Programming

Variables

The standard Basic variable types of Integer, String and Float are available for normal use within a program.

Integer
a 32 bit signed Integer

String [len]
a Basic string implemented as a 16 bit length followed by the string itself. Only fixed length strings are supported in this version.

Float
an 80 bit floating point number implemented using the Apple SANE libraries.

Additionally, Pascal type variables may be used for communication with the Macintosh Toolbox.

Byte
an 8 bit unsigned value

Word
a 16 bit signed value

Char [len]
a fixed length string of characters

Str255 [len]
a pascal string consisting of an 8 bit length followed by the string.

A structure type construct is provided for toolbox communication and for creating structured records for use in Random file access. (See GET and PUT).

Variables may have a LOCAL or GLOBAL scope. The examples below demonstrate the usage of the different variable types available.

```
Global Myint As Integer
Local Mystr As String [30]                .. maximum length=30 characters.
Global Mybyte As Byte
Local Myword As Word
```

```
Global Mychar As Char [20]                  .. fixed length= 20 characters.
Local Mypstring As Str255 [10]          .. maximum length=10 characters.

Local Mystruct As Structure
        Local Myint As Integer
        Local Mystr As String [10]
Endstruct
```

Arrays

Arrays may have up to 3 dimensions in this version of the compiler and are dynamic in the sense that the size of an array can be changed at any time. Arrays are given an initial dimension when declared with the GLOBAL or LOCAL statements and are resized with the REDIM statement.
The contents of an array may be saved and subsequently loaded between program invocations using the ARRAYLOAD and ARRAYSAVE statements. The contents of saved arrays are stored at DATA type resources in the applications resource file.
The maximum number of elements which can be stored in an array is 65000.

Arithmetic Functions and Commands

The following arithmetic functions are supported:

+	Add
-	Subtract or unary minus
*	Multiply
/	Divide

Additionally, the ADD and SUB commands may be used where an integer expression needs to be add to or subtracted from a variable and the INC and DEC commands provide a quick way to Increment or Decrement the value of an integer variable.

Logical Operators

The logical operators AND, OR, NOT and XOR (Exclusive OR) are available for use in arithmetic and conditional expressions.
A logical TRUE yields a result of -1 whereas a logical FALSE yields a result of zero.

Comparison Functions

Comparison functions, together with Logical Operators, resolve an expression down to a TRUE or FALSE value. The following comparison functions are implemented :-

>	Greater than
<	Less than
=	Equal to
<> or !=	Not equal to
>= or =>	Greater than or equal to
<= or =<	Less than or equal to

Program Flow

Several standard Basic constructs are available for controlling program flow :

The FOR / NEXT loop which steps through the loop a fixed number of times changing the value of a variable each time it passes through the loop.

FOR <variable name> = <start expression> TO <end expression> [STEP <expression>]
    statements
NEXT

The variable is initially set to <start expression> and each time through the loop, the STEP <expression> is added to it until it reaches the value of the ,end expression>. The STEP parameter defaults to 1.

The REPEAT construct loops until a condition is met.

```
REPEAT
    statements
UNTIL <conditional expression>
```

REPEAT always executes the statements between the REPEAT and UNTIL at least once.

The WHILE contruct also loops until a condition is met but in this case the condition is tested at the beginning of the loop.

```
WHILE <conditional expression>
    statements
WEND
```

The DO CASE construct allows multiple conditions to be tested and different

actions to be taken according to the results..

```
DO CASE
        CASE <conditional expression>
            statements
            [BREAK
```