FnxBasic Language Reference

LANGUAGE REFERENCE

SIGNS AND OPERATORS

Arithmetically:

=	is equal to
+	to add up
-	to subtract
*	to multiply
/	divide
^	power to

Boolean operators:

>	larger than
<	smaller than
=>	is equal and larger than
=<	is equal and smaller than
<>	is unequal to
AND	and
OR	or

Two or more comparisons can be added, for example: *If a>1 and a<10 then a=0*

Remaining:

‘	comment, remark, brake Ark can be used on each rule
:	a new command gives to (*a=10:b=10*) so that this can stand on a line
,	separation between claims e.g. *dim a as integer, b as integer*.
( )	parameters for ruts and array
“”	indicator for strings
&h	indicator for a hexadecimal value

COMMANDS AND FUNCTIONS

The syntax for command and function:

A function always returns a result and its parameters must be included with brackets.
Example: asc (A) ‘Returns value 65

A command is only on a line and its parameters are not in brackets.
Example: *kill mydoc.txt* ‘Erase the file mydoc.txt.

To Declare Variables

The claim of a variable must always stand for its use. The claim goes by means of DIM statement. If the claim is outside a sub-routine or a function then the claim is overall (global). Claims for a sub-routine or a function can only be used in the sub-routine or function. The dim statement can also be used to declare objects and Windows objects.

Self Defined Functions/Sub-Routines

These require brackets:

The claim:

```
Function test() as integer
   Print “ok”
   Result=210
End function
```

The call:

```
Dim x as integer
X=Test()   ‘ X now has the value 210.
```

The result is therefore indicated in the function as result.

For Functions, parameters can be given between the brackets or declared:

```
Function test (text as string, value as integer) as integer
  Print text+str$( value)
  Result=210
End function
```

The call:

```
X=test(“The value is”,10)   ‘ X now has the value 210.
```

For sub-routines, the parameters are passed on to the variable of the call.

The claim:

```
Sub test(s as string)
  S=” I am here”
  Print s
End sub
```

The call:

```
Dim a as string
Test(a)
```
'A contains the value "I am here"

If a composition of variables is used, then the parameter is not passed on:

```
Test("Piet"+A)
```

A has value now the same as before the call.
The value of s in the sub-routine has, however, the value of Piet +a.

A sub-routine or function can leave premature with the EXIT statement.

Loops

Program loops can be built in several manners:

The FOR NEXT loop

```
Dim t as integer
For t=1 to 100
  Print "ok"
Next t
```

After passed through the loop, t contains the value 101.

The WHILE loop:

```
While t<100
   Print "ok"
   Inc(t)
Wend
```

The DO loop:

```
Do
  Print "ok"
  Inc(t)
Loop until t>100
```

All loops are possible to leave pre-mature with the EXIT statement:

```
Do
  Print "ok"
  Inc(t)
  If t>100 then exit do
Loop
```

File Names

A file name with a path is written as *c:\myfolder\mydoc.txt*
If the path is not given, then the current map is used to find the file.

Arrays

Arrays are declared with a DIM statement:
Dim a(10) as integer

This declared array has 11 actual places because a (0) also exists.
Arrays can have a maximum of 5 dimensions (dimemsions are separated by commas):
Dim a(10,20,30,40,50) as integer

A special option is declaring sub-routines of functions as array:

```
Dim a(10) as function
Function test(a as integer) as string
        Result=" in the function"
End function
A(1)=test
```

The call looks as follows:
Print a(1)(10)

The result is therefore the value "in the function" and the parameter of the function contains the value 10. Windows objects can be also declared as an array.

Manifest For Windows XP

To show the application according to the new Windows XP an aid file is necessary, the manifest file. The manifest file must in the same folder as the application. As the application "myapp.exe" is called then must the manifest file "myapp.exe.manifest" be called. Using the editor, this file can be produced. One should include the file at program distribution. (See "Appendix -- Windows Vista Manifest" in this document for further details).

Windows Objects

AWindows object must be declared just like a variable by means of DIM statement:
Dim myform as form

Objects have properties, methods and events. These are separated from object name by a period.
Myform.left=100 'The object <u>Myform</u> with property <u>left</u> having value of 100

Now we declare a button:

Dim mybutton as button

However, we want the button on the form myform:
Mybutton.parent=myform

If this program is started up nothing happens, so a task must be given to show the form. This is the Method showmodal:
Myform.showmodal

The total code to show a form with button is:

```
Dim myform as form
Myform.left=100
Dim mybutton as button
Mybutton.parent=myform
Myform.showmodal
```

Now we want to let the button do something. This happens by means of Event. All events start with onclick event, which is activated if the button is pressed. Events refer to a sub-routine. The claim of the event must exist above showmodal of the program:

```
Dim myform as form
Myform.left=100
Dim mybutton as button
Mybutton.parent=myform
Mybutton.onclick=buttonclick
Myform.showmodal

Sub buttonclick()
  Print "the button is clicked"
End sub
```

To make the spelling more synoptic, there exists for all objects, which has the property parent, the statement object. In this manner, the first outer claim is to declare the parent. Thus, the above could be also written as:

```
Object myform as form
   Left=100
   Object mybutton as button
     Onclick=buttonclick
   End object
End object

Myform.showmodal

Sub buttonclick()
```

```
   Print "the button is clicked"
End sub
```

At the event assignment, the name of the sub-routine must be only mentioned without brackets.

Objects As Array

Objects can be also declared as an array; the object can also be passed on to a sub-routine. Using the property tag, it is possible to retrieve which object has been clicked as in example below:

```
Dim myform as form
Dim mybutton(2) as button
Dim t as integer
For t=0 to 2
  Mybutton(t).onclick=buttonclick
  Mybutton(t).top=t*30
  Mybutton(t).parent=myform
  Mybutton(t).tag=t
Next t
Myform.showmodal

Sub buttonclick(object as button)
  Select case object.tag
     Case 0 :  print "button 0 "
     Case 1 :  print "button 1 "
     Case 2 :  print "button 2 "
  End select
End sub
```

Defining A Type

It is possible to define types using the statement Type-End Type. Within a type, a function is a sub-routine. In fact, this is a Method of an Object. An overall variable of the type is in fact a Property. A type must be declared first before it can be used.

```
Type mytype
  tire as string
  tire="28 Inch"
  light as string
End Type

Dim bicycle as mytype
bicycle. light ="on"
print bicycle.tire     ' The print result is"28 Inch"
```

A Method to add:

```
Type mytype
  Sub light()
    Print “light on”
  End sub
End type

Dim bicycle as mytype
bicycle.light()      ‘Now “light on” is reflected on the screen.
```

A Type As Object

It is also possible to declare a type as object. This goes by means of the AS statement.

```
Type mytype as form
  Sub click()
    Showmessage “clicked”
  End sub
  Object mybutton as button
    Onclick=click
    Parent=mytype
  End object
End type

Dim box as mytype
Box.showmodal
```

It is also possible to declare an object within a type. The claims of subs do not have to exist above event assignments. A call to a sub-routine also works as the sub-routine outside the type has been declared.

Creating A Library

The *.lib file can be as ordinary as a fnxbasic file with the extension lib. The lib file must be placed in the folder where the compiler is. Lib file are generally a type of claim or API claim. The file is declared in the head of the program using the statement USES.

Example: *USES led,getfocus,setfocus*

API Or DLL Declaration

With the declare statement, DLL calls can be declared using a sub-routine or function.
DECLARE name AS functionname OF libraryname

For example the getfocus function, pay attention to the job title it is capital letter sensitive:

Declare mygetfocus as "GetFocus" of "user32.lib"
Dim result as integer
End declare

The call is as follows:
Mygetfocus
Print mygetfocus.result ' Here the handle of window receives the focus.

Now the same, but then with setfocus:

Declare mysetfocus as "SetFocus" of "user32.lib"
Dim hwnd as integer
Dim result as integer
End declare

The call, firstly, gives up the variables:

mysetfocus.hwnd=myform.hwnd
mysetfocus' after this call myform has the focus.

At string variables, the length can be given up, e.g. *dim mystring as string * 10*
If at the claim, an address must pass on to become a variable, then use the statement BYADDRESS, for example: *dim mystring as string byaddress*.

BLOCK COMMANDS

CASE

Syntax:

```
SELECT CASE argument
   CASE condition
   CASE condition
END SELECT
```

SELECT CASE searches the CASE statement for a met condition.

The condition can be numerically or a STRING. If no Boolean comparison is in the condition, then the condition is evaluated as equal to the argument.

The Boolean comparisons are:

= is equal to
> is larger than
< are smaller than
=> are equal and larger than
=< are equal and smaller than
<> are unequal as
AND and
OR or
XOR exclusive or

Example:

```
DIM x AS INTEGER
SELECT CASE x
  CASE 10
     PRINT x
  CASE >20
     X=x+100
END SELECT

DIM x AS STRING
SELECT CASE x
  CASE "test"
     PRINT x
 END SELECT
```

DECLARE

Syntax:

```
DECLARE name AS "function" of  "dllname"
   Variable declaration [BYADDRESS]
   STRING declaration [*length]
   RESULT AS TYPE
END DECLARE
```

With DECLARE for example API call can be done.

The additive of address at the claim of variables gives the address of the variable to that the address is used and not the variable itself. Result returns the outcome of the call.

Example1:

```
DECLARE setfocus  AS "SetFocus" of  "user32.dll"
```

```
    Hwnd AS INTEGER
    Result AS INTEGER
END DECLARE

Getfocus.hwnd=myform.hwnd
PRINT getfocus.result
```

Example 2:

```
OBJECT myform AS FORM
  OBJECT EDIT AS RICHEDIT
  END OBJECT
END OBJECT

DECLARE sendmessage AS "SendMessageA" of "user32.dll"
    hwndA AS INTEGER
    param AS INTEGER
    wparam AS INTEGER
    iparam AS STRING byaddress
    result AS INTEGER
END DECLARE

SUB settab()
  sendmessage.hwndA=edit.hwnd
  sendmessage.param=&h400+71
  sendmessage.wparam=0
  DIM n AS STRING
  n=mksrt$(156)+mksrt$(0)+mki$(&H10)+string$(chr$(0),18)+mksrt$(32)


FOR t=0 TO 31
    n=n+mki$(200*t)
  NEXT t
  sendmessage.iparam=n
  sendmessage
END SUB
settab()
myform .showmodal
```

DO

Syntax:

DO Boolean expression
Tasks
[EXIT DO]
Tasks

LOOP [UNTIL Boolean expression]

The Boolean expressions are expressive with the following signs:

= Is equal to
> Is larger than
< Are smaller than
=>Is right and larger than
=<Is right and smaller than
<>Is unequally as
AND and
OR or
XOR exclusive or

DO always passes through the same loop to LOOP and then starts again.

Example:

```
DIM a AS integer, b AS INTEGER
DO
  b=b+2
LOOP until b>10

DO
   IF a=b THEN EXIT DO
   PRINT a
   A=a+1
LOOP
```

IF/ELSE

Syntax:

IF Boolean expression THEN task [ELSE task]

Or:

```
IF Boolean expression THEN
  Tasks
[ELSE]
  [Different tasks]
END IF
```

The boolean expression is expressive with the following signs:

= Is equal to
> Is larger than
< Are smaller than
=> Are equal and larger than
=< Are equal and smaller than
<> Are unequal as
AND and
OR or
XOR exclusive or

Example:

```
DIM a AS INTEGER r, b AS INTEGER
IF a>b THEN b=0

IF b=>a AND b>3 THEN
    PRINT "okay"
ELSE
    PRINT "no"
END IF

IF (a>b AND b<>a) OR b=1 THEN w=false
```

ELSEIF

Syntax:

```
IF Boolean expression THEN
   Tasks
   ELSEIF Boolean expression
     [Tasks]
  END IF
END IF
```

The Boolean expressions are expressive with the following signs:

= is equal to
> Is larger than
< Are smaller than
=> Are equal and larger than
=< Are equal and smaller than
<> Are unequal as
AND and
OR or
XOR exclusive or

Example:

```
DIM a AS integer, b AS INTEGER
IF a=10 THEN
 B=1
 ELSEIF a=20 THEN
  B=2
  ELSEIF a=30 THEN
    B=3
  END IF
 END IF
END IF
```

EXIT

Syntax: EXIT argument

Prematurely leave a loop or block system. The following exits are:

- EXIT FOR: for-next leave a loop
- EXIT WHILE: leaves a while-wend loop
- EXIT DO: do-loop leave
- EXIT FUNCTION: leaves a function
- EXIT UNDER: leaves SUB

Example:

```
FUNCTION test(a AS integer) AS STRING
  DIM t AS INTEGER
  FOR t=1 TO 10
   IF t=a THEN
     Result="okay"
     EXIT FUNCTION
   END IF
  NEXT t
  Result=""
END FUNCTION
```

FOR/STEP

Syntax:

```
FOR counter variable = starts TO end [STEP size]
   Tasks
   [EXIT FOR]
NEXT counter variable
```

The <counter variable>, <starts>, and <end> are numerical types.
The STEP <size> must be given if step size is smaller than 0
Default step size is 1 .

With EXIT FOR , one can leave the loop prematurely.

Example:

```
DIM c AS INTEGER
FOR c=1 TO 10
   PRINT c
NEXT c

FOR c=10 TO 1 STEP –1
  PRINT c
   IF c=5 THEN EXIT FOR
NEXT c
```

FUNCTION/RESULT

Syntax:

```
FUNCTION name([parameter declarations]) AS TYPE
    [EXIT FUNCTION]
    Tasks
    [RESULT=value]
END FUNCTION
```

Declare a FUNCTION. Brackets are required with or without parameters. Also, TYPE of the function is given. The function can be declared both for and after a call. If a value is given to RESULT, the result of call is that value.

Example:

```
FUNCTION test(a AS integer, b AS integer) AS INTEGER
   `Tasks
   IF a=5 THEN EXIT FUNCTION
   Result=10
END FUNCTION

DIM x AS INTEGER
x= test (1.1)
```

X has now the value 10.

OBJECT

Syntax:

```
OBJECT name AS WINDOWS OBJECT
   OBJECT properties
END OBJECT
```

With OBJECT, one can declare a WINDOWS OBJECT. The properties can be given. (See Types – Windows Objects in this document).

Example:

```
OBJECT myform AS FORM
  Left=100
  OBJECT mybutton AS BUTTON
    Width=10
  END OBJECT
END OBJECT

Mybutton.left=10
Myform .showmodal
```

A form with a button is now shown.

SUB

Syntax:

```
SUB name([parameter declarations])
    [EXIT SUB]
    Tasks
END SUB
```

SUB declares a sub-routine. Brackets are required with or without parameters. The sub-routine can be declared both for and after a call. If while in the call, a variable has been given a value, then the value of it is passed on. Example:

```
SUB test (a AS INTEGER, b AS INTEGER)
   `Tasks
   IF a=5 THEN EXIT SUB
   A=10: b=20
END SUB

DIM x AS integer, y AS INTEGER
Test(x, y)    ' x has value 10 and y has value 20
Test (1, y)   ' y has value 20, a in sub-routine value 1, and X has value 10.
```

TYPE

Syntax:

```
TYPE name [AS FORM]
   Tasks
END TYPE
```

With TYPE, one can make defined objects or a record. Also sub-routines and functions can be applied within TYPE. TYPE must be declared, first with DIM or OBJECT before it can be used.

Example:

```
TYPE my
  DIM var AS INTEGER
  SUB is ()
     Var=var+1
  END SUB
END TYPE

DIM test AS my
Test.var=10
Test.is()

TYPE fmt AS FORM
   Left=100
   Caption="myform"
END TYPE

DIM myform AS fmt
Myform.width=20
```

WHILE

Syntax:

WHILE Boolean expression
[EXIT WHILE]
WEND

The Boolean expression is expressive with the following signs:

= is equal to
> is larger than
< are smaller than
=> Are equal and larger than
=< Are equal and smaller than
<> Are unequal as
AND and
OR or
XOR exclusive or

WHILE always passes through the same loop to WEND and then starts again.

Example:

```
DIM a AS integer, b AS INTEGER

WHILE a>b
  b=b+2
WEND

WHILE a>b
   IF a=b THEN EXIT WHILE
   PRINT a
WEND
```

WITH

Syntax:

WITH name
  Tasks
END WITH

WITH assumes the Tasks to be of the object. This can prevent extra coding work.

Example:

```
DIM myform AS FORM
Myform.left=100
Myform.width=100
Myform.parent=desktop
```

With the WITH statement it looks like this way:

```
DIM myform AS FORM
WITH myform
  Left=100
  Width=100
  Parent=desktop
END WITH
```

REMAINING COMMANDS

CONST

Syntax: CONST name = value

A constant is a variable with a fixed value. The TYPE is determined by the fixed value.

Example:

CONST x = 100 ‘ X has now the numerical value 100.

CONST x =”abc” ‘ X now contains “abc” as a string.

DIM

Syntax : DIM name [(parameter)] AS TYPE

Declares a variable, array, windows OBJECT, or array of functions. Parameter reflects the maximum value of array. TYPE can be also a self defined TYPE.

Example:

DIM x(10) AS STRING ‘declares a STRING array of 11 (0,1,2…10)

DIM x(9,4) AS INTEGER ‘declares an INTEGER array of 10 by 5’

DIM myform AS FORM ‘declares a windows form with name myform’

DIM x(10) AS FUNCTION ‘declares an array with functions’

INCLUDE

Syntax: INCLUDE filename

Adds a program part from a file. Filename is of the TYPE string.

Example:

INCLUDE “fnxbasic.bas”

KILLKEY

Syntax: KILLKEY keynumber

Erases a key press so that its windows are not processed. The key number is a numerical type.

Example:

Killkey 13

LOADLIBRARY

Syntax : LOADLIBRARY filename

Loads an external DLL file into memory. The DECLARE task does this automatically.
Filename is of the STRING type.

Example:

LOADLIBRARY “c:\windows\user32.dll”

LPEND

Syntax : LPEND

The command gives to the PRINTER the task to print the strings given with LPRINT.

Example:

```
LPRINT "hello"
LPRINT "world"
LPEND                ' Prints to the printer "hello world"
```

LPRINT

Syntax : LPRINT STRING

Print STRING on the line printer. Printing starts after LPEND command has been given.

Example:

```
LPRINT "hello"
LPRINT "world"
LPEND                ' Prints to the printer "hello world"
```

MAKESHORTCUT

Syntax: MAKESHORTCUT name, target

Makes a Windows program shortcut. Name and target are both of the STRING type.

Example:

```
Makeshortcut "myprogram.exe","c:\windows\desktop\myprg"
```

PLAYWAV

Syntax: PLAYWAV name, integer

Play Windows music file (file extension wav). The file name is TYPE string.
The INTEGER can have the next possible values:

- 0: Stops with playing a file.
- 1: The file takes place but the program does not wait.
- 2: The file takes place infinitely.
- 3: The file takes place and the program waits until the playing has stopped.

Example:

```
PLAYWAV "c:\windows\ding.wav",2
PLAYWAV "c:\windows\ding.wav",0
```

PRINT

Syntax : PRINT argument

PRINT prints the argument on the console screen. Can be used for debugging of a program.

Example:

```
PRINT “string”
PRINT 10
PRINT a+b
```

RESOURCE

Syntax: RESOURCE name AS file name

A file adds to the program during compiling. The program (i.e., for objects with the function loadresource) can use this file.

Example:

```
RESOURCE test AS “button.txt”
DIM s AS STRING
DIM stream AS MEMORYSTREAM
Stream.loadresource(test)
Stream.position=0
S=stream.read(stream.size)
Stream.close
```

S now contains the text of the file button.txt

SWAP

Syntax: SWAP variable, variable

The contents of a variable interchanges. Variable can be of the TYPE STRING or numerical.

Example:

```
DIM x AS INTEGER
DIM y AS INTEGER
X=10
Y=100
SWAP x,y      ‘ X now has value 100 and Y now has value 10
```

USES

Syntax : USES name,name,etc

Program part(s) from a file (with extension lib) adds to program.

Example:

USES led,myform

CONVERSION FUNCTIONS

BIN$

Syntax: BIN$(numeric value)

Convert NUMERIC value to binary STRING. (Front zeros are omitted.)

Example:

```
DIM X AS STRING
X=bin$(8)   ‘ X is now 1000
```

CVBIN

Syntax : CVBIN(string)

Convert STRING with a binary value to a decimal NUMBER.

Example:

```
DIM X AS INTEGER
X=cvbin(“1000”)    ‘ X is now 8
```

CVD

Syntax: CVD(string)

Convert eight bytes from STRING to a DOUBLE.

Example:

```
DIM X AS STRING
X=cvd("#@45fhZ~")
```

CVHEX

Syntax : CVHEX(string)

Convert STRING with hexadecimal value to a decimal NUMBER.

Example:

```
DIM X AS INTEGER
X=cvhex("FF")    ' X is now 255
```

CVI

Syntax : CVI(string)

Convert STRING with four bytes to an INTEGER.

Example:

```
DIM X AS INTEGER
X=cvi(";4#A")    ' X is now 1092826171
```

CVS

Syntax : CVS(string)

Convert 4 bytes STRING to SINGLE.

Example:

```
DIM X AS SINGLE
X=cvs("z6#A")   ' X is now 10,2008
```

CVSRT

Syntax : CVSRT(string)

Convert 2 bytes STRING to SHORT.

Example:

```
DIM X AS SHORT
X=cvsrt(“zz”)   ‘ X is now 31354
```

CVW

Syntax: CVW(string)

Convert 2 bytes STRING to WORD.

Example:

```
DIM X AS WORD
X=cvw(“zz”)    ‘ X is now 31354
```

HEX$

Syntax: HEX$(numeric expression)

Convert NUMBER (or expression) to hexadecimal STRING. (Front zeros are omitted)

Example:

```
DIM X AS STRING
X=hex(255)       ‘ X now FF
```

MKD$

Syntax : MKD$(double)

Convert eight bytes of a DOUBLE into a STRING.

Example:

```
DIM X AS STRING
X=mkd$(1092826171)
```

MKI$

Syntax : MKI$(integer)

Convert four bytes of an INTEGER into a STRING.

Example:

```
DIM X AS STRING
X=mki$(1092826171)       ' X is now ;4 # A
```

MKS$

Syntax: MKS$(single)

Convert 4 bytes of a SINGLE into a STRING.

Example:

```
DIM X AS STRING
X=mks$(10.2008)       ' X is now z6#A
```

MKSRT$

Syntax: MKSRT$(short)

Convert 2 bytes of a SHORT into a STRING.

Example:

```
DIM X AS STRING
X=mksrt$(31354)     ' X is now zz
```

MKW$

Syntax: MKW$(word)

Convert 2 bytes of a WORD into a STRING.

Example:

```
DIM X AS STRING
X=mkw$(31354)      ' X is now zz
```

RGB

Syntax: RGB(red value, green value, blue value)

Calculates the Color value. The values lie between 0 and 255.

Example:

```
DIM X AS INTEGER
X=rgb(10,100,60)      ' X is now 3957770
```

STR$

Syntax: STR$(numeric expression, [number of decimal places])

Convert NUMERICAL expression to STRING. Second parameter is number of decimal places.

Example:

```
DIM X AS STRING
X=str$(31.78)       ' X is now 31.78
X=str$(31.78,1)     ' X is now 31.7
```

VAL

Syntax: VAL(string)

Convert STRING to NUMERICAL value. The STRING must contain a number.

Example:

```
DIM X AS INTEGER
X=val("378")      ' X is now 378

X=val("20appel100")      ' X is now 20
```

STRING COMMANDS/FUNCTIONS

ASC FUNCTION

Syntax : ASC(string)

Returns the ASCII value of the first character in the string.

Example:

```
DIM X AS INTEGER
X=asc("A")          ' X is now 65

X=asc("Appel")    ' X is now 65
```

CHAR$ FUNCTION

Syntax: CHAR$(string, position)   CHAR$ also is a command.

Returns the character in the position of the STRING.

Example:

```
DIM X AS STRING
X=char$("abcdef",3)     ' X is now "c"
```

CHAR$ COMMAND

Syntax: CHAR$ string, index = value

Adds a character to STRING before the index (position). Index is INTEGER type.

Example:

```
DIM s AS STRING
S="apele"
Char$(s,3)="p"        ' S has now the value "apple"
```

CHR$ FUNCTION

Syntax : CHR$(numeric value)

Returns ASCII character of given ASCII numerical value. Value must be between 0 and 256.

Example:

```
DIM X AS STRING
X=chr$(65)     ' X now "A" is
```

DELETE$ FUNCTION

Syntax: DELETE$(string, start, length)

Remove part of the STRING, before start position, for length (number of characters).

Example:

```
DIM X AS STRING
X=delete$("abcdea",3,2)     ' X is now "abea"
```

EXISTS$ FUNCTION

Syntax: EXISTS$(keyword, string)

Check if key word(s) exists in a string. | sign is the separator for the key words.
Function is capital letter sensitive.

Example:

```
DIM X AS Boolean
X=exists$("ok|ab","abgfghk")     ' X now is true
X=exists$("ok|ab","gfokghk")     ' X now is true
```

FIELD$ FUNCTION

Syntax: FIELD$(string, field character separator, field position)

Returns characters of field position in STRING (using field separator character).
Function is capital letter sensitive.

Example:

```
DIM X AS STRING
X=field$("ab/cd/ea","/",2)             ' X is now "cd"

X=field$("applefffpeer","fff",1)       ' X now "apple" is
```

FIELD$ COMMAND

Syntax: FIELD$ string, field character separator, field position = value

Changes the value of a field position in STRING, using field character separator.
The function is capital letter sensitive.

Example:

```
DIM X AS STRING
X="ab/cd/ea"
FIELD$ x,"/",2 = hhh        ' X now has the value "ab/hhh/ea"
```

INSERT$ FUNCTION

Syntax : INSERT$(string, characters to add, position)

This function adds characters in the STRING before the position.

Example:

```
DIM X AS STRING
X=insert$("pietkees","/",5)        ' X is now "piet/kees"

X=insert$("appelpeer","fff",6)     ' X is now "appelfffpeer"
```

INSTR FUNCTION

Syntax: INSTR([start],string, character to find)

Returns first position of character found in string. If start given, start find before start position.
The function is capital letter sensitive and returns value 0 if character not found.

Example:

```
DIM X AS INTEGER
X=instr("Apple","p")         ' X is now 2

X=rinstr(3,"Apple","p")      ' X is now 3
```

LCASE$ FUNCTION

Syntax: LCASE$(string)

Put STRING in small characters (lower case).

Example:

```
DIM X AS STRING
X=lcase$("KEES")      ' X is now "kees"

X=lcase$("Appel)      ' X is now "appel"
```

LEFT$ FUNCTION

Syntax : LEFT$(string, number of characters)

Return characters from string starting at first position on left of string for number of characters.

Example:

```
DIM X AS STRING
X=left$("Kees",3)     ' X is now "Kee"
```

LEN FUNCTION

Syntax : LEN(string)

Return length of string (number of characters in string).

Example:

```
DIM X AS INTEGER
X=len("abcde")     '  X now has the value 5
```

LTRIM$ FUNCTION

Syntax : LTRIM$(string)

Removes the spaces from the left side of a string.

Example:

```
DIM X AS STRING
X=ltrim$("    Kees     ")      ' X is now "Kees     "
```

MID$ FUNCTION

Syntax : MID$(string,start, number) MID$ also exists as a command.

Return characters from string starting before start position for number of characters.

Example:

DIM X AS STRING
X=mid$("abcdefg",3,2) ' X is now "cd"

MID$ COMMAND

Syntax: MID$ stringvariable, index = string

Replace in stringvariable, starting before index position, the string.

Example:

DIM x AS STRING
x="appel"
mid$(x,2)="ooo" ' x is now "aoool" – "ooo" replaces "ppe"

PART$ FUNCTION

Syntax: PART$(string, field character separator,, position)

Return characters in field before position of string up to field character separator.

Example:

DIM X AS STRING
X=part$("abc/def/ter","/",6) ' X is now "def"

REPLACE$ FUNCTION

Syntax : REPLACE$(string, characters to be replaced, position)

This function replaces characters in the STRING before the position.

Example:

```
DIM X AS STRING
X=replace$("abcdefg","zq",3)      ' X is now "abzqefg" – "zq" replaced "cd"
```

REPLACESUBSTR$ **FUNCTION**

Syntax : REPLACESUBSTR$(string, substring1, substring2)

In string. replace substring1 with substring2.

Example:

```
DIM X AS STRING
X=replacesubstr$("abcdebcfg","bc","/")      ' X is now "a/de/fg"
```

RIGHT$ FUNCTION

Syntax: RIGHT$(string, number of characters)

Return characters from string starting at first position on right of string for number of characters.

Example:

```
DIM X AS STRING
X=right$("Kees",3)    ' X is now "ees"
```

RINSTR FUNCTION

Syntax: RINSTR([start],string, character to find)

Returns first position of character found in string starting from right side. If start given, start find before start position. The function is capital letter sensitive and returns value 0 if character not found.

Example:

```
DIM X AS INTEGER
X=rinstr("Appel","p")      ' X is now 3

X=rinstr(3,"Appel","p")    ' X is now 2
```

ROUND$ FUNCTION

Syntax: ROUND$(string, number)

Return STRING with number of characters from left of string. If string length is smaller than number, then add number of spaces after characters in string.

Example:

```
DIM X AS STRING
X=round$(“abcdefter”,3)   ‘ X is now “abc”
X=round$(“ab”,5)          ‘ X is now “ab   ” -- “ab<sp><sp><sp>”
```

RTRIM$ FUNCTION

Syntax : RTRIM$(string)

Removes the spaces from the right side of a string.

Example:

```
DIM X AS STRING
X=rtrim$(“   Kees   ”)   ‘ X is now “   Kees”
```

SORT$ COMMAND

Syntax: SORT$ array(start TO end),type

Sort string array from index start to index end. If start is larger than end, sort in expiring order. Type is the following values:

- StString: Sort array as a string.
- StNum: Sort array numerically.

Example:

```
Dim x(100) as string
sort$ x(0 to 100),ststring   ‘ Sort array x from 0 to 100 as a string type sort
```

SPACE$ FUNCTION

Syntax: SPACE$(number)

Return STRING with number of space characters.

Example:

DIM X AS STRING
X=space$(3) ‘ X is now “   ” -- “<sp><sp><sp>”

STRING$ FUNCTION

Syntax: STRING$(ASCII character, number)

Syntax: STRING$(ASCII value, number)

Returns STRING with the ASCII character (or ASCII value) the number of times.

Example:

DIM X AS STRING
X=string$(“A”,6) ‘ X is now “AAAAAA”

X=string(65,3) ‘ X is now “AAA”

TALLY FUNCTION

Syntax: TALLY(string, substring)

Return number of times substring is found in string. The function is capital letter sensitive.

Example:

DIM X AS INTEGER
X=tally(“abcdea”,”a”) ‘ X now is value 2

X=tally(“abhhhhab”,”ab”) ‘ X now is value 2

TRIM$ FUNCTION

Syntax : TRIM$(string)

Removes the spaces on both sides of string.

Example:

DIM X AS STRING
X=trim$(“ Kees ”) ‘ X is now “Kees”

UCASE$ FUNCTION

Syntax: UCASE$(string)

Put STRING in capital characters (upper case).

Example:

DIM X AS STRING
X=ucase$("kees") ' X is now "KEES"

X=ucase$("Apple") ' X is now "APPLE"

SYSTEM COMMANDS/FUNCTIONS

APPICON COMMAND

Syntax : AppIcon

Assign an icon to the application. The icon must be 32 x 32 pixels and 16 colors

Example:

Appicon "c:/windows/myapp.ico"

APPNAME$ FUNCTION

Syntax : APPNAME$

Returns filename and path of the application.

Example:

DIM x AS STRING
X=appname$ ' X is now for example "c:\test\app.exe".

BEEP COMMAND

Syntax : BEEP

The computer gives a sound signal.

Example:

BEEP

CHDIR COMMAND

Syntax: CHDIR name

Make name the current directory. Name is of the STRING type.

Example:

CHDIR "c:\windows\test"

COMMAND$ FUNCTION

Syntax : COMMAND$

Returns parameter string, which has been given at the call of the program.

Example:

C:\test.exe /apple

DIM x AS STRING
X=command$ ' X is now " /apple".

DATE$ FUNCTION

Syntax : DATE$(integer)

Return system date as a string. INTEGER has the following values:

- Dtsystem: Give date dependant on country.
- Dtdecode: Always give date in format DD-MM-YYYY

Example:

DIM x AS STRING
X=date$(dtdecode) ' X is now "28-12-1999"

DIR$ FUNCTION

Syntax : DIR$(string)

Return current directory of given drive in STRING. Function is capital letter sensitive.
If the STRING is @ then the current disk is used.

Example:

```
DIM x AS STRING
X=dir$("A")    ' X is now for example "A:\test\ok"
```

DIREXISTS FUNCTION

Syntax : DIREXISTS(string)

Returns false if directory mentioned in STRING does not exist.

Example:

```
DIM x AS Boolean
X=direxists("c:\test")    ' X now false or true depending on whether directory exists or not.
```

DISKFREE FUNCTION

Syntax: DISKFREE(string)

Returns unused number of bytes in drive character in STRING.

Example:

```
DIM x AS INTEGER
X=diskfree("c")    ' X now is number of free bytes on disk c.
```

DOEVENTS COMMAND

Syntax : DOEVENTS

Settles processes of windows and returns to program. Useful in long-term loops.

Example:

```
FOR t=1 TO 1000000
  Do something
  DOEVENTS
NEXT t
```

FILEEXISTS FUNCTION

Syntax: FILEEXISTS(string)

Returns false if file mentioned in STRING does not exist.

Example:

```
DIM x AS Boolean
X=fileexists("c:\test\b.exe")    ' X now false or true depending on whether file exists or not.
```

FILEINFO$ FUNCTION

Syntax : FILEINFO$(integer)

FILEINFO$ gives further information of file find with the FINDFIRST and FINDNEXT command.
THE INTEGER can have the values:

- 1 -- fileinfo$ returns the date
- 2 -- fileinfo$ return the time
- 3 -- fileinfo$ return the size of the
- 4 -- fileinfo$ return the attribute
- 5 -- fileinfo$ return the handle of the file

Example:

```
DIM x AS STRING
X=findfirst$(c:\test\*.*)
X=fileinfo$(3)    ' X is now "622"
```

FINDFIRST$ FUNCTION

Syntax: FINDFIRST$(string)

Returns name of first found file or directory. The STRING is path with wildcards -- if no path given, current directory is used. FINDNEXT$ is used to read following file names. If no file is found, result is empty string.

Example:

```
DIM x AS STRING
X=findfirst$(c:\test\*.*)
```

FINDNEXT$ FUNCTION

Syntax: FINDNEXT$

FINDNEXT$ gives name of file after use of FINDFIRST$.
The parameters in function FINDFIRST$ are used for this.

Example:

```
DIM x AS STRING
X=findfirst$(c:\test\*.*)
X=findnext$
```

INP FUNCTION

Syntax : INP(port)

Returns a value between &H00 and &HFF of a port read.
Port has a value between $H0000 and &HFFFF.

Example:

```
X=INP(10)  ‘ Reads port 10 of computer.
```

KILL COMMAND

Syntax : KILL filename

Removes a file from a drive.

Example:

```
KILL “c:\test.exe”
```

MESSAGEDLG FUNCTION

Syntax: MESSAGEDLG(text,caption , buttons+type)

Gives message box. The title and the caption are of a string type.

The values of the type box are:

- mbIconAsterisk
- mbIconExlamation
- mbIconHand
- mbIconQuestion

The values for the buttons are:

- mbOk
- mbOkCancel
- mbYesNo
- mbYesNoCancel
- mbAbortRetryIgnore
- mbRetryCancel

These values can be added.

Example:

```
DIM x AS INTEGER
X=messagedlg("Now stop?","Stop?",mbyes + mbiconquestion)
```

This gives messagebox with yes and a no button.

X may have the following values:

- IdYes 6
- IdNo 7
- IdOk 1
- IdCancel 2
- IdAbort 3
- IdRetry 4
- IdIgnore 5

Example:

```
IF messagedlg("Now stop?","Stop?",mbyesno) = idyes THEN
    'it has been clicked on the yes button.
ELSE
    'it has been clicked on the no button.
END IF
```

MKDIR$ FUNCTION

Syntax: MKDIR(string)

Create new directory. STRING contains the name of the new directory.
If no path is given, new directory is created in the current directory.

Example:

```
Mkdir$("c:\test\oke")
```

MOUSEX FUNCTION

Syntax : MOUSEX

Return horizontal position of the mouse.

Example:

```
DIM x AS INTEGER
X=mousex
```

MOUSEY FUNCTION

Syntax: MOUSEY

Return vertical position of the mouse.

Example:

```
DIM y AS INTEGER
Y=mousey
```

OPEN COMMAND

Syntax: OPEN filename, show

A file opens using a windows application. Filename is of the TYPE string.
Show can have the next possible value:

- SW_HIDE: the application hides.
- SW_SHOWNORMAL: shows the application

- SW_SHOWMINIMIZED: the application reduces to the taskbar.
- SW_SHOWMAXIMIZED: increased the application to the size of the screen.
- SW_SHOWNOACTIVATE: the application shows but application does not have focus.
- SW_SHOW: the application shows
- SW_MINIMIZE: reduces the application to the taskbar
- SW_SHOWMINNOACTIVE: the application does not show as active.
- SW_SHOWNA: the application shows but the application does not have the focus.
- SW_RESTORE: shows application and repaint.
- SW_SHOWDEFAULT: shows the application

Example:

OPEN “mydoc.txt”,SW_shownormal

Opens mydoc.txt using notepad.

OUT FUNCTION

Syntax: OUT(port, value)

A value between &H00 and &HFF is sent to a port.
Port is value between $H0000 and &HFFFF.

Example:

Out(10,2) ‘ Sends a 2 to port 10 of the computer.

PRINTFILE COMMAND

Syntax: PRINTFILE filename,show

A file prints application-using windows. File name is of TYPE string. Show is possible next value:

- SW_HIDE: the application hides.
- SW_SHOWNORMAL: shows the application
- SW_SHOWMINIMIZED: the application reduces to the taskbar.
- SW_SHOWMAXIMIZED: increased the application to the size of the baffle.
- SW_SHOWNOACTIVATE: the application shows but the application does not have focus.
- SW_SHOW: the application shows
- SW_MINIMIZE: reduces the application to the taskbar
- SW_SHOWMINNOACTIVE: the application does not show as active.
- SW_SHOWNA: the application shows but the application does not have the focus.

- SW_RESTORE: the baffle and shows application to repaint.
- SW_SHOWDEFAULT: shows the application

Example:

PRINTFILE “mydoc.txt”,SW_shownormal ‘ The mydoc.txt file prints using notepad.

RENAME COMMAND

Syntax: RENAME old name, new name

Change the name of a file to another name.

Example:

RENAME “c:\windows\test.exe”,”c:\windows\test.txt”

RENAME “c:\windows\test.exe”,”c:\test.exe” ‘ Changed not name but moves the file.

RMDIR COMMAND

Syntax: RMDIR name

Removes a directory, which is empty.

Example:

RMDIR ”c:\windows\test”

RUN COMMAND

Syntax: RUN file name [,parameter,show]

Calls another program and waits until has stopped. Parameter is STRING type and is parameter given to the called program. Show may be the next value:

- SW_HIDE: the application hides.
- SW_SHOWNORMAL: shows the application
- SW_SHOWMINIMIZED: the application reduces to the taskbar.
- SW_SHOWMAXIMIZED: increased the application to the size of the baffle.
- SW_SHOWNOACTIVATE: the application shows but the application does not have the focus.
- SW_SHOW: the application shows

- SW_MINIMIZE: reduces the application to the taskbar
- SW_SHOWMINNOACTIVE: the application does not show as active.
- SW_SHOWNA: the application shows but the application does not have the focus.
- SW_RESTORE: the baffle and shows application to repaint.
- SW_SHOWDEFAULT: shows the application

Example:

RUN "c:\windos\notepad.exe","mydoc.txt",SW_shownormal ' Show mydoc.txt using notepad.

SCREENHEIGHT FUNCTION

Syntax: SCREENHEIGHT

Return height (in pixels) of display device.

Example:

DIM x AS INTEGER
X=screenheight

SCREENWIDTH FUNCTION

Syntax : SCREENWIDTH

Return width (in pixels) of display device.

Example:

DIM x AS INTEGER
X=screenwidth

SHELL FUNCTION

Syntax: Handle=SHELL file name, parameter, show

Another application is called and continues with the next line of the program.

Parameter is STRING type and is parameter given to called program.
Show may be value:

- SW_HIDE: the application hides.
- SW_SHOWNORMAL: shows the application
- SW_SHOWMINIMIZED: the application reduces to the taskbar.

- SW_SHOWMAXIMIZED: increased the application to the size of the baffle.
- SW_SHOWNOACTIVATE: the application shows but application does not have focus.
- SW_SHOW: the application shows
- SW_MINIMIZE: reduces the application to the taskbar
- SW_SHOWMINNOACTIVE: the application does not show as active.
- SW_SHOWNA: the application shows but the application does not have the focus.
- SW_RESTORE: the baffle and shows application to repaint.
- SW_SHOWDEFAULT: shows the application

Example:

```
DIM handle AS INTEGER
Handle=shell “c:\windos\notepad.exe”,”mydoc.txt”,SW_shownormal
```

Show file mydoc.txt using notepad.

SHOWMESSAGE COMMAND

Syntax : SHOWMESSAGE text

Simple windows message box showing given text.

Example:

```
SHOWMESSAGE “Error occurred!”
```

SLEEP COMMAND

Syntax: SLEEP milliseconds

The program waits until the number of given millisecond(s) is past.

Example:

SLEEP 1000 : 1 second waits.
SLEEP 500 : a half second waits.

TIME$ FUNCTION

Syntax: TIME$

Return the current system time. The format is HH:MM:SS.

Example:

```
DIM x AS STRING
X=time$      ' X now for example has the value "10:26:09"
```

TIMER FUNCTION

Syntax : TIMER

Give number of second(s) that has expired after midnight.

Example:

```
DIM X AS double, t AS INTEGER
X=timer
FOR t=1 TO 10000
  PRINT t
NEXT t

X=timer-x      ' X now contains the number of seconds that the for-next loop has used.
```

TYPES – VARIABLES

VARIABLES OVERVIEW

NAME	VALUE	FORMAT
DOUBLE	5.0 x 10^–324. 1.7 x 10^308	8 BYTES
SINGLE	1.5 x 10^–45. 3.4 x 10^38	4 BYTES
INTEGER	–2147483648..2147483647	4 BYTES
WORD	0..65535	2 BYTES
SHORT	–32768..32767	2 BYTES
BYTE	0..255	1 BYTE
STRING	0..2 gigabyte	-------
BOOLEAN	FALSE or TRUE	-------

DOUBLE and SINGLE can contain decimal places separated by a point.
INTEGER, WORD, SHORT and BYTE are whole numbers.

Example:

```
DIM a AS DOUBLE
A=12.7256
```

BOOLEAN

Boolean has a value of TRUE or FALSE.

Example:

```
DIM existY AS BOOLEAN
existY =true

IF existY=true THEN PRINT "y exists"
```

BYTE

BYTE has value between 0 and 255. The format is 8 bits, in addition 1 byte.

DOUBLE

DOUBLE has value between 5.0 x 10^–324 and 1.7 x 10^308.
The format is 64 bits in addition 8 bytes.

INTEGER/LONG

INTEGER has value between – 2147483648 and 2147483647.
The format is 32 bits as well as 4 bytes.

SHORT

SHORT has value between – 32768 and 32767.
The format is 16 bits as well as 2 bytes.

SINGLE

SINGLE has value between 1.5 x 10^−45 and 3.4 x 10^38.
The format is 32 bits as well as 4 bytes.

STRING

A string can contain characters with length between 0 and 2 Gigabytes.

WORD

WORD has value between 0 and 65535. The format is 16 bits as well as 2 bytes.

TYPES – WINDOWS OBJECTS

APPLICATION

APPLICATION	
Properties:	
Cursor(cursor)	***STRING***. STRING contains the file name of the new cursor, which must replace the old cursor. Cursor could have the next possible value: **crDefault** **crNone** **crArrow** **crCross** **crIBeam** **crSizeNESW** **crSizeNS** **crSizeNWSE** **crSizeWE** **crUpArrow** **crHourGlass** **crDrag** **crNoDrop** **crHSplit** **crVSplit** **crMultiDrag**

	crSQLWait **CrNo** **CrAppStart** **CrHelp** **CrHandPoint** **crSizeAll**
ExeName	***STRING.*** The name of the application without the path.
Handle	***INTEGER.*** Windows handle of the application.
HintColor	***INTEGER.*** The Color of all hints in the program.
HintHidePause	***INTEGER.*** The time, which must expire until the hint disappears. The time is in milliseconds.
HintPause	***INTEGER.*** The time, which must expire before the hint appears. The given time is in milliseconds.
HintShortPause	***INTEGER.*** The time, which expires between two hints. Time in milliseconds.
ShowHint	***BOOLEAN.*** If true in the application no more hints are shown.
Methods:	
HintActivate(X,Y)	Shows the hint on position x,y.
Terminate	The program concludes.

BUTTON

BUTTON	
Properties:	
BMP	***STRING.*** File name of the BMP that must be shown on the button.
Caption	***STRING***. The text, which must be shown on the button.
Enabled	***BOOLEAN***. Decide if the button can be used or not.
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color or the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the button.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the button.
Kind	Kind, the next value property: ***BkCustom***:standard button ***BkOk***:okay ***BkCancel***: to annulated ***BkYes***: yes ***BkNo***: no ***BkHelp***: help

	BkClose: to close ***BkAbort***: to demolish ***BkRetry***:again try ***BkIgnore***:to ignore ***BkAll***
Layout	Layout is possible the next value has: ***BlGlyphLeft***:bmp left of the text ***blGlyphRight***:bmp Right of the text ***blGlyphTop***:bmp above the text ***blGlyphBottom***:bmp under the text
Left	***INTEGER***. Left position of the button.
Parent	***OBJECT:*** The parent or the button
Showhint	***BOOLEAN***. True shows the hint.
Taborder	***BOOLEAN***. The number for use with the tab key, if Taborder is activated.
Tag	***INTEGER***. Defining identification number for the user.
Top	***INTEGER***. The upper position or the button.
Visible	***BOOLEAN***. False, the button is not visible.
Width	***INTEGER***. The breadth of the button.
Methods:	
BMPResource(resource)	The same as property bmp but now it shows the RESOURCE file.
SetFocus	The gives the focus to the button.
Focused	True, if the button has the focus.

Events	
OnClick	**SUB([button])**. A sub-routine calls if there is clicked on the button.
OnKeyPress	**Sub([button]).** A sub-routine calls if a key is pressed.
OnMouseDown	**Sub([button,] button, X, Y, shift).** A sub-routine calls if a mouse button is pressed.
OnMouseMove	**Sub([button,] X, Y, shift).** A sub-routine calls if the mouse moves over the button.
OnMouseUp	**Sub([button,] button, X, Y, shift).** A sub-routine calls if mouse button is released.

CHECKBOX

CHECKBOX	
Properties:	
Caption	***STRING***. The text, which must be shown beside the CHECKBOX.
Checked	***BOOLEAN.*** True if the CHECKBOX is marked.
Color	***INTEGER***. Color of the CHECKBOX.
Cursor	Cursor can have one of the next values: **crDefault** **CrNone** **CrArrow** **CrCross** **CrIBeam** **CrSizeNESW**

	CrSizeNS **CrSizeNWSE** **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop** **CrHSplit** **CrVSplit** **CrMultiDrag** **CrSQLWait** **CrNo** **CrAppStart** **CrHelp** **CrHandPoint** **crSizeAll**
Enabled	***BOOLEAN***. Decided if the CHECKBOX can be used or not.
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color or the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the CHECKBOX.

Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the CHECKBOX.
Left	***INTEGER***. Left position of the CHECKBOX.
Parent	***OBJECT:*** The parent of the CHECKBOX
Showhint	***BOOLEAN***. True shows the hint.
Taborder	***BOOLEAN***. The number for use with the tab key, as Taborder is enabled.
Tag	***INTEGER***. By defining identification number for the user.
Top	***INTEGER***. The upper position of the CHECKBOX.
Visible	***BOOLEAN***. False, the CHECKBOX is not visible.
Width	***INTEGER***. The breadth of the CHECKBOX.
Methods:	
SetFocus	The CHECKBOX gets the focus.
Focused	True, if the CHECKBOX has the focus.
Events	
OnClick	**SUB([button])**. A sub-routine calls if there is clicked on the CHECKBOX.
OnMouseMove	**SUB([button,] shift, X, Y).** A sub-routine calls if the mouse moves over the CHECKBOX.

CHECKLISTBOX

CHECKLISTBOX	
Properties:	
Align	Align can have the next values: ***AlNone***: CHECKLISTBOX standard is shown. ***AlTop***: CHECKLISTBOX to the upper part of its parent it is put down. ***alBottom***: CHECKLISTBOX to the lower part of its parent it is put down. ***alLeft***: CHECKLISTBOX of its parent it is at the left-hand side put down. ***alRight***: CHECKLISTBOX of its parent it is at the right-hand side put down. ***alClient***: CHECKLISTBOX adapts to the format of the parent.
BorderStyle	***BsNone***. No border visible. ***BsSingle***. Border visible.
Color	***INTEGER.*** Background Color of CHECKLISTBOX.
Cursor	Cursor is possible the next value has: **crDefault** **crNone**

	crArrow **crCross** **crIBeam** **crSizeNESW** **crSizeNS** **crSizeNWSE** **crSizeWE** **crUpArrow** **crHourGlass** **crDrag** **crNoDrop** **crHSplit** **crVSplit** **crMultiDrag** **crSQLWait** **crNo** **crAppStart** **crHelp** **crHandPoint** **crSizeAll**
Enabled	***BOOLEAN***. Decided if CHECKLISTBOX can be used or not.
Flat	***BOOLEAN.*** If flat is true then the checkbox having a flat framework.
FontBold	***BOOLEAN***. True give fat text.

FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. Establishes the altitude of CHECKLISTBOX.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true activate these.
HWND	***INTEGER.*** Windows handle of CHECKLISTBOX.
Item(index)	***STRING.*** Index is of the INTEGER TYPE. With item the text can be given up or read. Index is the index of the item (0 is the first item).
ItemChecked(index)	***BOOLEAN.*** Index is of the INTEGER TYPE. Marked. Index 0 is the first item.
ItemCount	***INTEGER***. Itemcount are the number of items in CHECKLISTBOX.
ItemIndex	***INTEGER.*** Itemindex the index of the late item has been chosen. If no item has been selected, the itemindex – 1.
Left	***INTEGER***. Left position of CHECKLISTBOX.

Parent	***OBJECT.*** The parent of CHECKLISTBOX
Showhint	***BOOLEAN***. True shows the hint.
Sorted	***BOOLEAN.*** As sorted is true the list is sorted on alphabetical order.
Taborder	***BOOLEAN***. The number for use with the tab key.
Tag	***INTEGER***. By defining identification number for the user.
Top	***INTEGER***. The upper position of CHECKLISTBOX.
Visible	***BOOLEAN***. If false the CHECKLISTBOX not visible.
Width	***INTEGER***. The breadth of CHECKLISTBOX.
Methods:	
Clear	Clear all items of CHECKLISTBOX so that these can be used again.
Focused	Is true if CHECKLISTBOX has the focus.
Repaint	Paint the CHECKLISTBOX all over again.
SetFocus	Gives the CHECKLISTBOX the focus.
Events	
OnClick	**SUB([checklistbox])**. A sub-routine calls if it is clicked there on the button.

OnKeyDown	**SUB([checklistbox,] key, shift).** A sub-routine calls if a key is pressed.
OnMouseMove	**SUB([checklistbox,], shift, X, Y).** A sub-routine calls if the mouse moves over the CHECKLISTBOX.

CLIENTSOCKET

CLIENTSOCKET	
Properties:	
Active	***BOOLEAN.*** True make the connection active.
Address	***STRING.*** Here the ip address of the server can be given up.
ClientType	Can contain the next values: ***CtNonBlocking the*** program goes further. ***CtBlocking the*** program waits until CLIENTSOCKET is ready.
Connected	***BOOLEAN.*** Gives to if there is a connection or not.
Host	***STRING.*** Here a domain name can be given.
Port	***INTEGER.*** Port number can be given up.
Service	***STRING.*** The service can be given up for the socket. Sees Microsoft documentation for windows sockets.

Methods:	
Close	A connection concludes.
OPEN	Established a connection with the socket makes.
Receive	STRING returns the received text.
Send(Text)	Returns 0 if the text has successfully sent. Another INTEGER number return if the sending has postponed.
Events	
OnConnected	**Sub()**: A sub-routine call if the connection has come about.
OnDisconnect	**Sub().** A sub-routine calls if the connection on the point score concludes.
OnError	**Sub(error code).** A sub-routine calls if there wrongly act the error code is an INTEGER number.
OnRead	**Sub().** A sub-routine call as the socket text receives.

COLORDIALOG

COLORDIALOG	
Properties:	
AnyColor	***BOOLEAN.*** If this value is true the user can choose each color.
Color	***INTEGER.*** Reads and writes the Color witch has been selected.

CustomColor(index)	***INTEGER.*** Write and reads the 16 colors defined by the user. Index is a number or 0 to 15.
FullOpen	***BOOLEAN.*** A full Color dialog for the user.
PreventFullOpen	***BOOLEAN.*** Ensures that the user cannot define the user colors.
SolidColor	***BOOLEAN.*** If these are true the fixed Color becomes which uses the dense at the chosen Windows Color.
Methods:	
Show	This shows the Color dialog.

COMBOBOX

COMBOBOX	
Properties:	
Color	***INTEGER.*** The background Color of the COMBOBOX.
Cursor	Cursor is possible the next value has: **crDefault** **crNone** **CrArrow** **CrCross** **CrIBeam** **CrSizeNESW** **CrSizeNS**

	CrSizeNWSE **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop** **CrHSplit** **CrVSplit** **CrMultiDrag** **CrSQLWait** **CrNo** **CrAppStart** **CrHelp** **CrHandPoint** **CrSizeAll**
DropDownCount	***INTEGER***: The number or rules such as the COMBOBOX folds out.
EDIT	***BOOLEAN.*** If this is true then user can input text.
Enabled	***BOOLEAN***. Decided if the COMBOBOX can be used or not.
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color or the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.

FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the COMBOBOX establishes.
Hint	***STRING***. The text, which must be reflected if the hint are active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the COMBOBOX.
Item(index)	***STRING.*** The text of an item with the given up index returns, or establishes him.
ItemCount	***INTEGER***. The number or items in the COMBOBOX.
ItemIndex	***INTEGER.*** The chosen number from the trick gives, the first item are 0. If no item are chosen, the itemindex is – 1.
Left	***INTEGER***. Left position of the COMBOBOX.
Parent	***Object:*** The parent of the COMBOBOX
Showhint	***BOOLEAN***. True shows the hint.
Sorted	***BOOLEAN***: True sorts the COMBOBOX.
Taborder	***BOOLEAN***. The number for use with the tab key, if tab order is activated.
Tag	***INTEGER***. Defining an identification number for the user.
Text	***STRING:*** the content of the COMBOBOX.
Top	***INTEGER***. The upper position of the

	COMBOBOX.
Visible	***BOOLEAN***. False if the COMBOBOX is not visible.
Width	***INTEGER***. The breadth of the COMBOBOX.
Methods:	
Clear	Clears the COMBOBOX so that the COMBOBOX can be used again.
SetFocus	Gives the COMBOBOX the focus.
Focused	True if the COMBOBOX has the focus.
Events	
OnChange	**Sub([combobox])**. A sub-routine calls if the content is modified.
OnDropDown	**Sub([combobox]).** A sub-routine calls if the trick folds out.
OnKeyDown	**Sub([combobox,] key, shift).** A sub-routine calls if a key is pressed.

EDIT

EDIT	
Properties:	
Autoselect	***BOOLEAN.*** EDIT gets the focus.
BorderStyle	***BsNone***. No border visible. ***BsSingle***. Border visible.

CharCase	Charcase can have the next value: ***EcNormal:*** normal signs ***EcUppercase***:only capital characters ***EcLowercase:***only small characters
Color	***INTEGER.*** The Color of the EDIT.
Cursor	Cursor can have the next value: **CrDefault** **CrNone** **CrArrow** **CrCross** **CrIBeam** **CrSizeNESW** **CrSizeNS** **CrSizeNWSE** **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop** **CrHSplit** **CrVSplit** **CrMultiDrag** **CrSQLWait** **CrNo**

	CrAppStart **CrHelp** **CrHandPoint** **CrSizeAll**
Enabled	***BOOLEAN***. Decided if the EDIT can be used or not.
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT establish for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the EDIT.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the EDIT.
Left	***INTEGER***. Left position of the EDIT.
MaxLength	***INTEGER.*** Maximum number of to fill in signs.
Modified	***BOOLEAN.*** True, if the text has changed.
Parent	***OBJECT.*** The parent of the EDIT.
PassWordChar	***BOOLEAN.*** As this true is then another sign are placed there during typing, to occur to read.

Showhint	***BOOLEAN***. True shows the hint.
Taborder	***BOOLEAN***. The number for use with the tab key, if Taborder is activated.
Tag	***INTEGER***. Defining identification number for the user.
Text	***STRING***. The text of the EDIT.
Top	***INTEGER***. The upper position of the EDIT.
Visible	***BOOLEAN***. If false the EDIT is not visible.
Width	***INTEGER***. The breadth of the EDIT.
Methods:	
SetFocus	The EDIT gets the focus.
Focused	EDIT has the focus.
Events	
OnClick	**Sub([edit])**. A sub-routine calls if there is clicked on the EDIT.
OnChange	**Sub([edit]).** A sub-routine calls if the contents are modified.
OnKeyDown	**Sub([edit,] key, shift).** A sub-routine calls if a key is pressed.
OnKeyUp	**Sub([edit,] key, shift).** A sub-routine calls if a key is released.
OnKeyPress	**Sub([edit]).** A sub-routine calls if a key is pressed.

FACEBUTTON

FACEBUTTON	
Properties:	
Align	Align can have the next value: ***AlNone***: The FACEBUTTON is shown standard. ***AlTop***: The FACEBUTTON is well down to the upper part of its parent. ***AlBottom: The*** FACEBUTTON is well down to the lower part of its parent. ***AlLeft: The*** FACEBUTTON is at the left-hand side drawn of its parent. ***AlRight: The*** FACEBUTTON is at the right-hand side drawn of its parent. ***AlClient: The*** FACEBUTTON adapts to the format of the parent.
AllowAllUP	***BOOLEAN***: All unused FACEBUTTONS are up.
BMP	***STRING.*** File name of the BMP file that must be shown at the BUTTON.
Caption	***STRING***. The text, which must be shown on the BUTTON.
Cursor	Cursor can have the next value: **CrDefault** **CrNone** **CrArrow** **CrCross** **CrIBeam**

	CrSizeNESW **CrSizeNS** **CrSizeNWSE** **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop** **CrHSplit** **CrVSplit** **CrMultiDrag** **CrSQLWait** **CrNo** **CrAppStart** **CrHelp** **CrHandPoint** **crSizeAll**
Enabled	***BOOLEAN***. Decided if the BUTTON can be used or not.
Flat	***BOOLEAN.*** If the value is true, then there is difference between the active and not active BUTTON.
FontBold	***BOOLEAN***. True is fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True gives slanting pressed text.

FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the BUTTON establishes.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting SHOWHINT on true must activate these.
Layout	LAYOUT can have the next value: ***BlGlyphLeft***: Blade left of the text ***BlGlyphRight: Blade*** Right of the text ***BlGlyphTop: Blade*** above the text ***BlGlyphBottom: Blade*** under the text
Left	***INTEGER***. Left position of the button.
NumBMP	***INTEGER.*** The number or blades of what is in the BMP file, e.g. it active and inactive blade.
Parent	***OBJECT:*** The parent of the button
Showhint	***BOOLEAN***. True the hint shows.
Taborder	***BOOLEAN***. The number for use with the tab key, if Taborder is activated.
Tag	***INTEGER***. Identification number for the user.
Top	***INTEGER***. The upper position of the button.
Visible	***BOOLEAN***. False, if the button is not visible.

Width	***INTEGER***. The breadth of the bud.
Methods:	
BMPResource(resource)	The same as property BMP but is now showing the RESOURCE file.
Events	
OnClick	**SUB([button])**. A sub-routine calls if there is clicked on the mouse button.
OnMouseMove	**SUB([button,], shift, X, Y).** A sub-routine calls if the mouse moves over the button.

FILEDIALOG

FILEDIALOG	
Properties:	
Caption	***STRING***. Title of the dialog.
FileName	***STRING.*** The file name which appear in the dialog or the file name which the user has filled in.
Filter	***STRING.*** The filter that on the file list is applied, one can use with wild carts. The indexes are separated by the pipe symbol. E. g. filter= "all|*.*|text|* .txt"
FilterIndex	***INTEGER.*** The number of the index of the filter, which appears standard in the dialog.

HWND	***INTEGER.*** The Windows handle of the dialog.
InitialDir	***STRING.*** The directory, which are used in the file dialog.
OldStyle	***BOOLEAN.*** The old style dialog reflects if this value is true.
Methods:	
Show	This will shows file dialog.
Events	
OnFolderChange	**SUB().** A sub-routine calls if the user changes the directory.

FILESTREAM

FILESTREAM	
Properties:	
HWND	***INTEGER.*** Windows handle of the FILESTREAM.
LineCount	***INTEGER.*** The total number or lines in the FILESTREAM. A line is separated by a linefeed (chr$ (13) +chr$ (10)
Position	***INTEGER.*** Position of the pointer within the filestream, reading and writing happens from this position.
Size	***INTEGER.*** Total size of the FILESTREAM in bytes.

Methods:	
Close	The present open files closes that are used by the filestream, so that the FILESTREAM can be used with a new file.
LoadResource(resource)	RESOURCE is a RESOURCE file; this file is placed in its whole into the FILESTREAM.
Open(filename)	File name is of the STRING TYPE and contain the name of a file. If the file does not exist it is produced. The file must be opened firstly to use.
Read(aantalbytes)	Number of bytes is of the INTEGER TYPE. The function returns the number or read bytes in STRING. E. g. S=filestream.read (10)
ReadLine	STRING Returns the first next line of the FILESTREAM. The lines must be separated by linefeed.
Write(tekst)	Text is of the TYPE STRING. Write writes the STRING to the opened file. With conversion functions numerical values can be written to the file.

FONTDIALOG

FONTDIALOG	
Properties:	
AnsiOnly	***BOOLEAN.*** Only the ansi fonts are reflected within the dialog.
ApplyButton	***BOOLEAN.*** Applies button reflects if the value is true.

Effects	***BOOLEAN.*** Effects menu reflects if the value is true.
FixedPitch	***BOOLEAN***. Reflects only the fonts with a fixed breadth in the menu.
FontBold	***BOOLEAN***. If this is marked the value is true.
FontColor	***BOOLEAN***. The chosen Color of the FONT returns.
FontExists	***BOOLEAN.*** If the value is true there only fonts can be filled in which are in the trick.
FontItalic	***BOOLEAN***. If this is marked the value is true.
FontName	***STRING***. Returns the font name.
FontSize	***INTEGER***. Returns the size of the signs.
FontStrikeout	***BOOLEAN.*** If this is marked the value is true.
FontUnderline	***BOOLEAN***. If this is marked the value is true.
LimitSize	***BOOLEAN.*** This value must be true to use the maxfontsize and the minfontsize.
MaxFontsize	***INTEGER.*** Puts or the maximum size of the fonts. Only works if the limitsize is true.
MinFontsize	***INTEGER.*** Puts the minimum size or of the fonts. Only works if the limitsize is true.
NoFaceSel	***BOOLEAN.*** The dialog without the FONT examples.

NoOEMFonts	***BOOLEAN.*** If this value is true the trick or the dialog is shown without oem fonts.
NoSimulation	***BOOLEAN.*** If this value is true only fonts are shown which are within the definition file.
ScalableOnly	***BOOLEAN.*** If this value is true the bitmap fonts of the trick are removed.
TrueTypeOnly	***BOOLEAN.*** If this value is true then only the true TYPE fonts are shown in the trick.
Wysiwyg	***BOOLEAN.*** If this value is true all fonts are shown which the PRINTER and the posting device support.
Methods:	
Show	The fontdialoog shows.
Events	
OnApply	**SUB().** A sub-routine calls if a choice is made using the dialog.

FORM

FORM	
Properties:	
Autoscroll	***BOOLEAN***. Shows the scrollbars if the form is too small.
AutoSize	***BOOLEAN***. The format adapts to the contents.

BorderIcons	***BiMinimize*** OR ***BiMaximize*** OR ***BiHelp***. Shows the bordericons.
Bordestyle	***BsNone***. No border visible. ***BsSingle***. Not possible to set the dimensions by the user. ***BsSizeable***. Possible to set the dimensions by the user. ***BsDialog***. As a dialog box. ***BsToolWindow***. Tool form with a small caption. ***BsSizeToolWin***. As a tool window, dimensions adjustable.
Caption	***STRING***. Title of the form.
Center	***BOOLEAN***. The form centres on the screen.
ClientHeight	***INTEGER***. Altitude of usable form.
ClientWidth	***INTEGER***. Breadth of useable form.
Color	***INTEGER***. Color of the form.
Focused	***BOOLEAN***. True if the form has the focus.
Height	***INTEGER***. Total altitude of the form.
Hint	***STRING***. Text of the hint if the mouse comes on the form.
HWND	***INTEGER***. The handle of the form for use with an api call.
IconFile	***STRING***. File name of the icon for the form.
Left	***INTEGER***. Left position of the form.
Parent	***OBJECT.*** Assigns the parent to the form**.**
Showhint	***BOOLEAN***. True shows the hint.

Top	***INTEGER***. Upper position or the form.
Visible	***BOOLEAN***. False hides the form.
Width	***INTEGER***. The breadth of the form.
Windowstate	***WsNormal***. The form is reflected normally. ***wsMinimized***. The form minimizes. ***wsMaximized***. The form maximizes.
Methods:	
BMPResource(x1,y1,x2,y2,resource)	Draws a resource on the form. The uperleft corner is x1,y1. The right down corner is x2,y2. This function succeeds only if the form is visible.
DrawBMP(x1,y1,x2,y2,filename)	Draws a bmpfile on the form. The uperleft corner is x1,y1. The right down corner is x2,y2. This function succeeds only if the form is visible.
IconResource(resource).	Like property iconfile but now uses the RESOURCE file.
PRINT	The form prints.
Repaint	Repaints the form.
SetFocus	The form gets the focus
Show	The form shows and goes further with the program.
Showmodal	It shows the form and goes just further with the program if the form is concluded.
Events	
OnActivate	**SUB([form]).** A sub-routine calls if the form gets the focus.
OnClick	**SUB([form])**. A sub-routine calls if there is

	clicked on the form.
OnClose	**SUB([form,] action AS INTEGER)**. A sub-routine calls if the form is concluded. In the sub-routine a value can be given: 0:caNone the form cannot be concluded. 1:caHide the form are not concluded but hid. 2:caFree the form and the used memory are made free. 3:caMinimize the form is minimized.
OnDeactivate	**SUB([form]).** A sub-routine calls if the form loses the focus.
OnMouseDown	**SUB([form,] button, shift, X, Y).** A sub-routine calls if a mouse button is pressed.
OnMouseMove	**SUB([form,] shift, X, Y).** A sub-routine calls if the mouse moves over the form.
OnMouseUp	**SUB([form,] button, shift, X, Y).** A sub-routine calls if a mouse button is released.
OnPaint	**SUB([form]).** A sub-routine calls if the form is repaint.
OnResize	**SUB([form]).** A sub-routine calls if the dimensions of the form are modified.
OnShow	**SUB([form]).** A sub-routine calls if the form is shown.

GROUPBOX

GROUPBOX	
Properties:	
Caption	***STRING***. The title of the GROUPBOX.
Color	***INTEGER.*** The background Color of the GROUPBOX.
Cursor	Cursor can have the next possible value: **crDefault** **crNone** **crArrow** **crCross** **crIBeam** **crSizeNESW** **crSizeNS** **crSizeNWSE** **crSizeWE** **crUpArrow** **crHourGlass** **crDrag** **crNoDrop** **crHSplit** **crVSplit** **crMultiDrag**

	crSQLWait **crNo** **crAppStart** **crHelp** **crHandPoint** **crSizeAll**
Enabled	***BOOLEAN***. Decided if the GROUPBOX can be used or not.
FontBold	***BOOLEAN***. True gives fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the GROUPBOX.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the GROUPBOX.
Left	***INTEGER***. Left position of the GROUPBOX.
Parent	***OBJECT:*** The parent of the GROUPBOX
Showhint	***BOOLEAN***. True shows the hint.

Taborder	***BOOLEAN***. The number for use with the tab key, if taborder is activated.
Tag	***INTEGER***. Defining an identification number for the user.
Top	***INTEGER***. The upper position of the GROUPBOX.
Visible	***BOOLEAN***. False if the GROUPBOX is not visible.
Width	***INTEGER***. The breadth of the GROUPBOX.
Methods:	
Repaint	The GROUPBOX and the contents are repainting.
Events	
OnClick	**SUB([groupbox])**. A sub-routine calls if there is clicked on the form.
OnMouseDown	**SUB([groupbox,] button, shift, X, Y).** A sub-routine calls if a mouse button is pressed.
OnMouseMove	**SUB([groupbox,] shift, X, Y).** A sub-routine calls if the mouse moves over the GROUPBOX.
OnMouseUp	**SUB([groupbox,] button, shift, X, Y).** A sub-routine calls if a mouse button is released.

HASHTABLE

HASHTABLE	
Properties:	
Address	***INTEGER.*** Contains the address of the item after use of setaddress. Also the address can be put here.
Size	***INTEGER.*** Reads or sets the number of records of the HASHTABLE.
Methods:	
Clear	Erase all data of the HASHTABLE.
Delete	Removes the data of the HASHTABLE at the address, which is in the property address.
Density	Returns the density of the HASHTABLE in percents.
GetData	Returns a STRING, which contains the additional data. Used the address of the property address contains.
Put(string)	STRING contains the key word. The task returns the address where the STRING has been put.
PutData(string)	THE STRING contains additional data on the address, which stands in the property address.
SetAddress(string)	Puts the property address on the found address. STRING contains a key word.

IMAGE

IMAGE	
Properties:	
Align	Align can have the next value: ***AlNone***: the IMAGE is shown standard. ***AlTop***: the IMAGE is put down to the upper part of its parent. ***AlBottom: the*** IMAGE is put down to the lower part of its parent. ***AlLeft: the*** IMAGE is at the left-hand side put down of its parent. ***AlRight: the*** IMAGE is at the right-hand side put down of its parent. ***AlClient: the*** IMAGE adapt to the format of the parent.
AutoSize	***BOOLEAN.*** True modify automatically the dimensions of the IMAGE if the image changes him.
BMP	***STRING.*** The name of the bmp file that in the IMAGE must be loaded.
Center	***BOOLEAN.*** True centre the image.
Color	***INTEGER.*** The background Color of the IMAGE.
Cursor	Cursor can have the next value:

	CrDefault **CrNone** **CrArrow** **CrCross** **CrIBeam** **CrSizeNESW** **CrSizeNS** **CrSizeNWSE** **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop** **CrHSplit** **CrVSplit** **CrMultiDrag** **CrSQLWait** **CrNo** **CrAppStart** **CrHelp** **CrHandPoint** **CrSizeAll**
Enabled	***BOOLEAN***. Decides if the IMAGE can be used or not.
Height	***INTEGER***. The altitude of the

	IMAGE.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***Integer.*** The windows handle of the image.
Icon	***STRING.The*** file name of an icon.
IncrementalDisplay	***BOOLEAN.*** An image build by means of rendering if the value
Left	***INTEGER***. Left position of the IMAGE.
Monochrome	***BOOLEAN.*** At true the image is reflected monochrome***.***
Parent	***OBJECT:*** The parent of the IMAGE
PixelFormat	***Pixel format*** can have the next value: ***PfDevice:*** Classification to the properties of the apparatus. ***pf1Bit:*** 1 bit ***pf4Bit:*** 4 bits ***pf8Bit:*** 8 bits ***pf15Bit:*** 15 bits ***pf16Bit:*** 16 bits ***pf24Bit:*** 24 bits ***pf32Bit:*** 32 bits ***pfCustom***

Proportional	***BOOLEAN.*** True ensures that the proportions remain at use of the stretch property.
Showhint	***BOOLEAN***. True shows the hint.
Stretch	***BOOLEAN.*** At true the image is made being fit into the IMAGE.
Tag	***INTEGER***. An identification number for the user.
Top	***INTEGER***. The upper position of the IMAGE.
Transparent	***BOOLEAN.*** True if a transparent Color is used.
TransparentColor	***INTEGER.*** The Color here given up is the transparent Color. Only works as transparent true is.
Visible	***BOOLEAN***. At false the IMAGE is not visible.
Width	***INTEGER***. The breadth of the IMAGE.
Methods:	
BMPResource(resource)	As BMP now the RESOURCE file is used for the image.
Circle(X1,Y1,X2,Y2,color)	Drawing a circle on the IMAGE. X 1, Y1 is the left upper corner. X2, Y2 are the right lower corner.
DrawBMP(X,Y,filename)	Draws an image on the IMAGE of which pixel x, y is the left upper corner.
Fill(X,Y,color, bordercolor)	Paint an area in to the border Color is reached. X, Y is the pixel, which in the area is itself that

	must be coloured.
FillCircle(X1,Y1,X2,Y2,color)	As circle now the whole circle is only coloured.
FillRect(X1,Y1,X2,Y2,color)	Does the same such as rectangle now also the rectangle is paint in.
Line(X1,Y1,X2,Y2,color)	A line of x1, y1 draw to x2, y2.
LineTo(X,Y,color)	Draws a line as from the last point to x, y.
Peek(X,Y)	INTEGER. Returns the Color of the pixel X, Y.
Pset(X,Y,color)	Pixel on the IMAGE put.
Rectangle(X1,Y1,X2,Y2,color)	A rectangle on the IMAGE.X signs 1, Y1 is upper left corner. X2, Y2 is the lower right corner.
Repaint	THE IMAGE and the contents draw all over again.
RoundRect(X1,Y1,X2,Y2,X3,Y3,color)	As rectangle only now with round angles, X3, Y3 is the radius of the angles.
SaveToFile(filename)	The picture writes to a file.
STRING	STRING. Returns a STRING with the bytes of the image.
Text(X,Y,color,text)	Puts a text on the IMAGE starting at pixel x, y.
Events	
OnClick	**SUB([image])**. A sub-routine calls if there is clicked on the IMAGE.

OnMousDown	**SUB([image,] button, shift, X, Y).** A sub-routine calls if a mouse button is pressed.
OnMouseMove	**SUB([image,] shift, X, Y).** A sub-routine calls if the mouse moves over the RICHEDIT.
OnMousUp	**SUB([image,] button, shift, X, Y).** A sub-routine calls if mouse button is released.

INIFILE

IniFile	
Methods:	
DeleteKey("section \ key")	Removes a value and the key from the inifile.
Erase("section")	A group removal from the inifile.
Open(filename)	Opens or produce a new inifile.
ReadStr("section \ key")	Returns a STRING of the value, which is appropriate to the key.
WriteStr("section \ key \ value")	Writes a value to the inifile.

LABEL

LABEL	
Properties:	
AutoSize	***BOOLEAN.*** The length adapts automatically to the text if autosize is

	true.
Caption	***STRING***. The text of the LABEL.
Color	***INTEGER.*** The background Color of the LABEL.
Cursor	Cursor can have the next value: **CrDefault** **CrNone** **CrArrow** **CrCross** **CrIBeam** **CrSizeNESW** **CrSizeNS** **CrSizeNWSE** **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop** **CrHSplit** **CrVSplit** **CrMultiDrag** **CrSQLWait** **CrNo** **CrAppStart** **CrHelp**

	CrHandPoint **CrSizeAll**
Enabled	***BOOLEAN***. Decided if the LABEL can be used or not.
FontBold	***BOOLEAN***. True gives fat text.
FontColor	***INTEGER***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the LABEL.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
Left	***INTEGER***. Left position of the LABEL.
Parent	***OBJECT:*** The parent of the LABEL
Showhint	***BOOLEAN***. True shows the hint.
Taborder	***BOOLEAN***. The number for use with the tab key, if Taborder has been activated.
Tag	***INTEGER***. Defining an identification number for the user.
Top	***INTEGER***. The upper position of the LABEL.
Transparent	***BOOLEAN.*** If true the context of the LABEL is transparent.
Visible	***BOOLEAN***. If false the LABEL is not

	visible.
Width	***INTEGER***. The breadth of the LABEL.
Events	
OnClick	**SUB([label])**. A sub-routine calls if there is clicked on the LABEL.

LISTBOX

LISTBOX	
Properties:	
Align	Align, possible is the next value property: ***AlNone***: the LISTBOX is shown standard. ***AlTop***: the LISTBOX is draw down to the upper part of its parent. ***alBottom***: the LISTBOX is draw down to the lower part of its parent. ***alLeft***: the LISTBOX is at the left-hand side draw down of its parent. ***alRight***: the LISTBOX is at the right-hand side draw down of its parent. ***alClient***: the LISTBOX adapt to the format of the parent.
BorderStyle	***BsNone***. No border visible.

	BsSingle. Border visible.
Color	***INTEGER.*** The background Color of the LISTBOX.
Cursor	Cursor can have the next possible values: **CrDefault** **CrNone** **CrArrow** **CrCross** **CrIBeam** **CrSizeNESW** **CrSizeNS** **CrSizeNWSE** **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop** **CrHSplit** **CrVSplit** **CrMultiDrag** **CrSQLWait** **CrNo** **CrAppStart** **CrHelp**

	CrHandPoint **crSizeAll**
Enabled	***BOOLEAN***. Decided if the LISTBOX can be used or not.
FontBold	***BOOLEAN***. True gives fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True gives slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the LISTBOX.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the LISTBOX.
Item(index)	***STRING.*** The text of the concerning item. Item (0) is the first item.
ItemCount	***INTEGER.*** Total present items in the LISTBOX.
ItemIndex	***INTEGER.*** The current or selected item number. – 1 no item has been selected.
Left	***INTEGER***. Left position of the LISTBOX.

Parent	***OBJECT:*** The parent of the LISTBOX
Showhint	***BOOLEAN***. True shows the hint.
Sorted	***BOOLEAN.*** Inserts new given up items on alphabetical order.
Taborder	***BOOLEAN***. The number for use with the tab key, if Taborder it has been integrated.
Tag	***INTEGER***. Defines an identification number for the user.
Top	***INTEGER***. The upper position of the LISTBOX.
Visible	***BOOLEAN***. False if the LISTBOX is not visible.
Width	***INTEGER***. The breadth of the LISTBOX.
Methods:	
Clear	Clears the contents of the LISTBOX.
Focused	True if the LISTBOX has the focus.
Repaint	THE LISTBOX and the contents are repainting.
SetFocus	THE LISTBOX gives the focus.
Events	
OnActivate	**SUB([listbox])**. A sub-routine calls as the LISTBOX gets the focus.

OnClick	**SUB([listbox]).** A sub-routine calls if there is clicked on the LISTBOX.
OnKeyDown	**SUB([listbox,] key, shift).** A sub-routine calls if a key is pressed.

LISTVIEW

LISTVIEW	
Properties:	
Align	Align can have the next value: ***AlNone***: the LISTVIEW is shown standard. ***AlTop***: the LISTVIEW is put down to the upper part of its parent. ***AlBottom: the*** LISTVIEW is put down to the lower part of its parent. ***AlLeft: the*** LISTVIEW is at the left-hand side put down of its parent. ***AlRight: the*** LISTVIEW is at the right-hand side put down of its parent. ***AlClient: the*** LISTVIEW adapt to the format of the parent.
BMPIndex(index)	***INTEGER.*** Assigns an image from the IMAGELIST to the item with the value index.
BMPList(index)	***STRING.*** Assigns an image to a list. The image is of the bmp format.
BorderStyle	***BsNone***. No border visible.

	BsSingle. Border visible.
CheckBoxes	***BOOLEAN.*** Indicates if there must be checkboxes for the items.
ColCaption(column)	***STRING.*** The title of the concerning column.
ColCount	***INTEGER.*** The number of columns in the LISTVIEW.
Color	***INTEGER.*** The background Color of the LISTVIEW.
ColumnHeader	***BOOLEAN.*** At true a column header is used.
ColWidth(column)	***INTEGER.*** The breadth of the concerning column.
Cursor	Cursor can have the next value: **CrDefault** **CrNone** **CrArrow** **CrCross** **CrIBeam** **CrSizeNESW** **CrSizeNS** **CrSizeNWSE** **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop**

	CrHSplit **CrVSplit** **CrMultiDrag** **CrSQLWait** **CrNo** **CrAppStart** **CrHelp** **CrHandPoint** **CrSizeAll**
Enabled	***BOOLEAN***. Decided if the LISTVIEW can be used or not.
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
GridLines	***BOOLEAN.*** Indicates if there must be lines between the items.
Height	***INTEGER***. The altitude of the TREEVIEW.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the LISTVIEW.

ItemCaption([kolom,]regel)	***STRING.*** The text of the item, the item can be one or two-dimensional array.
ItemChecked(index)	***BOOLEAN.*** Gives the item with the number index is marked or not. Only works if checked is true.
ItemCount	***INTEGER.*** The total number of items of the LISTVIEW.
ItemIndex	***INTEGER.*** The index of the selected item. – 1 no item has been selected.
ItemSelected(index)	***BOOLEAN***. Indicates if the indicated item is selected or not.
Left	***INTEGER***. Left position of the LISTVIEW.
Parent	***OBJECT:*** The parent of the LISTVIEW
Showhint	***BOOLEAN***. True shows the hint.
Sorted	***BOOLEAN.*** At true the LISTVIEW have been sorted.
Taborder	***BOOLEAN***. The number for use with the tab key, if taborder has been activated.
Tag	***INTEGER***. Defining an identification number the user.
Top	***INTEGER***. The upper position of the LISTVIEW.
Viewstyle	Be possible the next value have: ***VsIcon:*** with large icons. ***VsSmallIcon:***with small icons. ***VsList:***As a list ***VsReport:*** As spreadsheet.

Visible	***BOOLEAN***. At false the LISTVIEW is not visible.
Width	***INTEGER***. The breadth of the LISTVIEW.
Methods:	
BMPResource(index, resource)	An image from RESOURCE puts the image into the IMAGELIST.
Clear	All contents of the LISTVIEW are erased.
Focused	True if the LISTVIEW has the focus.
Repaint	The LISTVIEW and the contents are paint all over again.
SetFocus	The LISTVIEW gets the focus.
Events	
OnActivate	**SUB([listview])**. A sub-routine calls as the LISTVIEW the gets focus.
OnClick	**SUB([listview])**. A sub-routine calls if there is clicked on the LISTVIEW.
OnColumnClick	**SUB([listview])**. A sub-routine calls if there is clicked on a column.
OnDblClick	**SUB([listview])**. A sub-routine calls if there is double clicked on the LISTVIEW.
OnKeyDown	**SUB([listview,] key, shift).** A sub-routine calls if a key is pressed.

MAINMENU

MAINMENU	
Properties:	
HWND	***INTEGER.*** Windows handle of the MAINMENU.
Parent	***OBJECT:*** The parent of the MAINMENU.

MEMORYSTREAM

MEMORYSTREAM	
Properties:	
HWND	***INTEGER.*** The windows handle of the stream.
LineCount	***INTEGER.*** The number of lines in the stream, which has separated by a linefeed**.**
Position	***INTEGER.*** The current position in the stream.
Size	***INTEGER.*** The size of the stream in bytes.
Methods:	
Clear	Clears the MEMORYSTREAM. And puts all values on 0.
LoadResource(resource)	Puts RESOURCE file into MEMORYSTREAM.

Read(bytes)	STRING return. These STRING contain the contents of MEMORYSTREAM as from the current position with the length bytes.
ReadLine	Returns a STRING, which contains the line as from the current position. The lines must be separated by a linefeed.
Write(string)	Writes the STRING at the current position to the stream.

MENUITEM

MENUITEM	
Properties:	
AutoCheck	***BOOLEAN.*** If this value is true then the MENUITEM functions as a CHECKBOX.
BMP	***STRING.*** The file name of the icon beside the text of the MENUITEM. The file is of the TYPE bmp.
Caption	***STRING.*** The text, which appears on the MENUITEM.
Checked	***BOOLEAN.*** At true there a mark appear beside the caption.
Enabled	***BOOLEAN***. Decided if the MENUITEM can be used or not.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.

HWND	***INTEGER.*** Windows handle of the MENUITEM.
ItemCount.	***INTEGER.*** The total number of menuitems.
ItemIndex	***INTEGER.*** The index of MENUITEM within the parent.
Parent	***OBJECT:*** The parent of the MENUITEM generally a MAINMENU or a POPUPMENU.
RadioItem	***BOOLEAN.*** The MENUITEM works as RADIOBUTTON. Only one of the menuitems from the group can be selected.
ShortCut	***STRING.*** These STRING contain the shortcuts of the concerning MENUITEM. CTRL+A are the control A key.
Tag	***INTEGER***. Defining an identification number for the user.
Visible	***BOOLEAN***. At false the MENUITEM is not visible.
Methods:	
BMPResource(Index, resource)	As BMP now the RESOURCE file is only used for the icon. The index is the index of the concerning MENUITEM.
Events	
OnClick	**SUB([menuitem])**. A sub-routine calls if there is clicked on the MENUITEM.

PANEL

PANEL	
Properties:	
Align	Align can have the next possible value: ***AlNone***: the PANEL is shown standard. ***AlTop***: the PANEL is put down to the upper part of its parent. ***AlBottom:*** the PANEL is put down to the lower part of its parent. ***AlLeft:*** the PANEL is at the left-hand side put down of its parent. ***AlRight:*** the PANEL is at the right-hand side put down of its parent. ***AlClient:*** the PANEL adapt to the format of the parent.
Alignment	Alignment can have the next possible the next value: ***TaLeftJustify*** Sets the text left. ***TaRightJustify:*** The text is put down at the Right. ***TaCenter:*** The text centres.
BevelInner	The interior puts part of the framework. Bevelinner can have the next possible value: ***BvNone***

	bvLowered ***bvRaised*** ***bvSpace***
Bevelouter	For the values see bevelinner, applying now to the outer part of the framework.
Bevelwidth	***INTEGER.*** The breadth of the framework.
Caption	***STRING***. The text of the PANEL.
ClientHeight	***INTEGER***. Altitude of the user part of the PANEL.
Clientwidth	***INTEGER***. Breadth of the user part of the PANEL.
Color	***INTEGER.*** The background Color of the PANEL.
Cursor	Cursor can have the next possible value: **crDefault** **crNone** **crArrow** **crCross** **crIBeam** **crSizeNESW** **crSizeNS** **crSizeNWSE** **crSizeWE** **crUpArrow**

	crHourGlass **crDrag** **crNoDrop** **crHSplit** **crVSplit** **crMultiDrag** **crSQLWait** **crNo** **crAppStart** **crHelp** **crHandPoint** **crSizeAll**
Enabled	***BOOLEAN***. Decided if the PANEL can be used or not.
FontBold	***BOOLEAN***. True gives fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True gives slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the PANEL.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.

HWND	***INTEGER.*** Windows handle of the PANEL.
Left	***INTEGER***. Left position of the PANEL.
Parent	***OBJECT:*** The parent of the PANEL
Showhint	***BOOLEAN***. True shows the hint.
Taborder	***BOOLEAN***. The number for use with the tab key, if taborder Is activated.
Top	***INTEGER***. The upper position of the PANEL.
Visible	***BOOLEAN***. At false the PANEL is not visible.
Width	***INTEGER***. The breadth of the PANEL.
Methods:	
Repaint	The PANEL and the contents are repainted.
Events	
OnClick	**SUB([panel])**. A sub-routine calls if there is clicked on the PANEL.
OnMouseDown	**SUB([panel,] button, shift, X, Y).** A sub-routine calls if a mouse button is pressed.
OnMouseMove	**SUB([panel,] shift, X, Y).** A sub-routine calls if the mouse moves

	over the PANEL.
OnMouseUp	**SUB([panel,] shift, button, X, Y).** A sub-routine calls if a mouse button is released.

POPUPMENU

POPUPMENU	
Properties:	
HWND	***INTEGER.*** Windows handle of the POPUPMENU.
Parent	***Object:*** The parent of the POPUPMENU.
Events:	
OnPopup	**SUB([popupmenu]).** Calls a sub-routine just before the popup appears.

PRINTDIALOG

PRINTDIALOG	
Properties:	
Collate	***BOOLEAN.*** Read and set the collate CHECKBOX.
FromPage	***INTEGER.*** By the user given page

	which as first must be printed.
HWND	***INTEGER.*** Windows handle of the dialog.
MaxPage	***INTEGER.*** Sets and reads the maximum page.
MinPage	***INTEGER.*** Sets and reads the minimum page.
PageNums	***BOOLEAN.*** Shows a RADIOBUTTON in the menu to select pages.
PrintSelection	***BOOLEAN.*** Reads and sets the RADIOBUTTON to choose a selection.
PrintToFile	***BOOLEAN.*** Read and sets the CHECKBOX so the user can print to a file.
ToPage	***INTEGER.*** The user can give up which page the printing stops.
Methods:	
Show	Shows the printdialog.

PRINTER

PRINTER	
Properties:	
Aborted	***BOOLEAN.*** True if the user has aborts the printing.
Copies	***INTEGER.*** The number of copies, which must be printed.

FontBold	***BOOLEAN***. True gives fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontCount	***INTEGER.*** The total number of fonts.
FontItalic	***BOOLEAN***. True gives slanting pressed text.
FontName	***STRING***. FONT for the text.
Fonts(index)	***STRING.*** Returns the name of the FONT with the given index.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Handle	***INTEGER.*** The windows handle of the PRINTER***.***
Orientation	Be possible to have the next values: ***poPortrait*** ***poLandscape***
PageHeight	***INTEGER.*** The page altitude***.***
PageWidth	***INTEGER.*** The page breadth***.***
PrinterCount	***INTEGER***. The total number of printers.
PrinterIndex	***INTEGER.*** The index of the selected PRINTER.
Printers(index)	***STRING.*** The name of the PRINTER with the given index.
Printing	***BOOLEAN.*** True if begindoc is called, and enddoc have not been called***.***

Title	***STRING.De*** title, which appear in the print manager of windows.
Methods:	
Abort	Abort printing.
BeginDoc	Starts the printing.
Circle(X1,Y1,X2,Y2, color)	Circle print. X 1, Y1 is the left upper corner. X2, Y2 lower right corner.
DrawBMP(X,Y,filename)	Puts an image on paper of which pixel x, y is the left upper corner.
DrawResource(resource)	As BMP now a RESOURCE file is used for the image.
EndDoc	Terminate the print session.
Fill(X,Y, Color, border Color)	Fills in an area until the border Color is reached. X, Y is the pixel in the area itself that must be coloured.
FillCircle(X1,Y1,X2,Y2, colore)	As circle now the whole circle is coloured.
FillRect(X1,Y1,X2,Y2, colore)	Does the same such as rectangle now also the rectangle is coloured in.
Line(X1,Y1,X2,Y2, Color)	A line from x1, y1 drawing to x2, y2.
LineTo(X,Y, Color)	Draws a line from the last point to x, y.
NewPage	Prints everything after this task on a new page.

PrintString(x,y,string)	STRING contain bitmap image and is printed on position x, y.
Pset(X,Y, Color)	Puts a pixel on the paper.
Rectangle(X1,Y1,X2,Y2, colore)	Draws a rectangle on the paper.X 1,Y1 is left upper corner. X2, Y2 is the lower right corner.
RoundRect(X1,Y1,X2,Y2,X3,Y3, colore)	As rectangle only now with round angles, X3, Y3 is the radius of the angles.
Text(X,Y, Color, text)	A text prints starting at pixel x, y.

PROGRESSBAR

PROGRESSBAR	
Properties:	
Align	Align can have the next possible value: ***AlNone***: The standard PROGRESSBAR is shown. ***AlTop***: PROGRESSBAR to the upper part of its parent. ***AlBottom:*** PROGRESSBAR to the lower part of its parent. ***AlLeft:*** The PROGRESSBAR is at the left-hand side put down. ***AlRight:*** The PROGRESSBAR is at the right-hand side put down. ***AlClient:*** PROGRESSBAR adapts to the format of the parent.

Borderwidth	***INTEGER.*** The framework breadth of PROGRESSBAR.
Height	***INTEGER***. The altitude of the PROGRESSBAR.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of PROGRESSBAR.
Left	***INTEGER***. Left position of the PROGRESSBAR.
Max	***INTEGER***. The maximum value of the PROGRESSBAR
Min	***INTEGER.*** The minimum value of the PROGRESSBAR.
Orientation	Orientation can have the next possible value: ***PbHorizontal***: gives a horizontal PROGRESSBAR ***PbVertical***: gives a vertical PROGRESSBAR
Parent	***OBJECT:*** The parent of the PROGRESSBAR
Position	***INTEGER***. Sets or gives the current position of the PROGRESSBAR
Showhint	***BOOLEAN***. True shows the hint.
Smooth	***BOOLEAN.*** False reflect the progress in segments.
Top	***INTEGER***. The upper position of the PROGRESSBAR.
Width	***INTEGER***. The breadth of the PROGRESSBAR.

RADIOBUTTON

RADIOBUTTON	
Properties:	
Caption	***STRING.*** This is the text beside the RADIOBUTTON.
Checked	***BOOLEAN.*** True, if the RADIOBUTTON is selected.
Enabled	***BOOLEAN***. Decided if RADIOBUTTON can be used or not.
FontBold	***BOOLEAN***. True gives fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True gives slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the RADIOBUTTON.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the RADIOBUTTON.
Left	***INTEGER***. Left position of the RADIOBUTTON.
Parent	***OBJECT:*** The parent of the RADIOBUTTON

Showhint	***BOOLEAN***. True shows the hint.
Taborder	***BOOLEAN***. The number for use with the tab key, if taborder has been activated.
Tag	***INTEGER***. By defining an identification number for the user.
Top	***INTEGER***. The upper position of the RADIOBUTTON.
Visible	***BOOLEAN***. At false the RADIOBUTTON is not visible.
Width	***INTEGER***. The breadth of the RADIOBUTTON.
Events	
OnClick	**SUB([radiobutton])**. A sub-routine calls if there is clicked on the RADIOBUTTON.

REGISTRY

REGISTRY	
Properties:	
CurrentKey	***INTEGER***. Index of the current key.
Path	***STRING.*** The registry path of the current key.
RootKey	***INTEGER.*** Sets the root key***.***
SubKey(index)	***STRING.*** The path on the current sub key with the value index.
SubKeys	***INTEGER. Returns the*** number of SUB

	keys of the current key.
Type(index)	The TYPE of the value with the concerning index. The following types can be used: ***rdInteger*** ***rdString*** ***rdExpandString*** ***rdbinary*** ***rdUnknown***
Value(index)	***STRING.*** The contents of the value with the concerning index.
ValueName(index)	***STRING.*** The name of the value with the concerning index***.***
Values	***INTEGER. The*** total number of values of the current key.
Methods:	
Close	Closes the registry session.
CreateKey(key name)	A new key adds to the current key. If this has succeeded the function returns true.
DeleteKey(key name)	Deletes a key. The function returns true if the task has succeeded
DeleteValue(value name)	Value name is the name of the value. The value himself is erased.
KeyExists(key name)	Looks at if a key with a certain name exists. True return if the key exists.
ValueExists(value name)	Looks at if a value with a certain name exists. The function returns true if these exist.

Write("value name|value|type")	A value writes to a key. Name|value|type is of the STRING TYPE. Value name is the name of the key. TYPE can be the next value: 1 INTEGER 2 STRING 3 binary The function returns true if the task has succeeded.

RICHEDIT

RICHEDIT	
Properties:	
Align	Align can have the next value: ***AlNone***: the RICHEDIT is shown standard. ***AlTop***: the RICHEDIT is put down to the upper part of its parent. ***AlBottom: the*** RICHEDIT is put down to the lower part of its parent. ***AlLeft: the*** RICHEDIT is at the left-hand side put down of its parent. ***alRight***: the RICHEDIT is at the right-hand side put down of its parent. ***alClient***: the RICHEDIT adapt to the format of the parent.

Alignment	Be possible to have the next value: ***TaLeftJustify***: text left aligns ***TaRightJustify***: text aligns Right ***TaCenter***: text centers
BorderStyle	***BsNone***. No border visible. ***BsSingle***. Border visible.
Col	***INTEGER.*** The position of the present column, there where the cursor stands.
Color	***INTEGER.*** The background Color of the RICHEDIT.
Cursor	Cursor is possible to have the next value: **crDefault** **crNone** **crArrow** **crCross** **crIBeam** **crSizeNESW** **crSizeNS** **crSizeNWSE** **crSizeWE** **crUpArrow** **crHourGlass** **crDrag** **crNoDrop**

	crHSplit **crVSplit** **crMultiDrag** **crSQLWait** **crNo** **crAppStart** **crHelp** **crHandPoint** **crSizeAll**
Enabled	***BOOLEAN***. Decided if the RICHEDIT can be used or not.
FirstLine	***INTEGER.*** The number of the first visible line.
FontBold	***BOOLEAN***. True gives fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the RICHEDIT.
HideScrollbars	***BOOLEAN.*** True hides the scrollbars.
HideSelection	***BOOLEAN.*** At true the selection disappears when the RICHEDIT loses the focus.
Hint	***STRING***. The text, which must be

	reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the RICHEDIT.
Left	***INTEGER***. Left position of the RICHEDIT.
Line(index)	***STRING.*** Index is the line number of which the text must be put in the STRING. Index 0 is the first line text in the RICHEDIT.
LineCount	***INTEGER.*** The number of lines of the RICHEDIT.
Parent	***OBJECT:*** The parent of the RICHEDIT
PlainText	***BOOLEAN.*** As these true is the text without attributes is reflected.
ReadOnly	***BOOLEAN.*** As read-only is true then the text can only be reflected and not to be changed.
Row	***INTEGER.*** The number of the present line there where the cursor stands.
RTFText	***STRING.*** Rtftext contains the text in RTF format.
ScrollBars	Scrollbars can have the next possible value: ***SsNone:*** no scrollbars ***ssBoth***:both scrollbars visible ***ssHorizontal:*** the horizontal scrollbar is only visible ***ssVertical:*** the vertical scrollbar is

	only visible
SelFontBold	***BOOLEAN***. True give fat selected text.
SelFontColor	***BOOLEAN***. The Color of the selected text.
SelFontItalic	***BOOLEAN***. True give slanting pressed selected text.
SelFontName	***STRING***. FONT for the selected text.
SelFontSize	***INTEGER***. Size of the selected signs.
SelFontUnderline	***BOOLEAN***. Underlines the selected text.
SelLength	***INTEGER.*** The number of selected signs of the text.
SelStart	***INTEGER.*** The index of the sign is where the selection starts.
SelText	***STRING.*** The selected part of the text.
Showhint	***BOOLEAN***. True show the hint.
Size	***INTEGER***. Standard the RICHEDIT have been limited on 64 kB establish the size of the RICHEDIT above can one the size in bytes up give***.***
Taborder	***BOOLEAN***. The number for use with the tab key, as Taborder has been integrated.
Tag	***INTEGER***. By defining an identification number for the user.
Text	***STRING.*** The total flat text of the RICHEDIT
Top	***INTEGER***. The upper position of the

	RICHEDIT.
Visible	***BOOLEAN***. At false the RICHEDIT is not visible.
WantTabs	***BOOLEAN.*** False if no tabs have been permitted.
Width	***INTEGER***. The breadth of the RICHEDIT.
WordWrap	***BOOLEAN.With*** true the line is demolished if these are longer than the breadth of the RICHEDIT. The rest of the line is put on the next line.
Methods:	
Clear	All text of the RICHEDIT is cleared.
CopyToClipboard	A selected piece text copies to the clipboard.
CutToClipboard	The selected text from the RICHEDIT cuts and puts him in the clipboard.
Focused	True if the RICHEDIT has the focus.
Lock	The richedit freeze, there no text can be modified and the changes are not visible until the command unlock has been given.
PasteFromClipboard	The contents of the clipboard are put in the RICHEDIT on the position of the cursor (selstart).
Print(title)	PRINT the whole contents of the RICHEDIT. Title is of the stringtype and contains the title of the document.

Repaint	Repaints the RICHEDIT all over again.
Scroll	Scrolls the RICHEDIT until the cursor falls in the visible part.
SetFocus	Gives the RICHEDIT the focus.
Undo	The last modification rectifies.
Unlock	THE RICHEDIT reflect rather for all changes. Changes, which have been carried out during the lock, are still done.
Events	
OnChange	**SUB([richedit]).** A sub-routine calls if the contents are modified.
OnKeyDown	**SUB([richedit,] key, shift).** A sub-routine calls if a key is pressed.
OnKeyPress	**SUB([richedit]).** A sub-routine calls as a key is used.
OnKeyUp	**SUB([richedit,] key, shift).** A sub-routine calls if the key is released.
OnMousDown	**SUB([richedit,] button, shift, X, Y).** A sub-routine calls if a mouse button is pressed.
OnMouseMove	**SUB([richedit,] shift, X, Y).** A sub-routine calls if the mouse moves over the RICHEDIT.
OnMousUp	**SUB([richedit,] button, shift, X, Y).** A sub-routine calls if a mouse button is released.

SERVERSOCKET

SERVERSOCKET	
Properties:	
Active	***BOOLEAN.*** True make the connection active.
ActiveConnections	***INTEGER.*** Gives the number of connections with the clientsockets.
ClientAddress(index)	***STRING.*** Give the ip number of concerning CLIENTSOCKET.
ClientHost(index)	***STRING. Gives the domain name*** of the concerning CLIENTSOCKET.
ClientPort(index)	***INTEGER.*** Gives the port number of the concerning CLIENTSOCKET.
Connected	***BOOLEAN.*** Gives to or if there is a connection or not.
Connection	***INTEGER. Gives the*** index of the active CLIENTSOCKET.
ServerType	The next value can contain. ***StNonBlocking the*** program goes further. ***StBlocking the*** program waits until SERVERSOCKET is finished
Service	***STRING.*** Here the service can be given up for the socket. See Microsoft documentation for windows sockets.
Methods:	
Close	A connection concludes.

OPEN	A connection with the socket is made.
Receive	STRING returns the received text.
Send(Index, Text)	Returns 0 if the text has been successfully sent. Another INTEGER number returns if the sending has been postponed. The text is sent to CLIENTSOCKET with the concerning index.
Listen	The socket is put to listen.
Events	
OnConnect	**SUB()**: A sub-routine calls if the connection to score is brought.
OnDisconnect	**SUB().** A sub-routine calls if the connection stands to conclude.
OnError	**SUB(error code).** A sub-routine calls if there was a wrongly action, the error code is an INTEGER number.
OnRead	**SUB().** A sub-routine calls if the socket text receives.

SPLITTER

SPLITTER	
Properties:	
Align	Align can have the possible value have: ***AlNone***: the SPLITTER is shown standard. ***AlTop***: the SPLITTER is put down to the

	upper part of its parent. ***AlBottom:the*** SPLITTER is put down to the lower part of its parent. ***AlLeft:the*** SPLITTER is at the left-hand side put down of its parent. ***AlRight: the*** SPLITTER is at the right-hand side put down of its parent. ***AlClient: the*** SPLITTER adapt to the format of the parent.
Beveled	***BOOLEAN.*** True give a 3D framework.
Color	***INTEGER***. The Color of the SPLITTER.
Enabled	***BOOLEAN***. Decided if the SPLITTER can be used or not.
Left	***INTEGER***. Left position of the SPLITTER.
MinSize	***INTEGER.*** The minimum dimension of the OBJECT beside the SPLITTER.
Parent	***OBJECT:*** The parent of the SPLITTER
Top	***INTEGER***. The upper position of the SPLITTER.
Visible	***BOOLEAN***. At false the SPLITTER is not visible.
Width	***INTEGER***. The breadth of the SPLITTER.
Events	
OnMoved	**SUB([splitter])**. A sub-routine calls if the SPLITTER has moved.

STATUSBAR

STATUSBAR	
Properties:	
Align	Align can have the next value: ***AlNone***: the STATUSBAR is shown standard. ***AlTop***: the STATUSBAR is put down to the upper part of its parent. ***AlBottom***: the STATUSBAR is put down to the lower part of its parent. ***AlLeft***: the STATUSBAR is at the left-hand side put down of its parent. ***AlRight***: the STATUSBAR is at the right-hand side put down of its parent. ***AlClient: the*** STATUSBAR adapt to the format of the parent.
Color	***INTEGER.*** The background Color of the STATUSBAR.
Cursor	Cursor is possible to have the next value: **CrDefault** **CrNone** **CrArrow** **CrCross** **CrIBeam** **CrSizeNESW** **CrSizeNS**

	CrSizeNWSE **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop** **CrHSplit** **CrVSplit** **CrMultiDrag** **CrSQLWait** **CrNo** **CrAppStart** **CrHelp** **CrHandPoint** **CrSizeAll**
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT establish for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the STATUSBAR.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting

	showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the STATUSBAR.
PanelAlignment(index)	At plural PANEL the text can be aligned as follows: ***TaLeftJustify***: left aligning ***TaRightJustify***: Right aligns ***TaCenter:*** to centre
PanelText(index)	***STRING.*** The text of plural PANEL. With index indicates which PANEL. Simpletext must be false. Zero is the first PANEL.
PanelWidth(index)	***INTEGER.*** The breadth of the PANEL, only at plural PANEL.
Parent	***OBJECT:*** The parent of the STATUSBAR
Showhint	***BOOLEAN***. True shows the hint.
SimplePanel	***BOOLEAN.*** If this value is true then a simple PANEL is shown.
SimpleText	***STRING.*** Here the text of simple PANEL can be given up, for use simplepanel must be put on true.
SizeGrip	***BOOLEAN.*** If the value is false the grip is not given and cannot change the format.
Visible	***BOOLEAN***. At false the STATUSBAR is not visible.
Width	***INTEGER***. The breadth of the STATUSBAR.

STRINGGRID

STRINGGRID	
Properties:	
Align	Align can have the next possible value: ***AlNone***: the STRINGGRID is shown standard. ***AlTop***: the STRINGGRID is put down to the upper part of its parent. ***AlBottom***: the STRINGGRID is put down to the lower part of its parent. ***AlLeft***: the STRINGGRID is at the left-hand side put down of its parent. ***AlRight***: the STRINGGRID is at the right-hand side put down of its parent. ***AlClient***: the STRINGGRID adapt to the format of the parent.
BorderStyle	***BsNone***. No border visible. ***BsSingle***. Border visible.
Cell(column, row)	***STRING.*** Reads or sets the text of the STRINGGRID on column, row.
Col	***INTEGER.*** Give the current column number.
ColCount	***INTEGER.*** Reads and sets the number of columns of the STRINGGRID.

ColMoving	***BOOLEAN***. Makes moving a column by the user possible**.**
Color	***INTEGER.*** The background Color of the STRINGGRID.
ColSizing	***BOOLEAN.*** Establishing the column makes breadth by the user possible.
ColWidth(column)	***INTEGER.*** Reads or sets the breadth of the column.
Cursor	Cursor can have the next value: **CrDefault** **CrNone** **CrArrow** **CrCross** **CrIBeam** **CrSizeNESW** **CrSizeNS** **CrSizeNWSE** **CrSizeWE** **CrUpArrow** **CrHourGlass** **CrDrag** **CrNoDrop** **CrHSplit** **CrVSplit** **CrMultiDrag**

	CrSQLWait **CrNo** **CrAppStart** **CrHelp** **CrHandPoint** **CrSizeAll**
DefaultColWidth	***INTEGER.*** Reads or sets the column breadth of all columns in the STRINGGRID***.***
DefaultRowHeight	***INTEGER.*** Reads or sets the row altitude of all ranges in the STRINGGRID***.***
DrawFocusSelected	***BOOLEAN.*** If this value is true then a cell, which has the focus, is coloured.
Editing	***BOOLEAN.*** At true the user can fill in data.
Enabled	***BOOLEAN***. Decided if the STRINGGRID can be used or not.
FixedColor	***INTEGER.*** The Color of the fixed columns and lines.
FixedCols	***INTEGER.*** The number of fixed columns.
FixedRows	***INTEGER.*** The number of fixed lines.
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.

FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the STRINGGRID.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the STRINGGRID.
Left	***INTEGER***. Left position of the STRINGGRID.
LineWidth	***INTEGER.*** The breadth of the lines between the cells.
Parent	***OBJECT:*** The parent of the STRINGGRID
Record(line)	***STRING***. Writes or reads a line. The separator separates the lines***.***
Row	***INTEGER.*** The current line number.
RowCount	***INTEGER.*** The total number of lines of the STRINGGRID***.***
RowHeight(line)	***INTEGER.*** The altitude of a specific line.
RowMoving	***BOOLEAN***. Makes moving a line by the user possible***.***
RowSelect	***BOOLEAN.*** At true the user can select a line.
RowSizing	***BOOLEAN.*** Making the line

	breadth by the user possible.
ScrollBars	Scrollbars can have the next value: ***SsNone:*** no scrollbars ***SsBoth***: both scrollbars visible ***SsHorizontal:*** the horizontal scrollbar is only visible ***SsVertical:*** the vertical scrollbar is only visible
Separator	***STRING.*** The sign, which reflects the separation between the cells of the text of the STRINGGRID.
Showhint	***BOOLEAN***. True shows the hint.
Taborder	***BOOLEAN***. The number for use with the tab key, as Taborder has been activated.
Tag	***INTEGER***. By defining identification number for the user.
Text	***STRING.*** Reads or writes the whole contents of the STRINGGRID. Lines are separated by linefeed and the separator separates the cells.
Top	***INTEGER***. The upper position of the STRINGGRID.
TopRow	***INTEGER.*** The number of the first visible line.
Visible	***BOOLEAN***. At false the STRINGGRID is not visible.
Width	***INTEGER***. The breadth of the STRINGGRID.

Methods:	
AutoWidth	The column width adapt to the current contents.
DelCol(column)	Removes a column. Zero is the first column.
DelRow(line)	Removed a line. Zero is the first line.
Focused	True if STRINGGRID has the focus.
Repaint	THE STRINGGRID and the contents are paint all over again.
SetFocus	THE STRINGGRID gets the focus.
SortCol(column, direction, type)	STRINGGRID sort. The column is the number of the column on which must be sorted: 0=ascending and 1 is descending. TYPE: 0=string apply and 1 is numerical.
TextHeight(text)	Returns the altitude of the text.
TextWidth(text)	Returns the breadth of the text.
Events	
OnClick	**SUB([stringgrid])**. A sub-routine calls if it there is clicked on the STRINGGRID.
OnKeyDown	**SUB([stringgrid,] key, shift).** A sub-routine calls if a key is pressed.
OnKeyPress	**SUB([stringgrid]).** A sub-routine calls as a key is used.

OnKeyUp	**SUB([stringgrid,] key, shift).** A sub-routine calls if the key is released.
OnSelectCell	**SUB([stringgrid,] col, row).** A sub-routine calls as a user selects a cell.
OnSetEditText	**SUB([stringgrid,] col, row).** A sub-routine calls if the user changes the contents of a cell.

TABCONTROL

TABCONTROL	
Properties:	
Align	Align can have the next possible value: ***AlNone***: the TABCONTROL is shown standard. ***AlTop***: the TABCONTROL is put down to the upper part of its parent. ***AlBottom***: the TABCONTROL is put down to the lower part of its parent. ***AlLeft***: the TABCONTROL is at the left-hand side put down of its parent. ***AlRight:*** the TABCONTROL is at the right-hand side put down of its parent.

	AlClient: the TABCONTROL adapt to the format of the parent.
BMP(index)	***STRING***. The file name of the icon for the tab with the concerning index. The file is of the TYPE bmp.
Cursor	Cursor can have the next value: **crDefault** **crNone** **crArrow** **crCross** **crIBeam** **crSizeNESW** **crSizeNS** **crSizeNWSE** **crSizeWE** **crUpArrow** **crHourGlass** **crDrag** **crNoDrop** **crHSplit** **crVSplit** **crMultiDrag** **crSQLWait** **crNo**

	crAppStart **crHelp** **crHandPoint** **crSizeAll**
Enabled	***BOOLEAN***. Decided if the TABCONTROL can be used or not.
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the TABCONTROL.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HotTrack	***BOOLEAN.*** If the value is true it is indicated which tab is selected as a user clicks on a tab***.***
HWND	***INTEGER.*** Windows handle of the TABCONTROL.
Left	***INTEGER***. Left position of the TABCONTROL.

Multiline	***BOOLEAN.*** True show several range with tabs.
Parent	***OBJECT:*** The parent of the TABCONTROL
Position	Position can have the next value: ***TpTop:*** the tabs put at the top***.*** ***TpBottom:*** the tabs put below***.*** ***TpLeft:*** put the tabs left ***TpRight:*** the tabs put Right***.***
RaggedRight	***BOOLEAN.*** True ensure that the tabs are divided concerning the whole length.
Showhint	***BOOLEAN***. True show the hint.
TabIndex	***INTEGER.*** Returns the number of the selected tab. – 1 no tab has been selected.
Taborder	***BOOLEAN***. The number for use with the tab key, if Taborder has been activated.
Tabs(index)	***STRING.*** Contain the title of the tab with the concerning index.
TabsCount	***INTEGER.*** The number of tabs in the TABCONTROL.
Tag	***INTEGER***. By defining identification number for the user.
Top	***INTEGER***. The upper position of the TABCONTROL.
Visible	***BOOLEAN***. At false the TABCONTROL is not visible.

Width	***INTEGER***. The breadth of the TABCONTROL.
Methods:	
BMPResource(index, resource)	As BMP now the RESOURCE file is used for the icon. The index is the index of the concerning tab.
Events	
OnChange	**SUB([tabcontrol])**. A sub-routine calls if another tab is chosen.

TIMEPICKER

TIMEPICKER	
Properties:	
Checked	***BOOLEAN.*** Gives to if the CHECKBOX is marked or not.
Color	***INTEGER.*** The background Color of the TIMEPICKER.
DateFormat	Be possible to have the next value: ***DfLong*** give a long date with the name of the day. ***DfShort*** give a shortened date e.g. 1/1/06. This function only works as kind stands on dtkDate.
DateTime	***STRING.*** The date or the time contains which is chosen depending on the kind that has been selected.

Enabled	***BOOLEAN***. Decided if the TIMEPICKER can be used or not.
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the TIMEPICKER.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the TIMEPICKER.
Kind	The TYPE of the TIMEPICKER indicate and can have the next value: ***DtkTime***: time ***DtkDate***: date
Left	***INTEGER***. Left position of the TIMEPICKER.
Parent	***OBJECT:*** The parent of the TIMEPICKER
ShowCheckBox	***BOOLEAN.*** At true the CHECKBOX is shown.
Showhint	***BOOLEAN***. True shows the hint.
Taborder	***BOOLEAN***. The number for use with the tab key, as Taborder has been

	activated.
Tag	***INTEGER***. By defining identification number for the user.
Text	***STRING***. The text of the TIMEPICKER.
Top	***INTEGER***. The upper position of the TIMEPICKER.
Visible	***BOOLEAN***. At false the TIMEPICKER is not visible.
Width	***INTEGER***. The breadth of the TIMEPICKER.
Methods:	
SetFocus	THE TIMEPICKER gets the focus.
Focused	True if the TIMEPICKER has the focus.
Events	
OnChange	**SUB([timepicker]).** A sub-routine calls if the contents are modified.
OnKeyDown	**SUB([timepicker,] key, shift).** A sub-routine calls if a key is pressed.

TIMER

TIMER	
Properties:	
Enabled	***BOOLEAN.*** At false the TIMER does not work.
Interval	***INTEGER.*** The interval with which

	the event ontimer is activated. The interval is given up in milliseconds.
Tag	***INTEGER.*** By the user giving identification number.
Events	
OnTimer	**SUB([timer])**. Every interval calls a sub-routine.

TRACKBAR

TRACKBAR	
Properties:	
Cursor	Cursor can have the next possible value: **crDefault** **crNone** **crArrow** **crCross** **crIBeam** **crSizeNESW** **crSizeNS** **crSizeNWSE** **crSizeWE** **crUpArrow**

	crHourGlass **crDrag** **crNoDrop** **crHSplit** **crVSplit** **crMultiDrag** **crSQLWait** **crNo** **crAppStart** **crHelp** **crHandPoint** **crSizeAll**
Enabled	***BOOLEAN***. Decided if the TRACKBAR can be used or not.
Frequentie	***INTEGER.*** As the tickstyle unequally to the value of frequency is no influence tsAuto has. If the value is for example 5 then 5 marks is put there.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the TRACKBAR.
Left	***INTEGER***. Left position of the TRACKBAR.
Max	***INTEGER***. The maximum value of the TRACKBAR

Min	***INTEGER.*** The minimum value of the TRACKBAR
Orientation	Orientation can have the next possible value have: ***TrHorizontal***: gives horizontal TRACKBAR ***TrVertical***: gives vertical TRACKBAR
Parent	***OBJECT:*** The parent of the TRACKBAR
Position	***INTEGER***. Sets or reads the current position of the TRACKBAR
Showhint	***BOOLEAN***. True shows the hint.
Taborder	***BOOLEAN***. The number for use with the tab key, if Taborder has been activated.
TickMarks	Tickmarks can have the next possible value: ***TmBottomRight*** put the marks to the lower part or at the right-hand side***.*** ***TmTopLeft*** put the marks to the upper part or at the left-hand side ***TmBoth*** put the marks on both sides.
Top	***INTEGER***. The upper position of the TRACKBAR.
Visible	***BOOLEAN***. At false the TRACKBAR is not visible.
Width	***INTEGER***. The breadth of the TRACKBAR.
Methods:	
SetFocus	The TRACKBAR gets the focus.

Focused	True if the TRACKBAR has the focus.
Events	
Onchange	***SUB([trackbar]).*** A sub-routine calls if the value has changed of the TRACKBAR.

TREEVIEW

TREEVIEW	
Properties:	
Align	Align can have the next possible value: ***AlNone***: the TREEVIEW is shown standard. ***AlTop***: the TREEVIEW is put down to the upper part of its parent. ***alBottom***: the TREEVIEW is put down to the lower part of its parent. ***alLeft***: the TREEVIEW is at the left-hand side put down of its parent. ***alRight***: the TREEVIEW is at the right-hand side put down of its parent. ***alClient***: the TREEVIEW adapt to the format of the parent.
BMPIndex(index)	***INTEGER.*** Assigns an image from the imagelist to the item with the value index.

BMPList(index)	***STRING.*** Assigns an image to a list. The image is of the bmp format.
BorderStyle	***BsNone***. No border visible. ***BsSingle***. Border visible.
Caption(index[,at index)	***STRING.*** An item with index adds on a certain index or returns the value of an item. THE STRING contains the text of the item.
Color	***INTEGER.*** The background Color of the TREEVIEW.
Cursor	Cursor can have the next value: **crDefault** **crNone** **crArrow** **crCross** **crIBeam** **crSizeNESW** **crSizeNS** **crSizeNWSE** **crSizeWE** **crUpArrow** **crHourGlass** **crDrag** **crNoDrop** **crHSplit** **crVSplit**

	crMultiDrag **crSQLWait** **crNo** **crAppStart** **crHelp** **crHandPoint** **crSizeAll**
Enabled	***BOOLEAN***. Decided if the TREEVIEW can be used or not.
FontBold	***BOOLEAN***. True give fat text.
FontColor	***BOOLEAN***. The Color of the text.
FontItalic	***BOOLEAN***. True give slanting pressed text.
FontName	***STRING***. FONT for the text.
FontSize	***INTEGER***. Size of the signs.
FontUnderline	***BOOLEAN***. Underlines the text.
Height	***INTEGER***. The altitude of the TREEVIEW.
Hint	***STRING***. The text, which must be reflected if the hint is active. Putting showhint on true must activate these.
HWND	***INTEGER.*** Windows handle of the TREEVIEW.
ItemCount	***INTEGER.*** The number of items in the TREEVIEW.
Left	***INTEGER***. Left position of the TREEVIEW.

Parent	***OBJECT:*** The parent of the TREEVIEW
Selected(index)	***BOOLEAN.*** Returns if the item with number index has been selected or not.
SelectedIndex	***INTEGER.*** Returns the index of the selected item.
Showhint	***BOOLEAN***. True shows the hint.
Sorted	***BOOLEAN.*** At true the TREEVIEW have been sorted.
Taborder	***BOOLEAN***. The number for use with the tab key, if Taborder has been activated.
Tag	***INTEGER***. By defining identification number for the user.
Top	***INTEGER***. The upper position of the TREEVIEW.
Visible	***BOOLEAN***. At false the TREEVIEW is not visible.
Width	***INTEGER***. The breadth of the TREEVIEW.
Methods:	
BMPResource(index, resource)	Puts an image from the RESOURCE file into the imagelist.
Clear	All contents of the TREEVIEW are erased.
Lock	The treeview freeze, so there no text can be modified and the changes are not visible until the command unlock has been given.

UnLock	THE TREEVIEW reflect rather for all changes. Changes, which have been carried out during the lock, are still done.
Events	
OnActivate	**SUB([treeview])**. A sub-routine calls as the TREEVIEW the focus gets.
Onchange	**SUB([treeview], index)**. A sub-routine calls if another item is chosen.
OnClick	**SUB([treeview])**. A sub-routine calls if there is clicked on the TREEVIEW.
OnDblClick	**SUB([treeview])**. A sub-routine calls if there is clicked twice on the TREEVIEW.
OnEditing	**SUB([treeview], index, string)**. A sub-routine calls if the text has changed of the item index.

MATHEMATICAL FUNCTIONS

ABS

Syntax : ABS(argument)

Return positive integer value of a numeric expression.

Example:

```
Dim X as integer
X=abs(-1000)        ' X now is value 1000
```

ACOS

Syntax : ACOS(argument)

Return (in radians) arccosine of a numerical expression. Argument must lie between – 1 and 1.

Example:

```
DIM X AS DOUBLE
X=acos(1)          ' X has value 3.14
X=deg(acos(1))     ' X is now the angle in degrees
```

ASIN

Syntax : ASIN(argument)

Return (in radians) arcsine of a numerical expression. Argument must lie between – 1 and 1.

Example:

```
DIM X AS DOUBLE
X=asin(1)          ' X has value 1.57
X=deg(asin(1))     ' X is now the angle in degrees
```

ATAN

Syntax : ATAN(argument)

Return (in radians) arctangent of a numerical expression. Argument must lie between – 1 and 1.

Example:

```
DIM X AS DOUBLE
X=atan(1)          ' X has value 0.785
X=deg(atan(1))     ' X is now the angle in degrees
```

CEIL

Syntax : CEIL(argument)

Round a number to an upper integer value. The argument is a numerical expression.

Example:

```
DIM X AS DOUBLE
X=ceil(5.2)          ‘ X has value 6
```

COS

Syntax : COS(argument)

Return (in radians) cosine of a numerical expression.

Example:

```
DIM X AS DOUBLE
X=cos(22/7)          ‘ X has value -1
X=cos(rad(180))      ‘ RAD converted the angle of degrees to radians.
```

DEC

Syntax : DEC(argument,[argument 2])

Return argument decreased by value of argument2. If argument2 is not given, value is 1.

Example:

```
DIM X AS INTEGER
X=dec(16)        ‘ X has value 15
X=dec(16,10)     ‘ X has value 6 – decreased by value of 10
```

DEG

Syntax : DEG(argument)

Return argument in degrees. The argument is a numerical expression in radians.

Example:

```
DIM X AS INTEGER
X=deg(3.14)      ‘ X is the angle in degrees.
```

EXP

Syntax : EXP(argument)

Return exponent e of the argument. The argument is a numerical expression.

Example:

```
DIM X AS DOUBLE
X=exp(2)       ' X has value 7.28
X=exp(1)       ' X has value of e.
```

FLOOR

Syntax: FLOOR(argument)

Return round of argument to smallest whole number. The argument is a numerical expression.

Example:

```
DIM X AS INTEGER
X=floor(34.6)      ' X has now the value 34.
X=floor(-34.6)     ' X has now the value –35.
```

FRAC

Syntax : FRAC(argument)

Return decimal part of a number. The argument is a numerical expression.

Example:

```
DIM X AS DOUBLE
X=frac(34.6)     ' X has now the value 0.6 .
```

INC

Syntax : INC(argument,[argument2])

Return argument increased by value of argument2. If argument2 is not given, value is 1.

Example:

```
DIM X AS INTEGER
X=inc(16)      ' X has value 17
X=inc(16,10)   ' X has value 26
```

INT

Syntax : INT(argument)

Return whole part of a number. The argument is a numerical expression.

Example:

```
DIM X AS DOUBLE
X=int(34.6)      ' X has value 34.
```

LOG

Syntax : LOG(argument,[argument2])

Return the logarithm of argument. If argument2 not given, the course logarithm is returned.

Example:

```
DIM X AS DOUBLE
X=log(10)           ' X has value 2.3
X=log(10,100)      ' X=10log100  --  X has value 10
```

NOT

Syntax: NOT(integer)

Return not value of an INTEGER.

Example:

```
DIM X AS INTEGER
X=not(4)      ' X has value -5

X=&hff000000
X=not(x)      ' X has hexadecimal value &h00ffffff
```

RAD

Syntax : RAD(argument)

Returns radians for argument in degrees. Argument is numerical expression in degrees.

Example:

```
DIM X AS INTEGER
X=rad(360)          ' X is angle in radians.
```

RND

Syntax : RND(argument)

Return random whole number between 0 and value of argument.
The argument is a numerical expression.

Example:

```
DIM X AS INTEGER
X=rnd(10)    ' X has value between 0 and 10.
```

ROUND

Syntax : ROUND(argument)

Return argument rounded to whole number. The argument is a numerical expression.

Example:

```
DIM X AS DOUBLE
X=round(10.4)     ' X has value 10.
X=round(10.5)     ' X has value 11.
```

SHL

Syntax: SHL(numeric expression, number bits)

Shift bit range of INTEGER number to left.

Example:

```
DIM X AS INTEGER
X=shl(1,1)      ' X has value 2.
X=shl(1,2)      ' X has value 4.
```

SHR

Syntax: SHR(numerical expression, number bits)

Shift bit range of INTEGER number to Right.

Example:

```
DIM X AS INTEGER
X=shr(2,1)       ' X has value 1.
X=shr(4,2)       ' X has value 1.
```

SIN

Syntax : SIN(angle)

Return sine of a numerical expression. The angle must lie between 0 and 2 * pi radians.

Example:

```
DIM X AS DOUBLE
X=sin(22/7)      ' X has value 0
X=sin(rad(90))   ' X has sine of angle in degrees.
```

SQR

Syntax : SQR(argument)

Return square root of a numerical expression. The argument must be 0 or larger.

Example:

```
DIM X AS DOUBLE
X=sqr(16)        ' X has value 4
```

TAN

Syntax : TAN(angle)

Return tangent of a numerical expression. The angle must lie between 0 and 2 * pi radians.

Example:

```
DIM X AS DOUBLE
X=tan(22/7)         ‘ X has value 0
X=tan(rad(90))      ‘ X has tangent of angle in degrees.
```

APPENDIX -- ASCII TABLE

0	16	32	48 0	64 @	80 P	96 `	112 p	128 Ç	144 É	160 á	176 _	192 "+"	208 ð	224 Ó	240 ""
1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129 ü	145 æ	161 í	177 _	193 "-"	209 Ð	225 ß	241 ±
2	18	34 "	50 2	66 B	82 R	98 b	114 r	130 é	146 Æ	162 ó	178 _	194 "-"	210 Ê	226 Ô	242 _
3	19	35 #	51 3	67 C	83 S	99 c	115 s	131 â	147 ô	163 ú	179 ¦	195 "+"	211 Ë	227 Ò	243 ¾
4	20 ¶	36 $	52 4	68 D	84 T	100 d	116 t	132 ä	148 ö	164 ñ	180 ¦	196 "-"	212 È	228 õ	244 ¶
5	21 §	37 %	53 5	69 E	85 U	101 e	117 u	133 à	149 ò	165 Ñ	181 Á	197 "+"	213 i	229 Õ	245 §
6	22	38 &	54 6	70 F	86 V	102 f	118 v	134 å	150 û	166 ª	182 Â	198 ã	214 Í	230 µ	246 ÷
7	23	39	55 7	71 G	87 W	103 g	119 w	135 ç	151 ù	167 º	183 À	199 Ã	215 Î	231 þ	247 ¸
8	24	40 (	56 8	72 H	88 X	104 h	120 x	136 ê	152 ÿ	168 ¿	184 ©	200 "+"	216 Ï	232 Þ	248 °
9	25	41)	57 9	73 I	89 Y	105 i	121 y	137 ë	153 Ö	169 ®	185 ¦	201 "+"	217 "+"	233 Ú	249 ¨
10	26	42 *	58 :	74 J	90 Z	106 j	122 z	138 è	154 Ü	170 ¬	186 ¦	202 "-"	218 "+"	234 Û	250 ·

11	27	43 "+"	59 ;	75 K	91 [	107 k	123 {	139 ï	155 ø	171 ½	187 "+"	203 "-"	219 _	235 Ù	251 ¹
12	28	44 ,	60 <	76 L	92 \	108 l	124 |	140 î	156 £	172 ¼	188 "+"	204 ¦	220 _	236 ý	252 ³
13	29	45 "-"	61 "="	77 M	93]	109 m	125 }	141 ì	157 Ø	173 ¡	189 ¢	205 "-"	221 ¦	237 Ý	253 ²
14	30	46 .	62 >	78 N	94 ^	110 n	126 ~	142 Ä	158 ×	174 «	190 ¥	206 "+"	222 Ì	238 ¯	254 _
15 ¤	31	47	63 ?	79 O	95 _	111 o	127	143 Å	159 ƒ	175 »	191 "+"	207 ¤	223 _	239 ´	255

Appendix -- Key Values

Key values of the events keydown and keyup (key as word,shift as integer)
Shift will give the following results: Ctrl : shift=1 Alt : shift=16 Shift : shift=256

	Ctrl 17	PgUp 33	1 ! 49	a A 65	q Q 81	Pad1 97	F2 113		Scrl Lock 145						
	Alt 18	PgDwn 34	2 @ 50	b B 66	r R 82	Pad2 98	F3 114								
	Pause 19	End 35	3 # 51	a C 67	s S 83	Pad3 99	F4 115								
	Caps Lock 20	Home 36	4 $ 52	d D 68	t T 84	Pad4 100	F5 116								
		Crsr Left 37	5 % 53	e E 69	u U 85	Pad5 101	F6 117								
		Crsr Up 38	6 ^ 54	f F 70	v V 86	Pad6 102	F7 118								
		Crsr Right 39	7 & 55	g G 71	wW 87	Pad7 103	F8 119								
Back Space 8		Crsr Down 40	8 * 56	h H 72	xX 88	Pad8 104	F9 120								
Tab 9			9 (57	i I 73	yY 89	Pad9 105	F10 121								
				j J 74	z Z 90	Pad* 106	F11 122				; : 186				
	Esc 27			k K 75		Pad+ 107	F12 123				= + 187		[{ 219		
		Del 46		n N 78		Pad . 110					. > 190		‘ “ 222		
				o O 79		Pad/ 111					/ ? 191				
Shift 16	Space 32	0) 48		p P 80	Pad0 96	F1 112		Num Lock 144			` ~ 192				

APPENDIX – WINDOWS VISTA MANIFEST

Under XP and Vista, FnxBasic will work; but do not use the themes of O/S to do that -- you will need a manifest file to tell windows the FnxBasic must use the themes.

Copy the below code into a txt file (notepad) and then rename that file (i.e., if your application is called "myapp.exe", then rename the txt file to myapp.exe.manifest. Put this file into the same folder as the application is.

```
<?xml version="1.0" encoding="utf-8" ?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity version="1.0.0.0"
   processorArchitecture="X86"
   name="Vista UAC Compat.Application"
   type="win32" />
  <description>WindowsVistaReadiness Application</description>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
   <security>
    <requestedPrivileges>
      <requestedExecutionLevel level="requireAdministrator"  />
    </requestedPrivileges>
   </security>
  </trustInfo>
</assembly>
```

APPENDIX – FNX COMPILER SWITCHES

The Fnx Compiler switches are:

fnxcompiler.exe [-LANGUAGE] appfilename [-w]

The -w option will force a window message log with the error description if there is any.

A language file always starts with error and the extension is the same as the language. ie ERROR.ENG the switch is -ENG. If there is no language option the language is english.

The errors are dumped in fnx.log

APPENDIX – SAMPLE CODE

ARRAY OF BUTTONS

```
''array of buttons by marco waterman
dim myform as form
dim btn(15) as button
dim col as integer,row as integer
dim c as integer
c=0
for col=0 to 2
	for row=0 to 4
		btn(c).parent=myform
		btn(c).top=btn(c).height*row
		btn(c).left=btn(c).width*col
		btn(c).tag=c
		btn(c).caption=str$(c)
		btn(c).onclick=button_click
		inc(c)
	next row
next col
myform.ShowModal

sub button_click(sender as button)
	showmessage "button "+str$(sender.tag)+" is clicked on"
end sub
```

CLOCK

```
''example of a analog clock need some adjustments
type clock as image
	sub adjust()
		dim TI as string,X as integer,Y as integer
		TI=time$
		fillrect(0,0,100,100,&hffffff)
		X=50+(COS(VAL(RIGHT$(TI,2))*(44/7/60)-11/7))*50
		Y=50+(SIN(VAL(RIGHT$(TI,2))*(44/7/60)-11/7))*50
		LINE(50,50,X,Y,0)
		X=50+(COS(VAL(MID$(TI,4,2))*(44/7/60)-11/7))*50
		Y=50+(SIN(VAL(MID$(TI,4,2))*(44/7/60)-11/7))*50
		LINE(50,50,X,Y,&HFF0000)
		X=50+(COS(( VAL(LEFT$(TI,2) )+VAL(MID$(TI,4,2))/60)*(44/7/12)-11/7))*40
		Y=50+(SIN((VAL(LEFT$(TI,2))+VAL(MID$(TI,4,2))/60)*(44/7/12)-11/7))*40
```

```
                LINE(50,50,X,Y,&H00FF00)
        end sub
end type
object myform as form
        object cl as clock
        end object
end object

dim t as timer
t.Interval=1000
t.ontimer=cl.adjust

myform.showmodal
```

CLOSE FORM

```
''close example of a form

object myform as form
caption="close me"
onclose=FormClose
end object

myform.ShowModal

sub FormClose(action as integer)
if messagedlg("do you really want to exit the program?","close?",mbYesNo+mbIconQuestion)=idYes
then
action=2
else
action=0
end if
end sub
```

DIAFORM

```
''simple diaform example by m.waterman
Declare SetWindow as "SetWindowRgn" of "user32"
        hwnd As integer
        hRgn As integer
        bRedraw As integer
end declare
declare elips as "CreateEllipticRgn" of "gdi32"
```

```
        x1 as integer
        y1 as integer
        x2 as integer
        y2 as integer
        result as integer
end declare

sub paint(x as integer,y as integer,x1 as integer,y1 as integer,myform as form)
        elips.x2=x
        elips.y2=y
        elips.x1=x1
        elips.y1=y1
        elips
        with setwindow:hwnd=myform.hwnd:hrgn=elips.result:bredraw=1:end with
        setwindow
end sub
dim stop as boolean

object dform as form
        onpaint=start
        left=(screenwidth-dform.width)/2
        top=(screenheight-dform.height)/2
end object
paint(0,0,0,0,dform)
dform.ShowModal

sub start()
        dim runonce as boolean
        if runonce=true then exit function
        runonce=true
        dim x as integer,y as integer
        dim x1 as integer,y1 as integer
        x=dform.width/2
        y=dform.height/2
        y1=y
        x1=x
        do
                sleep 8
                x=x+1
                y=y+1
                y1=y1-1
                x1=x1-1
                if (x^2+y^2)/2>(dform.height^2+dform.Width^2) then exit do
                paint(x,y,x1,y1,dform)
        loop
end sub
```

Form In Form

```
''a form into a form example
object myform1 as form
	object myform2 as form
		show
	end object
end object

myform1.ShowModal
```

Get Computer Name

```
''example get computer name by marco waterman
''uses an api call with pointers to the size and name

declare getcompname as "GetComputerNameA" of "kernel32.lib"
lpbuffer as string byaddress
lpsize as integer byaddress
result as integer
end declare

getcompname.lpsize=255
getcompname
end with
showmessage field$(getcompname.lpbuffer,chr$(0),1)
```

Graphics

```
''graphics by marco waterman
object myform as form
	onactivate=start
	caption="graphs"
	object grap as image
		align=alClient
	end object
end object
myform.showmodal

sub marks()
	dim x as integer,y as integer
```

```
    for x=10 to grap.width step 10
        grap.line(x,grap.Height-10,x,grap.Height,0)
    next x
    for y=grap.height to 10 step -10
        grap.line(0,y,10,y,0)
    next y
end sub
sub drawgrap()
    dim x as integer,y as integer
    for x=20 to grap.Width step 5
        y=rnd(grap.height)-10
        grap.lineto(x,y,&h0000ff)
    next x
end sub
sub start()
    marks()
    drawgrap()
end sub
```

LED

```
'Example of the led.lib file
uses led
object myform as form
    object btn as button
        caption="  Click me"
        onclick=clicked
        object ld as led
            left=6:top=6
            color=myform.Color
            ldcolor=ldRed
            size=btn.Height-12
            paint()
        end object
    end object
end object
myform.ShowModal

sub clicked()
    if ld.ldon=false then
        ld.ldon=true
    else
        ld.ldon=false
    end if
    ld.paint()
```

```
end sub
```

USING MYSQL

```
''example of the use of mysql
''instal mysql anonymious, without a password and user.
''for encription is needed more than this example.
''mysql is a freeware database server
uses mysql
object myform as form
object grid as stringgrid
align=alClient
separator=chr$(8)
end object
end object
dim db as mysql
db.separator=grid.separator
db.address="127.0.0.1"
db.port=3306
db.user="root"
db.password=""
showmessage db.query("connect")
db.query("create database if not exists mydbase")
db.query("use mydbase")
db.query("create table if not exists mytable(name CHAR(10),zipcode char(10),address char(20))")
db.query("INSERT INTO mytable (name,zipcode,address) VALUES('john','1234','bublestreet')")
db.query("INSERT INTO mytable (name,zipcode,address) VALUES('mike','5678','teststreet')")
grid.Text=db.query("Select * from mydbase.mytable")''*=everything
myform.showmodal
```

PLAY MIDI FILE

```
''playing a midi file by marco waterman
''this api can do more things like opening the cd tray or
''playing avi files.

Declare sendstring as "mciSendStringA" of "winmm.dll"
        lpstrCommand As String
        lpstrReturnString As String
        uReturnLength As integer
        hwndCallback As integer
        result as integer
end declare
```

```
object myform as form
	caption="play a mid file"
	object play as button
		caption="play"
		onclick=play_click
	end object
	object quit as button
		left=play.width
		caption="quit"
		onclick=quit_click
	end object
end object
dim file as filedialog
myform.ShowModal

dim alias as string
sub play_click()
	file.caption="select a midi file"
	file.Filter = "Midi files (*.mid)|*.mid"
	file.Show

	alias=part$(file.filename,"\",instr(file.filename,"."))
	alias=field$(alias,".",1)
	sendstring.lpstrcommand="OPEN "+chr$(34)+file.FileName+chr$(34)+" TYPE SEQUENCER
ALIAS "+alias
	sendstring
	sendstring.lpstrcommand="play "+alias+" from 0"
	sendstring
end sub

sub quit_click()
	sendstring.lpstrcommand="stop "+alias
	sendstring
end sub
```

Progress Bar In Status Bar

```
object myform as form
	caption="progressbar in statusbar"
	object status as statusbar
		simplepanel=True
		simpletext="progress"
		object progress as progressbar
			height=status.height-5
```

```
                top=4
                left=50
                position=progress.max/2
            end object
        end object
end object
myform.ShowModal
```

SORT

```
''example of binairely sort by marco waterman
''fnx can not handle recursive calls for a part,  every variable in a subroutine is overwritten by a
new call to this routine so it is possible with a work around with array's
Object myform as form
        onactivate=fill
        object myedit as richedit
                align=alleft
                scrollbars=ssVertical
                hidescrollbars=False
        end object
        object sort as button
                onclick=sort_click
                left=myform.clientwidth-sort.width
                caption="sort"
        end object
end object

myform.ShowModal

sub fill()
        dim counter as integer
        dim counter1 as integer,s as string
        for counter=1 to 1000
                s=chr$(65+rnd(26))
                for counter1=1 to 5
                        s=s+chr$(97+rnd(26))
                next counter1
                s=s+chr$(13)+chr$(10)
                myedit.text=myedit.text+s
        next counter
end sub

sub sort_data(leftparam as integer,rightparam as integer)
        ''fnx basic does not keeps a stack for local varibles
        ''these are global for recursive calls
```

```
    dim decider as string,buffer as string
    dim r as integer,l as integer,stackpointer as integer
    ''these are the stack locals
    dim left(500) as integer
    dim right(500) as integer
    dim leftstack(500) as integer
    dim rightstack(500) as integer

    inc(stackpointer)''this pointer keeps track on the locals
    left(stackpointer)=leftparam
    leftstack(stackpointer)=leftparam
    right(stackpointer)=rightparam
    rightstack(stackpointer)=rightparam
    decider=myedit.line(int((left(stackpointer)+right(stackpointer))/2))
    do
        while myedit.line(left(stackpointer))<decider
            inc(left(stackpointer))
        wend
        while myedit.line(right(stackpointer))>decider
            dec(right(stackpointer))
        wend
        if left(stackpointer)<=right(stackpointer) then
            ''swap
            buffer=myedit.line(left(stackpointer))
            l=left(stackpointer)
            myedit.line(l)=myedit.line(right(stackpointer))
            r=right(stackpointer)
            myedit.line(r)=buffer

            inc(left(stackpointer))
            dec(right(stackpointer))
        end if
    loop until left(stackpointer)>right(stackpointer)
    if leftstack(stackpointer)<right(stackpointer) then
sort_data(leftstack(stackpointer),right(stackpointer))
    if left(stackpointer)<rightstack(stackpointer) then
sort_data(left(stackpointer),rightstack(stackpointer))
    dec(stackpointer)''if finished go to the locals of the previous routine
end sub

sub sort_click()
    myedit.lock
    sort_data(0,1000)
    myedit.unlock
end sub
```

USING SPLITTER

```
object myform as form
	caption="use of splitter"
	object myedit as richedit
		align=alLeft
	end object
	object split as splitter
		left=myedit.Width
		width=10
	end object
	object grid as stringgrid
		align=alClient
	end object
end object
myform.ShowModal
```

CHAT CLIENT

```
''chat client example by marco waterman
''run the server first
object myform as form
	left=400
	caption="type some text and click send"
	object edit as richedit
		height=myform.clientheight-40
		width=myform.Clientwidth
	end object
	object but as button
		top=edit.height+10
		caption="send"
		onclick=send_click
	end object
end object
dim client as clientsocket
client.Port=13
client.Address="127.0.0.1"
client.Open
client.OnConnected=connected
myform.showmodal
client.close
sub send_click()
	client.Send(edit.text)
end sub
sub connected()
	showmessage "connected"
```

```
end sub
```

Chat Server

```
''simple chat example by marco waterman
''this part is the chat server
''run the exe first before making connection with the client

object myform as form
	caption="server"
	object edit as richedit
		align=alClient
	end object
end object
dim server as serversocket

server.Port=13
server.OnRead=server_read
server.Open

myform.ShowModal
server.Close
sub server_read()
	edit.text=edit.text+server.Receive+chr$(13)+chr$(10)
end sub
```

PopupMenu And MainMenu

```
object myform as form
  object fomenu as mainmenu
      object menu1 as menuitem
      AutoCheck=True
      Caption="menu_1"
      object menu3 as menuitem
      Caption="menu_3"
      onclick=menuclick3
      end object
      object menu4 as menuitem
      Caption="menu_4"
      onclick=menuclick4
      end object
      end object
      object menu2 as menuitem
```

```
        AutoCheck=True
        Caption="menu_2"
        onclick=menuclick2
        end object
    end object 'fomenu
    object m as  popupmenu
        object s as menuitem
            caption="sub1"
            onclick=clicked
        end object
        object s2 as menuitem
            caption="sub2"
                object s3 as menuitem
                    caption="subsub3"
                    onclick=clicked
                end object
        end object
    end object
end object' myform

 myform.showmodal

sub menuclick2(sender as menuitem)
    if sender=menu2 then showmessage "menu2 clicked"
end sub
sub menuclick3(sender as menuitem)
    if sender=menu2 then showmessage "menu3 clicked"
end sub
sub menuclick4(sender as menuitem)
    if sender=menu2 then showmessage "menu4 clicked"
end sub
sub clicked(sender as menuitem)
   showmessage sender.caption+" clicked"
end sub
```

ADD ITEMS TO LIST BOX

```
object myform as form
    center=true
    myform.AutoSize=true
    object LB_one as listbox
        width=200:height=200
        item(0)="Please click on this item!"
        item(1)="text 1"
        onclick=add_item
```

```
    end object
end object
myform.ShowModal
sub add_item()
    dim i as integer
    i=LB_one.ItemCount
    LB_one.item(i)="text "+str$(i)
end sub
```

SEND KEY PROGRAM

```
‘ API Call - SendKey Program -- Controls Another Windows App
'' Actual example (See botton of code)
‘
const KEYEVENTF_KEYUP = &H2 ' Release key
const VK_LBUTTON=&h01 'Left mouse button.
const VK_RBUTTON=&h02 'Right mouse button.
const VK_CANCEL=&h03 'Used for control-break processing.
const VK_MBUTTON=&h04h 'Middle mouse button (3-button mouse).
const VK_BACK=&h08
const VK_TAB=&h09
const VK_CLEAR=&h0C
const VK_RETURN=&h0D
const VK_SHIFT=&h10
const VK_CONTROL=&h11
const VK_MENU=&h12 '= alt key
const VK_PAUSE=&h13
const VK_CAPITAL=&h14
const VK_ESCAPE=&h1B
const VK_SPACE=&h20
const VK_PRIOR=&h21 'Page up.
const VK_NEXT=&h22 'Page down.
const VK_END=&h23
const VK_HOME=&h24
const VK_LEFT=&h25
const VK_UP=&h26
const VK_RIGHT=&h27
const VK_DOWN=&h28
const VK_SELECT=&h29
const VK_INSERT=&h2D
const VK_DELETE=&h2E
const VK_HELP=&h2F
const VK_0=&h30
const VK_1=&h31
const VK_2=&h32
```

```
const VK_3=&h33
const VK_4=&h34
const VK_5=&h35
const VK_6=&h36
const VK_7=&h37
const VK_8=&h38
const VK_9=&h39
const VK_A=&h41
const VK_B=&h42
const VK_C=&h43
const VK_D=&h44
const VK_E=&h45
const VK_F=&h46
const VK_G=&h47
const VK_H=&h48
const VK_I=&h49
const VK_J=&h4A
const VK_K=&h4B
const VK_L=&h4C
const VK_M=&h4D
const VK_N=&h4E
const VK_O=&h4F
const VK_P=&h50
const VK_Q=&h51
const VK_R=&h52
const VK_S=&h53
const VK_T=&h54
const VK_U=&h55
const VK_V=&h56
const VK_W=&h57
const VK_X=&h58
const VK_Y=&h59
const VK_Z=&h5A
const VK_NUMPAD0=&h60
const VK_NUMPAD1=&h61
const VK_NUMPAD2=&h62
const VK_NUMPAD3=&h63
const VK_NUMPAD4=&h64
const VK_NUMPAD5=&h65
const VK_NUMPAD6=&h66
const VK_NUMPAD7=&h67
const VK_NUMPAD8=&h68
const VK_NUMPAD9=&h69
const VK_MULTIPLY=&h6A
const VK_ADD=&h6B
const VK_SEPARATER=&h6C
const VK_SUBTRACT=&h6D
const VK_DECIMAL=&h6E
```

```
const VK_DIVIDE=&h6F
const VK_F1=&h70
const VK_F2=&h71
const VK_F3=&h72
const VK_F4=&h73
const VK_F5=&h74
const VK_F6=&h75
const VK_F7=&h76
const VK_F8=&h77
const VK_F9=&h78
const VK_F10=&h79
const VK_F11=&h7A
const VK_F12=&h7B
const VK_F13=&h7C
const VK_F14=&h7D
const VK_F15=&h7E
const VK_F16=&h7F
const VK_F17=&h80
const VK_F18=&h81
const VK_F19=&h82
const VK_F20=&h83
const VK_F21=&h84
const VK_F22=&h85
const VK_F23=&h86
const VK_F24=&h87

declare keybrd as "keybd_event" of "user32"
  bVk As Byte
  bScan As Byte
  dwFlags As Long
  dwExtraInfo As Long
end declare

sub keydown(key as integer)
keybrd.bvk=key
keybrd.dwExtraInfo=0
keybrd
end sub

sub keyup(key as integer)
   keybrd.bvk=key
   keybrd.dwExtraInfo=KEYEVENTF_KEYUP
   keybrd
end sub
''================ end declaration
''example
shell("c:\windows\notepad.exe","",1)''startup notepad
sleep 500'' wait
```

keydown(65)''65 is a to get an A fou must use the vk_shift first

---

Status Bar Panels

```
object myform as form
   width=400
   caption="statbar_test by PaPi"
   object mystatbar as statusbar
   end object
   object mybutton as button
      width=300
      caption="Press My!"
      onclick=mybutton_click
   end object
end object
myform.ShowModal

sub mybutton_click()
dim i as integer, j as integer
inc(j)
if j<6 then
   mybutton.caption="Press again, please!"
else
   mybutton.left=20:mybutton.top=20:mybutton.height=40
   mybutton.fontcolor=&hFF
   mybutton.caption="Do not press again, please!!!!!"
   exit sub
end if

for i=0 to 4
   mystatbar.panelwidth(i)=70
   if i<4 then
      mystatbar.paneltext(i)="Panel "+str$(i)+" here"
   else
      mystatbar.paneltext(i)="Your "+str$(j)+". press"
   end if
next i
end sub
```

TAB CONTROL OBJECT

```
object myform as form
    caption="tabcontrol test"
    myform.AutoSize=true
    object Tabs as tabcontrol
         width=400:height=200
        onChange=testing

        Tabs(0)="Test 1"
'     .
'     .
' you can to place here others object in tabs(0)
'     .
'     .
'     for example this:
                object label0 as label
                   top=30 :left=10':width=tabs.width/2-5 : height=tabs.height-5
                   visible=false
                   Color=&h80FF80
                   caption=" This is the tab Test1"
              end object ' label2

       Tabs(1)="Test 2"
'     .
'     .
'  you can to place here some object in tabs(1)
'     .
'     .
'     for example this:
           object label1 as label
                   top=40 :left=tabs.width/2':width=tabs.width/2 -5: height=tabs.height-5
                   visible=false
                   color=&h8080FF
                   caption="This is the tab Test2"
              end object ' label1

    end object ' tabs

end object 'myform

myform.ShowModal

Sub testing()
    IF Tabs.tabindex = 0 then
         label0.visible=true
    else
```

```
        label0.visible=false
    end if

    IF Tabs.tabindex = 1 then
         label1.visible=true
    else
        label1.visible=false
    end if

End Sub
```

App With One SysTray Button

PROBLEM:

```
dim f as form
f.Caption="test"
f.showmodal
```

When above program is run, 2 buttons appear in system tray.
A Form button and a button with form caption of code.

FIX:

FnxBasic Support reports: This is a known issue and no solution as yet.
Program runs with one button for applicasion and other for form.
For now on, asign application to main form with below code (place at top of program, then put the "setparent(<main form>.Hwnd" before <main form>.Showmodal):

```
' ----------------------------------------------------------
'  Fix -- Make Program Run With One System Tray Button
' ----------------------------------------------------------
' Note:
'    Add  "setparent(<main form>.Hwnd)"
'    before <main form>.Showmodal statement
' ----------------------------------------------------------
'
dim app as application
declare set_windowlong as "SetWindowLongA" of "user32.lib"
   hwnd As Long
nIndex As Long
dwNewLong As Long
end declare

sub setparent(fhwnd as integer)
```

```
set_windowlong.hwnd=app.handle
set_windowlong.nindex=-8
set_windowlong.dwnewlong=fhwnd
set_windowlong
end sub
'
' -----------------------------------------------------------
'
‘ Program Example using above code:
‘
dim f as form
f.caption="test"

setparent(f.hwnd)
f.showmodal
```

APPENDIX – SAMPLE PROGRAMS

FLAT DATABASE

USES: FileStream, Edit, RichEdit, Button, Select Case, MOD [remainder of division]
CONCEPT:

- The purpose is FileStream
- Load a "flat" [csv-comma separated value] database.
- Click load to bring it into sData() string array
- Use Up/Down to navigate.
- Clear blanks the edits so new data can be entered, then click ADD
- The sData() and RichEdit are updated.

```
dim myDB as FileStream
dim sData() as String
dim s1 as string
dim s as string
dim ndx as integer
dim dbCnt as integer
dim delim as string

dbCnt=0
delim = ","
OBJECT democust AS Form
 Left=271
 Top=106
 Caption="Demo Customer Flat Database"
 ClientHeight=393
 ClientWidth=384
 Color=&HFF00000F 'clBtnFace
 OBJECT Label1 AS Label
  Left=156
  Top=36
  Width=19
  Height=13
  Caption="First"
 END OBJECT '{Label1}
 OBJECT Label2 AS Label
  Left=155
  Top=68
  Width=20
  Height=13
  Caption="Last"
 END OBJECT '{Label2}
 OBJECT Label3 AS Label
```

```
  Left=144
  Top=100
  Width=31
  Height=13
  Caption="Phone"
END OBJECT '{Label3}
OBJECT Label4 AS Label
  Left=112
  Top=128
  Width=69
  Height=13
  Caption="Current Index: "
END OBJECT '{Label4}
OBJECT RichEdit1 AS RichEdit
  Left=16
  Top=288
  Width=185
  Height=89
  Line(0)="RichEdit1"
  PlainText=True
  TabOrder=0
END OBJECT '{RichEdit1}
OBJECT bLoadDB AS Button
  Left=16
  Top=16
  Width=75
  Height=25
  Caption="LoadDB"
  TabOrder=1
  OnClick=bLoadDBClick
END OBJECT '{bLoadDB}
OBJECT bPrev AS Button
  Left=16
  Top=80
  Width=75
  Height=25
  Caption="Prev"
  Enabled=False
  TabOrder=2
  OnClick=bPrevClick
END OBJECT '{bPrev}
OBJECT bNext AS Button
  Left=16
  Top=128
  Width=75
  Height=25
  Caption="Next"
  Enabled=False
  TabOrder=3
  OnClick=bNextClick
END OBJECT '{bNext}
OBJECT bAdd AS Button
  Left=16
  Top=176
```

```
    Width=75
    Height=25
    Caption="Add"
    Enabled=False
    TabOrder=4
    OnClick=bAddClick
  END OBJECT '{bAdd}
  OBJECT eFirst AS Edit
    Left=176
    Top=32
    Width=121
    Height=21
    TabOrder=5
    Text="eFirst"
    PasswordChar=False
  END OBJECT '{eFirst}
  OBJECT eLast AS Edit
    Left=176
    Top=64
    Width=121
    Height=21
    TabOrder=6
    Text="eLast"
    PasswordChar=False
  END OBJECT '{eLast}
  OBJECT ePhone AS Edit
    Left=176
    Top=96
    Width=121
    Height=21
    TabOrder=7
    Text="ePhone"
    PasswordChar=False
  END OBJECT '{ePhone}
  OBJECT bClear AS Button
    Left=112
    Top=176
    Width=75
    Height=25
    Caption="Clear"
    TabOrder=8
    OnClick=bClearClick
  END OBJECT '{bClear}
END OBJECT '{democust}

democust.ShowModal

SUB bLoadDBClick()
  'your code here
  RichEdit1.line(1)="test"
  myDB.Open("democust.db")
  ndx=0
  dbCnt=myDB.linecount
  dbCnt=dbCnt-1 ' 3 lines 0-1-2 so LineCount - 1
```

```
  s=""
  do
    's1=myDB.ReadLine
    's = s + s1
    s=myDB.ReadLine
    RichEdit1.line(ndx)=s
    sData(ndx)=s
    ndx=ndx+1
    if ndx>dbCnt then exit do
  loop
  myDB.Close
  ndx=0
  'pre-load the edit boxes
  eLast.text=field$(sData(ndx),delim,1)
  eFirst.text=field$(sData(ndx),delim,2)
  ePhone.text=field$(sData(ndx),delim,3)
  Label4.Caption="Current Index: "+str$(ndx)
  bLoadDB.enabled=False
  bNext.enabled=True
  bPrev.enabled=True
END SUB  '{bLoadDBClick}

SUB bPrevClick()
  'your code here
  if ndx-1<0 then exit sub
  ndx=ndx-1
  Label4.Caption="Current Index: "+str$(ndx)
  eLast.text=field$(sData(ndx),delim,1)
  eFirst.text=field$(sData(ndx),delim,2)
  ePhone.text=field$(sData(ndx),delim,3)
END SUB  '{bPrevClick}

SUB bNextClick()
  'your code here
  if ndx+1>dbCnt then exit sub
  ndx=ndx+1
  Label4.Caption="Current Index: "+str$(ndx)
  eLast.text=field$(sData(ndx),delim,1)
  eFirst.text=field$(sData(ndx),delim,2)
  ePhone.text=field$(sData(ndx),delim,3)
END SUB  '{bNextClick}

SUB bAddClick()
  'your code here
  dbCnt=dbCnt+1
  ndx=dbCnt-1
  DIM sData(dnx) as String
  sData(ndx)=eLast.Text+","+eFirst.Text+","+ePhone.Text
  RichEdit1.Line(ndx+1)=sData(ndx)
  bAdd.enabled=False
END SUB  '{bAddClick}

SUB bClearClick()
  'your code here
```

```
  eLast.Text=""
  eFirst.Text=""
  ePhone.Text=""
  bAdd.enabled=True
END SUB  '{bClearClick}
```

IMAGE LIST

USES: ImageList, Image, Panel, FaceButton, Edit, CheckBox, TabControl, Select Case
CONCEPT:

- The tabcontol has 3 tabs and 3 BMPresourse – take up space
- ImageList has 10 files from Glyphs
- Panel and 2 FaceButtons simulate a Window Spinner Button [up/down]
- Edit shows the File Name assigned to the Resource
- CheckBox to set the Image.Stretch
- When the spinner is inc/dec if the ndx [count] is 0 the TabControl default images are load otherwise using ndx mod 3 a select case sees the answer of 0,1,2 and load the TabControl with a different set of images from the ImageList

```
RESOURCE resImageList1_0 AS "..\Glyphs\abort.bmp"
RESOURCE resImageList1_1 AS "..\Glyphs\all.bmp"
RESOURCE resImageList1_2 AS "..\Glyphs\cancel.bmp"
RESOURCE resImageList1_3 AS "..\Glyphs\close.bmp"
RESOURCE resImageList1_4 AS "..\Glyphs\help.bmp"
RESOURCE resImageList1_5 AS "..\Glyphs\ignore.bmp"
RESOURCE resImageList1_6 AS "..\Glyphs\no.bmp"
RESOURCE resImageList1_7 AS "..\Glyphs\ok.bmp"
RESOURCE resImageList1_8 AS "..\Glyphs\retry.bmp"
RESOURCE resImageList1_9 AS "..\Glyphs\yes.bmp"
RESOURCE resTabControl1_0 AS "..\Glyphs\abort.bmp"
RESOURCE resTabControl1_1 AS "..\Glyphs\all.bmp"
RESOURCE resTabControl1_2 AS "..\Glyphs\close.bmp"

DIM Display(9) AS String
Display(0)="..\Images\save2.bmp"
Display(1)="..\Images\ShowAlignLine4.bmp"
Display(2)="..\Images\ShowAlignLine.bmp"
Display(3)="..\Images\shape.bmp"
Display(4)="..\Images\selector.bmp"
Display(5)="..\Images\scrollbox.bmp"
Display(6)="..\Images\scroll.bmp"
Display(7)="..\Images\ShowAlignLine-NO.bmp"
Display(8)="..\Glyphs\retry.bmp"
Display(9)="..\Glyphs\yes.bmp"

OBJECT NewForm AS Form
  Left=271
  Top=106
  Caption="New Form"
```

```
ClientHeight=393
ClientWidth=384
Color=&HFF00000F 'clBtnFace
OBJECT Image1 AS Image
 Left=29
 Top=67
 Width=105
 Height=105
 PixelFormat=pfDevice
END OBJECT '{Image1}
OBJECT Label1 AS Label
 Left=264
 Top=198
 Width=32
 Height=13
 Caption="Label1"
END OBJECT '{Label1}
OBJECT TabControl1 AS TabControl
 Left=32
 Top=16
 Width=289
 Height=26
 TabOrder=0
 Tabs(0)="1"
 Tabs(1)="2"
 Tabs(2)="3"
 TabIndex=0
 BMPresource(0,resTabControl1_0)
 BMPresource(1,resTabControl1_1)
 BMPresource(2,resTabControl1_2)
END OBJECT '{TabControl1}
OBJECT Panel1 AS Panel
 Left=240
 Top=194
 Width=18
 Height=21
 BevelOuter=bvSpace
 TabOrder=1
 OBJECT bUp AS FaceButton
  Left=1
  Top=1
  Width=16
  Height=9
  Align=alTop
  Caption="5"
  Flat=True
  FontColor=&HFF000008 'clWindowText
  FontSize=8
  FontName="Marlett"
  FontBold=True
  NumBMP=1
  OnClick=bUpClick
 END OBJECT '{bUp}
 OBJECT bDown AS FaceButton
```

```
      Left=1
      Top=11
      Width=16
      Height=9
      Align=alBottom
      Caption="6"
      Enabled=False
      Flat=True
      FontColor=&HFF000008 'clWindowText
      FontSize=8
      FontName="Marlett"
      FontBold=True
      NumBMP=1
      OnClick=bDownClick
    END OBJECT '{bDown}
  END OBJECT '{Panel1}
  OBJECT Edit1 AS Edit
    Left=32
    Top=194
    Width=208
    Height=21
    TabOrder=2
    Text="Edit1"
    PasswordChar=False
  END OBJECT '{Edit1}
  OBJECT cbStretch AS CheckBox
    Left=32
    Top=226
    Width=97
    Height=17
    Caption="Stretch"
    TabOrder=3
    OnClick=cbStretchClick
  END OBJECT '{cbStretch}
  OBJECT cbProportional AS CheckBox
    Left=32
    Top=254
    Width=97
    Height=17
    Caption="Proportional"
    TabOrder=4
    OnClick=cbProportionalClick
  END OBJECT '{cbProportional}
END OBJECT '{NewForm}

DIM ndx AS integer
ndx=0

Image1.BMPresource(resImageList1_0)
Edit1.Text=Display(0)
Label1.Caption=str$(ndx+1)+" of 10"

NewForm.ShowModal
```

```
SUB bUpClick()
  'your code here
  ndx=ndx+1
  if ndx>9 then ndx=9

  ChangeTabBMP()

  select case ndx
    case 9
      bUp.Enabled=False
      bDown.Enabled=True
    case <9
      bUp.Enabled=True
      bDown.Enabled=True
  end select
  select case ndx
    case 0
      Image1.BMPresource(resImageList1_0)
      Edit1.Text=Display(0)
    case 1
      Image1.BMPresource(resImageList1_1)
      Edit1.Text=Display(1)
    case 2
      Image1.BMPresource(resImageList1_2)
      Edit1.Text=Display(2)
    case 3
      Image1.BMPresource(resImageList1_3)
      Edit1.Text=Display(3)
    case 4
      Image1.BMPresource(resImageList1_4)
      Edit1.Text=Display(4)
    case 5
      Image1.BMPresource(resImageList1_5)
      Edit1.Text=Display(5)
    case 6
      Image1.BMPresource(resImageList1_6)
      Edit1.Text=Display(6)
    case 7
      Image1.BMPresource(resImageList1_7)
      Edit1.Text=Display(7)
    case 8
      Image1.BMPresource(resImageList1_8)
      Edit1.Text=Display(8)
    case 9
      Image1.BMPresource(resImageList1_9)
      Edit1.Text=Display(9)
  end select
  Label1.Caption=str$(ndx+1)+" of 10"
END SUB  '{bUpClick}

SUB bDownClick()
  'your code here
  ndx=ndx-1
```

```
  if ndx<0 then ndx=0

  ChangeTabBMP()

  select case ndx
    case 0
      bDown.Enabled=False
      bUp.Enabled=True
    case >0
      bDown.Enabled=True
      bUp.Enabled=True
  end select
  select case ndx
    case 0
      Image1.BMPresource(resImageList1_0)
      Edit1.Text=Display(0)
    case 1
      Image1.BMPresource(resImageList1_1)
      Edit1.Text=Display(1)
    case 2
      Image1.BMPresource(resImageList1_2)
      Edit1.Text=Display(2)
    case 3
      Image1.BMPresource(resImageList1_3)
      Edit1.Text=Display(3)
    case 4
      Image1.BMPresource(resImageList1_4)
      Edit1.Text=Display(4)
    case 5
      Image1.BMPresource(resImageList1_5)
      Edit1.Text=Display(5)
    case 6
      Image1.BMPresource(resImageList1_6)
      Edit1.Text=Display(6)
    case 7
      Image1.BMPresource(resImageList1_7)
      Edit1.Text=Display(7)
    case 8
      Image1.BMPresource(resImageList1_8)
      Edit1.Text=Display(8)
    case 9
      Image1.BMPresource(resImageList1_9)
      Edit1.Text=Display(9)
  end select
  Label1.Caption=str$(ndx+1)+" of 10"
END SUB  '{bDownClick}

SUB cbStretchClick()
  'your code here
  Image1.Stretch=cbStretch.Checked
END SUB  '{cbStretchClick}

SUB cbProportionalClick()
  'your code here
```

```
  Image1.Proportional=cbProportional.Checked
END SUB  '{cbProportionalClick}

SUB ChangeTabBMP()
  if ndx=0 then
    TabControl1.BmpResource(0,resImageList1_0)
    TabControl1.BmpResource(1,resImageList1_1)
    TabControl1.BmpResource(2,resImageList1_2)
  else
    select case ndx mod 3
      case 0
        TabControl1.BmpResource(0,resImageList1_0)
        TabControl1.BmpResource(1,resImageList1_1)
        TabControl1.BmpResource(2,resImageList1_2)
      case 1
        TabControl1.BmpResource(0,resImageList1_3)
        TabControl1.BmpResource(1,resImageList1_4)
        TabControl1.BmpResource(2,resImageList1_5)
      case 2
        TabControl1.BmpResource(0,resImageList1_6)
        TabControl1.BmpResource(1,resImageList1_7)
        TabControl1.BmpResource(2,resImageList1_8)
    end select
  end if
END SUB
```

IMAGE AND RESOURCE

USES: Image, CheckBox, Button
CONCEPT:

- Preload images in BMPresource and navigate them in the Image.
- Change the CheckBox and caption to reverse the indexing

```
RESOURCE Image1_0 AS "..\Glyphs\abort.bmp"
RESOURCE Image1_1 AS "..\Glyphs\all.bmp"
RESOURCE Image1_2 AS "..\Glyphs\cancel.bmp"
RESOURCE Image1_3 AS "..\Glyphs\close.bmp"
RESOURCE Image1_4 AS "..\Glyphs\help.bmp"
RESOURCE Image1_5 AS "..\Glyphs\ignore.bmp"
RESOURCE Image1_6 AS "..\Glyphs\no.bmp"
RESOURCE Image1_7 AS "..\Glyphs\ok.bmp"
RESOURCE Image1_8 AS "..\Glyphs\retry.bmp"
RESOURCE Image1_9 AS "..\Glyphs\yes.bmp"

OBJECT NewForm AS Form
  Left=271
  Top=106
```

```
  Caption="New Form"
  ClientHeight=393
  ClientWidth=384
  Color=&HFF00000F 'clBtnFace
  OBJECT Image1 AS Image
    Left=64
    Top=80
    Width=105
    Height=105
    PixelFormat=pfDevice
  END OBJECT '{Image1}
  OBJECT Label1 AS Label
    Left=32
    Top=38
    Width=32
    Height=13
    Caption="Label1"
  END OBJECT '{Label1}
  OBJECT Button1 AS Button
    Left=80
    Top=32
    Width=75
    Height=25
    Caption="Button1"
    TabOrder=0
    OnClick=Button1Click
  END OBJECT '{Button1}
  OBJECT cbForward AS CheckBox
    Left=176
    Top=32
    Width=97
    Height=17
    Caption="Forward"
    TabOrder=1
    OnClick=cbForwardClick
  END OBJECT '{cbForward}
END OBJECT '{NewForm}

dim x as integer
x=0

NewForm.ShowModal

SUB Button1Click()
  'your code here
```

```
  if x=9 then cbForward.Checked=True
  if x=0 then cbForward.Checked=False
  if cbForward.Checked=False then
    x=x+1
  else
    x=x-1
  end if
  Label1.caption=str$(x)
  select case x
    case 0
      Image1.BMPresource(Image1_0)
    case 1
      Image1.BMPresource(Image1_1)
    case 2
      Image1.BMPresource(Image1_2)
    case 3
      Image1.BMPresource(Image1_3)
    case 4
      Image1.BMPresource(Image1_4)
    case 5
      Image1.BMPresource(Image1_5)
    case 6
      Image1.BMPresource(Image1_6)
    case 7
      Image1.BMPresource(Image1_7)
    case 8
      Image1.BMPresource(Image1_8)
    case 9
      Image1.BMPresource(Image1_9)
  end select
END SUB  '{Button1Click}

SUB cbForwardClick()
  'your code here
  cbForward.caption="Reverse"
END SUB  '{cbForwardClick}
```

List View Vs Icon

USES: ListView
CONCEPT: Shows using BMPresource to load images for ListView in default mode

```
RESOURCE resListView1_0 AS "E:\Delphi2006-Working\FnxDesigner\Images\AlignBottom.bmp"
RESOURCE resListView1_1 AS "E:\Delphi2006-Working\FnxDesigner\Images\AlignCenter.bmp"
RESOURCE resListView1_2 AS "E:\Delphi2006-Working\FnxDesigner\Images\AlignLeft.bmp"

OBJECT NewForm AS Form
 Left=271
 Top=106
 Caption="New Form"
 ClientHeight=393
 ClientWidth=384
 Color=&HFF00000F
 OBJECT ListView1 AS ListView
  Left=10
  Top=60
  Width=354
  Height=212
  ItemCaption(0)="NewItem0"
  BMPresource(0,resListView1_2)
  ItemCaption(1)="NewItem1"
  BMPresource(1,resListView1_1)
  ItemCaption(2)="NewItem2"
  BMPresource(2,resListView1_0)
  TabOrder=0
  Sorted=False
 END OBJECT
END OBJECT

NewForm.ShowModal
```

List View Vs Report

USES: ListView
CONCEPT: Same as vsIcon but is now set for Report view and having check boxes

```
OBJECT NewForm AS Form
 Left=271
 Top=106
 Caption="New Form"
 ClientHeight=393
 ClientWidth=384
```

```
 Color=&HFF00000F
 OBJECT ListView1 AS ListView
  Left=20
  Top=90
  Width=354
  Height=198
  Checkboxes=True
  ColCaption(0)="c1"
  ColWidth(0)= 60
  ColCaption(1)="c2"
  ColWidth(1)= 60
  ColCaption(2)="c3"
  GridLines=True
  ItemCaption(0)="i1"
   ItemCaption(1,0)="i2"
   ItemCaption(2,0)="i3"
  ItemCaption(1)="i2"
  ItemCaption(2)="i3"
  TabOrder=0
  ViewStyle=vsReport
  Sorted=False
 END OBJECT
END OBJECT

NewForm.ShowModal
```

MULTI-PANEL

USES: Panels, StatusBar, RadioButton, CheckBox, TrackBar.
CONCEPT:

- Shows putting controls in “container” controls [panel] and last panel of StatusBar.
- 2 buttons hide and show Panel3
- Grouping of 4 RadioButtons

```
INCLUDE "E:\Delphi2006-Working\FnxDesigner\Demo\MultiPanel\MultiPanel.clr"

OBJECT NewForm AS Form
 Left=271
 Top=106
 Caption="New Form"
 ClientHeight=393
 ClientWidth=384
 Color=clBtnFace
 OBJECT Panel1 AS Panel
  Left=16
```

```
Top=48
Width=359
Height=269
Caption="Panel1"
TabOrder=0
OBJECT Panel2 AS Panel
 Left=16
 Top=16
 Width=337
 Height=106
 Caption="Panel2"
 TabOrder=0
 OBJECT Button1 AS Button
  Left=16
  Top=16
  Width=75
  Height=25
  Caption="Show Panel3"
  TabOrder=0
  OnClick=Button1Click
 END OBJECT '{Button1}
 OBJECT Button2 AS Button
  Left=16
  Top=48
  Width=75
  Height=25
  Caption="Hide Panel3"
  TabOrder=1
  OnClick=Button2Click
 END OBJECT '{Button2}
 OBJECT CheckBox1 AS CheckBox
  Left=192
  Top=16
  Width=97
  Height=17
  Caption="CheckBox1"
  TabOrder=2
 END OBJECT '{CheckBox1}
END OBJECT '{Panel2}
OBJECT Panel3 AS Panel
 Left=16
 Top=144
 Width=333
 Height=98
 Caption="Panel3"
 TabOrder=1
 OBJECT RadioButton1 AS RadioButton
  Left=16
```

```
    Top=16
    Width=113
    Height=17
    Caption="RadioButton1"
    TabOrder=0
   END OBJECT '{RadioButton1}
   OBJECT RadioButton2 AS RadioButton
    Left=16
    Top=32
    Width=113
    Height=17
    Caption="RadioButton2"
    TabOrder=1
   END OBJECT '{RadioButton2}
   OBJECT RadioButton3 AS RadioButton
    Left=16
    Top=48
    Width=113
    Height=17
    Caption="RadioButton3"
    TabOrder=2
   END OBJECT '{RadioButton3}
   OBJECT RadioButton4 AS RadioButton
    Left=16
    Top=64
    Width=113
    Height=16
    Caption="RadioButton4"
    TabOrder=3
   END OBJECT '{RadioButton4}
  END OBJECT '{Panel3}
 END OBJECT '{Panel1}
 OBJECT StatusBar1 AS StatusBar
  Width=384
  Height=19
  PanelWidth(0)=50
  PanelWidth(1)=50
  PanelWidth(2)=50
  PanelWidth(3)=50
  SimplePanel=True
  OBJECT ProgressBar1 AS ProgressBar
   Left=153
   Top=3
   Width=209
   Height=15
   Position=50
  END OBJECT '{ProgressBar1}
 END OBJECT '{StatusBar1}
```

```
END OBJECT '{NewForm}

NewForm.ShowModal

SUB Button1Click()
 'your code here
 Panel3.visible=true
END SUB  '{Button1Click}

SUB Button2Click()
 'your code here
 Panel3.visible=False
END SUB  '{Button2Click}
```

MULTI-POPUP

USES: PopUpMenu, RichEdit
CONCEPT: Reparent the 2nd PopUp to the RichEdit by moving the code out of form Object...End Object [now has no parent] and setting the Parent to RichEdit.

- R-Click the form for its popup
- R-Click the RichEdit for its popup

```
OBJECT NewForm AS Form
 Left=271
 Top=106
 Caption="New Form"
 ClientHeight=393
 ClientWidth=384
 Color=&HFF00000F 'clBtnFace
 OBJECT RichEdit1 AS RichEdit
  Left=48
  Top=160
  Width=185
  Height=89
  Line(0)="RichEdit1"
  TabOrder=0
 END OBJECT '{RichEdit1}
 OBJECT PopUpMenu1 AS PopUpMenu
  OBJECT NewItem1 AS MenuItem
   Caption="NewItem1"
  END OBJECT '{NewItem1}
 END OBJECT '{PopUpMenu1}
END OBJECT '{NewForm}
OBJECT PopUpMenu2 AS PopUpMenu
 Parent=RichEdit1
```

```
 OBJECT MenuItem2 AS MenuItem
  Caption="cut"
 END OBJECT '{MenuItem2}
END OBJECT '{PopUpMenu2}

NewForm.ShowModal
```

STATUS BAR OBJECTS

USES: StatusBar, Button, Label, ProgressBar, CheckBox
CONCEPT:

- The buttons and label do nothing
- The CheckBox and ProgressBar are owned by the StatusBar
- Clicking the CheckBox uses OnClick to show a message

```
OBJECT NewForm AS Form
 Left=252
 Top=106
 Caption="New Form"
 ClientHeight=374
 ClientWidth=384
 Color=&HFF00000F
 OBJECT Label1 AS Label
  Left=112
  Top=208
  Width=31
  Height=13
  Caption="Label1"
 END OBJECT
 OBJECT Button1 AS Button
  Left=96
  Top=82
  Width=75
  Height=25
  Caption="Button1"
  TabOrder=0
 END OBJECT
 OBJECT Button2 AS Button
  Left=96
  Top=128
  Width=75
  Height=25
  Caption="Button2"
  TabOrder=1
 END OBJECT
```

```
 OBJECT StatusBar1 AS StatusBar
  Align=alBottom
  Height=19
  Width=384
  Color=&HFF00000F
  FontName="Tahoma"
  PanelWidth(0)=50
  PanelWidth(1)=50
  PanelWidth(2)=50
  PanelWidth(3)=50
   OBJECT cbShowProg AS CheckBox
    Left=163
    Top=-1
    Width=25
    Height=17
    Hint="Show Progress"
    Checked=True
    ShowHint=True
    Checked=True
    TabOrder=0
    OnClick=cbShowProgClick
   END OBJECT
   OBJECT ProgressBar1 AS ProgressBar
    Left=179
    Top=0
    Width=185
    Height=16
    Position=50
   END OBJECT
 END OBJECT
END OBJECT

NewForm.ShowModal

SUB cbShowProgClick()
 showmessage "cbShowProg click"
END SUB
```

TREE VIEW

USES: TreeView, Button, Panel
CONCEPT:

- Good coding practice to indent your code.
- The “nodes” of the tree are indented to each other just like they show.
- How the array of elements actually work

- (ndx, level) where ndx is the number from top to bottom of all elements and level is the amount of indentation [do not skip any]

```
OBJECT NewForm AS Form
 Left=271
 Top=106
 Caption="New Form"
 ClientHeight=393
 ClientWidth=384
 Color=&HFF00000F
 OBJECT TreeView1 AS TreeView
  Left=10
  Top=60
  Width=361
  Height=151
  TabOrder=0
  Caption(0)="0"
   Caption(1,0)="0-0"
    Caption(2,1)="0-1"
     Caption(3,2)="0-2"
      Caption(4,3)="0-3"
      Caption(5,3)="0-3a"
    Caption(6,1)="0-1a"
  Caption(7)="1"
  Sorted=False
  OnClick=TreeView1Click
 END OBJECT
 OBJECT Button1 AS Button
  Left=10
  Top=230
  Width=111
  Height=25
  Caption="Show Item #4 0-3"
  TabOrder=1
  OnClick=Button1Click
 END OBJECT
 OBJECT Panel1 AS Panel
  Left=140
  Top=230
  Width=231
  Height=25
  Caption="Panel1"
  TabOrder=2
 END OBJECT
END OBJECT

NewForm.ShowModal
```

```
SUB Button1Click()
  'your code here
  showmessage TreeView1.Caption ( 4 )
  TreeView1.Caption(5)="ItemIndex #5"
END SUB  '{Button1Click}

SUB TreeView1Click()
  'your code here
  Panel1.Caption = TreeView1.Caption ( TreeView1.SelectedIndex )

END SUB  '{TreeView1Click}
```