
>
```

AFTER RUNNING +BAS.MOD

```
T 3BB0 3CBF 10
3BB0 4E 43 48 CB 42 44 CD 55 4C 24 CE 4F 54 D0 49 D3 NCHKBDMUL$NOTFIS
3BC0 49 5A 45 80 FD 2C 94 2C 99 2C 85 2C A4 2E 95 2E IZE.),.,.,.,$...
3BD0 51 2E D7 2C 7D 2D 00 00 C7 34 D9 34 65 2D 14 2E Q.W,}-..G4Y4e-..
3BE0 7A 34 A7 34 8C 34 B4 34 62 30 E6 34 E1 2B 89 2F z4'4.4.4b0f4a+./
3BF0 1D 2C 35 2C F8 35 00 00 00 00 00 00 00 00 00 00 .,5,x5..........
3C00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C60 00 00 00 00 00 00 26 10 D8 0A 0C 0B 8B 29 4A 2C ......&.X....)J,
3C70 68 0E 85 29 81 0E 80 0C 91 10 A7 18 4B 04 1E 2A h..).......'.K..*
3C80 81 0E 70 0A 71 0D 88 0B 9A 0B CB 10 29 0E 94 0E ..p.q.....K.)...
3C90 2D 0C 55 09 43 28 F3 10 D9 2B 94 0A 09 0D A1 06 -.U.C(s.Y+....!.
3CA0 E1 0B 7F 29 7C 04 3B 04 B8 0B A1 0F B9 0E 81 0E a..)|.;.8.!.9...
3CB0 69 11 E3 2C 4D 0E CC 0B E7 0A 94 28 E5 2C 63 0A i.c,M.L.g..(e,c.
>
```

```
T 3E00 3EFF 10
3E00 19 C4 52 41 57 D4 43 4F 4C C7 43 4F 4C C2 43 4F .DRAWTCOLGCOLBCO
3E10 4C D3 50 52 49 54 45 CD 41 47 D3 48 41 50 45 CF LSPRITEMAGSHAPEO
3E20 52 49 47 49 4E C5 4C 4C 49 50 53 45 C4 4F 53 D2 RIGINELLIPSEDOSR
3E30 41 44 28 C4 45 47 28 D0 4F 4C 59 C6 49 4C 4C D6 AD(DEG(POLYFILLV
3E40 50 4F 4B 45 D6 50 45 45 4B 28 D6 44 4F 4B 45 D6 POKEVPEEK(VDOKEV
3E50 44 45 45 4B 28 C2 45 45 50 C2 49 4E 24 28 D2 53 DEEK(BEEPBIN$(RS
3E60 54 CB 45 59 C1 44 43 28 C2 54 4E 28 D0 53 57 80 TKEYADC(BTN(PSW.
3E70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3E80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3E90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EA0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EB0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3ED0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3EF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
>
```

(4.12)

AFTER INSERTING USER ROUTINES +BASIC V4.

```
T 3BB0 3CBF 10
3BB0 4E 43 48 CB 42 44 CD 55 4C 24 CE 4F 54 D0 49 D3 NCHKBDMUL$NOTPIS
3BC0 49 5A 45 80 FD 2C 94 2C 99 2C 85 2C A4 2E 95 2E IZE.),.,.,.,$...
3BD0 51 2E D7 2C 7D 2D 00 00 C7 34 D9 34 65 2D 14 2E Q.W,}-..G4Y4e-..
3BE0 7A 34 A7 34 8C 34 B4 34 62 30 E6 34 E1 2B 89 2F z4'4.444b0f4a+./
3BF0 1D 2C 35 2C F8 35 91 3E 96 3E 89 3E C5 3E D6 3E .,5,x5.>.>9>E>V>
3C00 E1 3E 2B 3F 45 3F 00 00 00 00 00 00 00 00 00 00 a>(?E?..........
3C10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
3C60 00 00 00 00 00 00 26 10 D8 0A 0C 0B 8B 29 4A 2C ......&.X....)J,
3C70 68 0E 85 29 81 0E 80 0C 91 10 A7 18 4B 04 1E 2A h..).......'.K..*
3C80 81 0E 70 0A 71 0D 88 0B 9A 0B CB 10 29 0E 94 0E ..p.q.....K.)...
3C90 2D 0C 55 09 43 28 F3 10 D9 2B 94 0A 09 0D A1 06 -.U.C(s.Y+....!.
3CA0 E1 0B 7F 29 7C 04 3B 04 B8 0B A1 0F B9 0E 81 0E a..)!.;.8.!.9...
3CB0 69 11 E3 2C 4D 0E CC 0B E7 0A 94 28 E5 2C 63 0A i.c,M.L.g..(e,c.
>


T 3E00 3F7F 10
3E00 2F C4 52 41 57 D4 43 4F 4C C7 43 4F 4C C2 43 4F .DRAWTCOLGCOLBCO
3E10 4C D3 50 52 49 54 45 CD 41 47 D3 48 41 50 45 CF LSPRITEMAGSHAPEO
3E20 52 49 47 49 4E C5 4C 4C 49 50 53 45 C4 4F 53 D2 RIGINELLIPSEDOSR
3E30 41 44 28 C4 45 47 28 D0 4F 4C 59 C6 49 4C 4C D6 AD(DEG(POLYFILLV
3E40 50 4F 4B 45 D6 50 45 45 4B 28 D6 44 4F 4B 45 D6 POKEVPEEK(VDOKEV
3E50 44 45 45 4B 28 C2 45 45 50 C2 49 4E 24 28 D2 53 DEEK(BEEPBIN$(RS
3E60 54 CB 45 59 C1 44 43 28 C2 54 4E 28 D0 53 57 C8 TKEYADC(BTN(PSWH
3E70 4E 4D 41 C1 53 AE 28 CC 43 53 28 C8 53 4E 28 C8 OMEASN(ACS(HSN(H
3E80 43 53 28 C8 54 4E 28 CC 4F 43 28 D5 43 53 24 28 CS(HTN(LOC(UCS$(
3E90 80 3E 1E C3 FE 25 CD 07 3F CD 80 1D CD 8D 1D CD .>.C~%M.?M..M..M
3EA0 CD 1B 21 FA 1F CD 96 1A CD F6 1D C1 D1 3A 83 01 M.!z.M..Mv.AQ:..
3EB0 B7 28 0C CD 1F 1C C3 83 1E CD 07 3F CD 99 3E 21 7(.M.C..M.?M.>!
3EC0 38 20 C3 96 1A CD 07 3F CD FB 3E CD 9C 1A 21 83 8 C..M.?M{>M..!.
3ED0 01 7E B7 C8 35 C9 CD 07 3F CD FB 3E CD 9F 1A 18 .~7H5IM.?M{>M...
3EE0 ED CD 07 3F CD 1B 3F CD 43 1E 21 FA 1F E5 CD 91 mM.?M.?MC.!z.eM.
3EF0 1A CD 13 3F CD 1B 3F E1 C3 96 1A CD 43 1E CD 80 .M.?M.?aC..MC.M.
3F00 1D CD 13 3F C1 D1 C9 E1 E3 23 CD 7C 16 11 93 02 .M.?AQIac#M|....
3F10 E3 D5 E9 01 00 81 51 59 C3 1F 1C 21 83 01 7E B7 cUi...QYC..!..~7
3F20 C8 34 C0 1E 06 C3 D0 06 E1 23 CD AC 18 E5 3A 66 H4@..CP.a#M,.e:f
3F30 01 B7 EB 28 0A 23 23 7E 23 66 6F AF 32 66 01 CD .7k(.##~#fo/2f.M
3F40 DD 04 C3 93 02 E1 23 CD 95 16 CD 94 02 E5 CD 8B ].C..a#M..M..eM.
3F50 14 2B 2B 2B 7E D5 CD AF 12 E1 47 7E CD D2 02 12 .+++~UM/.aG~MR..
3F60 13 23 10 F7 C3 E2 12 1A 00 00 00 00 00 00 00 00 .#.wCb..........
3F70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
>
```

```
                        ;*********************************************;
                        ;* RESERVED WORD EXTENSIONS TO XTAL BASIC 4.12  *;
                        ;*            J.RANGELEY JANUARY 1985          *;
                        ;*             TATUNG (UK) LIMITED             *;
                        ;*********************************************;

                              .Z80
0000'                         ASEG

                        ;* EQUATES

25FE                    PR      EQU     25FEH
1DF6                    SQR     EQU     1DF6H
1C1F                    DIV1    EQU     1C1FH
1BCD                    MULT1   EQU     1BCDH
1E83                    ATN     EQU     1E83H
1A96                    SUBN    EQU     1A96H
1A9C                    SUB1    EQU     1A9CH
1A9F                    ADD1    EQU     1A9FH
148B                    ASC1    EQU     148BH
1E43                    EXP     EQU     1E43H
1D80                    STKFPA  EQU     1D80H
167C                    EXNMCK  EQU     167CH
0293                    FNEND   EQU     0293H
06D0                    ERROR   EQU     06D0H
1D8D                    LDFPR   EQU     1D8DH
1A91                    ADDN    EQU     1A91H
18AC                    FNDVAR  EQU     18ACH
04DD                    FORMPOS EQU     04DDH
1695                    EXPR    EQU     1695H
0294                    RPARN   EQU     0294H
12AF                    ASNSTR  EQU     12AFH
02D2                    LWRTST  EQU     02D2H
12E2                    STREND  EQU     12E2H
                        ;
0166                    NTYPE   EQU     0166H
0180                    FPA     EQU     0180H
2038                    HALFPI  EQU     2038H
1FFA                    ONE     EQU     1FFAH

                              ORG     3BF6H             ; START OF USER AUX ADDR TABLE
3BF6    3E91                  DW      HOME
3BF8    3E96                  DW      ASN
3BFA    3EB9                  DW      ACS
3BFC    3EC5                  DW      HSN
3BFE    3ED6                  DW      HCS
3C00    3EE1                  DW      HTN
3C02    3F28                  DW      LOC
3C04    3F45                  DW      UCS$

                              ORG     3E00H             ; START OF AUX CMD/FN TABLE
3E00    21                    DB      19H+8             ; NUMBER OF WORDS

                              ORG     3E6FH             ; START OF USER AUX CMD/FN TABLE
3E6F    C8                    DC      'H'
3E70    4F 4D 45              DB      'OME'             ; HOME CURSOR
```

```
3E73  C1                  DC    'A'
3E74  53 4E 28            DB    'SN('          ; ARC SIN
3E77  C1                  DC    'A'
3E78  43 53 28            DB    'CS('          ; ARC COSINE
3E7B  C8                  DC    'H'
3E7C  53 4E 28            DB    'SN('          ; SINH
3E7F  C8                  DC    'H'
3E80  43 53 28            DB    'CS('          ; COSH
3E83  C8                  DC    'H'
3E84  54 4E 28            DB    'TN('          ; TANH
3E87  CC                  DC    'L'
3E88  4F 43 28            DB    'OC('          ; LOCATION OF VARIABLE
3E8B  D5                  DC    'U'
3E8C  43 53 24 28         DB    'CS$('         ; LOWER CASE TO UPPER CASE
3E90  80                  DB    80H            ; END OF TABLE

                    ; MACHINE CODE ROUTINES

3E91  3E 1E         HOME: LD    A,30           ; <HOME> CODE IN A
3E93  C3 25FE             JP    PR             ; OUTPUT IT

3E96  CD 3F07       ASN:  CALL  TFN            ; ASN(X)
3E99  CD 1D80       ASN1: CALL  STKFPA         ; STACK X
3E9C  CD 1D8D             CALL  LDFPR
3E9F  CD 1BCD             CALL  MULT1          ; X^2
3EA2  21 1FFA             LD    HL,ONE
3EA5  CD 1A96             CALL  SUBN           ; 1-X^2
3EA8  CD 1DF6             CALL  SQR            ; SQR(1-X^2)
3EAB  C1                  POP   BC
3EAC  D1                  POP   DE             ; UNSTACK X
3EAD  3A 0183             LD    A,(FPA+3)      ; SPECIAL CASE FOR ASN(1)=PI/2
3EB0  B7                  OR    A
3EB1  28 0C               JR    Z,ACS2
3EB3  CD 1C1F             CALL  DIV1           ; X/SQR(1-X^2)
3EB6  C3 1E83             JP    ATN            ; ATN(X/SQR(1-X^2))

3EB9  CD 3F07       ACS:  CALL  TFN
3EBC  CD 3E99       ACS1: CALL  ASN1
3EBF  21 2830       ACS2: LD    HL,HALFPI
3EC2  C3 1A96             JP    SUBN           ; PI/2-ASN(X)

3EC5  CD 3F07       HSN:  CALL  TFN
3EC8  CD 3EFB       HSN1: CALL  HSN2
3ECB  CD 1A9C             CALL  SUB1           ; EXP(X)-EXP(-X)
3ECE  21 0183       HALVE: LD   HL,FPA+3       ; DIVIDE-BY-2 BY JUST
3ED1  7E                  LD    A,(HL)         ; DECREMENTING EXPONENT
3ED2  B7                  OR    A
3ED3  C8                  RET   Z              ; NOT IF FPA=0
3ED4  35                  DEC   (HL)
3ED5  C9                  RET

3ED6  CD 3F07       HCS:  CALL  TFN
3ED9  CD 3EFB       HCS1: CALL  HSN2
3EDC  CD 1A9F             CALL  ADD1           ; EXP(X)+EXP(-X)
3EDF  18 ED               JR    HALVE
```

```
3EE1  CD 3F07          HTN:   CALL   TFN
3EE4  CD 3F18          HTN1:  CALL   DOUBLE          ; X^2
3EE7  CD 1E43                 CALL   EXP             ; EXP(X^2)
3EEA  21 1FFA                 LD     HL,ONE
3EED  E5                      PUSH   HL
3EEE  CD 1A91                 CALL   ADDN            ; 1+EXP(X^2)
3EF1  CD 3F13                 CALL   RECIP           ; 1/(1+EXP(X^2))
3EF4  CD 3F18                 CALL   DOUBLE          ; 2/(1+EXP(X^2))
3EF7  E1                      POP    HL
3EF8  C3 1A96                 JP     SUBN            ; 1-2/(1+EXP(X^2))
3EFB  CD 1E43          HSN2:  CALL   EXP             ; GET EXP(X) AND EXP(-X)
3EFE  CD 1D80                 CALL   STKFPA
3F01  CD 3F13                 CALL   RECIP           ; DO EXP(-X) AS 1/EXP(X)
3F04  C1                      POP    BC
3F05  D1                      POP    DE
3F06  C9                      RET

3F07  E1               TFN:   POP    HL              ; ROUTINE TO EVALUATE THE
3F08  E3                      EX     (SP),HL         ; EXPRESSION BETWEEN THE
3F09  23                      INC    HL              ; BRACKETS, FOR USER-DEFINED
3F0A  CD 167C                 CALL   EXNMCK          ; FUNCTIONS.
3F0D  11 0293                 LD     DE,FNEND        ; WILL EVENTUALLY RETURN TO
3F10  E3                      EX     (SP),HL         ; FNEND
3F11  D5                      PUSH   DE
3F12  E9                      JP     (HL)            ; JUMP TO RETURN ADDRESS

3F13  01 8100          RECIP: LD     BC,8100H        ; CALCULATE RECIPROCAL
3F16  51                      LD     D,C
3F17  59                      LD     E,C             ; FPR=1
3F18  C3 1C1F                 JP     DIV1

3F1B  21 0183          DOUBLE: LD    HL,FPA+3        ; DOUBLE FPA BY INCREMENTING
3F1E  7E                      LD     A,(HL)          ; EXPONENT
3F1F  B7                      OR     A
3F20  C8                      RET    Z               ; NOT IF FPA=0!
3F21  34                      INC    (HL)
3F22  C0                      RET    NZ
3F23  1E 06                   LD     E,06
3F25  C3 06DB                 JP     ERROR           ; OVERFLOW IF EXPONENT=FF

3F28  E1               LOC:   POP    HL
3F29  23                      INC    HL
3F2A  CD 18AC                 CALL   FNDVAR
3F2D  E5                      PUSH   HL
3F2E  3A 0166                 LD     A,(NTYPE)
3F31  B7                      OR     A
3F32  E8                      EX     DE,HL
3F33  28 0A                   JR     Z,LOC1
3F35  23                      INC    HL              ; IF STRING, GET ACTUAL STRING
3F36  23                      INC    HL              ; ADDRESS, NOT JUST POINTER
3F37  7E                      LD     A,(HL)
3F38  23                      INC    HL
3F39  66                      LD     H,(HL)
3F3A  6F                      LD     L,A
3F3B  AF                      XOR    A
3F3C  32 0166                 LD     (NTYPE),A
```

```
3F3F  CD 04DD          LOC1:  CALL  FORMPOS
3F42  C3 0293                 JP    FNEND

3F45  E1               UCS$:  POP   HL
3F46  23                      INC   HL
3F47  CD 1695                 CALL  EXPR
3F4A  CD 0294                 CALL  RPARN          ; GET CLOSING BRACKET
3F4D  E5                      PUSH  HL             ; AND PUSH TEXT POINTER
3F4E  CD 1488                 CALL  ASC1
3F51  2B                      DEC   HL
3F52  2B                      DEC   HL
3F53  2B                      DEC   HL
3F54  7E                      LD    A,(HL)
3F55  D5                      PUSH  DE
3F56  CD 12AF                 CALL  ASNSTR         ; MAKE NEW STRING
3F59  E1                      POP   HL
3F5A  47                      LD    B,A
3F5B  7E               UC1:   LD    A,(HL)
3F5C  CD 02D2                 CALL  LWRTST         ; CONVERT LOWER-CASE LETTER
3F5F  12                      LD    (DE),A         ; AND PLACE IN NEW STRING
3F60  13                      INC   DE
3F61  23                      INC   HL
3F62  10 F7                   DJNZ  UC1
3F64  C3 12E2                 JP    STREND

                              END
```

Macros:

Symbols:

3EB9	ACS	3EBC	ACS1	3EBF	ACS2
1A9F	ADD1	1A91	ADDN	1488	ASC1
3E96	ASN	3E99	ASN1	12AF	ASNSTR
1E83	ATN	1C1F	DIV1	3F18	DOUBLE
06D0	ERROR	167C	EXNMCK	1E43	EXP
1695	EXPR	18AC	FNDVAR	0293	FNEND
040D	FORMPOS	0180	FPA	2038	HALFPI
3ECE	HALVE	3ED6	HCS	3ED9	HCS1
3E91	HOME	3EC5	HSN	3EC8	HSN1
3EFB	HSN2	3EE1	HTN	3EE4	HTN1
1D8D	LDFPR	3F28	LOC	3F3F	LOC1
02D2	LWRTST	1BCD	MULT1	0166	NTYPE
1FFA	ONE	25FE	PR	3F13	RECIP
0294	RPARN	1DF6	SQR	1D80	STKFPA
12E2	STREND	1A9C	SUB1	1A96	SUBN
3F07	TFN	3F5B	UC1	3F45	UCS$

No Fatal error(s)

A>

```
GO    10 REM PROGRAM TO MODIFY BASIC V4.12
      20 REM TO ALLOW ADDITION OF USER RESERVED WORDS
      30 RST:BCOL1:TCOL15,4
      40 PRINT@6,2;"BASIC MODIFICATION PROGRAM"
      50 CLEAR &6100
      60 REN"XBAS.COM"TO"XBAS.OBJ"
      70 PRINT@0,4;"LOADING BASIC"
      80 LOAD "XBAS.OBJ"
      90 REN"XBAS.OBJ"TO"XBAS.COM"
     100 PRINT@0,6;"BASIC LOADED"
     110 :
     120 REM CLEAR NEW RAM AREA
     130 FOR I=0 TO 511:POKE &9E00+I,0:NEXT
     140 :
     150 REM MOVE COMMAND/FUNCTION TABLE TO NEW RAM
     160 FOR I=0 TO 111:POKE &9E00+I,PEEK(&9BC4+I):NEXT
     170 :
     180 REM MOVE ADDRESS TABLE TO OLD WORD TABLE
     190 FOR I=0 TO 49:POKE &9BC4+I,PEEK(&9C34+I):NEXT
     200 :
     210 REM SET UP NEW POINTERS IN HARD SCRATCH
     220 DOKE &8B8A,&4001:REM NEW START OF BASIC
     230 DOKE &8B8E,&3E00:REM NEW AUX COMMAND/FUNCTION
     240 DOKE &8B98,&3BC4:REM NEW ADDRESS TABLE
     250 :
     260 REM CLEAR OLD AUXILIARY TABLE
     270 FOR I=0 TO 111:POKE &9BF6+I,0:NEXT
     280 :
     290 PRINT@0,8;"ENTER VERSION NUMBER SUFFIX(1 OR 2 DIGITS) ";:INPUT X$
     300 IF LEN(X$)>3 THEN GOTO 290:ELSE POKE &9DBB,ASC(X$)
     310 PRINT@0,10;"ENTER FILE NAME OF NEW BASIC ";:INPUT F$
     320 IF F$="" THEN F$="NEWXBAS"
     330 G$=F$+".OBJ":H$=F$+".COM"
     340 PRINT@0,12;"SAVING NEW BASIC"
     350 SAVE G$,&6100,&9FFF
     360 REN G$ TO H$
     370 PRINT@0,14;"NEW BASIC SAVED"
     380 END
```

```
 10 REM PROGRAM TO MODIFY BASIC V4.20
 20 REM TO ALLOW ADDITION OF USER RESERVED WORDS
 30 RST:TCOL15,4
 40 PRINT@6,2;"BASIC MODIFICATION PROGRAM"
 50 CLEAR &6100
 55 UNLOCK "XBAS.COM"
 60 REN"XBAS.COM"TO"XBAS.OBJ"
 70 PRINT@0,4;"LOADING BASIC"
 80 LOAD "XBAS.OBJ"
 90 REN"XBAS.OBJ"TO"XBAS.COM"
100 PRINT@0,6;"BASIC LOADED"
110 :
120 REM CLEAR NEW RAM AREA
130 FOR I=0 TO 511:POKE &9E00+I,0:NEXT
140 :
150 REM MOVE COMMAND/FUNCTION TABLE TO NEW RAM
160 FOR I=0 TO 111:POKE &9E00+I,PEEK(&9BC5+I):NEXT
170 :
180 REM MOVE ADDRESS TABLE TO OLD WORD TABLE
190 FOR I=0 TO 49:POKE &9BC4+I,PEEK(&9C35+I):NEXT
200 :
210 REM SET UP NEW POINTERS IN HARD SCRATCH
220 DOKE &8B84,&4001:REM NEW START OF BASIC
230 DOKE &8B88,&3E00:REM NEW AUX COMMAND/FUNCTION
240 DOKE &8B92,&3BC4:REM NEW ADDRESS TABLE
250 :
260 REM CLEAR OLD AUXILIARY TABLE
270 FOR I=0 TO 111:POKE &9BF7+I,0:NEXT
280 :
290 PRINT@0,8;"Enter Version No. SUFFIX: ";:INPUT X$
300 IF LEN(X$)>3 THEN GOTO 290:ELSE POKE &9DBB,ASC(X$)
310 PRINT@0,10;"ENTER NEW VERSION TITLE  ";:INPUT F$
320 IF F$="" THEN F$="XBASXT"
330 G$=F$+".OBJ":H$=F$+".COM"
340 PRINT@0,12;"SAVING NEW BASIC"
350 SAVE G$,&6100,&9FFF
360 REN G$ TO H$
370 PRINT@0,14;"NEW BASIC SAVED"
380 PRINT" Under name : ";F$
390 LOCK "XBAS.COM"
400 PRINT"File Locked."
410 PRINT"Job Done."
420 END
```

XBASX

DESCRIPTION OF EXTRA COMMANDS AVAILABLE FOR TATUNG/XTAL BASIC V4.2X

GENERAL

The following is a description of a number of Reserved word extensions that have been added to TATUNG/Xtal Basic V4.2. It should be noted that in the case of the **functions** (i.e. those words which return a value) the left parenthesis should immediately follow the last letter of the word, i.e. no space should be included. The same restriction does not apply to the right parenthesis.

Example:

PRINT ACS(.1) will return the value 1.47063
whereas:
PRINT ACS (.1) will return the value 0

In some of the examples that follow it will be noted that certain values are not absolutely accurate e.g. arc sine of 0.5 = 29.9997 degrees, where the correct value is 30 degrees. This is due to a combination of the following factors:-

1. Certain real numbers cannot be represented accurately by a binary code.

2. Trigonometric functions are evaluated by an infinite series which has to be truncated.

3. TATUNG/Xtal Basic works to an accuracy of seven significant figures and displays to six.

HOME (Home Cursor)

Syntax: HOME

Purpose: This is a command to return the cursor to the top left corner of the screen. This command is similar to the CLS command but does not perform a clear screen operation.

This command performs the same function as the following statement:

```
PRINT CHR$(30);
```

Related Keywords: CLS

ASN (Arc Sine)

Syntax: ASN(N)

Purpose: This is a function which returns the angle whose sine is N, and is expressed in radians.

Example 1:
```
PRINT ASN(0.5)
0.523599
```

Example 2:
```
PRINT DEG(ASN(0.5))
29.9997
```

Example 1 shows the angle whose sine is 0.5 is 0.523599 radians.
Example 2 shows the angle whose sine is 0.5 is 29.9997 degrees.

Related Keywords: ACS ATN

ACS (Arc Cosine)

Syntax: ACS(N)

Purpose: This is a function which returns the angle whose cosine is N, expressed in radians.

```
Example 1:    PRINT ACS(0.5)
              1.0472

Example 2:    PRINT DEG(ACS(0.5))
              59.9995
```

Example 1 shows the angle whose cosine is 0.5 is 1.0472 radians.
Example 2 shows the angle whose cosine is 0.5 is 59.9995 degrees.

Related Keywords: ASN ATN

HCS (Hyperbolic Cosine)

Syntax: HCS(N)

Purpose: This is a function which returns the hyperbolic cosine (cosh) of the argument N.

$$\cosh(x) = (EXP(x) + EXP(-x))/2$$

```
Example:      PRINT HCS(0.5)
              1.12763
```

Related Keywords: HSN HTN EXP

HSN (Hyperbolic Sine)

Syntax: HSN(N)

Purpose: This is a function which returns the hyperbolic sine (sinh) of the argument N.

sinh(x) = (EXP(x)-EXP(-x))/2

Example: PRINT HSN(0.5)
0.521095

Related Keywords: HCS HTN EXP

HTN (Hyperbolic Tangent)

Syntax: HTN(N)

Purpose: This is a function which returns the hyperbolic tangent (tanh) of the argument N.

tanh(x) = 1-2/(1+EXP(x 2))

Example: PRINT HTN(0.5)
0.462117

Related Keywords: HCS HSN EXP

LOC (Location)

Syntax: LOC(V)

Purpose: This is a function that returns the location in memory of the start of the variable V. A variable name or array element must be specified as the argument.

Example: A$ = "HELLO":PRINT LOC(A$)
59195

Related Keywords:

UCS$ (Upper Case String)

Syntax: UCS$(V$)

Purpose: This is a function which returns the string corresponding to the argument V (which is also a string) with all lower-case letters converted to upper-case.

Example: PRINT UCS$("hello")
HELLO

Related Keywords:

BAUD (Baud Rate)

Syntax: BAUD R,T

Purpose: This is a command to set the serial interface Transmit and Receive rates. The values of R and T are in the range 0 to 8 and represent the Receive and Transmit rates, respectively as in the following table.

0	75 baud	5	1200 baud
1	110 baud	6	2400 baud
2	150 baud	7	4800 baud
3	300 baud	8	9600 baud
4	600 baud		

These codes correspond to those of the 'B' command which operates in the MOS enviroment. The default values are 9600 baud transmit and receive.

Example: BAUD 5,0

This sets up the serial interface to 1200 baud receive, 75 baud transmit.

Related Keywords: MODE