This is a feature update to Carl's port of Tiny Basic. There is also a version for Run Basic which is posted in the Run Basic wikispace.

The main purpose of this release was to add enough features to Tiny Basic to run the Sieve benchmark. Here are the new features:

1. Added **seconds** and **ms** functions for benchmarking purposes. In Run Basic, however, if you use the **seconds** function, it will still display milliseconds. Try "print seconds".
2. Added one array to the interpreter environment. In addition to the numeric variables "a" through "z", there is also the numeric array "a(1)" through "a(7001). There is an existing bug in the range checking which will be fixed later. Range checking works for assignment, but not for printing.
3. For LB4 only, LOAD, SAVE and AutoRun are fixed. I also added KILL and DIR commands for file management. For the Run Basic release, this code is just commented out.

I will post results of the Sieve benchmark test in the Liberty Basic forum and on my blog. Needless to say that Tiny Basic is a lot slower than Run Basic or Liberty Basic. Regardless, it has a lot of potential as an embedded interpreter in a Run Basic or Liberty Basic application. Best of all, it's fun to play with!

Enjoy!

David den Haring.
Simple Computing Blog

```
        'TinyBasic.bas


' tinyBasic v1.1 - A very small and simple BASIC interpreter written i
n
    ' Run BASIC.  WARNING:  For uber geeks only.  :-p

    '   Copyleft 2005 by Laurent DUVEAU
    '   http://www.aldweb.com/
    ' an iziBasic sample program

    ' Ported to Run BASIC by Carl Gundel and Scott McLaughlin
    '   http://www.libertybasic.com

    ' Version 1.1 (Will run on Liberty Basic 4.03 and Run Basic 2.27)
    ' by David den Haring (8/15/2007)

' The purpose of this release is to add enough features to Tiny Basic
in order to run the Sieve benchmark.
```

```
    ' Added a fixed array accessible from the interpreter (i.e. a() )

' Added the functions "seconds" and "ms" or "milliseconds" for benchma
rking purposes

' (LB4) Fixed LOAD and SAVE commands because it's tiring to have to ty
pe in the Sieve program by hand all the time. I'm having flashbacks to
 my VIC-20 before the Datasette (i.e. cassette tape storage)

' (LB4) Added DIR and KILL commands for file management. DIR has no pa
rameters. KILL works like LOAD and SAVE.

' (LB4) Fixed AutoRun feature. If file tinyBas0 exists, it will be loa
ded and run automatically when the interpreter starts.
    ' A          temp
    ' B          temp
    ' C          character index in line
    ' E          line number for error msg
    ' I          temp (loops)
    ' L          number of lines. Index into A$ array
    ' N          number
    ' S          expression stack index
    ' T          temp
    ' V          variable index

    ' A$         temp
    ' B$         temp
    ' C$         character
    ' D$         single statement
    ' E$         error message
    ' G$         string code (")
    ' H$         HALT code (Line Feed)
    ' I$-R$      Help
    ' Z$=A$(26)  statement input

    DIM A$(125) ' [27-125] = 99 program lines
    DIM A(82)   ' [27-52] = 26 variables
    ' [53-82] = 30 items math stack

    ArraySize = 7001 'Size of user array
    DIM Array(ArraySize) 'Fixed-sized array available to user program
    DIM info$(10, 10)

    A$(9)="BYE, CLEAR, CLS, END"
    A$(10)="HELP, MEM, NEW, RUN, DIR"
    A$(11)="GOTO | LOAD | SAVE | KILL <exp>"
```

```
    A$(12)="IF <exp> THEN <statement>"
    A$(13)="INPUT <var>"
    A$(14)="[LET] <var>=<exp>"
    A$(15)="LIST [<exp>|PAUSE]"
    A$(16)="PRINT <exp|str>[,<exp|str>][;]"
    A$(17)="REM <any>"
    A$(18)="Numeric variables available from 'a' to 'z'"
    A$(19)=
"One array variable available -- a(1) to a("; STR$(ArraySize); ")"
    A$(20)="No file access for Run Basic yet!"
    A$(21)="Functions available: seconds, milliseconds or ms"
    TRUE = 1
    FALSE = 0
    arrayFlag = FALSE
'this flag indicates whether a variable is an array reference or not
    simpleFlag = FALSE
'this flag indicates a simple variable on the left side of an expressi
on

    'BEGIN
    G$=CHR$(34) : H$=CHR$(10) 'double quote and linefeed, respectively

    'if the file "tinyBas0" exists, load and run it automatically
    B = fileExists(DefaultDir$, "tinyBas0")
    IF B = TRUE THEN
        Z$="LOAD 0: RUN"
        GOTO [AutoRun]
    END IF

    PRINT "Tiny BASIC v1.1 for Run BASIC and Liberty BASIC"
    PRINT "Type HELP for commands."
[Ready] 'display any errors and show ready prompt
    simpleFlag = FALSE
    arrayFlag = FALSE
    IF E$<>"" THEN 'print the error
        IF E>0 THEN
            E$="#Err in "+STR$(E)+": "+E$
        ELSE
            E$="#Err: "+E$
        END IF
        PRINT E$ : E$=""
    END IF
    PRINT "Ready"
[Input] 'get user input
    LINE INPUT Z$
[AutoRun] 'parse the user input
```

```
    A$(26)=Z$
    L=26 : C=1 : GOSUB [GetNumber] : E=N
    IF N=0 THEN 'no line number
        IF C$="" THEN [Ready] 'user just hit the return key
        GOTO [NextStatement] 'get the next token
    ELSE 'line number
        GOSUB [EnterLine] 'enter line of code into program
        IF E$<>"" THEN [Ready] 'branch and display error
        GOTO [Input] 'get user input
    END IF
[Exec] 'get the next number
    GOSUB [GetNumber] : E=N
[NextStatement]
    'This commented-
out code is for stopping the execution of a running program
    'A$=INKEY$
    'IF A$=H$ THEN
    ' A$="Break in "+STR$(E,0)
    ' PRINT A$ : GOTO [Ready]
    'END IF
    GOSUB [GetLabel]
    IF E$<>"" THEN [Ready] 'branch and display error
    SELECT CASE D$
        CASE "if"
        GOSUB [GetExpression]
        IF E$<>"" THEN [Ready]
        IF N<1 THEN
            B$=A$(L) : C=LEN(B$)+1
            GOTO [FinishStatement]
        END IF
        GOSUB [GetLabel]
        IF E$<>"" THEN [Ready]
        IF D$<>"then" THEN
            E$="'THEN' expected"
            GOTO [Ready]
        END IF
        GOTO [NextStatement]
        CASE "rem"
        B$=A$(L) : C=LEN(B$)+1
        GOTO [FinishStatement]
        CASE "input"
        GOSUB [GetVar]
        IF E$<>"" THEN [Ready]
        INPUT N
        IF arrayFlag THEN
            Array(V)=N 'set the value of the array
```

```
        ELSE
            A(V)=N 'set the value of simple variable
        END IF
        GOTO [FinishStatement]
        CASE "print"
[Print]
        GOSUB [SkipSpace]
        GOSUB [GetChar]
        IF C$=G$ THEN 'print quoted string
            B$=""
[NextChar]
            C = C + 1 : C$=MID$(A$,C,1)
            IF C$="" THEN
                E$="Unterminated string"
                GOTO [Ready]
            ELSE
                IF C$<>G$ THEN
                    B$=B$+C$
                    GOTO [NextChar]
                END IF
            END IF
            C = C + 1 : C$=MID$(A$,C,1)
            IF C$=G$ THEN
                B$=B$+C$
                GOTO [NextChar]
            END IF
            PRINT B$;
        ELSE 'print expression
            GOSUB [GetExpression]
            IF E$<>"" THEN [Ready]
            B=0
            IF B=N THEN 'if variable is 0, then print a trailing "*"
                PRINT N;"*";
            ELSE
                PRINT N;
            END IF
        END IF
        GOSUB [SkipSpace]
        GOSUB [GetChar]
        IF C$="," THEN C = C + 1 : GOTO [Print]
        GOSUB [SkipSpace]
        GOSUB [GetChar]
        IF C$<>";" THEN
            PRINT
        ELSE
            C = C + 1
```

```
        END IF
        GOTO [FinishStatement]
        CASE "clear"
        FOR I=27 TO 52 : A(I)=0 : NEXT I 'clear variables a -- z
        FOR I = 0 TO ArraySize : Array(I) = 0: NEXT I
'clear the array -- can't REDIM becuase Run Basic doesn't support it y
et
        GOTO [FinishStatement]
        CASE "run"
        FOR I=27 TO 52 : A(I)=0 : NEXT I
        L=27 : C=1
        GOTO [FinishStatement2]
        CASE "goto"
        GOSUB [GetExpression]
        IF E$<>"" THEN [Ready]
        IF E>=N THEN L=27
        C=1 : T=N
[NextGoto]
        IF L=126 THEN
            E$="Line not found"
            GOTO [Ready]
        END IF
        GOSUB [GetNumber]
        IF N=T THEN E=N : GOTO [NextStatement]
        L = L + 1 : C=1
        GOTO [NextGoto]
        CASE "new"
        FOR I=27 TO 125 : A$(I)="" : NEXT I 'clear the program
        FOR I=27 TO 52 : A(I)=0 : NEXT I 'clear variables a -- z
        FOR I = 0 TO ArraySize : Array(I) = 0: NEXT I
'clear the array -- can't REDIM becuase Run Basic doesn't support it y
et
        IF E=0 THEN [FinishStatement]
        GOTO [Ready]
        CASE "cls"
        CLS : GOTO [FinishStatement]
        CASE "help"
        FOR I=9 TO 21
            B$=A$(I) : PRINT B$
        NEXT I
        GOTO [FinishStatement]
        CASE "mem"
        B=126
        FOR I=27 TO 125
            diffI = 152 - I  'Cheating here
            B$=A$(diffI) : IF B$="" THEN B=diffI
```

```
        NEXT I
        B=126-B : PRINT B;"*";
        PRINT " lines free"
        GOTO [FinishStatement]
        CASE "end"
        GOTO [Ready]
        CASE "bye"
        GOTO [ExitTinyBAS]
        CASE "list"
        GOSUB [GetNumber] : T=N : A=L : I=C
        IF T=0 THEN
            GOSUB [GetLabel]
            IF E$="" AND D$="pause" THEN I=C
            E$=""
        END IF
        FOR L=27 TO 125
            C=1 : GOSUB [GetNumber]
            B=(T=0) OR (N=T)
            IF B=TRUE THEN
                IF A$<>"" THEN
                    PRINT A$
                    IF D$="pause" THEN
                        B = (L-26) mod 10
                        IF B=0 THEN PRINT "Pause..."; : INPUT AAA$
                    END IF
                END IF
            END IF
        NEXT L
        L=A : C=I
        GOTO [FinishStatement]
        CASE "save" 'save a program file
        GOSUB [GetExpression]
        IF E$<>"" THEN [Ready]
        A$="tinyBas"+STR$(N) : A=FALSE
        OPEN A$ FOR OUTPUT AS #1
        FOR I=27 TO 125
            B$=A$(I)
            IF B$<>"" THEN PRINT #1,B$ : A=TRUE
        NEXT I
        CLOSE #1
        IF A=FALSE THEN KILL A$
        GOTO [FinishStatement]
        CASE "load"
'load a program file, but does not account for a program on disk that'
s too large!
        GOSUB [GetExpression]
```

```
        IF E$<>"" THEN GOTO [Ready]
        A$="tinyBas"+STR$(N)
        B=fileExists(DefaultDir$, A$)
        IF B=FALSE THEN
            E$="File "+A$+" not found"
            GOTO [Ready]
        END IF
        OPEN A$ FOR INPUT AS #1
        B=FALSE : I=27
        B=EOF(#1)
        WHILE B <> (-1)
            LINE INPUT #1,B$ : A$(I)=B$ : I=I+1
            B=EOF(#1)
        WEND
        CLOSE #1
        WHILE I<=125
            A$(I)="" : I=I+1
        WEND
        IF E=0 THEN [FinishStatement]
        GOTO [Ready]
        CASE "kill" 'delete a program file
        GOSUB [GetExpression]
        IF E$<>"" THEN [Ready]
        A$="tinyBas"+STR$(N)
        Kill A$
        GOTO [FinishStatement]
        CASE "dir" 'list all program files
        FILES DefaultDir$, "tinyBas*", info$()
        IF VAL(info$(0,0)) > 0 THEN 'list the files
            FOR I = 1 to VAL(info$(0,0))
                print info$(I, 0), info$(I, 2)
'print filename and date/times stamp
            NEXT I
        ELSE
            PRINT "No files!"
        END IF
        GOTO [Ready]
        CASE "let"
        GOSUB [GetLabel]
        IF E$<>"" THEN [Ready]
    END SELECT
    'it must be an expression
    GOSUB [ReturnVar]
    IF E$<>"" THEN [Ready]
    GOSUB [SkipSpace]
    GOSUB [GetChar]
```

```
    IF C$<>"=" THEN
        E$="'=' expected"
        GOTO [Ready]
    END IF
    C = C + 1 : T=V
    IF NOT(arrayFlag) THEN simpleFlag = TRUE
'left side is a simple variable
    GOSUB [GetExpression]
'get the right side which could be an expression
    IF E$<>"" THEN [Ready]
    IF simpleFlag THEN
        A(T)=N 'set the value of simple variable
    ELSE
        Array(T)=N 'set the value of the array
    END IF
    arrayFlag = FALSE : simpleFlag = FALSE
[FinishStatement]
    GOSUB [SkipSpace]
    GOSUB [GetChar]
    IF C$=":" THEN
        C = C + 1 : GOTO [NextStatement]
    ELSE
        IF C$<>"" THEN
            E$="End of statement expected"
            GOTO [Ready]
        END IF
    END IF
    IF L=26 THEN [Ready]
    L = L + 1 : C=1
    IF L=126 THEN
        E$="Program Overflow"
        GOTO [Ready]
    END IF
[FinishStatement2]
    B$=A$(L)
    IF B$="" THEN [Ready]
    GOTO [Exec]
[ExitTinyBAS] 'end the program
    PRINT "Thanks for using Tiny BASIC."
    END

[EnterLine]
    L=27 : C=1 : T=N
[NextLine]
    GOSUB [GetNumber]
    B=(N<T) AND (N<>0) AND (L<126)
```

```
    IF B=TRUE THEN
        L = L + 1 : C=1 : GOTO [NextLine]
    END IF
    IF L=126 THEN
        E$="Program Overflow"
        GOTO [EndEnterLine]
    END IF
    IF T<>N THEN
        FOR I=L to 125
            diffI = (125+L)-I
            B=diffI-1 : A$(diffI)=A$(B)
        NEXT I
    END IF
    A$(L)=Z$
    GOSUB [SkipSpace]
    IF C$="" THEN
        FOR I=L TO 124
            B=I+1 : A$(I)=A$(B)
        NEXT I
    END IF
[EndEnterLine]
    RETURN

[GetExpression] 'numeric expressions only
    A(53)=0 : S=53
    GOSUB [BoolExpression]
    N=A(S) : GOTO [EndExpression]
[BoolExpression]
    GOSUB [AddExpression]
    GOSUB [SkipSpace]
    GOSUB [GetChar]
[NextBool]
    SELECT CASE C$
        CASE "="
        C = C + 1 : GOSUB [AddExpression]
        B=S-1 : A(B)=A(B)=A(S) : S = S - 1
        CASE ">"
        C = C + 1 : GOSUB [GetChar]
        IF C$="=" THEN
            C = C + 1 : GOSUB [AddExpression]
            B=S-1 : A(B)=A(B)>=A(S) : S = S - 1
        ELSE
            GOSUB [AddExpression]
            B=S-1 : A(B)=A(B)>A(S) : S = S - 1
        END IF
        CASE "<"
```

```
        C = C + 1 : GOSUB [GetChar]
        SELECT CASE C$
            CASE "="
            C = C + 1 : GOSUB [AddExpression]
            B=S-1 : A(B)=A(B)<=A(S) : S = S - 1
            CASE ">"
            C = C + 1 : GOSUB [AddExpression]
            B=S-1 : A(B)=A(B)<>A(S) : S = S - 1
            CASE ELSE
            GOSUB [AddExpression]
            B=S-1 : A(B)=A(B)<A(S) : S = S - 1
        END SELECT
    END SELECT
    GOSUB [SkipSpace]
    GOSUB [GetChar]
    B=ASC(C$) : B=(B>=60) AND (B<=62)
    IF B=TRUE THEN [NextBool]
    GOTO [EndExpression]
[AddExpression]
    GOSUB [MulExpression]
    GOSUB [SkipSpace]
    GOSUB [GetChar]
[NextAdd]
    SELECT CASE C$
        CASE "+"
        C = C + 1 : GOSUB [MulExpression]
        B=S-1 : A(B)=A(B)+A(S) : S = S - 1
        CASE "-"
        C = C + 1 : GOSUB [MulExpression]
        B=S-1 : A(B)=A(B)-A(S) : S = S - 1
    END SELECT
    GOSUB [SkipSpace]
    GOSUB [GetChar]
    B=ASC(C$) : B=(B=43) OR (B=45)
    IF B=TRUE THEN [NextAdd]
    GOTO [EndExpression]
[MulExpression]
    GOSUB [GroupExpression]
    GOSUB [SkipSpace]
    GOSUB [GetChar]
[NextMul]
    SELECT CASE C$
        CASE "*"
        C = C + 1 : GOSUB [GroupExpression]
        B=S-1 : A(B)=A(B)*A(S) : S = S - 1
        CASE "/"
```

```
        C = C + 1 : GOSUB [GroupExpression]
        B=A(S)
        IF B=0 THEN
            IF E$="" THEN E$="Division by zero"
            S = S - 1 : GOTO [EndExpression]
        ELSE
            B=S-1 : A(B)=A(B)/A(S) : S = S - 1
        END IF
        CASE " "
        C = C + 1 : GOSUB [GroupExpression]
        B=A(S)
        IF B=0 THEN
            IF E$="" THEN E$="Division by zero"
            S = S - 1 : GOTO [EndExpression]
        ELSE
            B=S-1 : A(B)=A(B) mod A(S) : S = S - 1
        END IF
    END SELECT
    GOSUB [SkipSpace]
    GOSUB [GetChar]
    B=ASC(C$)
    B=(B=42) OR (B=47)  OR (B=92)
    IF B=TRUE THEN [NextMul]
    GOTO [EndExpression]
[GroupExpression]
    GOSUB [SkipSpace]
    GOSUB [GetChar]
    SELECT CASE C$
        CASE "("
        C = C + 1 : GOSUB [BoolExpression]
        GOSUB [SkipSpace]
        GOSUB [GetChar]
        IF C$<>")" THEN
            IF E$="" THEN E$="Missing ')'"
            GOTO [EndExpression]
        END IF
        C = C + 1
        CASE ""
        IF E$="" THEN E$="Invalid Factor"
        CASE ELSE
        B=ASC(C$) : B=(B<48) OR (B>57)
        IF B=FALSE THEN
            GOSUB [GetNumber]
            S = S + 1 : A(S)=N
        ELSE
            GOSUB [GetLabel]
```

```
            IF E$<>"" THEN [EndExpression]
            B=LEN(D$)
            IF B = 1 OR (LEFT$(D$, 2) = "a(" AND RIGHT$(D$,1) = ")")
THEN
                GOSUB [ReturnVar]
                S = S + 1
                IF arrayFlag THEN
                    A(S)=Array(V)
                ELSE
                    A(S)=A(V)
                END IF
            ELSE
                SELECT CASE D$
                    CASE "milliseconds", "ms"
                    S = S + 1 : A(S)=time$("ms")
'return number of milliseconds since midnight
                    CASE "seconds"
                    S = S + 1 : A(S)=time$("seconds")
'return number of seconds since midnight
                    CASE ELSE
                    IF E$="" THEN E$="Function expected"
                END SELECT
            END IF
        END IF
    END SELECT
[EndExpression]
    RETURN

[GetNumber] 'get the line number if it exists
    GOSUB [SkipSpace] 'skip leading spaces
    B$=""
[NextNumber]
    GOSUB [GetChar] 'get the next character
    IF C$="" THEN [GetNumberExit]
    B=ASC(C$) 'convert to ASCII character code
    B=((B<48) OR (B>57)) AND (B<>46)
    IF B=TRUE THEN [GetNumberExit]
'Abort if not a digit or decimal point
    B$=B$+C$ : C = C + 1 : GOTO [NextNumber] 'Build the number
[GetNumberExit]
    N=VAL(B$) 'convert the assembled string to a number
    RETURN

[GetVar]
    GOSUB [GetLabel]
    IF E$<>"" THEN [GetVarExit]
```

```
[ReturnVar]
    arrayFlag = FALSE
    IF LEFT$(D$, 2) = "a(" and RIGHT$(D$,1) = ")" THEN
'possible array variable (i.e. a() )
        GOSUB [ExtractArrayIndex]
    ELSE 'simple variable (i.e. a-z)
        B=ASC(D$) : A=LEN(D$)
        A=(A<>1) OR (B<97) OR (B>122)
        IF A=FALSE THEN
            V=B-70
        ELSE
            IF E$="" THEN E$="Variable expected"
        END IF
    END IF
[GetVarExit]
    RETURN

[GetLabel]
    GOSUB [SkipSpace]
    GOSUB [GetChar]
    D$="" 'single statement
    IF C$="" THEN [GetLabelError]
    B=ASC(C$) : B=((B>=97) AND (B<=122)) OR (B=40) OR (B=41) OR ((B
>=48) AND (B<=57)) 'letters, parentheses and digits are OK
    IF B=FALSE THEN [GetLabelError]
[GetNextLabel]
    D$=D$+C$ : C = C + 1
    GOSUB [GetChar]
    IF C$="" THEN [GetLabelExit]
    B=ASC(C$) : B=((B>=97) AND (B<=122)) OR (B=40) OR (B=41) OR ((B
>=48) AND (B<=57)) 'letters, parentheses and digits are OK
    IF B=TRUE THEN [GetNextLabel]
    GOTO [GetLabelExit]
[GetLabelError]
    IF E$="" THEN E$="Invalid label"
[GetLabelExit]
    RETURN

[SkipSpace] 'skip leading spaces in input
    GOSUB [GetChar]
    IF C$=" " THEN C = C + 1 : GOTO [SkipSpace]
'skip all leading spaces
    RETURN

[GetChar] 'get the next character and change it to lowercase
    A$=A$(L)
```

```
'efficient? This assignment is made before retrieving each character.
    C$=MID$(A$,C,1) : C$=LOWER$(C$)
    RETURN

[ExtractArrayIndex] 'get the array index
    'the index can be a number or a simple variable
    A$ = MID$(D$, 3, (LEN(D$) - 3))
    A = VAL(A$)
    IF A = 0 THEN 'index is 0 or it's a variable
        B = ASC(LOWER$(A$))
        IF (B >= 97) AND (B <= 122) THEN 'it's a variable
            arrayFlag = TRUE
            V = A(B-70) 'get the index value
            IF V = 0 THEN
                IF E$="" THEN E$=
"Array index should be between 1 and "; STR$(ArraySize)
            END IF
        ELSE
            IF E$="" THEN E$=
"Array index should be between 1 and "; STR$(ArraySize)
        END IF
    ELSE
        IF A > ArraySize THEN
            IF E$="" THEN E$=
"Array index should be between 1 and "; STR$(ArraySize)
        ELSE
            arrayFlag = TRUE
            V = A
'must use array flag to determine where to use the V index
        END IF
    END IF
    RETURN

function fileExists(path$, filename$)
    'this function is from the LB4 help file
    'dimension the array info$() at the beginning of your program
    files path$, filename$, info$()
    fileExists = val(info$(0, 0))  'non zero is true
end function
```