IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

In This Issue

IBasic Monthly -Volume 1, Issue 1
Editor – Tony Jones

Submission Guidelines
If you would like to have your work considered for publication in IBasic Monthly, please forward your submission in RTF, DOC, HTML or TXT format to ibasicmonthly@pyxia.com. IBasic Monthly is published on the 15th of every month. All submissions must be in by the 8th of the month in order to be considered for the current issue.

IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Editors REMarks

Hello and welcome to the premiere issue of IBasic Monthly, the one and only magazine dedicated to everything IBasic! Paul Turley has created a wonderful development tool that is both easy for beginners to pick up, yet contains the power and features needed for advanced programming and it is my hope that this magazine will, in time, evolve into a useful resource for both new and veteran IBasic programmers alike.

As IBasic itself has evolved so has a dedicated and helpful user community. It is this community that IBasic Monthly is written for and by. Every month you'll find articles, tips and code submitted and written by you, the IBasic user. And without you, this first issue of IBasic Monthly would not exist. I would like to thank everyone that submitted articles for this issue, along with the IBasic community for their support when I volunteered to take on this project. Without your encouragement this first issue would not have been possible. And I would also like to extend a very special thank you to Paul Turley for his support of IBasic Monthly.

Inside this first issue Kludge introduces us to pointers in IBasic, while Bizzy takes a look at the mysteries behind linked list. And don't miss the article by Ferdinand Schinkel, where he takes us on an in-depth tour of IBasic's MESSAGEBOX function. Ever wondered how to create those neat gradient filled windows like you find in many installer programs? Well, wonder no more! Bizzy has written an article, along with a sample program, on how to achieve this great effect. Paul Love tells us all about developing electronic forms for use in web pages with IBasic. And if you are new to IBasic and programming in general then be sure to read Rick Lett's article where he takes a look at IBasic from a "new" perspective. Don't know what a wiki is and how to use one? Jerry Muelver, the resident wiki expert, has the inside scoop for you. Internet apps, written in IBasic? Sure! Bizzy show's the possibilities with his article on developing an ftp client. Want to learn about the Windows API, but don't know where to start? How about starting with our regular, monthly column, Inside The Windows API? And for an inside and in-depth look at IBasic and the man behind it, you'll definitely want to read the interview with the man himself, Paul Turley!

So, get comfortable and enjoy reading the premiere issue of IBasic Monthly!

Sincerely,

Tony Jones

Tony Jones

Editor – IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

The Man Behind IBasic

An Interview With Paul Turley

By

Tony Jones

How did you get your start in computers and programming?
In 6th grade my brother had a Heathkit H-8 computer with H-9 terminal. The only language it came with was Benton Harbor BASIC. Wrote quite a few games on it. Never stopped writing code from that point on.

What inspired you to develop IBasic?
At the time I was really into the Amiga line of computers. While there were a few C compilers for the Amiga, AmigaBASIC written by none other than Microsoft was simply horrid. It was so bad that it inspired many programmers to write their own languages. IBasic was among them, so was Blitz BASIC and True BASIC. My vision was different as I wanted a language that was both low level enough to do all the cool things, yet had a lot of high level code for the mundane tasks such as opening a window.

What can we expect to see in the next major release of IBasic?
That's a secret :) Actually the CLI version of the compiler will be done soon. Also adding native MOD support back into the language (Music MOD's).

How is it being a solo developer, do you feel overwhelmed at times?
Sure, all the time. The forums are extremely busy lately and I couldn't manage it without the great help of Vikki and Ian.

It's been said by a lot of people that IBasic is a steal at only $24.95, why such a low price for such a powerful tool?
You should be asking Microsoft why such a high price for VS.NET ($799.00) Prices for software development tools have skyrocketed while quality seems to go out the window. Everyone seems to be following the trend set by the 'big boys'. Personally I think the pricing is fair for a development tool.

How many hours do you spend a day working on IBasic, what's a typical day like for you?

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

I spend about 10-12 hours a day coding, answering questions, and managing the site. I don't have typical days but when I do I will be sure to let you know :)

After it's all said and done, is it worth it?
Most of the time. There are days when the whiners and complainers make me just want to sell the source code to somebody else.

IBasic has developed a wonderful community around it. What are your thoughts on this?
This behavior is by design. If you read back 10000 or so posts on the forum you will see that developing the community, freely sharing ideas, was all part of the IBasic equation.

How did you come up with the name IBasic?
The 'I' originally stood for Intuition. Part of the Amiga OS. Code name during development was IMM (Intuition Macro Machine).

What tools do you use to produce IBasic?
C++, assembler, blood, sweat and tears.

Would you consider IBasic a hobbyist tool or a professional tool?
Both. That's why it's so appealing.

Where would you like to see IBasic to go?
More schools would be nice. It's a great learning tool.

Do you use IBasic yourself? What kind of programs do you write with it?
Yes. I find it much easier to code up an IBasic app then using C anymore. I write everything from utilities to games. Parts of the IBasic IDE are actually written in IBasic too.

Are you ever surprised at how far people "push" IBasic or they types of apps they produce with it?
IBasic users have come up with some pretty unique ideas. Ideas that would never have seen the light of day if it wasn't for IBasic.

IBasic is the most stable compiler I've ever used. Would it be fair to say that you're a "perfectionist" when it comes to IBasic?
Actually I am a perfectionist when coding. Not only IBasic but programming in any language. Back in the early days of Windows (3.0 and 3.1) you had to be a perfectionist if you wanted your program to run more than once without a reboot.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

During the course of developing IBasic, you had to make certain decisions regarding features and overall design of the language. How did you arrive at these decisions? Did you ask yourself what you wanted in a language?
While developing IBasic originally I would look at all of the languages available at the time and use the features I liked the most, when writing my own IBasic programs I would say "gee it would be nice if it did..." and I would incorporate it into the root language. Many of the features in the language today are the result of feedback from users. As I have always said if you want it, and its a good idea, then it will end up in the language. IBasic is now a community project. Every time you post a question on the forums your helping design the future of the language.

What prompted you to develop IBasic around the BASIC language and syntax as opposed to say Pascal or some other language?
Its the easiest for beginners to learn. Has less punctuation to get in the way than other languages and allows freeform programming. I like to just sit down and start coding without having to worry about the implementation of a language getting in the way. PASCAL's colon is annoying A:=1 for example is one too many characters to type.

When you're not programming and working on IBasic, what do you do for fun and relaxation?
Play guitar, watch movies, and spend time with the family.

Finally, any other thoughts you would like to share with our readers?
Never stop learning, never stop asking questions. Reading is the best tool a programmer has; it's also the most underused.

I would like to take a moment and extend my gratitude to Paul Turley for taking the time to do this interview for IBasic Monthly and to thank him for all of the support he provides, for both the magazine and IBasic. – Tony Jones

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Understanding Pointers Part I

By

Kludge

Pointers are particularly scary for newbie IBasic programmers so let's try to explain them in more or less simple terms.

Variables

While an IBasic program runs the variables used in it are stored somewhere in computer memory. So when a program DEFs (or uses an automatic) variable...

```
DEF avar:INT
avar = 1
```

...IBasic creates an INT variable. An INT requires 4 bytes of storage space in computer memory and IBasic automatically finds a suitable unused memory slot to store it in. When the program changes the value of variable avar...

```
avar = 99
```

...IBasic already knows where avar is stored in memory and so can make the requested change in it's value.

Enter the Pointer

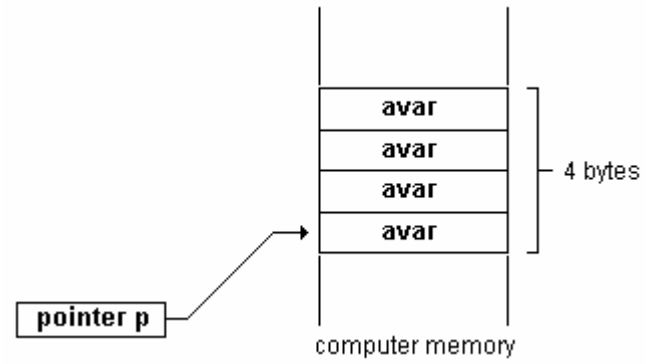
Suppose for some reason a program needs to know *where* in memory variable avar is stored? With that information (a *memory address*) a program can *directly modify* the value of avar, bypassing the more usual method of avar = somenumber. By defining a pointer and telling it to *point to* avar...

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1


```
DEF p: POINTER
DEF avar: INT
p = avar
```



...p can now directly access the *memory address* of avar (so *pointer* is another way of saying *memory address*). To use the pointer to directly access the value of avar at it's memory address...

```
#p = 100
'now avar = 100

DEF anothervar: INT
anothervar = #p
'now anothervar = 100
```

...the # symbol tells IBasic that the value held in *memory address p* should be set or retrieved, depending on which side of the equals sign #p appears. IBasic 'knows' that p was made to point to an INT and so 4 bytes in memory need to be updated or fetched.

Why would anyone want to do that?

The example above uses an INT to simplify the explanation and, more often than not, pointers are not required just to modify the value of Integer variables. Another use of pointers is as arguments to SUBs.

Consider the following code...

```
DEF i: INT
i = 9
GOSUB SquareInt
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
SUB SquareInt
      i = i * i
RETURN
```

...which would result in **i** being set to 9 * 9 = 81. This is all very well if the SUB is part of your IBasic program, but what if the SUB is to go into a component? Other component users may not want to call their INT **i** so a 'generic' version of SquareInt is required that can process any INT, not just one named **i**...

```
DECLARE SquareThis(p: POINTER)

DEF x, y: INT
x = 9
y = 4
SquareThis(x)
SquareThis(y)

'now x = 81, y = 16

SUB SquareThis(p: POINTER)
      #p = #p * #p
RETURN
```

What is happening here? For example, when SquareThis(x) is called IBasic automatically creates a *pointer to* x, because the argument p in SUB SquareThis was *declared as* a pointer. So inside SUB SquareThis we can use pointer p to *work directly* on what it points to - that is, the original Integer variable x at it's memory location. Note also that since SUB SquareThis can work with (pointers to) any type of variable that can be squared, it is possible to redeclare x and/or y as a FLOAT, CHAR, WORD etc. So SUB SquareThis is really a generic squaring mechanism that will work with any numeric variable type.

There are other ways of writing a SUB to square a number (and RETURN a result), but hopefully the example above has provided some insight into the workings of pointers in IBasic. In future articles we will look into other useful pointer techniques.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Graduation Tints

By

Bizzy

A simple method of putting a Graduation Tint on a Window using IBasic commands

To make the Graduation Tint - like you see in the Installation Package Window when installing using the Installshield Program - the IBasic RECT function and the RGB color function are all that are needed!

STEP 1

Define the Window

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
DEF Main:WINDOW
SETID "WM_EXITSIZEMOVE",0x232

WINDOW Main,150,150,400,350,@CAPTION|@SYSMENU|@SIZE,0,"Graduation
Tint",MainWindow
MENU Main,"T,&File,0,0","I,&Quit,0,1"
```

STEP 2

Start Window and create Tint at start up of program.

```
run = 1
' initialise the tint on start up
MakeBackground

waituntil run=0
closewindow Main
END
```

STEP 3

Code the Main Window Message Event Handler

WM_EXITSIZEMOVE
is used to re-paint the window when a window is re-sized by user by calling the SUB **MakeBackground**.

Window is centered in the @IDCREATE Message

```
SUB MainWindow
SELECT @CLASS
  CASE @IDCLOSEWINDOW
      run = 0
  CASE @IDMENUPICK
    SELECT@MENUNUM
      CASE 1
        run = 0
    ENDSELECT
  ' check for an > exit size move < from Window Message
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
   CASE @WM_EXITSIZEMOVE
      ' if finished changing the Window size then repaint Tint
      MakeBackground
   CASE @IDCREATE
      CENTERWINDOW Main
ENDSELECT
RETURN
```

STEP 4 (a)

Code the Sub MakeBackground

Part (a) of the Sub is where the Graduation Tint is made and applied to the Client Area of the Window.

Actually it is applied to a RECT that covers the Client Area of the Window.

Comments explaining each process are made in **GREEN** print throughout the code.

Basically the Screen Height is divided into 255 RECTangles (dif = height / 255), **which are drawn one above the other - each one having a different color - so as to give the Graduation Effect.**

```
SUB MakeBackground
   DEF left, top, width, height:INT
   DEF dif:FLOAT
   DEF newtop, newheight, dz, fx, gx, bt:INT

   ' get Client Area to place Tint into
   GETCLIENTSIZE(Main, left, top, width, height)
   ' obtain the amount of change - we have 255 shades in a color
   ' divide the HEIGHT by the color shades (255) to obtain the number of
   ' different RECT we will need to paint
   dif = height / 255
   ' work through all tint colors and RECTs - one by one in the for loop
   FOR dz = 0 to 255
      ' work TOP for each Tint and RECT
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
        newtop = dz * dif
        ' work HEIGHT for each Tint and RECT
        newheight = (dz + 1) * dif
        ' Work Color for RGB change for each Tint and RECT
        ' Below three different combination variables are made
        ' use as you wish to cause different Graduation combinations
        bt = 255 - dz : fx = 0 + dz : gx = 0 + dz
        ' Set the Rect to the desired Tint - ...
        RECT Main, left, newtop, width, newheight, RGB(0,bt,bt),RGB(0,bt,bt)
        ' Any of the following lines may be uncommented to be tested
        ' RECT Main, left, newtop, width, newheight, RGB(gx,fx,bt),RGB(gx,fx,bt)
        ' RECT Main, left, newtop, width, newheight, RGB(fx,0,bt),RGB(fx,0,bt)
        ' RECT Main, left, newtop, width, newheight, RGB(fx,bt,gx),RGB(fx,bt,gx)
        ' RECT Main, left, newtop, width, newheight, RGB(bt,0,0),RGB(bt,0,0)
        ' RECT Main, left, newtop, width, newheight, RGB(0,bt,0),RGB(0,bt,0)
        ' RECT Main, left, newtop, width, newheight, RGB(bt,bt,0),RGB(bt,bt,0)
        ' Make other combinations that you wish to try
    NEXT dz
    ' The Graduation TINT is now finished
. . . the rest of this SUB follows below . . .
```

STEP 4 (b)

Continuing with the Sub, Text is added on top of the Graduation Tinted RECT.

First two **RECT** are Drawn to give a **Drop Shadow Effect** to the Box where the type will be placed.

Next the **Text** is Printed too the above **RECT** as **Black** Print, which will be used to give **Shadow** to the Type.

Finally the **MOVE** statement offsets the type to **PRINT** the text in **RED** above the **Black** Text.

```
 Continued from the SUB above
    ' put rect to hold text on screen -  Black RECT for Shadow
    RECT Main, left+22, top+12, width-40, 33, RGB(0,0,0),RGB(0,0,0)
    ' Yellow Rect with Red Border - offset on top of Black RECT
    RECT Main, left+20, top+10, width-40, 33, RGB(255,0,0),RGB(255,255,0)
    ' Set the font to use for text
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
    SETFONT Main, "Arial", 18, 900, @SFITALIC
    ' Draw Mode set to Transparent so type uses existing background
    DRAWMODE Main, @TRANSPARENT
    ' set color for the typeface - black for drop shadow
    FRONTPEN Main, RGB(0,0,0)
    ' Position on screen to print text
    MOVE Main, 26,13
    ' Place the text on screen
    PRINT Main, "Graduation Tint  -  by Bizzy"
    ' set color for the typeface - red for type
    FRONTPEN Main, RGB(255,0,0)
    ' Position on screen to print text -
    ' offset from above text to produce shadow
    MOVE Main, 25,12
    ' Place the text on screen
    PRINT Main, "Graduation Tint  -  by Bizzy"
RETURN
```

Use the Mouse to Re-Size the Window and see the Graduation Tint re-painted to the screen.

Graduation Tints can also be added to any area on the window in the same way simply be specifying the RECT area of the Screen to work with. For Example the RECT where the Type was printed could have its own Graduation Tint instead of the Solid Yellow simply by working another Tint to the size of the RECT.

Listed below is the source code demonstrating graduation effects. Copy and paste it in the IBasic IDE and run it. I think you'll like what you see! - Editor

```
' Graduation Tint - by Bizzy
' The client area of the Screen is filled with a graduating tint.
' Different combos can be set in the RGB for different color tints
' Any defined area such as Client Height and Width can be used
' For Example the RECT where the text is PRINTed to the Screen could be
' filled with a Graduation, using the same technique
' Transitions and Effects - also from this method - experiment!
DEF Main:WINDOW
SETID "WM_EXITSIZEMOVE",0x232
WINDOW Main,150,150,400,350,@CAPTION|@SYSMENU|@SIZE,0,"Graduation
Tint",MainWindow
MENU Main,"T,&File,0,0","I,&Quit,0,1"
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
run = 1
' initialise the tint on start up
MakeBackground
waituntil run=0
closewindow Main
END
SUB MainWindow
select @CLASS
case @IDCLOSEWINDOW
run = 0
case @IDMENUPICK
select @MENUNUM
case 1
run = 0
endselect
' check for an > exit size move < from Window Message
case @WM_EXITSIZEMOVE
' if finished changing the Window size then repaint Tint
MakeBackground
case @IDCREATE
CENTERWINDOW Main
endselect
return
SUB MakeBackground
DEF left,top,width,height:INT
DEF dif:FLOAT
DEF newtop, newheight, dz, fx, gx, bt:INT
' get Client Area to place Tint into
GETCLIENTSIZE(Main, left, top, width, height)
' obtain the amount of change - we have 255 shades in a color
' divide the HEIGHT by the color shades (255) to obtain the number of
' different RECT we will need to paint
dif = height / 255
' work through all tint colors and RECTs - one by one in the for loop
for dz = 0 to 255
' work TOP for each Tint and RECT
newtop = dz * dif
' work HEIGHT for each Tint and RECT
newheight = (dz + 1) * dif
' Work Color for RGB change for each Tint and RECT
' Below three different combination variables are made
' use as you wish to cause different Graduation combinations
bt = 255 - dz
fx = 0 + dz
gx = 0 + dz
' Set the Rect to the desired Tint - ...
RECT Main, left, newtop, width, newheight, RGB(0,bt,bt),RGB(0,bt,bt)
'RECT Main, left, newtop, width, newheight, RGB(gx,fx,bt),RGB(gx,fx,bt)
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
'RECT Main, left, newtop, width, newheight, RGB(fx,0,bt),RGB(fx,0,bt)
'RECT Main, left, newtop, width, newheight, RGB(fx,bt,gx),RGB(fx,bt,gx)
'RECT Main, left, newtop, width, newheight, RGB(bt,0,0),RGB(bt,0,0)
'RECT Main, left, newtop, width, newheight, RGB(0,bt,0),RGB(0,bt,0)
'RECT Main, left, newtop, width, newheight, RGB(bt,bt,0),RGB(bt,bt,0)
next dz
' The Graduation TINT is now finished
'
' put rect to hold text on screen
' Black RECT for Shadow
RECT Main, left+22, top+12, width-40, 33, RGB(0,0,0),RGB(0,0,0)
' Yellow Rect with Red Border - offset on top of Black RECt
RECT Main, left+20, top+10, width-40, 33, RGB(255,0,0),RGB(255,255,0)
' Set the font to use for text
SETFONT Main, "Arial", 18, 900, @SFITALIC
' Draw Mode set to Transparent so type uses existing background
DRAWMODE Main, @TRANSPARENT
' set color for the typeface - black for drop shadow
FRONTPEN Main, RGB(0,0,0)
' Position on screen to print text
MOVE Main, 26,13
' Place the text on screen
PRINT Main, "Graduation Tint - by Bizzy"
' set color for the typeface - red for type
FRONTPEN Main, RGB(255,0,0)
' Position on screen to print text - offset from above text to produce shadow
MOVE Main, 25,12
' Place the text on screen
PRINT Main, "Graduation Tint - by Bizzy"
RETURN
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

My Adventures With IBasic

(Or how to frustrate yourself for fun!!)

By

Rick Lett
(Newbie)

OK, you've decided to try your hand at programming, and you've looked at several different basic languages and decided on IBasic. First of all congrats on your choice!!
You've picked a friendly community to learn from, an online resource and of course articles like this one to try and help you along.
Now if you have never programmed before (and it has been a long time for me until recently) then the users guide and forum stuff must seem overwhelming, however most of the old hands honestly do want you to have a good experience with IBasic. It's just that they have forgotten about being new!

Since I am still new, and still struggling some, I thought I'd share a few things that I've learned since starting IBasic.

Well with the intro out of the way, lets get to it!!

Now I know that you've been researching programming on line and you're pretty tired of "HELLO WORLD" programs. Their nice but you feel unsatisfied by it don't you?

Well I thought we could start with something that actually does something and talks about some important parts of a program. It's a console application and here it is:

```
DEF j, mov1:INT

OPENCONSOLE
j=1
mov1 = 1

DO
j = j +mov1
```

IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
LOCATE 10,j
color 14,0
print "IBASIC RULES!!"
for t = 1 to 30000:next t
locate 10,j
color 0,0
print "                "
if j > 65 then mov1 = -1

if j < 5 then mov1 = 1

UNTIL INKEY$<>""
CLOSECONSOLE
END
```

This program opens a console window prints "IBASIC RULES" and moves it right to left then right again.

Hey Now! I thought it was cool, and it has important parts of a program we can talk about.

Let's break it down starting with `DEF j,mov1:INT`

Ok **DEF** means define and what you are doing is telling IBasic that j and mov1 are **INT**egers or whole numbers (*By the way these terms are discussed in the IBasic users guide)*
j and mov1 are variables **DEF**ined as **INT**egers.

Now we need to **OPENCONSOLE** or open the device we are going to display our message in. Now this may seem premature but it is important to **CLOSECONSOLE**, even though it's at the end of our program, I think we should talk about it now.

The last three lines control closing out our program, in cahoots with the **DO** statement near the beginning. Our program will execute the statements between **DO** and **UNTIL** until the command **INKEY$** detects a *not equal to* (this means not equal to **<>**) space value (whats between " and ").
What this means is that IBasic will execute the lines between **DO** and **UNTIL** until you press any key, then the program will go to the next line **CLOSECONSOLE** and then **END.**
Whew that's confusing some even for me, but let's move on and see if we can make sense of it.

```
j=1
mov1 = 1
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Wow what is this? Well we need to initialize our variables, we're performing some math and we can't add 1 to zero, we can but it's not useful for this program cause if you look at the other line right after **DO** (*this is where it gets neat)*

```
j = j +mov1
```

Remember when we gave our variables a value? This is where our program adds these numbers together so it will do something. Each time IBasic loops back to this line(***UNTIL*** *you press any key huh,huh,get it)* it will add 1 to j which starts as 1 which equals 2 then 3 then 4, well you get it.... then this number is sent to the command **LOCATE,** actually

```
 LOCATE 10,j
```

10 places our **PRINT** statement in the approximately in the center of the console. j is what actually controls movement.

```
color 14,0
print "IBASIC RULES!!"
```

COLOR determines the message color (*it actually does fore and background, experiment!! You'll have great fun do* ***COLOR 12,10*** *or* ***COLOR 8,13*** *have fun with this stuff!!!)*
PRINT places our message on the screen provided it is in quotes(" IBASIC RULES")

```
for t = 1 to 30000:next t
```

OK, this one, wellllll, I kind of cheated, what we're doing here is waiting, there are better ways to do this, but you really need to read up more, lets just say we've stopped to count to 3000.(***FOR NEXT*** *loops are covered in the users manual, since I'm using it to count or time out I'd rather not go into it because the example is bad, sorry!)*

```
locate 10,j
color 0,0
print "                "
```

What we're doing here is after a short wait we are coming behind and setting our **COLOR** to black (*notice our* ***LOCATE*** *is the same, just after a slight delay provided by our* ***FOR NEXT*** *loop)* and **PRINT** statement is placing a space where our "IBASIC RULES" was, or is ,or whatever!?
Anyway you get the drift.

```
 if j > 65 then mov1 = -1
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
if j < 5 then mov1 = 1
```

AW-right this is where programming slaps the pavement!!!

We are making decisions, **IF** j is **Greater than** 65 **THEN** mov1 equals -1

Wow, we just changed mov1 from a positive number to a negative number
But hold on, look at this!

```
if j < 5 then mov1 = 1
```

IF j is **Less than** 5 **THEN** mov1 equals 1

Woo-ee now we just made it a positive number; this is what determines the boundaries of our message in our console application.
Copy and paste it, change it, play with, do what you can with it.
If you make it better post it on the forum, but remember to **HAVE FUN WITH IT.**

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Using The Messagebox

By

Ferdinand Schinkel

This is what you find in the User's Guide:

```
MESSAGEBOX window, text, caption {, flags}
```

Displays a message box in the window. Text and caption are strings that define what is shown in the box and caption. The optional flags parameter controls the creation of the message box.

```
MESSAGEBOX returns standard ID's of @IDOK and @IDCANCEL.
```

The messagebox in IBasic has many more features than you might surmise from the information in the User's Guide. First of all concerning the "flags" you can set at the end of the statement. If you *don't* use the flags you get the standard messagebox with an exclamation mark and an OK button. But if you *do* use the flags you will get a messagebox whose appearance is determined by the value you use for the flags. In fact, the flags consist of a number of bits that are set to either on or off (1 or 0). These bits are what influence the appearance of the messagebox. In the rest of this article I will use the normal decimal values that you can use for the "flags" parameter.

Buttons In The Box

There are 7 basic types of buttons combinations available for use in a messagebox.. If you look at the *basic output format*, without any pictorial sign (like a big exclamation, question mark, etc.) on the left side of the message box, these button types are:

Flags	Appearance
0 (zero)	An OK button
1	OK and Cancel buttons
2	Abort, Retry, Ignore buttons
3	Yes, No, Cancel buttons
4	Yes and No buttons
5	Retry and Cancel buttons
6	Cancel, Retry, and Continue buttons

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Information Types And Output Format

So, what about the big exclamation marks, question marks, etc.? You can see these as showing what kind of information you are giving with the messagebox. There are 4 output formats for *information types* that you can choose from and each one also has all the button types mentioned at the basic output format above. So, per output format there are the 7 different button-output types as seen above. The values for the "flags" parameter for these output formats per type of information are:

Flags:	Output Format
16 - 22	Big "X" in red ("don't do this!", "danger" or "are you really sure?")
32 - 38	Big question mark in light blue ("make a choice")
48 - 54	Big exclamation mark in yellow ("attention please!")
64 - 70	Big "i" in light blue ("information")

Return Values

And then, of course, it's also important to be able to know which button was pressed. To determine this, you call the routine as a function as demonstrated below:

```
def a:int
a=messagebox (0, "test in box", "test caption",1)
```

So here the arguments of the messagebox are between parentheses! Something to keep in mind! You don't have to use parentheses. The following is also legal, but you don't get a return value for the button that was pressed.

```
messagebox 0, "test in box", "test caption",1
```

Here's an overview of the button return values per basic output format type:

Flags	Button Return Values
0 (zero)	OK: 1
1	OK: 1, Cancel: 2
2	Abort: 3, Retry: 4, Ignore: 5
3	Yes: 6, No: 7, Cancel: 2
4	Yes: 6, No: 7
5	Retry: 4, Cancel: 2
6	Cancel: 2, Retry: 10,Continue: 11

Of course the same output values apply for the other output format types mentioned in *Information Types And Output Format* above.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

One Liner's

Did you know that with the messagebox statement you can write the smallest IBasic programs with output? For instance, the following is a complete program , and it shows something too.

```
messagebox 0,"Hi",""
```

So is this:

```
a=messagebox (0,"Hi","",1):if a=1 then messagebox 0,"you pressed OK!",""
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Linked Lists Made Easy

The Foundations - Part I

By

Bizzy

Program Languages have many different types of variables, such as, Integer, Float, Double, String, UDT (User Defined Type), and Arrays, by which a programmer can build an application. Some areas of the program may only need simple variables where other parts of a program may need complex data structures to manipulate and store data. The Linked List is much like Arrays except the Linked List keeps their own address reference to the next item in the list.

Linked Lists are not difficult to understand, however, when it comes time to program them the logics can be quite challenging. To make it easy to use Linked Lists it is necessary to use a set of rules to apply the code to each task in the Linked List. With a set of rules to follow using Linked Lists it becomes as easy as any other part of coding in the program.

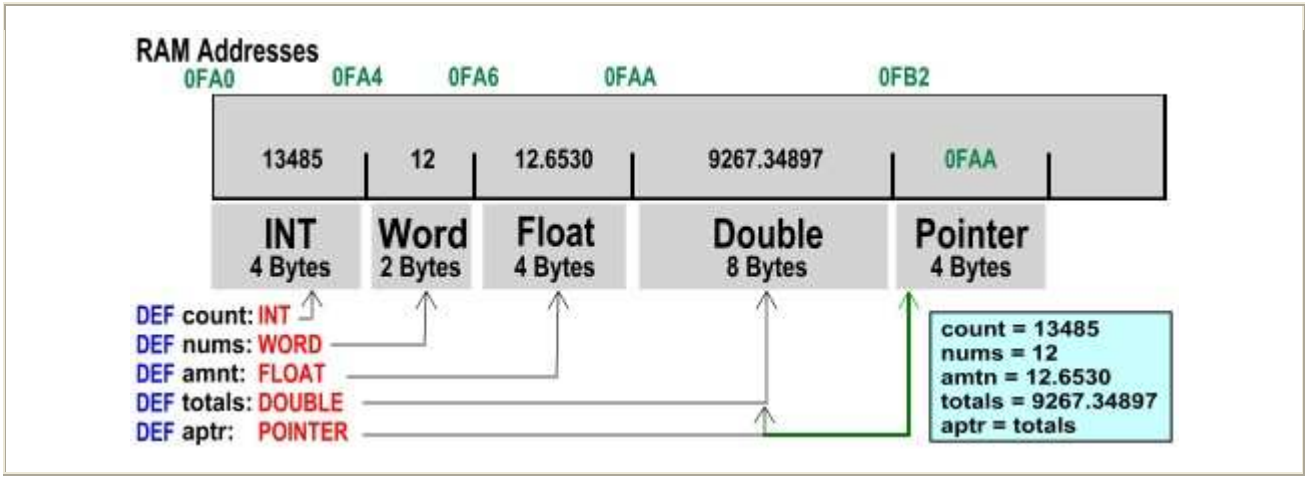
Before we start working on the Linked Lists we will look at the RAM and how variables are used so that we can understand the addressing used in **Linked Lists**.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

VARIABLES IN MEMORY (RAM - Random Access Memory)



The diagram above shows RAM and addresses across the top of the image. Under the rectangle representing RAM are different variables, INT, Word, Float, Double. Variables are shown at the bottom left of the image.

(1) When variables are defined the program allocates RAM for each variable defined.
(2) When values are assigned to the variables the program puts the values into the allocated RAM

DEF count:**INT** - Defines the variable name **count** as an Integer (**INT**). The Integer takes up four Bytes of RAM.
DEF nums:**WORD** - Defines the variable name **nums** as a **Word**. The Word variable uses two Bytes of RAM.
DEF amnt:**FLOAT** - **DEF totals**:**DOUBLE** and **DEF aptr**:**POINTER** are also defined.

In the blue colored panel to the right of the image the variables are assigned values. These values are stored in the RAM allocated for each variable. Study the diagram!

IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

POINTERS - Simply

The **Pointer** **aptr** is assigned the address of the **totals**: **DOUBLE** variable. That means **aptr** addresses **totals.** Both **totals** and **aptr** are now addressing the **same RAM (0FAA).**

You can assign values to **totals (totals = 2345.634)** and values can also be assigned by using **aptr**, **(#aptr = 2345.634).**

If you were to get the contents of **aptr** it would be the address of **totals 0FFA** but by using the **dereferencing sign #** before **aptr** you get the contents of **0FFA (totals),** which is **9267.34897** in the diagram.

Example of #aptr overwriting the value assigned to totals:-

totals = 768.23
#aptr = **452.12**

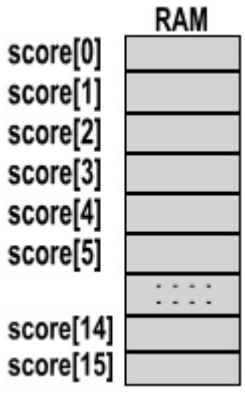
PRINT window, **totals** - would print **452.12.**

ARRAYS

Arrays are used when you wish to store a lot of data like song titles, movie titles, scores and so on. The array is ideal when you know how many items you wish to use. First the array has to be defined, then initialized and then values assigned to it.

DEF score[15]:**INT**
score[0] = 25
score[1] = 40
score[2] = 34

The array is stored in RAM as shown in diagram to the right.
Arrays are useful when the number of items is known but often where you do not know how many items you will need; it is better to use Linked Lists.



IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

USER DEFINED TYPE

DECLARING A USER DEFINED TYPE

```
TYPE list
   DEF FName:STRING
   DEF LName:STRING
   DEF Age:INT
   DEF AreaCode:INT
ENDTYPE

DEF MyList:list
```

User Defined Types as shown above can also be used in Arrays and in Linked Lists. UDTs are a great way to put related data together in one variable.
MyList uses **518 bytes** of RAM when it is defined. **List** can also be used in defining an **Array**.

To use the MyList variable: -

```
MyList.FName = "Fred"
MyList.LName = "Jones"
MyList.Age = 21
MyList.AreaCode = 41278
```

DEF MyList [100]:**list** would make an Array of 100 items using **51800 bytes** of RAM. To use the MyList[] Array: -

```
MyList[0].FName = "Fred"
MyList[0].LName = "Jones"
MyList[0].Age = 21
MyList[0].AreaCode = 41278
```

MyArray[1] will be 518 bytes after MyList[0] in your RAM and the program knows that and how to get to each set of UDT items in RAM.
Using too large an Array of a UDT can take up a lot of **Memory**; remember a user may wish to run other programs as well!
A more economical way of using an Array of a UDT would be to create the Array dynamically and keep an Array of Pointers to the dynamically declared UDTs.

A SINGLE LINKED LIST

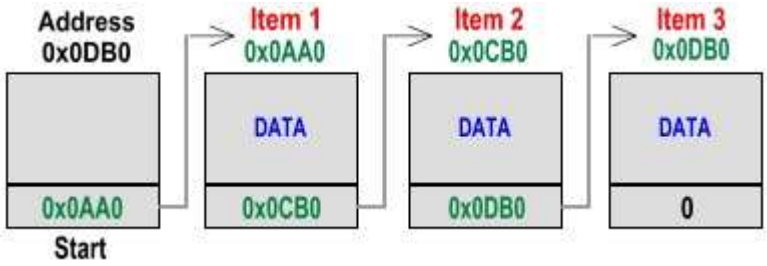
Well that has quickly covered the foundations of variables, as we need to understand them to use Linked Lists. First, the generally used variables (INT, Float, Double, String) and then the UDT and Arrays. Pointers were also used in

IBasic Monthly
IBasic Monthly
Dedicated to everything IBasic!
November 15th, 2002 Volume 1, Issue 1

conjunction with the variables to explain how they work generally. Now let's take a look at the Linked List.

A SINGLE LINKED LIST

(1) The top row in the Diagram shows the **Items** in the **Linked List** and their **Addresses** in RAM.
(2) DATA shows that the **Item** is saved into the above **address**.
(3) The **hexadecimal number** in the bottom row shows the **addresses** of the next **Item** in the **Linked List**.



(1) DEFINE A UDT

```
TYPE list
   DEF Name:STRING
   DEF LName:STRING
   DEF Age:INT
   DEF AreaCode:INT
   DEF Nxt:POINTER
ENDTYPE
```

I have used the same **UDT** as shown above in the arrays. However, there is one more item in the **UDT:**
DEF Nxt:POINTER.

The **Nxt Pointer** is where we will keep the **Address** of the **Next Item** in the **Linked List** (as shown in the Diagram above).

(2) DEFINE THE VARIABLES

```
DEF start: list
DEF node: POINTER
```

(3) ASSIGN START.NXT AS NULL TO MAKE AN EMPTY LIST

```
start.Nxt = 0
```

Starts with empty list pointing to NULL - No items in Linked List.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

node = start.Nxt **Assigns the node pointer an address from the address in start.Nxt. We learned earlier that assigning a pointer to a variable assigns both the pointer and the variable to address the SAME location in RAM. This is done so we can traverse through the node list from the start**

The **Start** variable is used to **address the first Item** in the **Linked List**. *See Diagram:* **start.Nxt** has **0x0AA0** as the address of **Item 1** in the **Linked List**. I use the **start** variable as a place to begin the **Linked List**. However, to begin with **start.Nxt** is assigned 0 (NULL) showing that there are no **Items** in the **Linked List**.

(4) CREATE A NEW ITEM - Rules to follow

(1) Find end of the list (#node.Nxt will be equal to 0)

```
WHILE (#node.Nxt <> 0) Looks for the end of the list
    node = #node.Nxt        Puts the address in #node.Nxt into the node
                            (increments node to the next Item)
```

(2) Allocate RAM for the Item and assign the RAM address to node.Nxt

```
#node.Nxt = new(list,1)
    #node.Nxt was = 0 so create a new Item and assign the address to #node.Nxt
    Remember in (1) above we traversed the list until #node.Nxt = 0
    Now #node.Nxt contains the address of the new Item just created
```

(3) Assign node the value of #node.Nxt

```
node = #node.Nxt
    To work with the node we just created we need to assign the address to the node
    pointer
```

(4) Assign the UDT values to node (#node.FName, #node.LName, #node.Age, #node.AreaCode)

```
#node.FName = "Willie"
#node.LName = "What"
#node.Age = 21
#node.AreaCode = 42135
    Puts the DATA into node
```

(5) Assign to #node.Nxt the value 0. It is the Last Item so it should have

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

#node.Nxt = 0

#node.Nxt = 0

Assign this node a value in its #node.Nxt equal to 0 (it is now last in list)

NOTE: When the first Item is created its address is assigned to #node.Nxt. Remember we set node to = start.Nxt at the beginning so that the first item will have the address in start.Nxt. This is the same as shown in the beginning of this article where aptr = totals was discussed. (node = start.Nxt, same thing). You may wonder why there is no # before start.Nxt in the above text? *start is defined as a variable not a pointer.*

Drawing diagrams makes that easy to understand and the first diagram in this article shows the aptr assigned the address of totals.

That is all for this article as it is enough to get to grips with. In the next article we will build in IBasic software a Single Linked List with diagrams and explanations on each step. Also we will use code to traverse through the list and print it out. Also to come is software in IBasic to SORT Linked Lists and save to a Binary File; Delete items from the List and save to a file; Sort Linked List on more than one variable; Extract a RANGE of items from a Linked List and Export to a Bfile and Double Linked Lists.

Happy programming until then - Bizzy

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Electronic Forms And IBasic

By

Paul Love (pel)

What are Electronic forms and what do they have to do with IBasic?

Electronic forms are a marriage of computerized form filling and database technology --- they allow the user to fill out a form on screen, print, e-mail or fax the completed document, and save the form for reference, reporting and analysis. Benefits of using electronic (rather than paper) forms include the ability to provide on-screen help, automate calculations, provide built-in edits, warning messages and self-checking routines, and capture form data for indexing, searching and filtering.

Currently there are a several rather high-priced packages available to handle electronic forms processing. IBasic however provides a quick, easy (and cheap) alternative through the use of its embedded browser control. Anyone with knowledge of HTML (and, in some cases, a scanned image of the form) can create his or her own electronic form. For example, let's say you want to set up a computerized version of the Federal 1040 form and you have a full-size image of the front page of the 1040. You can create a web page with the 1040 as a background, place text input boxes wherever there's a box or line on the form for you to enter information, include a "SUBMIT" button, display the page in an IBasic embedded browser window and collect the form data with the @GETPOSTDATA command. You can also design your own custom forms (inventory sheets, invoices, etc.) without using a background image.

In any case, once you've captured the data off the form, the data string can be parsed to get the information for the individual fields. For example, the string returned from an address book by the @GETPOSTDATA command might look like:

```
"FirstName=Michael&LastName=Johnson&Address=777+Seventh+Street"
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

You can use a simple parsing routine to separate and save the individual pieces of this data:

```
def data$,data1$:STRING
def pos,pos1,pos2:INT

data$="FirstName=Michael&LastName=Johnson&Address=777+Seventh+Street"
data1$ = data$

do
      pos = instr(data1$,"=")
      if pos <> 0
            field$ = mid$(data1$,1,pos-1)
            pos1 = instr(data1$,"&")
            if pos1 = 0 then pos1 = len(data1$)+ 1
            value$ = mid$(data1$,pos+1,pos1-pos-1)
            do
                  pos2 = instr(value$,"+")
                        if pos2 <> 0
                              value$ = mid$(value$,1,pos2-1) + " " +
                              mid$(value$,pos2+1)
                        endif
            until pos2 = 0
            data1$= mid$(data1$,pos1+1)
            messagebox 0,field$+" = "+value$,""
      endif
until pos = 0

messagebox 0,"Done",""
end
```

Of course it's easy to incorporate other features into this type of program by using IBasic to add menus, custom command buttons, etc. to do things like dumping posted data directly into a database file.

Why create a form this way instead of a using a normal IBasic program with a window or dialog screen and standard controls? The flexibility of a browser window for one thing -- forms can be full size since the window is fully scrollable horizontally and vertically. Plus you can include anything that would work in a normal web page, such as animated gifs, javascript coding or video clips (and e-forms like this can be easily ported over for internet hosting).

And maybe most important, as mentioned above, the end user with some knowledge of HMTL coding can easily create or modify his or her own forms.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

A Wiki Is An Anyone-Can-Edit Web Site.... What?!

By

Jerry Muelver

In this day and age, with “security” on the tip of everyone's tongue and “hacker attacks” in the background of every geek conversation, the notion of a web site that is completely open for anyone to edit, add, revise, and delete at will is a strangeness beyond reason. How could such a thing possibly work? And why would anyone try to make it work in the first place? The IBasic Wiki at http://hytext.com/cgi-bin/ibasic.pl will serve as the model for this question-and-answer article about the ways of the wikians.

Is a wiki really “anyone can edit”?
Yes. Every page on a wiki site has an “Edit” link. Click the link, and you are looking at an editor (or a form's text area editing box) showing the page's wiki-syntax source code. Change, delete, add what you want, click on “Save”, and you have edited the web page. It's as simple as that.

How do you prevent people from just trashing the page?
We don't. There is nothing to prevent someone from having their way with a page. Anyone can make any changes they want.

Isn’t that stupid, or at least dangerous?
Not at all. A wiki is a dynamic experiment in community trust. The wiki community takes care of itself, and the members take care of each other. If a page becomes... disenhanced, someone in the community will come along soon and fix it, for the good of the wiki and of the community.

How would anyone know how to fix a trashed page, or what it looked like before the trashing?
Wiki engines, the software that displays the pages and handles the editing, keep track of their pages. Each page has a “diff” or “Differences” link. Clicking the “diff” link will show the changes and deletions and additions performed by the last edit. It's easy to see what was done. The “diff” link is very useful for highlighting and finding the last edits on a lengthy page. Finding the revisions would be an arduous task without such a tool.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Wikis also store page versions, usually running back two weeks or so. Click on the "See other revisions" link on a page, and you can choose to view any of the stored versions of the page. Click on "Edit" to bring it up in the editor, then on "Save" to save the page over the trashed version, and everything's fixed. The process is common enough that it has it's own name among wikians -- "reverting".

What if it takes more than two weeks for someone to find out a page has been trashed?
Experienced wikians check the "Recent Changes" page of their wikis at least once a day, sometimes several times a day. That page shows what pages have been changed, when they were changed, who changed them (by name, if the editor set his or her "Preferences", or by ISP address otherwise), and what the change was about (the information from the "Summary" field during editing). With a global community, it is unlikely that a change would be unnoticed for more than a few hours. In fact, they are often noted within minutes.

You must spend a lot of time "reverting"!
Not really. It's only necessary maybe once or twice a month on large-community wikis, and hardly ever at all on tight communities like the IBasic Wiki. Since their adventurous exploits require no particular skill or secret knowledge, have no lasting effect, create no outrage, and hardly make a stir at all, those with trasher mentalities quickly lose interest in the sport, and move on to more exciting targets. It's a social solution to a social problem -- a technological solution would simply offer more challenge and excitement. As a result, wikis have a zen-like invulnerability.

The IBasic wiki pages look gorgeous. I don't know enough HTML to edit something like that!
You don't have to know any HTML at all to edit a wiki page. There are just a handful of tricks to know, and they are all laid out on the wiki's "How To Edit" or "Markup Syntax" pages. To create a heading, put some equal-signs and a space at the beginning of the line with the heading text -- one "=" for heading level 1, two for heading level two, etc. To make a list item, put an asterisk or star ("*") at the beginning of a line, or two stars if it's a sub list item. Use a pound sign or hash ("#") instead of a star, and you've got a numbered list. A blank line ends the list. Just type regular paragraphs for the text like you'd type them in a text editor. That's it! That's all you have to know. You're a wiki editor, already!

What about links and stuff like that?
The hallmark of all wikis is the ease of creating links and new pages -- it's their defining wiki characteristic. To make a link in the original wiki created by Ward Cunningham, all you had to do was use a smashed-together phrase with some internal capital letters in it -- ThisIsaLink, or HowToEdit -- and the phrase ThisIsaLink or HowToEdit would show on the page as an active

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Web link. That's called the "camel-back" convention, because of the ups and downs caused by the internal capitals.
The IBasic Wiki uses a bracketing convention -- put the intended link page title in double parentheses -- ((This is a link)) or ((How to edit)) -- which shows up as This is a link or How to edit, a little easier to read than the camel-back style, and less of an odd-ball appearance. Some wikis use [[double-brackets]] instead of parentheses, but the effect is the same.

How do I create and upload a page to link to in the first place?
In all wikis, creating a link also creates a page if one is needed to make the link work. If you make a link to a non-existing page, a page is created immediately. When you "Save" the page with your new link, it will show up with an underlined (linked) question mark after the page name. Click on the question mark, and you get the page editor running, all ready for your new page content. Say what you want to say, click on "Save", and the page is created.

What can you use wikis for?
Anything your imagination generates. I have wikis for HTML FAQs, for blues harmonica players, for Northwood's Wisconsin community news and gossip exchange, for ebook technology discussions, and even one for Esperanto language fans. There are dozens of wikis that I follow, and hundreds (maybe thousands!) that I have yet to see. The IBasic Wiki is for IBasic topics, questions and answers, project descriptions and collaborations, code snippets and examples, tips, bugs, and gotchas. Since wikis can be edited instantly, it only takes me 20 seconds to open the wiki, find the right page, edit a code enhancement (all right -- bug fix), and save the page to make the coding update. Better yet, anyone else who finds a bug or figures out an improvement can drop it right into the page themselves, or even create a page with their own version on it to give readers a choice!

Well, how can I get started? How will I know what to do?
Since you've read this far, you already know what to. Just go to the IBasic Wiki Home Page, and explore whatever interests you. "How to Join", "Guest Register", and "Markup Syntax" are good starting places. Experiment in the "SandBox", ask question in "Talk to Me". There's always room for another contributor on a wiki!

IBasic Monthly

Dedicated to everything IBasic!

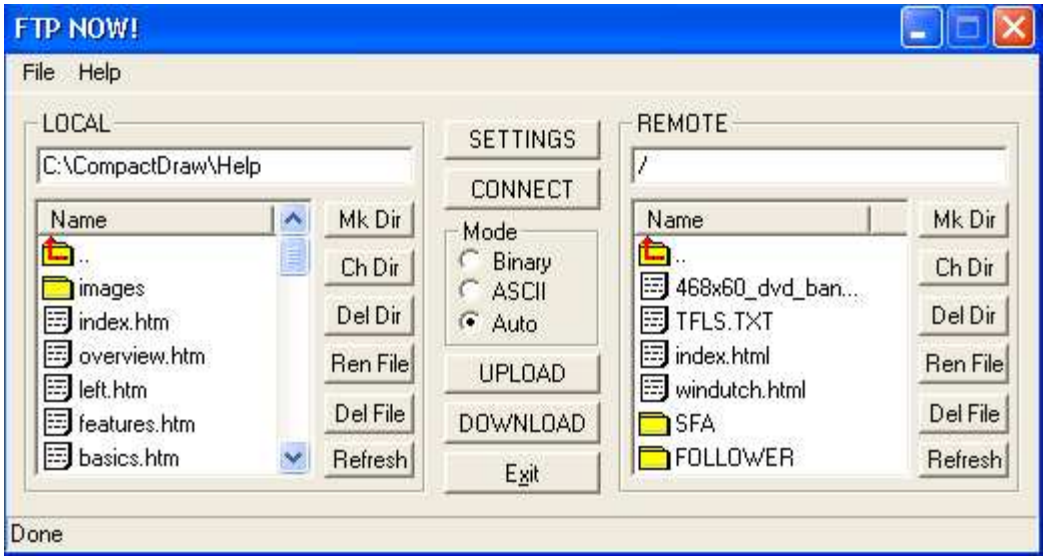
November 15th, 2002 Volume 1, Issue 1

FTP NOW!
Creating a Windows Internet Application with IBasic

Part I

By

Bizzy



To create this program references were obtained from Pyxia Examples, MS SDK, and AllAPI.net. In this article we will start by planning the Application and then building an outline of the FTP Now! program. Finally all the code will be added to make a fully functional Internet Application.

DESIGNING THE PROGRAM

IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

The first thing to do is to write down what the program is to do:

1. The FTP Now Program to be able to set up paramaters to connect to a specified Internet Web Address.
   This will require a seperate Dialog Window to obtain the input data and save it to a File for future use.
2. To be able to Connect to the Internet with the above parameters.
3. To be able to decide how to Transfer Files - Binary, ASCII, and Automatically
4. The program needs to be able to work with the Local Hard Drive.
   (a) Display a Folder Address
   (b) Display a File List
   (c) Create a new Folder
   (d) Change to a different Folder
   (e) Remove a Folder
   (f) Rename a File
   (g) Delete a File
5. The program also needs to be able to work with the Remote Web Address
   (a) Display a Folder Address
   (b) Display a File List
   (c) Create a new Folder
   (d) Change to a different Folder
   (e) Remove a Folder
   (f) Rename a File
   (g) Delete a File
6. Upload Files from the Local Computer to the Remote Internet Web Address
7. Download Files from the Remote Internet Web Address to the Local Computer
8. A Menu option to show About Box, and Help File Info.
9. A Status Bar to show information on current process.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

SET UP PARAMETERS TO CONNECT TO THE INTERNET WEB ADDRESS

SETUP FOR INTERNET PARAMETERS

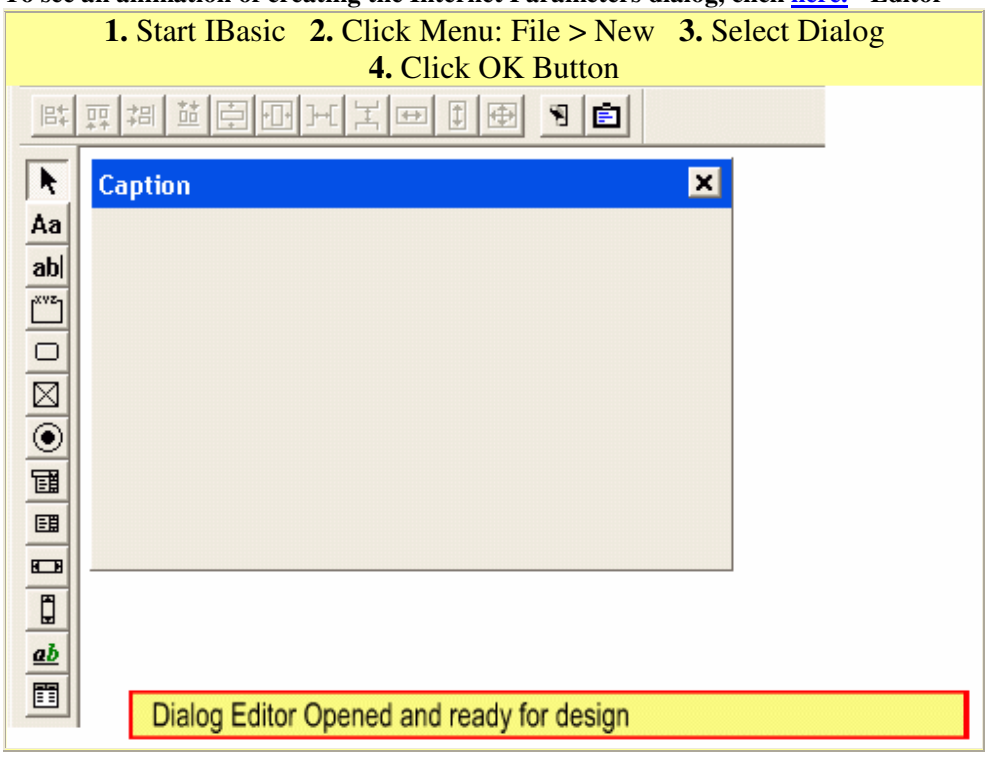
We will need Fields to receive the Input Info, so we set up Edit Boxes to obtain the Data.

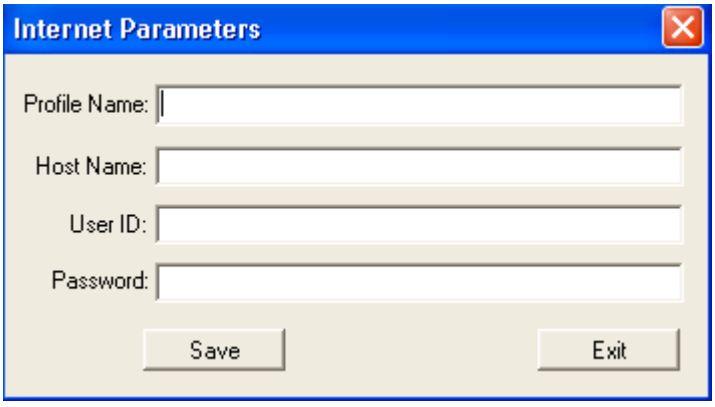
1.	**Profile Name:** Will be used to put an identifiable name for the Web Site
2.	**Host Name:** You can obtain this info from your ISP
3.	**User ID:** Is usually your ID for your Internet

CREATING THE DIALOG IN IBASIC

To see an animation of creating the Internet Parameters dialog, click here. - Editor

1. Start IBasic **2.** Click Menu: File > New **3.** Select Dialog
4. Click OK Button





IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

	Account.
4.	**Password:** Is usually the Password used for your Internet Account.

If you wish to connect with Web Sites other than your own Web Page then you need to get the above info from the relative ISP's.

A **User ID** for many sites is usually **anonymous**.

A **Password** can sometimes be your **Email Address**

The SAVE Button is used to save the parameters to a file.

The EXIT Button is used to close the Dialog Window.

```
DIALOG d2,0,0,352,171,0x80C80080,0,"Internet Parameters",DiagTwo
CONTROL d2,"T,Profile Name:,10,18,64,15,0x5000010B,1"
CONTROL d2,"T,Host Name:,15,50,58,15,0x5000010B,2"
CONTROL d2,"T,User ID:,32,79,42,14,0x5000010B,3"
CONTROL d2,"T,Password:,23,106,51,16,0x5000010B,4"
CONTROL d2,"E,,75,16,265,20,0x50810080,5"
CONTROL d2,"E,,75,46,265,20,0x50810080,6"
CONTROL d2,"E,,75,75,265,20,0x50810080,7"
CONTROL d2,"E,,75,104,265,21,0x508100A0,8"
CONTROL d2,"B,Save,70,138,70,20,0x50010000,12"
CONTROL d2,"B,Exit,266,138,70,20,0x50010000,13"
CONTROL d2,"B,B,11,134,32,20,0x40080001,14"
```

CREATE A NEW IBASIC PROGRAM

IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

To save the Internet Parameters Dialog just created we need to open a New Ibasic Program to put the Dialog Code into.

1. Click: Menu > File New > IBasic Program
2. Click: Edit > Paste - and the Dialog will be pasted into the Code Editor.
3. There are other items that need to be inserted in the Main Program that are associated with Internet Parameters Dialog.
(a) A **definition variable** for the Dialog, which we have named '**d2**'
(b) A **Sub Procedure** for Windows to pass Messages to the Dialog Window, which we have named '**DiagTwo**'

```
' FTP NOW - from BizzyPak
' FTP program to Upload and Download files via the Internet
' Uses Wininet.dll - Internet Explorer

DEF d2:DIALOG

'----------------------------------PARAMETERS DIALOG----------------------
---------
DIALOG d2,0,0,352,171,0x80C80080,0,"Internet Parameters", DiagTwo
CONTROL d2,"T,Profile Name:,10,18,64,15,0x5000010B,1"
CONTROL d2,"T,Host Name:,15,50,58,15,0x5000010B,2"
CONTROL d2,"T,User ID:,32,79,42,14,0x5000010B,3"
CONTROL d2,"T,Password:,23,106,51,16,0x5000010B,4"
CONTROL d2,"E,,75,16,265,20,0x50810080,5"
CONTROL d2,"E,,75,46,265,20,0x50810080,6"
CONTROL d2,"E,,75,75,265,20,0x50810080,7"
CONTROL d2,"E,,75,104,265,21,0x508100A0,8"
CONTROL d2,"B,Save,70,138,70,20,0x50010000,12"
CONTROL d2,"B,Exit,266,138,70,20,0x50010000,13"
CONTROL d2,"B,B,11,134,32,20,0x40080001,14"    :'(Used as Invisible
Default Button)

'-----------------------PARAMETERS DIALOG MESSAGES---------------
SUB DiagTwo
  SELECT @CLASS
    CASE @IDCONTROL
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1


```
        SELECT @CONTROLID
         CASE 12
              ' Save Settings
              WriteParamFile   :' (This is a SUB that will be written to Save
the Parameters)
         CASE 13
            CLOSEDIALOG d2,@IDOK
        ENDSELECT
     CASE @IDINITDIALOG
        CENTERWINDOW d2
        ' read file and put data into Edits for FTP Settings
        ObtainParamFile     :' (This is a SUB that will be written to Read
the Parameters)
  ENDSELECT
RETURN

'-----------------------------------------CODE-------------------------------
---------------

' The Two SUBs needed for the Internet Parameters
SUB ObtainParamFile
REETURN

SUB WriteParamFile
RETURN
```

SAVE THE IBASIC PROGRAM

NEXT STEP

CREATE THE MAIN FTP NOW PROGRAM USER INTERFACE

When we create this Dialog Window, the program will use the First Control we create in the Dialog, to set the Focus too. I want the Focus to be on the SETTINGS Button, so we will create the Controls at the Center of the Dialog first.

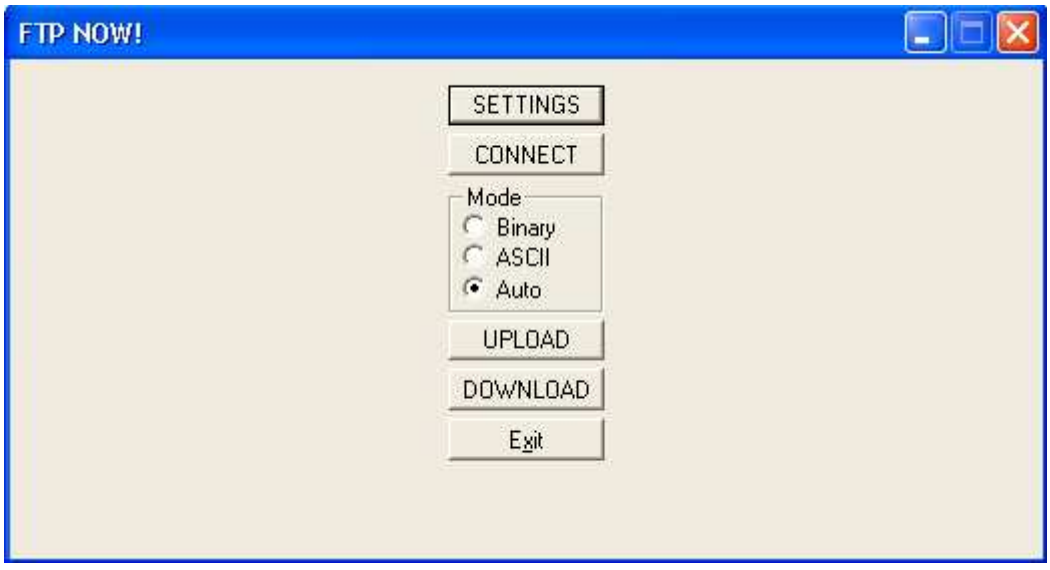
IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

1. Start IBasic
2. Click Menu: File > New
3. Select Dialog
4. Click OK Button

1.	**Right Click Dialog** and type in **Caption**. Drag Dialog to approximate size.
2.	**Select Button Control** from **Toolbar** and place on Dialog
3.	**Double Click on Button** and input the **Caption** and **Tab stop** settings.
4.	Do same thing for the **CONNECT** Button.
5.	**Select Group Control** and place in Dialog.
6.	**Select Radio Button** and place inside **Group**.
7.	**Double Click Radio Button** and type in **Caption** and put **Check** in the **Group Box**
8.	**Select Radio Button** from **Toolbar** and place on dialog for **ASCII** and **Auto**.



Use the Alignment Icons at top of Dialog Editor to line up the Controls the same as was done for the Internet Parameters Dialog (above)

IBasic Monthly

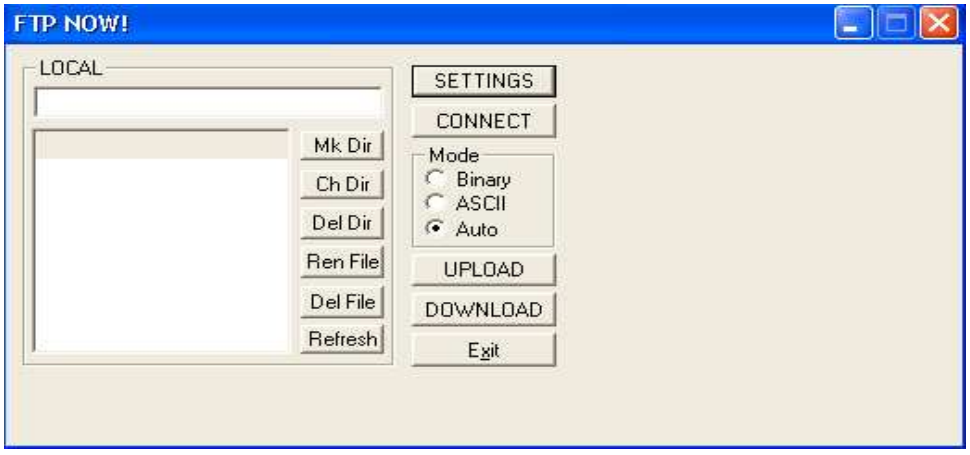
Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

NEXT STEP

The Next Step is to place the Controls on the left side of the Dialog for the LOCAL section of our FTP Now program.

1.	**Select the Group Box Control** from the **Toolbar** and place it on the Dialog Set Caption **LOCAL**
2.	**Select the Edit Control** from the **Toolbar** and place it on the Dialog. **Select Edit Control and Stretch the length.**
3.	**Double Click the Edit Control** and check the **Tab stop** and **AutoHScroll**
4.	**Select List View Control** from **Toolbar** and place on Dialog. **Stretch size.**
5.	**Select Button Control** from **Toolbar** and place on Dialog.
6.	**Double Click Button** and make settings, **Tab stop** and **CaptionMk Dir**
7.	There are **FIVE more Buttons** to place in the **LOCAL Group Box** so repeat **step 6**, and name each **Button** as shown in the View above.



Use the Alignment Icons to line up the Buttons. The Arrow Keys can also be used for small adjustments to the position of selected Controls.

NEXT STEP

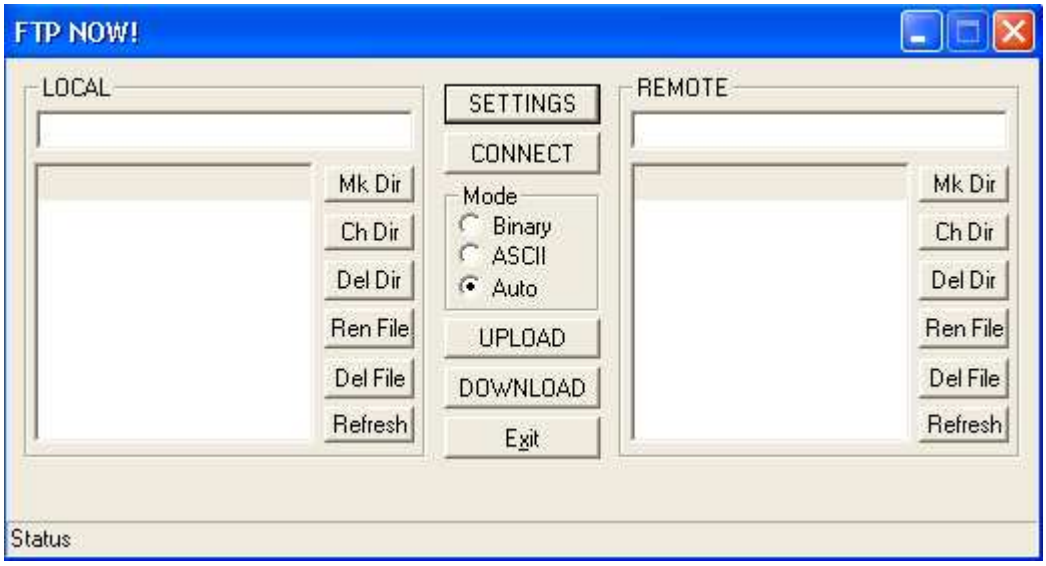
The Next Step is to place the Controls on the right side of the Dialog for the REMOTE section of our FTP Now program.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

1.	**Select the Group Box Control** from the **Toolbar** and place it on the Dialog. Set Caption **REMOTE**
2.	**Select the Edit Control** from the **Toolbar** and place it on the Dialog. **Select Edit Control and Stretch the length.**
3.	**Double Click the Edit Control** and check the **Tab stop** and **AutoHScroll**
4.	**Select List View Control** from **Toolbar** and place on Dialog. **Stretch size.**
5.	**Select Button Control** from **Toolbar** and place on Dialog.
6.	**Double Click Button** and make settings, **Tab stop** and **CaptionMk Dir**
7.	There are **FIVE more Buttons** to place in the **REMOTE Group Box** so repeat **step 6**, and name each **Button** as shown in the View above.



Use the Alignment Icons to line up the Buttons. The Arrow Keys can also be used for small adjustments to the position of selected Controls. Furthermore by Selecting the Group Box for LOCAL and holding down the Shift Key and clicking on REMOTE Group Box you can then use the Alignment Icons to make both these Group Boxes the same Size and Position. Same can be done with all Controls in LOCAL and REMOTE.

IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

ONLY THE MENU TO ADD

The Menu is added in the Initializing of the Dialog. That can be done after the Code Generated by the Dialog Editor is pasted into the Main Program. Also the Status Bar and List View need settings in the Initializing of the Dialog.

Below is the Code with both dialogs and definitions. Note comments on Code Lines. The List Views both have additional variables put in below (in Red) and the Status Bar is also initialized in the Initializing of the Main Dialog. See SUB DiagOne.

```
' FTP NOW - from BizzyPak
' FTP program to Upload and Download files via the Internet
' Uses Wininet.dll - Internet Explorer

' These two variables will be used in the List Views to enable them to run as
required
SETID "LVS_SHAREIMAGELISTS", 0x40
SETID "LVS_SHOWSELALWAYS", 0x8

DEF d1:DIALOG
DEF d2:DIALOG
' Set variable for Status Bar - 1 pane to be shown
DEF panes[1]:INT

'------------------------MAIN DIALOG-----------------------------------
DIALOG d1,0,0,517,250,0x80CA0080,0,"FTP NOW!",DiagOne
CONTROL d1,"B,SETTINGS,219,13,78,21,0x50010000,1"
CONTROL d1,"B,CONNECT,219,38,78,21,0x50010000,2"
CONTROL d1,"C,Mode,220,62,78,65,0x50000007,28"
CONTROL d1,"R,Binary,226,77,54,16,0x50030009,3"
CONTROL d1,"R,ASCII,226,92,54,16,0x50010009,4"
CONTROL d1,"R,Auto,226,107,54,17,0x50010009,5"
CONTROL d1,"B,UPLOAD,219,130,78,21,0x50010000,6"
CONTROL d1,"B,DOWNLOAD,219,155,78,21,0x50010000,7"
CONTROL d1,"B,E&xit,219,180,78,21,0x50010000,8"
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

```
CONTROL d1,"C,LOCAL,10,9,200,191,0x50000007,27"
CONTROL d1,"E,,16,27,188,20,0x50810080,9"
CONTROL
d1,"LV,,14,52,139,140,@TABSTOP|@LVSREPORT|@VSCROLL|@HSCROLL|
@LVS_SHOWSELALWAYS|@LVS_SHAREIMAGELISTS,10"
CONTROL d1,"B,Mk Dir,159,54,45,19,0x50010000,11"
CONTROL d1,"B,Ch Dir,159,78,45,19,0x50010000,12"
CONTROL d1,"B,Del Dir,159,102,45,19,0x50010000,13"
CONTROL d1,"B,Ren File,159,126,45,19,0x50010000,14"
CONTROL d1,"B,Del File,159,150,45,19,0x50010000,15"
CONTROL d1,"B,Refresh,159,174,45,19,0x50010000,16"
CONTROL d1,"C,REMOTE,307,9,200,191,0x50000007,18"
CONTROL d1,"E,,313,27,188,20,0x50810080,19"
CONTROL
d1,"LV,,312,52,140,140,@TABSTOP|@LVSREPORT|@VSCROLL|@HSCROLL|
@LVS_SHOWSELALWAYS|@LVS_SHAREIMAGELISTS,20"
CONTROL d1,"B,Mk Dir,456,54,45,19,0x50010000,21"
CONTROL d1,"B,Ch Dir,456,78,45,19,0x50010000,22"
CONTROL d1,"B,Del Dir,456,102,45,19,0x50010000,23"
CONTROL d1,"B,Ren File,456,126,45,19,0x50010000,24"
CONTROL d1,"B,Del File,456,150,45,19,0x50010000,25"
CONTROL d1,"B,Refresh,456,174,45,19,0x50010000,26"
CONTROL d1,"SW,Status,0,0,0,0,0,40"
CONTROL d1,"B,Button4,207,0,18,14,0x40080001,30"  :' Invisible Default Button

'----------------------------------PARAMETERS DIALOG-------------------------------
DIALOG d2,0,0,352,171,0x80C80080,0,"Internet Parameters", DiagTwo
CONTROL d2,"T,Profile Name:,10,18,64,15,0x5000010B,1"
CONTROL d2,"T,Host Name:,15,50,58,15,0x5000010B,2"
CONTROL d2,"T,User ID:,32,79,42,14,0x5000010B,3"
CONTROL d2,"T,Password:,23,106,51,16,0x5000010B,4"
CONTROL d2,"E,,75,16,265,20,0x50810080,5"
CONTROL d2,"E,,75,46,265,20,0x50810080,6"
CONTROL d2,"E,,75,75,265,20,0x50810080,7"
CONTROL d2,"E,,75,104,265,21,0x508100A0,8"
CONTROL d2,"B,Save,70,138,70,20,0x50010000,12"
CONTROL d2,"B,Exit,266,138,70,20,0x50010000,13"
CONTROL d2,"B,B,11,134,32,20,0x40080001,14"      :'(Used as Invisible Default
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1


```
Button)

'----------------------------START PROGRAM-----------------------------
SHOWDIALOG d1
run = 1
WAITUNTIL run=0
CLOSEDIALOG d1
END

'----------------------------MAIN DIALOG MESSAGES----------------------------
SUB DiagOne
    ' Variables needed in program
    DEF stattext:STRING             :' Variable for Status Bar Text
    DEF left,top,width,height:INT    :' Variables used to obtain Dialog Size for
Status Bar
    DEF mem:MEMORY             :' Used in Windows Messages (to come)

   SELECT @CLASS
     CASE@IDCLOSEWINDOW
        run = 0
     CASE @IDSIZE
       ' If Main Dialog is Resized - which it is when Menu added then adjust
        ' the Status Bar to fit the Dialog - (see Statusbar.iba in Examples)
        IFCONTROLEXISTS(d1,40)
          ' Tell the status window we are sizing
         CONTROLCMD d1,40,@SWRESIZE
         ' get the client size of the window and
          ' calculate the size of the pane.
         GETCLIENTSIZE d1,left,top,width,height
          panes = -1
         CONTROLCMD d1,40,@SWSETPANES,1,panes
          stattext = "Ready - "
         CONTROLCMD d1,40,@SWSETPANETEXT,0,stattext
      ENDIF
     CASE @IDINITDIALOG
       ' Add Menu to Dialog
       MENU d1, "T,File, 0, 0", "I,Internet Settings, 0, 1", "I,-,0, 0", "I,Quit, 0 ,2"
       INSERTMENU d1,1, "T,Help,0,0", "I,Help Content,0,3", "I,About,0,4"
```

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1


```
        ' Instruct Dialog to Center the Dialog Window
        CENTERWINDOW d1
        ' Set Group Radio Buttons to AUTO
        SETSTATE d1, 5, 1
        'get the windows client size and set up 1 pane for the status window
        GETCLIENTSIZE d1,left,top,width,height
        panes = -1    :' Sets Status Bar pane to width of Dialog Window
        CONTROLCMD d1,40,@SWSETPANES,1,panes
        'set the initial pane text
        stattext = "Ready - "
        CONTROLCMD d1,40,@SWSETPANETEXT,0,stattext
   ENDSELECT
RETURN

'------------------------PARAMETERS DIALOG MESSAGES-------------------------
SUB DiagTwo
  SELECT @CLASS
    CASE @IDCONTROL
      SELECT @CONTROLID
       CASE 12
            ' Save Settings
            WriteParamFile  :' (This is a SUB that will be written to Save the
Parameters)
       CASE 13
          CLOSEDIALOG d2,@IDOK
      ENDSELECT
    CASE @IDINITDIALOG
      CENTERWINDOW d2
      ' read file and put data into Edits for FTP Settings
      ObtainParamFile    :' (This is a SUB that will be written to Read the
Parameters)
  ENDSELECT
RETURN

'-----------------------------------------------CODE-----------------------------------------------

' The Two SUBs needed for the Internet Parameters
SUB ObtainParamFile
```

IBasic Monthly

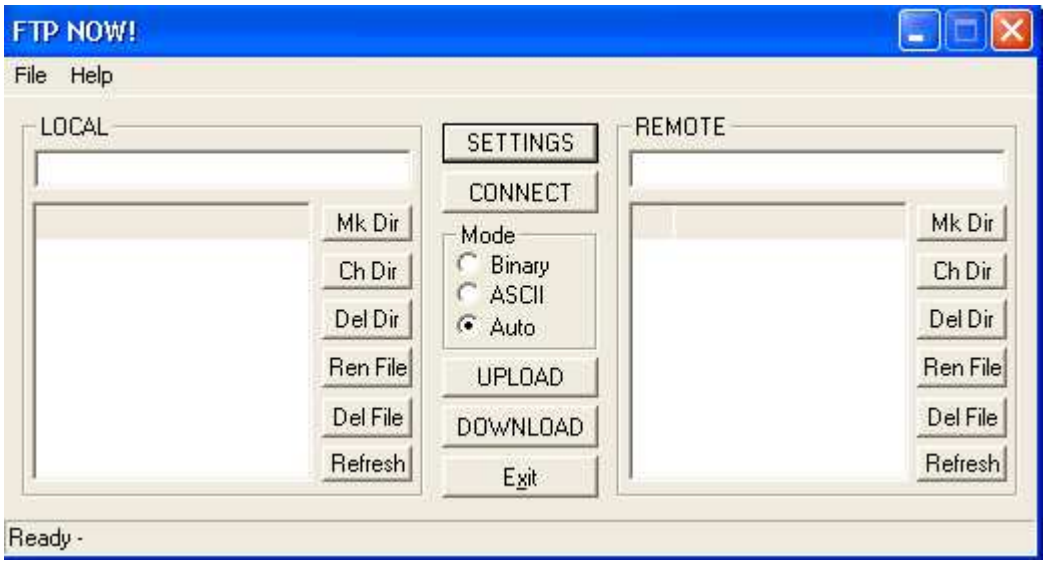
Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1


```
REETURN

SUB WriteParamFile
RETURN
```

If you copy the code above into IBasic Editor you will need to make sure the List View Control lines are on ONE Line!!



The Designing of the FTP NOW! program is complete - Only the Coding is left to do.
Part II will contain the rest of the Code and complete the Project of building the FTP NOW! Program.

For a sneak peak at FTP NOW, you can download the final, compiled version of it from the IBasic Monthly web site right here. ***- Editor***

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Inside The Windows API

By

Tony Jones

Welcome to Inside The Windows API! Every month this column will feature and take an in-depth look at calling various API functions from IBasic. For those of you who may not have had the opportunity to work with the Windows API, I hope this column will shed some light on the "mysteries" surrounding the API. And for the more experienced of you, you can help too! How? By taking part and sharing your knowledge of the API with an article on one of your favorite API functions. For details on submitting an article, please see the Submission Guidelines.

Pyxia Development has given us a powerful, yet easy to use BASIC compiler for creating Windows applications. And among its many features is the ability to call Dynamic Link Libraries, including those that are part of the Windows API.

A Dynamic Link Library or DLL for short is nothing more than a set of functions or subroutines that a program can use at run-time. The advantage of DLL's is that they are loaded on demand and used at run-time, thus saving memory. Another advantage is if you have multiple programs that use the same routines, rather than compiling the same code into the each program, the code can be compiled once as a DLL and called at run-time by the main program. If you are familiar with C or C++, a DLL is comparable in concept to the libraries used by those languages.

In this column we are only interested in DLL's as they relate to the Windows API or Application Program Interface. The API is the method by which you, as a programmer, can make request to the Windows operating system to carry out specific task, such as creating a window, dialog, button, etc. IBasic takes care of the details of calling the various API routines behind the scenes for you, but there may be times when you want or need to use an API routine for something that isn't available in IBasic.

A good place to start is with the IBasic User's Guide, under the section *Calling External DLL's*. The User's Guide gives an excellent overview of using DLL's with IBasic. For any function or subroutine that you want call that resides in a DLL you must first let IBasic know about it by declaring it with the *DECLARE* statement as illustrated below. We'll use the API routine *Sleep,* which suspends execution of your program for a specified time, in milliseconds.

```
DECLARE "kernel32", Sleep(dwMilliseconds:INT)
```

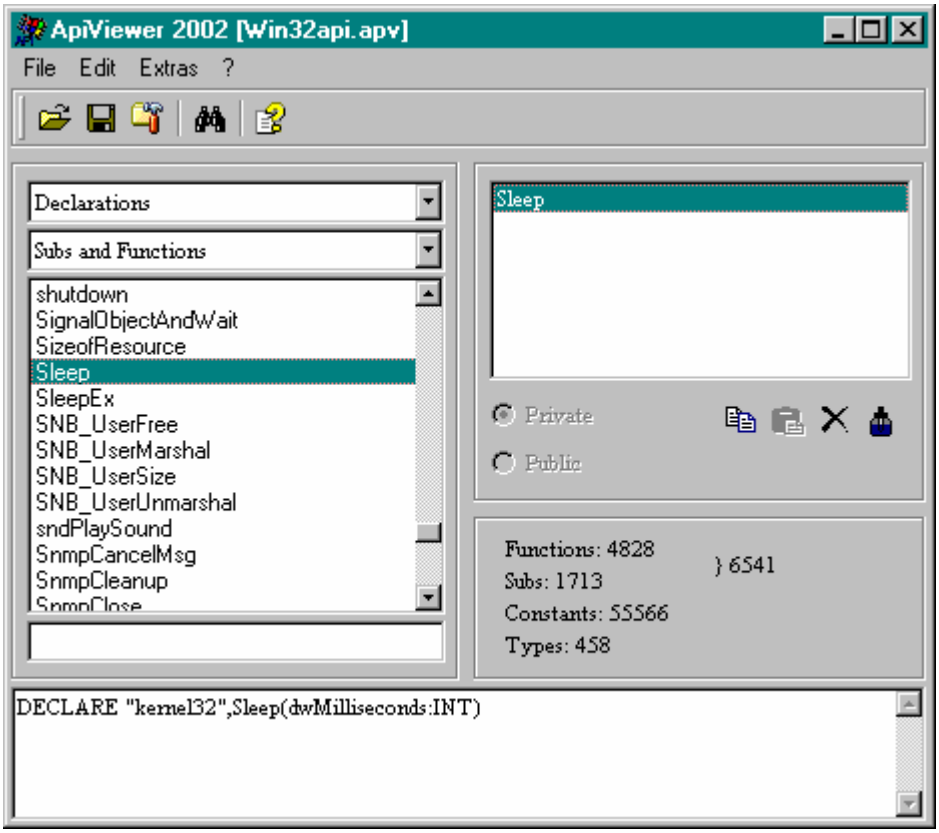
Let's take a closer look at the above line. The first part *DECLARE* tells IBasic that what follows is a declaration for a routine that resides in an external DLL. The next part "kernel32", is

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

telling IBasic which DLL the *Sleep* routine is in. In this case it's in the Kernel32.dll.Windows is made up of numerous DLL's, such as User32.dll, Kernel32.dll, etc. The next part is the actual declaration of the routine, which is for the most part similar to the way you would declare a routine that was written in IBasic. We see the name of the subroutine, and that it takes a variable of the integer type, specifying the number of milliseconds to sleep. You may be asking where you can get the declarations for calling API routines. You could go to the MSDN Developers Library which is a great resource describing all of the API routines available to you. However, the declarations there in C syntax, so you need to translate them to IBasic syntax. But there is an easier way. If you plan to use the API with IBasic then a must have tool is ApiViewer 2002. ApiViewer contains the most up to date API declarations, constants and types. Make sure you download the IBasic syntax plug-in also, which allows ApiViewer to generate the declarations in the correct syntax for IBasic. Using ApiViewer is a simple matter of choosing which routine you want to use and then simply copying and pasting the declaration into the IBasic IDE.



ApiViewer 2002 showing the Sleep declaration.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Now that we have the declaration for the *Sleep* routine declared, let's write a short example program illustrating how to use it.

```
' Sample IBasic program illustrating
' the use of the Sleep API call.

' The API declaration
DECLARE "kernel32",Sleep(dwMilliseconds:INT)

OPENCONSOLE

CLS

PRINT "Press a key to put me to sleep."

WHILE INKEY$ = ""
ENDWHILE

PRINT "Going to sleep for 3 seconds."

' The sleep routine use milliseconds
' 1000 milliseconds = 1 second.
Sleep(3000)


PRINT "Now I'm awake!"
PRINT "Press a key to exit."

WHILE INKEY$ = ""
ENDWHILE

CLOSECONSOLE
END
```

Copy and paste the above program into the IBasic editor and run it. I admit that it doesn't do much and is simplistic, but the important thing is it shows the use of the *Sleep* API routine. When you first run the program you are prompted to press a key to put the program to sleep. Once you press a key, the program tells you that it's going to sleep for 3 seconds. Next, you see our call to the API routine *Sleep*.

```
Sleep(3000)
```

Here is where the action, or rather the lack of action happens. We're telling the Windows operating system to stop or suspend our program for 3000 milliseconds or 3 seconds. Once 3 seconds have passed our little program resumes execution and waits for you to press a key to exit.

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Pretty simple, don't you think? Granted, not all API calls are this easy, but it gives you a good start and a basic foundation to build upon. That's all for this month. Next month we will continue our journey by using the API to determine what version of Windows a user is running.

IBasic Monthly

IBasic Monthly

Dedicated to everything IBasic!

November 15th, 2002 Volume 1, Issue 1

Coming Next Month!

Kludge continues bringing us up to speed on pointers with part two of his article, *Understanding Pointers*. Bizzy tells us more about linked lists as he continues with his series, *Linked Lists Made Easy*, plus he has all the inside details and how-to on his *FTP NOW!* program. We continue our adventures with Rick Lett as he explains SELECT/CASE to us in *My Adventures With IBasic* and Jerry Muelver takes us on a tour of *IBHashes – Faking Associative Arrays With ISTRING*. Plus, much, much more! So, be sure and download your copy of IBasic Monthly December 15th!

Tony Jones - Editor