MicroBASIC

A

tiny BASIC

for the

6809

Revised: 02-Dec-05

Dunfield Development Systems

----------------------------

High quality tools for

Embedded Development

at low prices.

http://www.dunfield.com

Copyright 1982-2005 Dave Dunfield

All rights Reserved

MicroBASIC USERS GUIDE

TABLE OF CONTENTS

Page

1. INTRODUCTION 1

1.1 Command syntax 1

1.2 Interactive mode 1

1.3 Program editing 1

2. LANGUAGE SUMMARY 2

2.1 Numbers 2

2.2 Character strings 2

2.3 Variables 2

2.4 Functions 3

2.5 Operators 4

2.6 Statements 5

3. Error Messages 13

3.1 Illegal program error 13

3.2 Illegal direct error 13

3.3 Syntax error 13

3.4 Line number error 13

3.5 Wrong type error 13

3.6 Divide by zero error 13

3.7 Nesting error 13

3.8 File not open error 13

3.9 File already open error 13

3.10 Input error 13

3.11 Dimension error 14

3.12 Data error 14

3.13 Out of memory error 14

4. Control-C 14

MicroBASIC USERS GUIDE Page: 1

1. INTRODUCTION

MicroBASIC is tiny interpreter for the BASIC computer languag, which

runs on the 6809 under CUBIX.

1.1 Command syntax

To launch BASIC, simply type BASIC at the CUBIX command prompt.

You may also specify a program name on the command line, which

will cause MicroBASIC to automatically load the .BAS program file

with that name and run it.

\*BASIC MYPROG

1.2 Interactive mode

When MicroBASIC issues the 'Ready' prompt, it is waiting for a

command. Most BASIC statements can be executed as an interactive

command.

1.3 Program editing

Entering a command beginning with a number will perform program

editing. Any existing line in the program with the same line

number will be replaced by the command line. If the line number

does not exist in the program, the new line will be added to the

program. If you enter a command containing a number only, that

line from the program will be deleted.

MicroBASIC USERS GUIDE Page: 2

2. LANGUAGE SUMMARY

2.1 Numbers

MicroBASIC uses "16 bit integer" numbers. This means that any

numeric values used in a program must fall in the range of -32768

to 32767. Within expressions, numbers can be expressed in

hexidecimal by preceeding them with '$'.

2.2 Character strings

MicroBASIC can also manipulate character strings. Character

strings are identified by the double-quote (") character, and are

limited to 100 characters or less in length.

2.3 Variables

MicroBASIC provides a total of 780 "variables" which may be used

to store numbers and character strings. These are divided into

three "types" of 260 variables each. The variables of each type

are identified with a single character from the alphabet followed

by digit from 0 to 9; ie: A0-Z9.

2.3.1 Simple numeric variables: A0 - Z9

These variables are the most commonly used, and can each store

one number (-32768 to 32767). They are used for loop counters,

input values, and general purpose calculations.

2.3.2 String variables: A0$ - Z9$

These variables can hold character strings. These "string"

variables are identified by a '$' appended to the name. They

are normally used for input strings, names, and any other

non-numeric data.

String variables allow indexing to extract individual

characters:

A0$(n) = Return character n (0 to length-1)

2.3.3 Numeric arrays: A0(index) - Z9(index)

These variables can each hold multiple numbers. Unlike the

others, variables of this type must be explicitly created (with

the 'DIM' statement). They are typically used for tables, lists

and any applications which use a group of numbers.

As a matter of convience, the '0' variables can be referenced by

letter only. Ie: A is equivalent to A0, B$ is equivalent to B0$,

and Z(index) is equivalent to Z0(index)

MicroBASIC USERS GUIDE Page: 3

2.4 Functions

MicroBASIC includes a number of "built in" functions which can be

used as values in an expression:

2.4.1 CHR$(value)

Converts a number into an ASCII character. For example: the

ASCII value of the character 'A' is 65. Coding CHR$(65) would

be equivalent to "A". This is the reverse of the ASC()

function.

2.4.2 STR$(value)

Converts a number into a character string. Coding STR$(1234)

would be equivalent to "1234". This is reverse of the NUM()

function.

2.4.3 ABS(value)

Returns the "absolute value" of a number, which a number of the

same magnitude, with a positive sign. For example: ABS(5) and

ABS(-5) would both evaluate to a number of 5.

2.4.4 ASC(char)

Converts a character into its numeric ASCII value. For example:

coding ASC("A") would be equivalent to 65. This is the reverse

of the CHR$() function.

2.4.5 PEEK(address)

Reads a byte of memory at the absolute address given.

2.4.6 KEY()

Tests to see if a key has been pressed on the PC keyboard and

if so, returns its ASCII value. If no key has been pressed,

KEY() returns 0.

2.4.7 LEN(variable)

Returns the length of the specified string or array variable.

If the variable has a '$' suffix, the length return is the

number of characters in the character variable. If no '$', the

length returned is the number of elements in the array

variable.

2.4.8 NUM(string)

Converts a character string containing a number into a numeric

value. Coding NUM("1234") would be equivalent to 1234. This is

the reverse of the STR$() function.

MicroBASIC USERS GUIDE Page: 4

2.4.9 RND(value)

Generates a "random" number which will fall in the range of 0

to (value-1). For example: RND(10) would generate a number

ranging from 0 to 9.

2.4.10 USR(address[,parm1...])

Calls a machine language subroutine at the specified address.

When the subroutine is called, the 6809 'X' register will point

to the top of a stack frame containing any parameters which

were passed. If a character string is passed, the stack value

contains the address of the MicroBASIC internal buffer where

the string is stored (NOTE: Only ONE character string may occur

in the parameter list). The value returned by the USR function

is the value of the 6809 16 bit accumulator (A:B) when the

subroutine returns.

; Example machine language subroutine two add two numbers

ADD2 LDD ,--X ; Get first parameter

ADDD ,--X ; Add second parameter

RTS ; Return with value

All 6809 registers (except SP) may be modified by the

subroutine.

2.5 Operators

Numbers, Strings, Variables and Function may be formed into

expressions and manipulated with the following operators:

+ Addition and String concatination ($)

- Subtraction

\* Multiplication

/ Division

% Modulus (remainder)

& Bitwise AND

| Bitwise OR

^ Bitwise EXCLUSIVE OR

= Assignment / Test EQUAL ($)

<> Test NOT EQUAL ($)

< Test LESS THAN

<= Test LESS THAN or EQUAL

> Test GREATER THAN

>= Test GREATER THAN or EQUAL

! Bitwise NOT (complement)

Operators denoted with ($) can be applied to strings.

When applied to strings, '+' performs string concatination, which

is the operation of joining two strings into one. For example:

"ABC"+"DEF" is equivalent to "ABCDEF".

The "test" operators (=, <>, <, <=, >, >=) can be used in any

expression, and evaluate to 1 of the test is TRUE, and 0 if it is

FALSE.

MicroBASIC USERS GUIDE Page: 5

2.6 Statements

All actions performed by the MicroBASIC program are coded as

"statements", which are commands directing the interpreter.

Statements often contain expressions as described above to perform

calculations.

Most statements can be used either from within a program, or

interactively from the command line "Ready" prompt. Those

statements which are only available in one or the other mode are

noted.

2.6.1 CLEAR

Erases all variables. All numeric variables are set to zero.

All string variables are set to null (0 length), and all arrays

are removed.

10 REM clear all variables

20 CLEAR

2.6.2 CLOSE#unit

Closes a previously opened file (0-9). See 'OPEN#unit'.

10 REM Close file # 5

20 CLOSE#5

2.6.3 DATA

Codes numeric and string data in the program text such that it

can be read with the READ statement. See 'READ' and 'ORDER'.

DATA statements encountered during the normal flow of the

program are ignored. They are only used as sources of data for

the READ statement.

DATA can only be used within a program (non-interactive).

10 REM Data for READ

20 DATA 1,2,3,"text"

30 DATA 4,5,6,"more text"

MicroBASIC USERS GUIDE Page: 6

2.6.4 DIM variable(size)[, ...]

This command "dimensions" (creates) a numeric array variable.

The size parameter defines the maximum index value which can be

used with the array.

If an array of the same name was previously defined, it is

erased first. All elements of the new array are set to zero.

Note that MicroBASIC starts numbering its elements from 0,

while most other BASIC's start numbering elements from 1.

10 REM Define arrays 11 elements (0-10) and 100 elements

20 DIM A(10), A1(99)

2.6.5 END

Terminate execution of the program. No message is issued.

10 REM End program quietly

20 END

2.6.6 EXIT

Terminates MicroBASIC and exits back to DOS.

10 REM Terminate MicroBASIC

20 EXIT

2.6.7 FOR variable=value TO limit [STEP increment]

Begins a counted loop. The variable begins with the specified

value. The program segment between FOR and a corresponding NEXT

statement is repeated until the variable exceeds the specified

limit. After each loop, the variable is incremented by the

increment value. If no increment value is specified, a default

value of 1 is used.

FOR can only be used within a program (non-interactive).

10 REM Count to 10 and print it

20 FOR i=1 to 10

30 PRINT I

40 NEXT I

MicroBASIC USERS GUIDE Page: 7

2.6.8 GOSUB line

Calls a subroutine. The program segment beginning at the

specified line number is executed until a RETURN statement is

encountered, after which the program continues at the next

statement after GOSUB.

GOSUB can only be used within a program (non-interactive).

10 REM Call a subroutine

20 PRINT "start"

30 GOSUB 80

40 PRINT "middle"

50 GOSUB 80

60 PRINT "end"

70 END

80 PRINT "subroutine"

90 RETURN

2.6.9 GOTO line

Causes a "jump" to the specified line number. The program will

continue at the first statement in the indicated line.

GOTO can only be used within a program (non-interactive).

10 REM goto example

20 PRINT "Hello"

30 GOTO 20

2.6.10 IF test THEN statement/line

Performs a statement conditionaly upon a test. The indicated

test expression is evaluated, and if the result is non-zero,

the specified statement is performed. As a shorthand

convienece, a statement consisting of only a number is assumed

to be "GOTO number".

10 REM IF example

20 FOR i=1 TO 10

30 IF i=5 THEN PRINT "I is five"

40 IF i<>8 THEN 60

50 PRINT "I is eight"

60 NEXT i

MicroBASIC USERS GUIDE Page: 8

2.6.11 INPUT ["prompt",]variable

Prompts for a variable. The specified prompt is displayed, and

a single line of input is accepted from the console, which is

assigned to the indicated variable. String variables receive

the input unchanged. Numeric variables cause the input line to

be evaluated to a number.

If no "prompt" string is given, a default of "?" is assumed.

Note that you cannot directly use a character variable as the

prompt (Input assumes that you are inputting into the

variable), but you can use one concatinated to a string.

10 REM INPUT example

20 INPUT A

30 PRINT "A=",A

40 INPUT A$

50 INPUT ""+A$,B

60 PRINT "B=",B

2.6.12 INPUT#unit,variable

Reads the contents of a variable from a file previously opened

with OPEN#unit.

10 REM open example

20 OPEN#1,"TEST.FIL","r"

30 INPUT#1,A$

40 CLOSE#1

50 PRINT "A$=",A$

2.6.13 LET variable = expression

Assigns the value of the expression to the specified variable.

The "LET" keyword is optional, since the interpreter will

assume LET for any statement which it does not recognize.

10 REM LET example

20 LET a=10

30 A$="test string"

2.6.14 LIF test THEN statements

Performs a "long IF". This statement behaves exactly like the

IF statement, except that ALL statements on the line are

conditional on the test.

10 REM IF/LIF example

20 INPUT "Enter a number?",A

30 IF A = 1 THEN PRINT "A=1" : PRINT "TEST1"

40 LIF A = 2 THEN PRINT "A=2" : PRINT "TEST2"

50 IF A<>0 THEN 20

MicroBASIC USERS GUIDE Page: 9

2.6.15 LIST [start,[end]]

Displays the program on the console. If one line number is

given, only that line is listed. If both start and end lines

are given, only that range is listed.

LIST

LIST 10

LIST 10,50

2.6.16 LIST#unit[,start,[end]]

Behaves exactly like LIST except that the program is written to

a previously opened file (0-9).

10 REM List to file demo

20 OPEN#1,"PGM.BAS","wv"

30 LIST#1,100,200

40 CLOSE#1

2.6.17 LOAD "file"

Loads the specified MicroBASIC program file into memory. Any

previous program is erased. The extension ".BAS" is assumed.

When used interactively, all variables are erased prior to

loading the new program.

When used within a program, LOAD performs a "chaining"

function. The new program is loaded, and immediately begins to

run. In this case, the variables are NOT erased, and may be

used as a means of communication between the two programs.

NOTE: If you do not need to pass variables to a chained

program, it is a good idea to execute CLEAR just prior to LOAD.

This makes it easier for MicroBASIC to manage memory, and also

insures that the new program will start with a clean set of

variables.

LOAD "program"

10 REM LOAD demo

20 CLEAR

30 A$="String to pass"

40 LOAD "next"

2.6.18 NEW

Erases the program and all variables.

NEW

MicroBASIC USERS GUIDE Page: 10

2.6.19 NEXT [variable]

Terminates a FOR loop (see FOR). If a variable is specified it

must match the one in the corresponding FOR statement.

NEXT can only be used within a program (non-interactive).

10 REM FOR/NEXT demo

20 FOR i=1 TO 100 STEP 2

30 PRINT I

40 NEXT

2.6.20 OPEN#unit,"filename",R|W]

Opens a file for subsequent 'R'ead or 'W'rite

10 REM OPEN/PRINT to file

20 OPEN#1,"test.fil",W

30 PRINT#1,"This is a test"

40 CLOSE#1

50 REM OPEN/INPUT from file

60 OPEN#2,"test.fil",R

70 INPUT#2,A$

80 PRINT "A$=",A$

90 CLOSE#2

2.6.21 OPEN#unit,"\*device"

Opens a CUBIX serial device for I/O.

10 REM OPEN/READ/WRITE device 2

20 OPEN#1,"\*2"

30 INPUT#1,A$

40 PRINT#1,"STRING=",A$

50 CLOSE#1

2.6.22 ORDER line

Positions the DATA/READ pointer to a specific line. The line

MUST begin with a DATA statement. This command must be given

before a READ statement can be used at the beginning of a

program, and again after all data has been read from a data

block.

Once the data pointer has been set, READ will automatically

accept data from multiple DATA statements, but it will not

automatically skip over non-DATA statements.

10 REM ORDER/DATA demo

20 ORDER 30

30 DATA 1,9,2,8,3

40 DATA 7,4,6,5,0

50 PRINT "Nothing much happening yet"

MicroBASIC USERS GUIDE Page: 11

2.6.23 POKE address,value

Writes a byte value to the specified memory address.

10 REM Zero LPT data port

20 POKE 10,0

2.6.24 PRINT expression[,expression][,]

Prints the values of expressions on the console. Either numeric

values or character strings may be printed. Unless a trailing

',' is supplied, PRINT will output a Line-feed/Carriage-return

to reposition output to a new line after the expressions are

printed.

NOTE: Print automatically preceeds numbers with a space,

however it does no such special processing for character

strings. The STR$ function can be used to convert a number into

a string and avoid this space.

10 REM PRINT demo

20 FOR i=0 TO 10

30 PRINT "I=",i,"!"

40 PRINT "I=",STR$(i),"!"

50 NEXT I

2.6.25 PRINT#unit,expression[,expression],

Behaves exactly like the PRINT command above except that the

output is directed to a previously opened file.

10 REM PRINT#unit demo

20 OPEN#1,"TEST.FIL",W

30 FOR i=0 TO 10

40 PRINT#1,"I=",I

50 NEXT I

60 CLOSE#1

2.6.26 READ variable[,variable ...]

Reads data into a variable from DATA statements in the program.

An ORDER statement must be used to set the data pointer before

READ can be used.

10 REM ORDER/READ/DATA demo

20 ORDER 60

30 READ A

40 PRINT "A=",A

50 IF A <> 0 THEN 30

60 DATA 1,9,2,8,3

70 DATA 7,4,6,5,0

MicroBASIC USERS GUIDE Page: 12

2.6.27 RETURN

Returns program execution from a subroutine to the statement

immediately following the GOSUB that invoked the subroutine.

RETURN can only be used within a program (non-interactive).

10 REM Call a subroutine

20 PRINT "start"

30 GOSUB 80

40 PRINT "middle"

50 GOSUB 80

60 PRINT "end"

70 END

80 PRINT "subroutine"

90 RETURN

2.6.28 REM

Inserts a comment in the program. The remainder of the line is

ignored by the interpreter. Execution continues on the next

line.

10 REM this is a comment

2.6.29 RUN [line]

Starts the program executing. All user variables are cleared

first. If a line number is specified then the program will be

started at that line, otherwise execution begins on the first

line of the program.

RUN can only be used from the interactive command line

(non-program)

RUN

2.6.30 SAVE "name"

Saves the program in the specified file.

SAVE can only be used from the interactive command line

(non-program).

SAVE "test"

2.6.31 STOP

Terminate execution of the program with an informational

message.

10 REM End program with an announcement

20 STOP

MicroBASIC USERS GUIDE Page: 13

3. Error Messages

The following error messages are produced by MicroBASIC. If the error

occurs during the execution of a program, it will be followed by "in

line n" where n is the offending line unmber.

3.1 Illegal program error

Results from an attempt to use a statement within a program which

is only allowed as an interactive command.

3.2 Illegal direct error

Results from an attempt to use a statement as an interactve

command which is only available within a program.

3.3 Syntax error

Results from any statement which does not follow the MicroBASIC

syntax rules.

3.4 Line number error

Results from an attempt to reference a line number which does not

exist

3.5 Wrong type error

Results from an attempt to use a value in a context where a

different type (number or string) is expected.

3.6 Divide by zero error

Results from an attempt to perform division by zero.

3.7 Nesting error

Results from incorrect nesting of FOR/NEXT of GOSUB/RETURN

constructs, or from overflow of the internal control stack which

manages these constructs.

3.8 File not open error

Results from an attempt to perform I/O to a unit number that does

not have a corresponding open file.

3.9 File already open error

Results from an attempt to open a file using a unit number that

already has a corresponding file opened.

3.10 Input error

Results an incorrect numeric entry in response to an input request

for a numeric variable.

MicroBASIC USERS GUIDE Page: 14

3.11 Dimension error

Results from an attempt to access a numeric array that has not

been defined, or use of an index value that is outside of the

range declared for the array.

3.12 Data error

Results from attempt to READ when the data pointer does not point

to a data statement containing the correct data type for the

variable you are attempting to use.

3.13 Out of memory error

Results when MicroBASIC's available memory pool has been used up.

Available memory is depleted by program lines, character strings,

and numeric arrays.

4. Control-C

MicroBASIC uses the keyboard Ctrl-C character (Press 'C' while

holding CTRL) to signal an ABORT condition. MicroBASIC will stop it's

current activity and return to the interactive prompt as quickly as

possible when this key combination is detected.