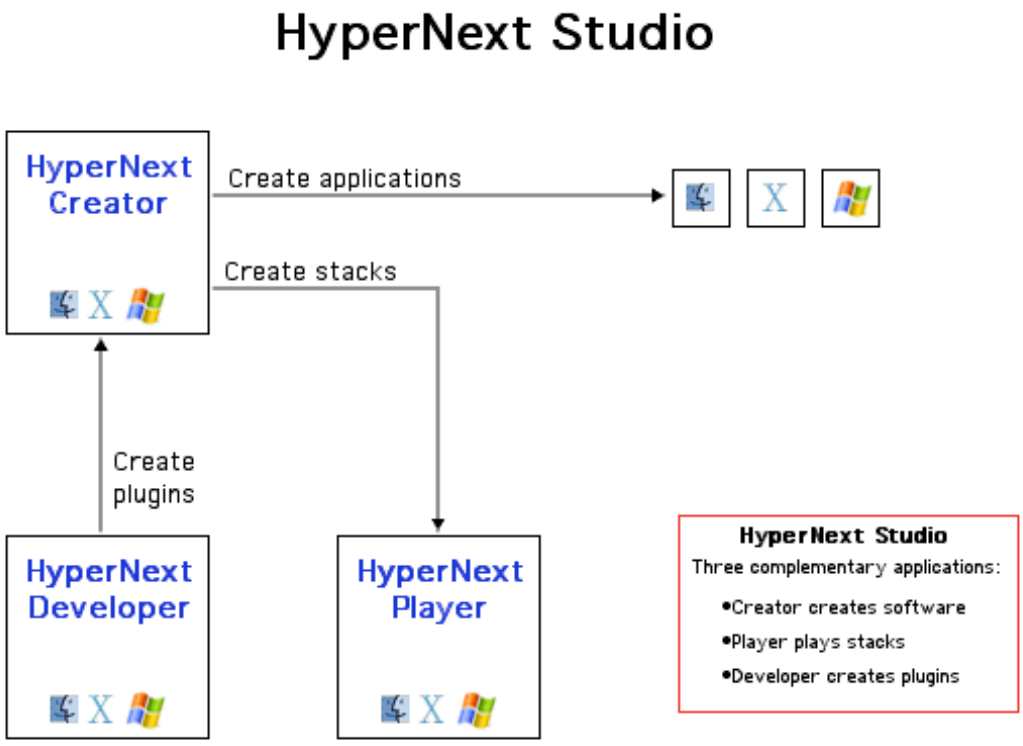
HyperNext Studio

Language Reference

v3.83

updated 31st December 2007



TigaByte Software 2007. http://www.tigabyte.com info@tigabyte.com

Contents

1 INTRODUCTION 5

Welcome 5
Resources 6

2 HYPERNEXT CREATOR 7

Using HyperNext Creator 7
Running Project 8
Compiling Project 8
Build Stack 8
Build Classic Application 8
Build OS X Application 9
Build Windows XP/Vista Application 9
Building for Macintosh on Windows XP/Vista 9
Script Editor 9
Menu Designer 10
About Box 10
Splash Screen 11
Sound Library 11
Image Library 11
Movie Library 11

3 HYPERNEXT DEVELOPER 13

Plugins 13
Using Developer 13
Compile Plugin 14
Run Plugin 14
Build Plugin 14
Plugin Names 14
Neural Networks 15

4 HYPERNEXT PLAYER 16

Using HyperNext Player 16
Distributing Stacks 16

5 HYPERNEXT LANGUAGE 17

Introduction 17
Features 17
Limitations 18
HyperNext Events 18
Control Statements 20
User Interaction 24
Clipboard 28
Fonts and System 29
Main Timer 30
Runtime Errors 31
Compile Errors 32
Localization 33

6 VARIABLES 35

Global & Local 35
Simple 36

COMPLEX 36
BOOLEAN 37
COLOURS 37

7 PROCEDURES & HANDLERS 38

MAINCODE 38
CONTROL HANDLERS 39
PARAMETERS 39

8 OPERATIONS 40

GENERAL 40
BINARY 43
MATHS 46
STRING 50
SORTING 55
DATE & TIME 58
ARRAYS 61

9 CARDS, WINDOWS & SCREENS 65

SCREEN & SCREENSHOTS 65
CARD COMMANDS 67

10 CONTROL TYPES 72

BUTTONS 72
FIELDS 79
Field Events 85
CANVASES 87
Canvas Events 92
TEXTS 95
CHECK BOXES 99
RADIO BUTTONS 103
POPUP MENUS 108
SCROLL BARS 112
MOVIES 116
LISTBOXES 121
Listbox Events 124
SLIDERS 134
SCROLLBARS 137
PROGRESS BARS 141
TIMERS 144

11 CANVAS GRAPHICS 146

EXAMPLES 146

12 PLAYING SOUNDS & MUSIC 149

COMMANDS & FUNCTIONS 149

13 NOTE PLAYER 151

COMMANDS & FUNCTIONS 151
INSTRUMENT LIST 155

14 PRINTING 158

COMMANDS & FUNCTIONS 159

15 FILES 164

GENERAL FILES 164
FOLDER/FILE DETAILS 170

Binary Files 171
Text Files 176
Reading example 177
Writing example 178
Graphics Files 179

16 NETWORKS 181

Easy Networks 181
Web Server 189

17 RBSCRIPT 190

Introduction 190
Example 1 190
Example 2 190
RBscript Graphics 194
Neural Network Interface 198

18 IMAGE BANKS 202

Commands & Functions 202

19 SPRITE ANIMATION 208

Commands & Functions 208

20 SPRITE OBJECTS 211

Commands & Functions 211

21 APPLESCRIPT 214

22 APPLE EVENTS 217

Example - Eudora 217
Example - String Tx 218

23 SERIAL PORTS 219

24 MENUS 225

25 REGISTRATION 229

26 ENCRYPTION 232

Blowfish example 233

27 RECEIVING EMAILS (POP3) 234

28 SENDING EMAILS (SMTP) 241

29 USB HID COMMS 245

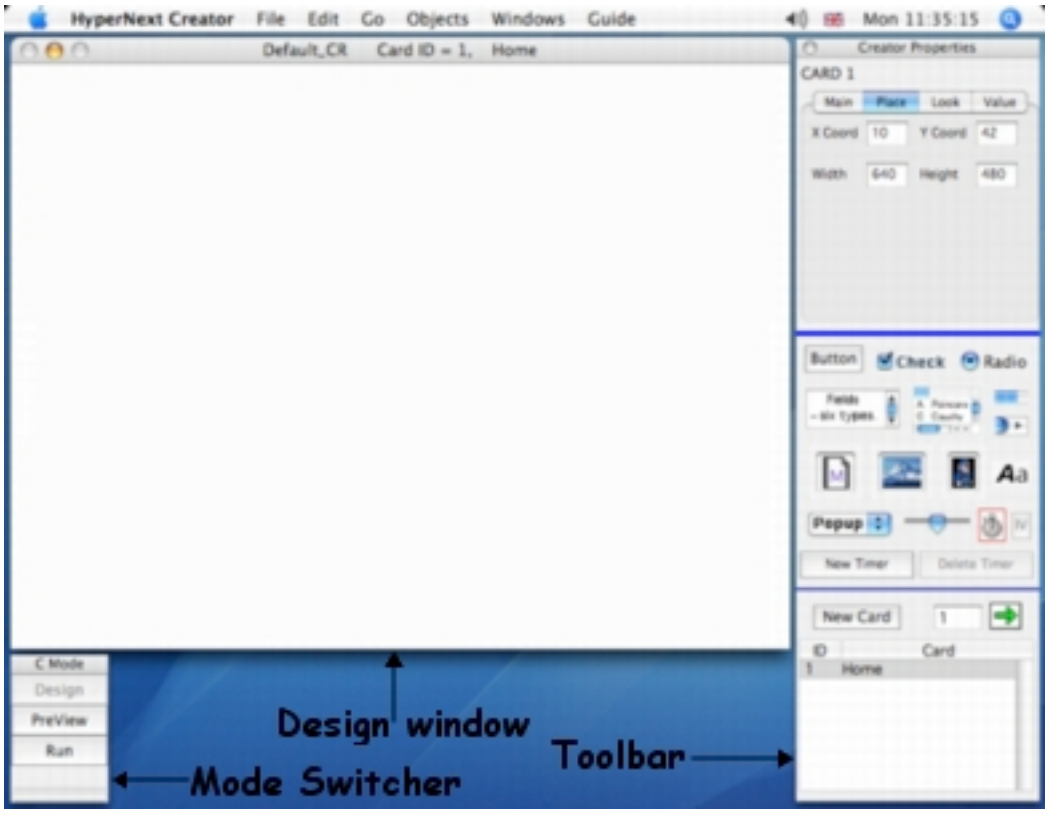
30 LIGHTSTONE BIO-FEEDBACK 251

1 Introduction

Welcome

HyperNext Studio comprises three complementary applications that help you create, develop and distribute your own software. Together they make up a powerful but easy to use cross-platform programming system that allows hobbyists, students, educators, in fact anyone, to quickly start building their own software.

Unlike most modern visual programming systems HyperNext has a very simple interface comprising just a main design window and one tool bar. It also has three easy modes of Design, Preview and Run so making it very easy for novice programmers to use.



Both HyperNext Creator and Developer have online help that makes exploring the HyperNext language so easy and contains examples that can be copied and pasted into scripts.

HyperNext Studio is not only great for beginners but is powerful enough to produce more complex software. With its flexible HyperNext scripting language, built-in networking and internal fast RBscripting it offers lots of potential.

Installation

Before installing HyperNext on your computer please ensure that you are using the correct version. Currently there are versions for Windows XP, Macintosh OS X and OS 9. Please note the OS X (Carbon) versions are not intended to work on Macintosh OS 9.

Windows XP

To install HyperNext Studio simply run the installer and it will place the HyperNext Studio folder into the Programs folder and the HyperNext Examples folder into your douments folder.
Apple Quicktime needs to be installed for full Hypernext functionality, for instance displaying images on buttons.

Macintosh OS X & OS 9

To install HyperNext Creator, Developer or Player simply mount the Disk Copy image on your desktop and then copy the HyperNext Studio folder to your preferred location on your hard disk. On OS X it is best to drag the examples folder into your documents folder.

Registering

HyperNext Creator and Developer will not offer full functionality until valid registration details are entered into their registration screens. Please note that the trial versions cannot be registered as they are simply restricted versions. Only the version of HyperNext Studio as supplied by Direct Download is fully functional.

Resources

If you need further information or help with using HyperNext Creator, Developer or Player then try our free HyperNext forum. Registration is and details can be found on our web site. There are also projects and documentation on our web site.

2 HyperNext Creator

HyperNext Creator is a self contained and easy to use software creation program. It has an easy to use interface and can build standalone applications as well as stacks for our freeware HyperNext Player. HyperNext Creator allows full projects to be developed and tested. It shares many general features of a typical developer application in that it can save, load, and create new projects.

The main aspects of the Creator GUI(Graphical User Interface) are the Design(Card) window, Tool Bar and Mode switcher. There is also an Editor for editing scripts (handlers and procedures) attached to cards and controls.

There are three main modes in HyperNext Creator

1. Design - for creating a card and its layout - adding controls and editing code.

2. Preview - displays a card as it would appear at runtime.

3. Run - for running the project within the Creator.

Using HyperNext Creator

Once HyperNext Creator has loaded it will display main two windows, the Design window and the Tool Bar which together allow cards to be created and controls placed upon them. A third global window called the Mode Switcher will also appear and shows three buttons that allow quick movement between the Design, Preview and Run modes. When switching from Design to Run mode the project is automatically saved, and when switching back the saved project is reloaded.

Some frequently used shortcuts employing the Command Key(CK) plus another key are:

CK+D - Preview current card (to revert back use CK+D)

CK+K - Compile and test project - it does not run the project.

CK+R - Run the project. To end the run simple use CK+Q

CK+Q - either Quits Creator OR ends a run and returns to Creator.

CK+E – opens the Script Editor

CK+S – Save

On Windows XP/Vista use the Control Key instead of the Command Key.

Running Project

The following sequence occurs when a project is run within the Creator

(1) The project is automatically saved.

(2) An attempt is made to compile the project.

(3) The Home card is then loaded and run.

Compiling Project

Compilation checks the whole project for errors and if any are found the Editor window is opened and the first error line is highlighted.

Compiling a program is a much faster way to find compile time errors than running it.

Build Stack

The current project is compiled and a stack capable of running on the HyperNext Player is built. The resulting stack will be placed in the current project folder and can then be run on Windows XP, Macintosh OS X and Classic versions HyperNext Player.

When distributing a stack you must remember to bundle the project Data folder along with it. The Data folder will contain any multimedia needed by the application. The project Resource folder is only needed by the Creator while working on projects.

Build Classic Application

Builds a standalone application for OS Classic, that is OS 9.x.

The current project is compiled and a Classic standalone application is built. The resulting application plus its accompanying data file are placed in the current project folder.

When distributing a standalone application you must remember to bundle the project Data folder along with it. The Data folder will contain any multimedia needed by the application. The project Resource folder is only needed by the Creator while working on projects.

Build OS X Application

The current project is compiled and an OS X (Carbon) standalone application is built. The resulting application plus its accompanying data file are placed in the current project folder.

When distributing a standalone application you must remember to bundle the project Data folder along with it. The Data folder will contain any multimedia needed by the application. The project Resource folder is only needed by the Creator while working on projects.

Build Windows XP/Vista Application

The current project is compiled and a Windows (.exe) standalone application is built. The resulting application plus its accompanying data file are placed in the current project folder.

When distributing a standalone application you must remember to bundle the project Data folder along with it. The Data folder will contain any multimedia needed by the application. The project Resource folder is only needed by the Creator while working on projects.

Building for Macintosh on Windows XP/Vista

In some respects Macintosh and Windows operating systems have quite different file systems and therefore the built standalone for Macintosh is actually contained within a binary file in order to ensure that its components are not separated before reaching the target Macintosh computer. Once on the target Macintosh computer a utility such as Stuffit Lite by Aladdin Systems can be used to extract the standalone.

Script Editor

HyperNext Creator and Developer each have an editor for creating and editing scripts.

The Editor is used for editing the MainCode, AppleEvents, Menus and other handlers such as those associated with cards and controls. It can be opened via the Edit menu and for cards/controls via the Script button located in the Properties section of the Toolbar. The editor is also automatically invoked when an error is found during compilation.

The Editor can be opened quickly via the keyboard with the CK+E keys as mentioned above.

At any one time the Editor can be in one of four editing modes:

1 - Cards/Controls
2 - MainCode
3 - Specials such as Animation
4 - User-defined Menu scripts.

Except for the MainCode mode, each mode can have several scripts, each associated with a card, control, event handler or menu action. Each script has its own action handler but can also have its own local procedures. The editor has four buttons for working with each script and they have the following functions:-

NEW - create a new handler.

EDIT - edit the selected handler's name and parameters.

DEL - delete the currently selected handler.

CLOSE - save any changes and close the editor.

Note, when the editor is open and the program is compiled or run then all scripts are saved to memory.
It is also possible to copy scripts from the online-help into the script editor.Both

Menu Designer

The Menu Designer allows user defined menu to be built and handlers defined for each menu item. In addition command key shortcuts can be defined for each menu item.

The HyperNext language has a set of commands for manipulating menu items and calling their associated handlers as described later in the Menu section.

WARNING - if you remove the Quit command from the Quit menu item then users will not be able to quit your stack or built application.

About Box

This allows either text or an image to be displayed in an About Box which the user can select via the About Menu of your stack or application.

If no image is present then only the specified text will be displayed.

An image can only be displayed if it has previously have been added to the image library of the project. Checking the scale image check box will ensure that the image's proportions will be retained otherwise the image will completely fill the About Box.

The current About Box has a fixed size of 280x210 pixels.

Splash Screen

This allows either text or an image to be displayed in the Splash Screen which is the first window visible when your stack or application starts up.

If no image is present then only the specified text will be displayed.

An image can only be displayed if it has previously have been added to the image library of the project. Unlike the About Box, the Splash Screen will automatically change its size to match that of the image.

Sound Library

Sounds can be made available in the Properties Window by first dragging and dropping them onto the list box of the Sound Library window. These sounds can then be assigned to buttons

HyperNext currently supports the following sound formats and extensions:

aif, aiff, au, midi, mp3, snd, wav

If you have a sound that the resource manager will not accept then try changing its extension to one of the above and see if it will play in the sound preview.

Image Library

Images can be made available in the Properties Window by first dragging and dropping them onto the list box of the Image Library window. These images can then be assigned to canvases.

HyperNext currently supports the following image formats and their ewxtensions:

jpg, jpeg, pict, pct, gif, png, tiff, tif and bmp

If you have a image that the resource manager will not accept then try changing its extension to one of the above and see if it will show in the image preview.

Movie Library

Movies can be made available in the Properties Window by first dragging and dropping them onto the list box of the Movie Library window. Any movie in the movie resource manager can be assigned to a movie control.

Acceptable movies are:

Quicktime, mov, avi, mpg, mpeg and swf

Note, only flash movies, i.e swf, of version 6 and lower are fully playable.

If you have a movie that the resource manager will not accept then try changing its extension to one of the above and see if it will play in the movie preview.

3 HyperNext Developer

HyperNext Developer builds libraries/plugins for HyperNext Creator. Plugins can provide extra functionality ranging from maths functions to complex algorithms such as neural networks. Plugins are automatically encrypted so making it safer to deploy and sell them to other HyperNext Studio users.

Plugins, once placed in the Plugins folder become available to any project being developed by the HyperNext Creator.

Plugins

Plugins are like libraries as they consist of a self contained set of procedures. There are no cards, controls or multimedia saved with a plugin although a plugin can access cards and other controls throughout any stack or application containing it. Plugins usually access controls on the focus card although because they have access to the full range of HyperNext commands they can interact with controls on any card.

Plugin development is generally undertaken by intermediate to advanced programmers. The HyperNext language gives access to Rbscript, a fast High Level language used in REALbasic by REAL Software. Unlike HyperNext, Rbscript is a strongly typed language that due to its strict type checking runs extremely quickly. For further details see the section on Rbscript towards the end of this guide.

To help protect HyperNext Creator users from errant or malicious plugins, a plugin cannot automatically initialise itself, any required Initialisation procedure must therefore be called by the Creator stack/application.

To protect plugin authors from having their plugins reverse engineered, HyperNext Developer produces a pCode memory dump and then encrypts critical sections in order to build the plugin. Even without encryption, pCode memory dumps are themselves cryptic and non trivial to reverse engineer.

Using Developer

When HyperNext Developer is loaded its GUI layout is very similar to HyperNext Creator's. However, certain aspects of the Creator are missing from the Developer because the Developer does not bundle cards and controls when it builds a plugin.

Many types of plugin can be tested by simply running them within the Developer but some interactive types can only be fully tested within a Creator project.

If you need to test such interactive plugins then have both the Developer and Creator open together. This arrangement works because the Creator only scans the Plugins folder when it has to compile, run or build a stack/application.
When your plugin is ready, simply build it, then switch over to the Creator and depending upon your needs either compile the Creator project or else run it. Next, switch back to the Developer and further develop your plug-in.

Compile Plugin

Compilation checks the whole plugin for errors and if any are found the Editor window is opened and the first error line highlighted.

As with the Creator, compiling a project is a much quicker way to find compile time errors than actually trying to run it.

Run Plugin

The following sequence occurs when the plug-in is run.

(1) The project is automatically saved.

(2) An attempt is made to compile the plugin and its Home card.

(3) If, successful, the Home card is loaded and then run.

(4) You can then interact with and test your plugin.

Build Plugin

The current project is compiled and then the built plugin is placed into the Plugin folder of the HyperNext Creator folder.

Plugin Names

If you have two different plugins each sharing the same name, then one of them must have its name changed as clearly they both cannot reside together in the same

Plugins folder. It is probably easiest to rename the one with the fewest calls and then drop it into the Plugins folder.

For instance if your built Plugin is called NewGraphics and has a procedure called DrawCircle, the usual method of referring to it from within a Creator project is

```
Call NewGraphics.DrawCircle
```

However, if you rename your plugin to NewGraphics1 you can simply refer to it by

```
Call NewGraphics1.DrawCircle
```

Neural Networks

As neural networks are inherently computationally intensive they are best implemented in Rbscript. However, in order to control and communicate with an Rbscript neural network an interface is required. The later section on Rbscript covers this interface.

Usually it will be best for plugin users if the plugin transparently manages the interface between itself and the Rbscript neural network. Of course this causes more work for the plugin developer but as described later HyperNext helps with the interface implementation.

4 HyperNext Player

HyperNext Player is a freeware and self contained application that runs stacks created by HyperNext Creator. Stacks can be as complex as any HyperNext Creator built application but are far smaller in size so making it easier to distribute them by email and the internet. Stacks are encrypted and cannot be modified so making them an alternative way to deploy your software.

Stacks are not modifiable in the sense that their structure can change. However, both the number of cards in a stack and the data can change as the user interacts with it. For instance, a medical analysis system stack might grow over time as the number of patient records builds up.

Note, unlike most applications, stacks automatically save their state whenever they quit. Most of the time this feature makes life easier for stack users. However, the automatic save can be overridden by the stack designer.

Using HyperNext Player

Either run the Player and use its File Open command to load the required stack or else simply double click on the stack itself.

It is important to note that the Player usually saves its state before quitting. Unless the stack designer chooses to disable this a stack user has no way to prevent this happening. The Player saves links to images, texts and all variables, even local variables. If need be, it is easy to create a stack employing its own pre-save handler and then place a call to this handler in the File Quit handler.

Distributing Stacks

Stacks are much smaller than standalone applications because they do not contain the overheads of a runtime engine, instead they rely on the HyperNext Player to provide it.
Furthermore, stacks are cross-platform and a stack created on OS X can be run by the OS 9 or Windows XP HyperNext Player and visa versa.

5 HyperNext Language

Introduction

The HyperNext programming language is similar to Hypertalk as used by Hypercard. Both languages have English-like statements and do not need their variables to be designated as having a specific type. Variables might be thought of as named locations within computer memory where data is stored. In HyperNext the vast majority of variables are stored as strings.

In most programming languages the type of a variable must be specified so that the compiler will know in advance which type of data is to be stored and which operations can be performed on that variable. There are many different types of data, but two of the most basic types are numeric and text. Typed variables make it much easier to develop and debug large programs as the compiler can do a great deal of checking before the program is even run. However, beginner programmers usually find it easier to get started if they can simply give the variables relevant names and concentrate on giving commands in order to make things happen rather than on being preoccupied with data types.

Features

These are the main features of the HyperNext programming language.

* English-like statements.

* Software generally has a card based organization.

* Variables are type-less and are all stored as strings(text).

* Variables are either Global or Local.

* Global procedures are declared in the MainCode section.

* Control handlers and their procedures are local and hence protected.

* Each control can have many local procedures.

* Specific commands for numeric and string processing.

* Variables can be single line, multi line, or array-like.

* Runtime error reporting can be dynamically switched on or off.

Limitations

The current version of the compiler has a number of limitations which will immediately become apparent to more advanced programmers. However, except for recursion, the limitations will be removed as the compiler is further developed.

* No recursion in HyperNext although Rbscript fully supports recursion.

* No user-defined functions although Rbscript supports user-defined functions

* Restrictions on where Functions can be placed
  - not allowed within Boolean terms as in IF THEN, WHILE
  - not allowed as parameters

Compile time error messages are sometimes terse.

HyperNext Events

In HyperNext certain events are handled by placing them into a queue and servicing the queue on a first in first out basis. Example of events are:-

Button Pressed, Canvas Mousedown Control Timer firing, Main Timer firing

HyperNext's event queue is managed by a one millisecond timer that tries to process one event per millisecond but on slower machines or when events occur which require heavy computation then this may not be possible. Apart from memory limitations there is no limit to how long the event queue may become but once several dozen events build up then the stack will become sluggish in responding to button presses and timers etc.

Note, AppleEvents have higher priority and so have their own event queue.

Notation Type

This refers to whether the keyword is C or F, a command or function respectively.

Name	Type	Description
EventsCountFN	F	This function returns the number of unprocessed events in the event queue. Put EventsCountFN into field 1
FlushQueue	C	This disposes of all events in the event queue. If this command is called from within a button script it will not be executed until its button event enters the event queue and is processed.
ControlTypeFN	F	Returns the number of the currently active control or interrupt. Possible values are:- 1 – Button 2 – Canvas 3 – Field 4 – Main Timer 5 – Menu Bar 6 – Card Timer 7 – Socket 8 – Checkbox 9 – Popup Menu 10 – Radio Button 11 – Sprite Surface 12 – Progress Bar 13 – Listbox 14 – Slider 15 - KeyDown
ControlIndexFN	F	Returns the index number of the currently active control.
KeyDownFN	F	Returns 1 if the key matching the passed key code was pressed but otherwise returns 0. Put KeyDownFN(code) into okay
KeyCharFN	F	Returns the character from the pressed key Put KeyCharFN into char

Control Statements

Name	Description
Comments	Lines preceded by the @ character are treated as comments by the compiler, as shown below @ *** Assign vars *** Local x,y,z
Quit	This command will cause the stack or program to quit although before quitting the state of the stack/program will be saved. Quit
QuitSave	This command will cause a stack/program to quit but will only save if the value is true. Use this command to disable automatic saving when quitting. QuitSave(value)
GotoStackPath	Loads a stack pointed to by the given absolute path name. The save parameter specifies whether the current stack should be saved before loading the target stack, if non zero then saving occurs. GotoStackPath(homestack,saveflag) If the path name is valid then the target stack will be loaded otherwise an error is raised. This command works only within the HyperNext Player and not when running a project within the Creator, Developer or a Standalone application. The path name can also be obtained using the FileGet or FileAsk commands.
GotoStackAsk	Loads a stack pointed to by the path name returned from the file dialog box. The save parameter specifies whether the current stack should be saved, before loading the target stack, if non zero then saving occurs. If the path name is valid then the target stack loaded otherwise an error is raised. This command works only in the Player and not when running a project within the Creator, Developer or a Standalone application. GotoStackAsk(saveflag)
Call	This command executes a user defined procedure or plugin procedure. @ with no parameters Call MyBeep

	@ with parameters Call MyBetterBeep(x,y,z) @ Plugin with parameters Call Plugin23.MyBetterBeep(x,y,z)

GotoLabel	This cause the program to jump to the labeled line. Gotos are local and cannot jump outside the current handler/procedure.
	
	If x=2 Then
	GotoLabel 1
	Else
	Beep
	End If
	
	Label 1

If Then
Else
EndIf
	This statement can evaluate multiple terms. Depending upon whether the result is true or false then one of two branches will be taken.
	
	If x=2 Then
	Message ok
	Else
	Message notok
	End If
	
	If Thens can be nested almost indefinitely.
	
	Note 1
	To tell the compiler that text is being compared the $ operator is used as follows.
	
	If res1$=res2 Then
	...
	EndIf
	
	Note 2
	Currently, if AND or OR are being used they must be the same, either all AND or all OR. AND and OR cannot be mixed.
	
	@ a valid if then
	IF (x=2) AND (y=2) AND (z=2) AND (w=2) Then
	
	EndIf
	
	@ an invalid if then
	IF (x=2) AND (y=2) OR (z=2) AND (w=2) Then
	
	EndIf

	Note 3 The runtime engine uses short circuit evaluation so that if the first term is false then the remaining terms are not evaluated.

For
EndFor
ExitFor
	The FOR loop executes a series of statements. The loop can be exited by using the EXITFOR statement.
	
	Whether the loop goes up or down is dependent upon the step variable. If the step variable is omitted then it is assumed to be plus 1.
	
	For x=1 to 10 step 1.2
	...
	put line x of field 1 into y
	If y=2 then
	ExitFor
	EndIf
	...
	EndFor
	
	Note, it is currently not possible to use procedure/function parameter as one of the limit variables.

While (term)
EndWhile
ExitWhile
	The WHILE loop executes a series of statements until the terms evaluate to false.
	The loop can be exited at any time by using the EXITWHILE statement.
	
	While (x<10)
	...
	put line x of field 1 into y
	If y=2 then
	ExitWhile
	EndIf
	...
	EndWhile

DoEvents	Allows background and other tasks to run. It also allows HyperNext or HyperNext built applications and stacks to update their displays. It can be called when a lot of processing is taking place so that the user interface does not freeze up. DoEvents
Wait	Forces the program to wait for the specified number of milliseconds before moving on to the next instruction. When the yield parameter is set to 1 then the program will give some time to the user interface and background applications. Wait(yield,period)

	@ wait for 5 seconds Wait(1,5000)

User Interaction

Name	Type	Description
Beep	C	Simply plays the currently designated OS system beep.
Message	C	Displays a dialogue box with the specified message. Message ('hello') Message (value)
Say	C	Speaks the given text in the default voice. If text is currently being spoken the message will be added onto the end of the queue. Say text If speech is not installed on the operating system then at runtime an error flag will be set allowing your program to handle the situation.
SayNow	C	Speaks the given text in the default voice. If text is currently being spoken it will be interrupted. SayNow text If speech is not installed on the operating system then at runtime an error flag will be set allowing your program to handle the situation.

Dialog	C	Presents a dialog box and returns a value indicating which of two buttons was pressed. The return value can be retrieved using the AnswerFN function
		
		There are three parameters. The first is the message to be shown in the dialog box. The next two are used as the text in the buttons of the dialog box.
		
		Dialog mess,button1,button2
		Put AnswerFN into ans
		If ans=1 Then
		...
		Else
		...
		EndIf

AnswerFN	F	Returns the answer returned after a DialogBox has been called. It indicates which of the two buttons were pressed. Put AnswerFN into choice

ChoiceFN	F	Allows the programmer to ask the user to make a choice between two values. The function passes a question and two options in the form of a dialog box to the user. The return value is the option choosen by the user.
		
		Put ChoiceFN(question,option1,option2) into choice
		If choice=option1 Then
		Dosomething1
		Else
		Dosomething2
		EndIf

ColorPicker	C	Calls a colour picker so that the user can select a colour. The following example calls the color picker and then sets the draw colour of Canvas 5 to that returned by colour picker. Local r,g,b ColorPicker(message,r,g,b) SetDrawColor(5,r,g,b)
SetMouseCursor	C	Changes the mouse cursor to either a watch, ibeam or arrow using the values 1, 2 or 3 respectively. The watch cursor is useful to let users of your program know when it is busy and not responding to user input. @ Set to watch SetMouseCursor 1 @ Set to ibeam SetMouseCursor 2 @ Set to arrow SetMouseCursor 3
MouseMove	C	Moves the mouse pointer to the specified x y coordinates on the screen. MouseMove(x,y)
MousePress	C	Sends a left mouse button pressed signal to the

		Operating System. MousePress
PressKeyReset	C	Resets the state of the simulated key press routine clearing any error flag. PressKeyReset
PressKeySetShift	C	Sets the state of the Shift key to either up or down using the values 0 or 1 respectively. After the simulated key press the Shift key will automatically return to the up position. PressKeySetShift
PressKeySetCommand	C	Sets the state of the Command key to either up or down using the values 0 or 1 respectively. After the simulated key press the Command key will automatically return to the up position. PressKeySetCommand
PressKeySetControl	C	Sets the state of the Control key to either up or down using the values 0 or 1 respectively. After the simulated key press the Control key will automatically return to the up position. PressKeySetControl
PressKeySetOption	C	Sets the state of the Option key to either up or down using the values 0 or 1 respectively. After the simulated key press the Option key will automatically return to the up position. PressKeySetOption(state)
PressKeyCharAuto	C	Simulates a key press with only the ascii character being specified. The routine automatically tries to find the matching key code. PressKeyCharAuto(char)
PressKeyCharCode	C	Simulates a key press with both the ascii character and key code being specified. PressKeyCharCode(char,code)
PressKeyAsciiAuto	C	Simulates a key press with only the ascii character number being specified. The routine automatically tries to find the matching key code.

		PressKeyAsciiAuto(cnumber)
PressKeyAsciiCode	C	Simulates a key press with both the ascii character number and key code being specified. PressKeyAsciiCode(cnumber,code)
KeyPressErrorFN	F	Returns the last reported error. A value of 0 indicates no error occured. Error value explanations are platform dependent and can be found on the Microsoft and Apple web sites. Calling this function resets the error value to zero. KeyPressErrorFN Put KeypressErrorFN into errnumn
KeyConvertCodeFN	F	Takes a key code and returns the matching Ascii character. KeyConvertCodeFN(integer) Put KeyConvertCodeFN(25) into char
KeyConvertAsciiFN	F	Takes an Ascii character and returns the matching key code. KeyConvertAsciiFN(char) Put KeyConvertAsciiFN(c1) into kcode
KeyboardNameFN	F	Returns the name of the keyboard layout. For instance English USA KeyboardNameFN Put KeyboardNameFN into ktype

Clipboard

These commands allow text to be copied to and from the clipboard using literals, variables and fields.

Name	Type	Description
ClipboardFN	F	Used for putting the text contents of the clipboard into a variable or field. Put ClipboardFN into field 1
SetClipboard	C	Used for putting a literal or the contents of a variable into the clipboard. SetClipboard cvar SetClipboard 25
ClipboardFN	F	Used for putting the contents of a field into the clipboard. @ field 1 SetClipboard 1 @ field 3 SetClipboard 3 @ field fid SetClipboard fid

Fonts and System

This section contains adhoc system related commands.

Name	Type	Description
FontCountFN	F	Returns the number of fonts currently installed in the system. Put FontCountFN into nfonts
FontListFN	F	Returns the names of the currently installed fonts in list form. Put FontListFN into fonts
PlatformFN	F	Returns the platform upon which the program is currently running. 1 - Mactel Rosetta 2 - Mactel native 3 - G3 PPC 4 - G4 PPC 5 - G5 PPC 6 - Windows 32 7 - Linux X86 Put PlatformFN into option

Main Timer

This is a global timer that operates independently of whichever card is in focus. It can be started/stopped, can execute its own script and change its own mode. It also access to both global handlers and variables. The HyperNext language currently has only one main timer and if more are needed they must be provided by setting up some local variables to handle the different scenarios. Each card though can have its own timer which work independently of other times.

The Main Timer's script can be edited via the Main Timer menu item of the Edit menu.

Note, the firing state of a timer cannot be saved and when a stack is first loaded all timers are switched off and their periods set to a default value of 1000 ms. Therefore if the Main Timer needs to be working immediately after loading it must be setup early on, perhaps either in the MainCode startup script or a card's open handler.

Timers are not yet supported in plugins. Although timers will work within the Developer they will not be saved or incorporated into the built plugin.

Name	Type	Description
MainTimerSet	C	This sets the main timer into one of three modes, off, single and multi. In single mode when its count down reaches zero the timer will fire just once and then enter off mode. In multi mode it will fire and then restart its countdown again. MainTimerSet mode,period mode : 0 = off 1 = single 2 = multi period: measured in milliseconds, i.e 1220 ms = 1.220 seconds
MainTimerOff	C	This switches the main timer into off mode.

Runtime Errors

There are two types of runtime error. The first type are caused by a mistake in the program or else by invalid input data so causing the program to fail. The second type occurs when an unknown command is encountered as can arise when a stack created on a more modern version of HyperNext Creator is run on an old version of HyperNext Player.

Type one runtime errors all have their keywords starting with ERROR while the second type all start with KEYWORD.

During runtime it is possible to either ignore a runtime error and try to continue or else display an error message and let the code in the handler try to cope with it. Error reporting is turned off by default because it can be quite annoying to a program user if frequent dialogue boxes are displayed after a single error. Generally, during program development it is best if the programmer turns on error reporting as the error dialogs are quite explicit. However, when the program is released to users the error reporting should be quietly controlled by the program itself and it should display user-friendly error messages when appropriate.

Name	Type	Description
ErrorReport	C	Sets error reporting either on or off. ErrorReport(value)
ErrorClear	C	Clears the error flag.
ErrorNumberFN	F	Returns the number of the last runtime error.
ErrorTypeFN	F	Returns the type of the last runtime error.
ErrorLineFN	F	Returns the line number of the last runtime error. This may be helpful in debugging the program.
ErrorSourceFN	F	Returns the source line causing the error.
KeywordErrorReport	C	Sets the state of keyword error reporting. By default this is set to silently ignore errors caused by unknown keywords. The programmer can modify the text of an error message dialogue box so making it more user-friendly. KeywordErrorReport(number) KeywordErrorReport(3) where number can be 0 - Silent, no action occurs.

		1 - Beep, beeps when an error occurs. 2 - Dialogue and Continue, displays an error message and continues. 3 - Dialogue and Quit, displays an error message and quits.
KeywordErrorMessage	C	Sets the error message seen by the user. The default message simply gives a program line number, the source line and the type of error. The user-defined message can be multi-line text. KeywordErrorMessage(text) KeywordErrorMessage(mess)
KeywordErrorRaise	C	This raises an error allowing the programmer to modify the programs reaction to possible errors and to produce a user-friendly dialogue message. KeywordErrorRaise
KeywordErrorFN	F	Returns the current error state. If the result is zero then no error occurred, if 1 then it was caused by an unknown command, else if 2 by an unknown function. Once the KeywordErrorFN function is called then the error state is reset back to zero. KeywordErrorFN Put KeywordErrorFN into etype
KeywordErrorSettingFN	F	Returns the state of the keyword error reporting. When 0 then error reporting is silent, 1 will beep on an error, 2 is in dialogue box continue mode, and 3 in dialogue box quit mode. KeywordErrorSettingFN Put KeywordErrorSettingFN into estate

Compile Errors

When the compiler encounters an error it will open the script editor and indicate the line and a message detailing the possible error.

Localization

The default language used in HyperNext is English but both stacks and applications can be localized to work in another language. This localisation must be set in the Creator project before the software is built. It is also possible to define data within a plugin for a specific localization but the localized text must be used in a project set up specifically for that language otherwise it will probably be displayed as garbage characters.

The following can be localized:-
Controls - buttons, fields, texts, canvases(indirectly via DrawText)
Menu titles and items.
Message and dialog boxes.
Text files can save/load in correct encoding.
String literals inside the script editor.
Variables store correct encoding.
About box.
Splash screen.

HyperNext Creator

To start a localized project the default font should be set in the Preferences. The default font is used in editing text and setting controls etc.

Example - Japanese

Assumes that you have just created a new project.

(1) From the Edit menu select the Preferences menu item.

(2) From the settings box select the Localization tab.

(3) From the font pop-up menu select Osaka.

(4) Close the settings box.

From now on this project will use Osaka as its default font for controls etc.

HyperNext Developer

To develop a localized plug-in open the Developer, create a new project and then follow these steps

Example - Japanese

(1) From the Edit menu select the Preferences menu item.

(2) From the settings box select the Localization tab.

(3) From the font pop-up menu select Osaka.

(4) Close the settings box.

From now on this project will use Osaka as its default font for controls etc.

6 Variables

Variables are used to store numbers, text, lists of names etc. In HyperNext, unlike most other programming languages, all variables are basically text, often referred to as strings, and generally the compiler makes no distinction between data types.

All variables must be declared, either within the local procedure/handler otherwise the compiler will flag an error. However, variables can be used before they are declared , something that can make a procedure algorithm easier to follow.

Global & Local

Name	Description
Global	This is used to declares global variables. Such variables are visible anywhere within the program. Global x,y Note, by default, a stack or standalone saves all global variables before quitting.
Local	This is used to declare local variables. Such variables are only visible within the current procedure and retain their value even after the procedure has exited. Local a,b,c Note, by default, a stack or standalone saves all local variables before quitting.

Variables must be either Global or Local. Global variables can be accessed from any handler/procedure within a stack whereas Local variables can only be accessed from within the handler/procedure in which they are declared. Plugins can also have global variables so that data can be passed to and from the plugin.

The following procedure shows how both global and local variables are declared.

```
Local x1,y1
```

```
Global name,job
Local z,w
```

A variable name must start with a letter and it does not matter whether lower or upper case letters are used

```
name      name1      name2
```

```
name      Name
```

Note
Before a HyperNext stack quits it usually save all variables, both global and local, including text stored in field controls. This automatic save can be disabled though.

Simple

A simple variable contains a piece of text with no <Carriage Return> characters. Here, a Carriage Return is denoted by CR.

For example a variable might contain a number, a set of numbers separated by commas, or a sentence.

Complex

In HyperNext a variable is considered complex when it contains one or more CR characters and then can be treated as a list or an array.

For instance a variable might contain a list of three lines such as

```
Apples<CR>
Oranges<CR>
Bananas
```

Usually the <CRs> are not displayed as HyperNext functions/commands operate transparently on lists.

Boolean

A Boolean variable can be either true or false. HyperNext, in common with most programming languages uses 0 to represent false and 1 to represent true. However, when parameters to HyperNext predefined procedure/function are expected to be Boolean, any non zero value will be interpreted as true by that procedure/function.

Colours

Colors in HyperNext have three components - red, green and blue.
Their component values range from 0 to 255.
Black equals 0,0,0 and white equals 255,255,255

For example in setting the paper colour of field 1 to red the following could be used.

```
Local r,g,b
Put  255 into r
Put 0 into g
Put 0 into b
FldPaperColor(1,r,g,b)

or more simply

FldPaperColor(1,255,0,0)
```

7 Procedures & Handlers

A procedure is a self contained set of instructions that usually has local variables while a handler is just another name for a procedure that is attached to a control such as a card, button or other control. In HyperNext, handlers can have an unlimited number of local procedures and can also access global procedures declared in the MainCode block and variables declared as global.

HyperNext has various blocks of code such as MainCode, AppleEvents, Main Timer, as well as the handlers that are attached to controls.

There are two main classes of procedure, procedures that simply perform some action and procedures which perform an action and then return a value. In HyperNext, as in most languages, a procedure that returns a value is called a Function. In HyperNext there are hundreds of functions although user defined functions are currently NOT supported.

The method of using procedures and functions are quite different. For instance in the following, MyBeep is a procedure that plays a sound, whereas MyBeepFN plays a sound and returns a number that is subsequently placed into the variable num.

```
Call MyBeep

Put CountFN into num
```

MainCode

This is where global procedures are declared and defined. Your global variables can also be declared here and perhaps even given starting values although actually global variables need only be declared in the handler which uses them.

The MainCode block is a convenient place for declaring your global variables as it is easier to keep track of them here especially when a program becomes quite large with many global variables.

The MainCode block includes a procedure called Startup that runs before any cards are loaded and is intended to include initialisation code.

The following procedures are visible in the left hand list of the editor. When a stack or application is first run they are executed in the order below.

???? - used by HyperNext and cannot be modified.

Constants - for future use when HyperNext allows constants

Variables - where global variables can be declared along with suitable comments.

StartUp - any instructions placed here will be executed before the first card is loaded. Calls to other global procedures can be made from here.

Control Handlers

Procedures within controls are local to that control and cannot be accessed from outside that control.

Parameters

Parameters are variables that are passed directly to the procedure/function when it is called. For instance a DrawRectangle procedure might need both the start coordinates and the dimensions as below

```
Call DrawRectangle(x,y,width,height)
```

or the HyperNext function CardActiveFN

```
Put CardActiveFN(12) into state
```

Currently in HyperNext variables are passed by value which means that the value of the variable is protected and cannot be changed by the procedure/function called. If it is essential that the value of the parameter be changed by the procedure then the only option is to use a global variable and declare it within that procedure.

Parameters have a further restriction placed upon their usage - within the target procedure they cannot themselves be passed as a parameter to another procedure deeper down the call chain.

8 Operations

This section lists and details keywords associated with the processing of variables.

General

Name	Type	Description
Put	C	The Put command is probably the most powerful command in HyperNext as it has a number of variations to cope with simple variables holding one line or more complex variables acting as lists and arrays. There are three basic type of Put - (1) Put x into y (Puts a variable directly into another) Put x into field 1 Put x into line 23 of field 1 Put w into z (2) Put x before y (Puts a variable before the contents of the target variable) Put x before line 23 of field 1 (3) Put x after y (Puts a variable after the contents of the target variable) Put x after line 23 of field 1 There are also variations to deal with chunks of text. A chunk can be a single character, a word or a line. Put char 3 of x into char 7 of field 1 Put word 3 of x into word 7 of field 1 Put line 3 of x into line 7 of field 1 At the present time HyperNext cannot cope with more complex Puts, eg

		Put word 2 of line 7 into ... will cause the compiler to reject the statement.
Add	C	Adds one variable to another. Values can be doubles, singles or integers. Add x to y
Subtract	C	Subtracts one variable from another. Values can be doubles, singles or integers. Subtract x from y
Multiply	C	Multiplies one variable by another. Values can be doubles, singles or integer. Multiply x by y
Divide	C	Divides one variable by another. Values can be doubles, singles or integers. Divide x by y
Decrement	C	Decrements a variable or field by one. Values can be doubles, singles or integers. Decrement x Decrement field 12
Increment	C	Increments a variable or field by one. Values can be doubles, singles or integers. Increment x Increment field 12
Clear	C	Clears a target variable or field. Variables can be global or local. Clear x Clear field 1
Set	C	Sets the variable or field to 1. Set x Set field 1

Reset	C	Sets the variable or field to 0. Reset x Reset field 1
SwapLinesVar	C	Swaps two lines in a target variable. The variable can be either local or global. SwapLinesVar data,10,15
SwapLinesField	C	Swaps two lines in a target field. SwapLinesField 1,10,15
DeleteLinevar	C	Deletes the specified line in the target variable. All higher lines are moved down by one line. DeleteLineVar data,5
DeleteLineField	C	Deletes the specified line in the target field. All higher lines are moved down by one line. DeleteLineField 1,5

Binary

These allow the underlying binary digits or bits within a number to be taken into account in an operation.

Name	Type	Description
BinFN	F	Returns a binary version of the given number. The result is expressed in all zeros and ones. Put BinFN(num) into res
BinFormatFN	F	Returns a binary version of the given number. The result is expressed in all zeros and ones and can be padded with leading zeros using the width parameter Put BinFormatFN(num,width) into res
BitAndFN	F	Returns the result of the AND operation on two given numbers. Put BitAndFN(num1,num2) into res
BitOrFN	F	Returns the result of the OR operation on two given numbers. Put BitOrFN(num1,num2) into res
BitXorFN	F	Returns the result of the XOR operation on two given numbers. Put BitXorFN(num1,num2) into res
BitOnesFN	F	Returns the result of a ones complement operation on the given number. Put BitOnesFN(num) into res
BitLeftFN	F	Returns the result of a shift left operation on the given number. The number is shifted left by nshift bits and the field width is specified by fwidth. Put BitLeftFN(num,nshift,fwidth) into res
BitRightFN	F	Returns the result of a shift right operation on the given number. The number is shifted right by nshift bits and the field width is specified by fwidth.

		Put BitRightFN(num,nshift,fwidth) into res
HexFN	F	Returns a hexadecimal version of the given number. Put HexFN(num) into res
OctFN	F	Returns an octal version of the given number. Put OctFN(num) into res
BinToIntegerFN	F	Converts a string of binary digits into an integer. The binary digits are all zeros and ones. Put BinToIntegerFN(bnum) into res
ByteToHexFN	F	Returns a hexadecimal version of the given byte. Put ByteToHexFN(bnum) into res
HexToByteFN	F	Returns a byte version of the given hexadecimal number. Put HexToByteFN(hnum) into res
ZipCompressFN	F	This function takes data and returns it in zip compressed form. It has 10 levels of compression ranging from zero to 9. ZipCompressFN(data,level) @ Fully compress data Put ZipCompressFN(svar,9) into res
ZipDecompressFN	F	This function takes zip compressed data and returns it in decompressed form. It requires an allocation of buffer space of at least 10 times, usually 10 is sufficent. ZipDecompressFN(zippeddata,buffersize) Put ZipDecompressFN(svar,10) into res
Base64EncodeFN	F	This function takes data and returns it in Base64 form. It us useful when binary data needs converting to a safe form for transmission across a network. Base64EncodeFN(data)

		Put Base64EncodeFN(svar) into res
Base64DecodeFN	F	This function takes Base64 format data and returns it in decoded form. It us useful when an email attachment or other network data needs decoding. Base64DecodeFN(b64data) Put Base64DecodeFN(svar) into res
QPEncodeFN	F	This function takes data and returns it in Quote Printable form. QP form is an encoding using ASCII characters for non ASCII text and is often used in email transfers. QPEncodeFN(data) Put QPEncodeFN(svar) into res
QPDecodeFN	F	This function takes Quote Printable format data and returns it in decoded form. It us useful when QP email data needs decoding. QPDecodeFN(QPdata) Put QPDecodeFN(svar) into res

Maths

These are the mathematical keywords and functions supported by HyperNext. Most of them treat the numbers involved as extended floating point. Both the precision of the calculations and the length of the output string can be changed.

Name	Type	Description
IsNumericFN	F	Returns 1 if the given string is numeric else returns 0. Put IsNumericFN(s1) into res
MathsSetPrecision	C	Sets the number of bits used for processing each floating point number. The default is 64 bits. Normally the precision is 4 times the output length. MathsSetPrecision(100)
MathsSetOutput	C	Sets the number of characters used for holding the output of a floating point operation. The default is 16 characters. MathsSetOutput(25)
MathsPrecisionFN	F	Returns the number of bits used for processing each floating point number. Put MathsPrecisionFN into nbits
MathsOutputFN	F	Returns the number of characters used for holding the output of a floating point operation. Put MathsOutputFN into nchars
AbsFN	F	Returns the absolute value of the given number. Put AbsFN(num) into res
AcosFN	F	Returns the arccosine of the given number with the result being in radians. Put AcosFN(num) into res
AsinFN	F	Returns the arcsine of the given number with the result being in radians. Put AsinFN(num) into res
AtanFN	F	Returns the arctan of the given number with the

		result being in radians. Put AtanFN(num) into res
Atan2FN	F	Returns the arctangent of two points x and y with the result being in radians. Put Atan2FN(x,y) into res
CeilFN	F	Returns the given number rounded up to the nearest integer. Put CeilFN(num) into res
CosFN	F	Returns the cosine of the given number with the given number being in radians. Put CosFN(num) into res
DivFN	F	Returns the result of integer division between two numbers. Put DivFN(num1,num2) into res
ExpFN	F	Returns the exponential of the given number. That is e raised to the power of the given number. Put ExpFN(num) into res
FloorFN	F	Returns the the given number rounded down to the nearest integer. Put FloorFN(num) into res
LogFN	F	Returns the natural logarithm of the given number. Put LogFN(num) into res
MaxFN	F	Returns the maximum of two given numbers. Put MaxFN(num1,num2) into res
MinFN	F	Returns the minimum of two given numbers. Put MinFN(num1,num2) into res
ModFN	F	Returns the remainder of the division between the two given numbers. Put ModFN(num1,num2) into res

PiFN	F	Returns the value PI. Put PiFN() into res
PowerFN	F	Returns the given number raised to the given power. Put PowerFN(num,pnum) into res
RndDbFN	F	Returns a double random number: 0 <= r < 1 Put RndDbFN into res
RndFN	F	Returns an integer random number within the range specified by the two numbers: num1 <= r <= num2 Put RndFN(num1,num2) into res
RoundFN	F	Rounds the given number to the nearest integer. Put RoundFN(num) into res
SeedFN	F	Returns the current random seed. Put SeedFN into res
SetSeed	P	Sets the seed of the HyperNext random number generator where seed is an integer. If seed is zero then the current time is used. SetSeed num
SignFN	F	Returns the sign of the given number. The results can be, -1, 0, or 1 if the given number is negative, zero or positive respectively. Put SignFN(num) into res
SinFN	F	Returns the sine of the given number with the given number being in radians. Put SinFN(num) into res
SqrtFN	F	Returns the square root of the given number. Put SqrtFN(num) into res

TanFN	F	Returns the tangent of the given number with the given number being in radians. Put TanFN(num) into res
TruncFN	F	Truncates the fractional part of the given number. Put TruncFN(num) into res

String

These keywords and functions all operate upon strings. Strings are considered to be just sequences of characters.

Name	Type	Description
Append	C	Appends the first text to the end of the second text. Append s1 onto s2
ReplaceOne	C	Replaces the first specified word in the given text with the second specified word. Word1 and word2 do not have to be single words, they can be any text. ReplaceOne word1 with word2 in source
ReplaceAll	C	Replaces all occurrences of the specified word in the given text with the second specified word. Word1 and word2 do not have to be single words, they can be any text. ReplaceAll word1 with word2 in source
AscFN	F	Returns an integer representing the first character of the given text. @ If s1 contains an A then res equals 65 Put AscFN(s1) into res
ChrFN	F	Returns the character value of the given number. @ Put a space into s1 Put ChrFN(32) into s1
ChunksFN	F	Counts the number of chunks in the specified string that are separated by the specified separator. The separator can be a space, comma, full stop etc and can be specified by quotes or a by a variable. Put ChunksFN(s1,sep) into ncount
EndLineFN	F	Returns the the end of line character for the specified system 1 - Macintosh - Chr(13) 2 - Unix - Chr(10) 3 - Windows - Chr(13) +Chr(10)

FindWordFN	F	Returns the position of the given word assuming that it can be separated from other word by punctuation. Put FindWordFN(source,word) into wpos
FormatFN	F	Returns a text version of a given number. The number format used is specified in the System Number Control Panel but the text is formatted according to the format specifiers given below. The format string can be made up of three separate specifiers, for positive, negative and zero numbers respectively. Character Description # If the digit is present it is displayed. 0 As for # but if the digit is not present then a 0 is displayed. . The position of the decimal point. , Specifies that the number should be displayed with thousand separators. % Displays the number multiplied by 100. + If the number is positive a + sign is displayed on the left of the number and if negative a - sign is displayed. - If the number is negative a - sign is displayed on the left of the number, otherwise nothing. E or e The number is displayed using scientific notation. \char The character following the backslash is displayed. (An open parenthesis is displayed.) A closed parenthesis is displayed. Examples Format Number Output #.## 3.142 3.14 #.0000 3.143 3.1420 #% 0.35 35% Put FormatFN(number,'#.##') into output
InStrFN	F	Returns the position of the first occurrence of the specified word in the given text and starts the search at the specified position. Word does not have to be a single word, it can be any text.

		Put InStrFN(pos,source,word) into wpos
LeftFN	F	Returns the left n characters of the given text. Put LeftFN(s1,n) into s2
LenFN	F	Returns the number of characters in the given text. Put LenFN(s1) into slen
LinesFN	F	Returns the number of lines in the given text where the line separators are Chr(13) as used on the Macintosh. Put LinesFN(s1) into nlines
LowerFN	F	Returns the lower case version of the given text. Put LowerFN(s1) into s2
LtrimFN	F	Returns the given text with all left leading spaces removed. Put LtrimFN(s1) into s2
MiddleFN	F	Returns the text comprising n characters from position p of the given text. Put MiddleFN(source,p,n) into s2
NthChunkFN	F	Returns the nth chunk from the given text using the given separator. Put NthChunkFN(source,sep,place) into res
NStringsFN	F	Returns the specified number of strings concatenated together. Put NStringsFN(source,count) into res
ReplaceEndsFN	F	Returns the given text with all line endings changed to the new line ending. It will automatically find Macintosh, Unix and Windows line endings and change them as appropriate. The end type specifies the ending to be used. @ Change to UNIX Put ChrFN(10) into end type Put ReplaceEndsFN(s1,end type) into s2

		Line endings 1 - Macintosh - Chr(13) 2 - Unix - Chr(10) 3 - Windows - Chr(13) +Chr(10)
RightFN	F	Returns the n rightmost characters from the given text. Put RightFN(s1,n) into s2
StrCompFN	F	Returns the resulting comparison of two texts. If ctype is 0 then case-sensitive comparison is used and if 1 then case is ignored. Put StrCompFN(s1,s2,ctype) into res Results -1 string 1 < string 2 0 string 1 = string 2 1 string 1 > string 2
TitleFN	F	Returns the titled version of the given text. This converts all characters in the text to lowercase and then converts the first character of each word to uppercase. Put TitleFN(s1) into s1
TrimFN	F	Returns the given text with both left and right spaces removed. Put TrimFN(s1) into s1
UpperFN	F	Returns the uppercase version of the given text. Put UpperFN(s1) into s1
ValFN	F	Returns the text representation of the given number with the separators being as defined in the System Numbers Control Panel. This recognizes binary, octal and hexadecimal using the ampersand sign using &b, &O and &H respectively. Put ValFN(num) into s1
WordsFN	F	Returns the number of times the specified word occurs in the given text. The word can be any text and need not be delimited by spaces. Put WordsFN(text,word) into nwords

AscBFN	F	Returns an integer representing the first character of the given byte text. @ If s1 contains an A then res equals 65 Put AscBFN(s1) into res
ChrBFN	F	Returns the byte string value of the given number. @ Put a space into s1 Put ChrBFN(32) into s1
InStrBFN	F	Returns the position of the first occurrence of the specified word in the given byte text and starts the search at the specified byte position. Word does not have to be a single word, it can be any text. Put InStrBFN(pos,source,word) into wpos
LeftBFN	F	Returns the left n bytes of the given text. Put LeftBFN(s1,n) into s2
LenBFN	F	Returns the number of bytes in the given text. Put LenBFN(s1) into slen
MiddleBFN	F	Returns the text comprising n bytes from position p of the given text. Put MiddleBFN(source,p,n) into s2
RightBFN	F	Returns the n rightmost bytes from the given text. Put RightBFN(s1,n) into s2
LineEndingFN	F	Returns the the end of line character for the current platform Macintosh - Chr(13) Unix - Chr(10) Windows - Chr(13) +Chr(10) @ Example Put LineEndingFN into sep

Sorting

Both variables and fields can have their contents sorted in a number of ways

Depending upon the structure of the source contents sorting can be carried out using either lines or words. The type of sort can also be specified, such as text, numeric or date.

An error flag can be raised if the sort data is incompatible with the specified sort format. For instance if the data does not have a valid date format when type date is specified.

Name	Type	Description
SortLinesVar	C	This assumes that the data in the specified variable is organized with a line format, as when separated by Chr(13), the normal format in HyperNext. SortLinesVar sdata,type,direction where type: 1 - text, 2 - numeric, 3 - date direction: 1 - descending, 2 - ascending.
SortLinesField	C	This assumes that the data in the specified field is organized with a line format, that is every line is separated by Chr(13), the normal format in HyperNext. SortLinesField sdata,type,direction where type: 1 - text, 2 - numeric, 3 - date direction: 1 - descending, 2 - ascending.
SortWordsVar	C	This assumes that the data in the specified variable is organized with a line format and that each line is

		composed of a number of words separated by a specified separator. Each line is separated by Chr(13), the normal format in HyperNext. The number of words in each line should be the same and the word or column that acts as the key is specified by wnum below. SortWordsVar src,typ,dir,sep,wnum where type: 1 - text, 2 - numeric, 3 - date direction: 1 - descending, 2 - ascending. separator eg , (comma) wnum eg 6 for column 6 Example This example takes the parameters from various fields. Local src,typ,dir,sep,wnum Put field 5 into src Put field 2 into typ Put field 3 into dir Put ',' into sep Put field 6 into wnum SortWordsVar src,typ,dir,sep,wnum Put src into field 5
SortWordsField	C	This assumes that the data in the specified field is organized with a line format and that each line is composed of a number of words separated by a specified separator. Each line is separated by Chr(13), the normal format in HyperNext. The number of words in each line should be the same and the word or column that acts as the key is specified by wnum below. SortWordsFieldB src,typ,dir,sep,wnum where type: 1 - text,

		2 - numeric, 3 - date direction: 1 - descending, 2 - ascending. separator eg , (comma) wnum eg 6 for column 6 Example This example takes the parameters from various fields. Local src,typ,dir,sep,wnum Put 5 into src Put field 2 into typ Put field 3 into dir Put ',' into sep Put field 6 into wnum SortWordsField src,typ,dir,sep,wnum

Date & Time

These are a related set of commands and functions for setting and reading dates and times.

The central value representing a date is the total number of seconds since a specific date namely January 1, 1904.

To use a date one can create a date value using the DateTotalSecsFN function and then modify the value using the DateSet commands. Various portions of the date such as month, minute etc can also be accessed.

Name	Type	Description
TicksFN	F	Returns the number of ticks since the computer was started. Each tick is 1/60th of a second. Put TicksFN into tnow
MicroSecondsFN	F	Returns the number of microseconds since the computer was started. A microsecond is 1/1000000th or one millionth of a second. Put MicroSecondsFN into tnow
DateNowFN	F	Creates a date value for the current date and time. Put DateNowFN into totSecs1
DateTotalSecsFN	F	This parses a date string and returns the total number of seconds representing that date. If an invalid date is passed to the function then an empty value will be returned. Once the date is set the actual time on that date can be set using the DateSet commands. Put DateTotalSecsFN(dvar) into date1 Valid formats for dvar are: 15/10/2004, 15/10/04, 15Oct2004,
DateAbbrevDateFN	F	Takes a variable holding the total number of seconds and returns an abbreviated date such as Mon, May 5, 2003 Put DateAbbrevDateFN(tsecs) into adate
DateDayOfWeekFN	F	Takes a variable holding the total number of

		seconds and returns the corresponding day of the week in integer form where Sunday is 1, Monday is 2, ... and Saturday is 7. Put DateDayOfWeekFN(tsecs) into adate
DateDayOfYearFN	F	Takes a variable holding the total number of seconds and returns the corresponding day of the year in integer form, for example day 285. Put DateDayOfYearFN(tsecs) into adate
DateLongDateFN	F	Takes a variable holding the total number of seconds and returns a date having the form Tuesday, May 6, 2003 Put DateLongDateFN(tsecs) into adate
DateShortDateFN	F	Takes a variable holding the total number of seconds and returns a date having the form 6/5/2003 Put DateShortDateFN(tsecs) into adate
DateLongTimeFN	F	Takes a variable holding the total number of seconds and returns a time having the form 7:15:21 PM Put DateLongTimeFN(tsecs) into atime
DateShortTimeFN	F	Takes a variable holding the total number of seconds and returns a time having the form 7:15 PM Put DateShortTimeFN(tsecs) into atime
DateWeekOfYearFN	F	Takes a variable holding the total number of seconds and returns an integer for the week that the date falls on, for instance 31 Put DateWeekOfYearFN(tsecs) into adate
DateDayFN	F	Takes a variable holding the total number of seconds and returns the day portion of that date. Put DateDayFN(tsecs) into adate
DateHourFN	F	Takes a variable holding the total number of seconds and returns the hour portion of that date. Put DateHourFN(tsecs) into adate

DateMinuteFN	F	Takes a variable holding the total number of seconds and returns the minute portion of that date. Put DateMinuteFN(tsecs) into adate
DateMonthFN	F	Takes a variable holding the total number of seconds and returns the month portion of that date. Put DateMonthFN(tsecs) into adate
DateSecondFN	F	Takes a variable holding the total number of seconds and returns the second portion of that date. Put DateSecondsFN(tsecs) into adate
DateYearFN	F	Takes a variable holding the total number of seconds and returns the year portion of that date. Put DateYearFN(tsecs) into adate
DateSetDay	C	Sets the day portion of the given date. DateSetDay adate,1
DateSetHour	C	Sets the hour portion of the given date. DateSetHour adate,13
DateSetMinute	C	Sets the minute portion of the given date. DateSetMinute adate,45
DateSetMonth	C	Sets the month portion of the given date. DateSetMonth adate,11
DateSetSecond	C	Sets the second portion of the given date. DateSetSecond adate,23
DateSetYear	C	Sets the year portion of the given date. DateSetYear adate,2003

Arrays

Although the list based data structures in HyperNext are fine for most problems there are times where an array is a more natural and efficient representation. Arrays are named at creation where the name is expected to be a string although it can be 1, 2 etc.

Array values are still strings so that they can operate seamlessly with HyperNext lists and variables.
Both row and column subscripts start at 1 just as other lists in HyperNext do.

Warning, unlike global and local variables, arrays are not saved when HyperNext quits. However, as they are strings they can be saved using the text file commands.

Name	Type	Description
ArrayCreateFN	F	Creates a named array having the specified number of rows and columns. ArrayCreateFN(aname,nrows,ncolumns) Put ArrayCreateFN(name,10,20) into okay Function return numbers - 0 - successfully created. 1 - ** not used ** 2 - name already exists 3 - ** not used ** 4 - insufficient memory.
ArrayResizeFN	F	Resizes the named array to the specified number of rows and columns. The data within the resized array remains unchanged except for those elements lost. ArrayResizeFN(aname,nrows,ncolumns) Put ArrayResizeFN(name,100,200) into okay Function return numbers - 0 - successfully resized. 1 - name not found 2 - ** not used ** 3 - ** not used ** 4 - insufficient memory.
ArrayDeleteAll	C	This simply deletes all arrays from memory and reclaims their storage space.

		ArrayDeleteAll
ArrayExistsFN	F	Returns 1 if the named array exists otherwise it returns 0. ArrayExistsFN(aname) Put ArrayExistsFN(name) into num
ArrayBytesStoredFN	F	Returns the number of bytes used by the named array. ArrayBytesStoredFN(aname) Put ArrayBytesStoredFN(name) into num
ArrayCountFN	F	Returns the number of arrays is existance. ArrayCountFN Put ArrayCountFN into num
ArrayStatsFN	F	Returns a list detailing the arrays in existance. The stats for each array occupy one line of the list and they are array name, number of rows, number of columns. The entries on each line are separated by commas. ArrayStatsFN Put ArrayStatsFN into slist
ArrayPutValue	C	This puts the specified value into the named array location designated by the row and column. ArrayPutValue(aname,row,col,value) ArrayPutValue(name,5,10,value)
ArrayValueFN	F	This returns the value from the named array location designated by the row and column. ArrayValueFN(aname,row,col) Put ArrayValueFN(name,row,col) into value
ArrayFill	C	This fills the entire arrays with the specified value. ArrayFill(aname,value)

		ArrayFill(name,25)
ArrayFillRow	C	This fills one row of the array with the specified value. ArrayFillRow(aname,row,value) ArrayFillRow(name,10,25)
ArrayFillColumn	C	This fills one column of the array with the specified value. ArrayFillColumn(aname,col,value) ArrayFillColumn(name,10,25)
ArrayRowCountFN	F	Returns the number of rows in the named array. ArrayRowCountFN(aname) Put ArrayRowCountFN(name) into num
ArrayColumnCountFN	F	Returns the number of columns in the named array. ArrayColumnCountFN(aname) Put ArrayColumnCountFN(name) into num
ArrayRowListFN	F	Returns the specified row from the array as a list. ArrayRowListFN(aname,row) Put ArrayRowListFN(name,10) into list
ArrayColumnListFN	F	Returns the specified column from the array as a list. ArrayColumnListFN(aname,row) Put ArrayColumnListFN(name,10) into list
ArrayPutRow	C	Puts the list into the specified row of the array. The separator describes the separator used in the list, whether a comma, tab etc. ArrayPutRow(aname,row,list,sep) ArrayPutRow(aname,25,list,comma)
ArrayPutColumn	C	Puts the list into the specified column of the

		array. The separator describes the separator used in the list, whether a comma, tab etc. ArrayPutColumn(aname,col,list,sep) ArrayPutColumn(aname,25,list,comma)
ArrayWholeFN	F	Returns the whole contents of the array as a list. Each line in the list will contain the items from a row separated by the specified separator. ArrayWholeFN(aname,sep) Put ArrayWholeFN(name,comma) into list
ArrayPutWhole	C	This puts the contents of the list into the named array. Each line in the list will contain the items for a row separated by the specified separator. ArrayPutWhole(aname,list,sep) ArrayPutWhole(name,list,comma)
ArrayCopy	C	Copies one array into a second array. The size of the second array must be equal to or larger than the first otherwise no operation will be performed. ArrayCopy(aname1,aname2) ArrayCopy(xmatrix,ymatrix)

9 Cards, Windows & Screens

Within both HyperNext Creator and the HyperNext scripting language cards are the main organizational unit. Only one card may be visible at a time and each card can hold many different types of control so allowing the user to interact with the application. By changing the focus card different aspects of the stack/application can be made accessible to users.

By treating cards as windows it can become easier to visualize the structure of an application. Consider the creation of a simple neural network and how the user may interact with it. Generally a neural network system has three main stages, data pre-processing, training and querying. This can be easily implemented by a HyperNext Creator project having three cards which are navigateable using push buttons. The first card allows the user to drop in data, set up the number of data rows and then pre-process it. The second card deals with training the neural network on the data and typically would display a graph of training error over time. The third card uses the trained neural network to accept further user data as a query and produce an evaluation of it. In this example, three main functions equates with three cards. To deploy the system in the field further cards would probably be added, such as a front screen with information, a context help card and a preferences card.

Screen & ScreenShots

Name	Type	Description
ScreenWidthFN	F	Returns the width of the first screen. Put ScreenWidthFN into width
ScreenHeightFN	F	Returns the height of the first screen. Put ScreenHeightFN into height
SystemGetPixel	C	Places the appropriate red, green and blue values of the pixel at the specified screen coordinates. SystemGetPixel(x,y,r,g,b)
SystemMouseXFN	F	Returns the x coordinate of the mouse cursor on the screen. Put SystemMouseXFN into mx

SystemMouseYFN	F	Returns the y coordinate of the mouse cursor on the screen. Put SystemMouseYFN into my
SystemMouseDownFN	F	Returns 1 when the mouse is pressed and 0 otherwise. Put SystemMouseDownFN into mdown
SystemCommandLineFN	F	On the Windows platform this returns the parameters passed to the program when it was launched via the command line. On the Macintosh platform it always returns 0. Put SystemCommandLineFN into cline
ScreenShotCanvas	C	Takes a screen shot of the whole screen and places it in the specified canvas. When the scale parameter is 0 then the image is placed in the canvas unscaled and when 1 or nonzero is scaled to fit the canvas. ScreenShotCanvas(cid,scale)
ScreenRectCanvas	C	Takes a screen shot of a rectangular area within the screen and places it in the specified canvas. When the scale parameter is 0 then the image is placed in the canvas unscaled and when 1 or nonzero is scaled to fit the canvas. ScreenRectCanvas(cid,scale,x,y,w,h)
ScreenShotImageBank	C	Takes a screen shot of the whole screen and places it in the specified imagebank. When the scale parameter is 0 then the image is placed in the imagebank unscaled and when 1 or nonzero is scaled to fit the imagebank. ScreenShotImageBank(ibnum,scale)
ScreenRectImageBank	C	Takes a screen shot of a rectangular area within the screen and places it in the specified imagebank. When the scale parameter is 0 then the image is placed in the imagebank unscaled and when 1 or nonzero is scaled to fit the imagebank. ScreenRectImageBank(ibnum,scale,x,y,w,h)

Card Commands

The HyperNext language has a rich set of commands for moving between cards.

The first card in a stack is recognized as the Home card by HyperNext and it cannot be deleted. Whenever a stack or built application starts up it will always load the Home card first. It can though be forced into moving immediately to another card by placing redirection code in the Home card start up handler.

Note
When any Goto card command is executed, all other commands following it are ignored.

```
Global x,y
Put 2 into x
GotoCard 2
@ this line and following are not executed
Put 10 into y
```

Name	Type	Description
GotoHome	C	Goes to the first card. In HyperNext the first card called is always called Home and cannot be deleted.
GotoCard	C	Goto the specified card, either a number or a name. Goto 14 Goto finances
GotoThisCard	C	Goes to the current card - it reloads the card and executes its Open handler.
GotoPriorCard	C	Goto the previous card. If some cards have been deleted this command will search downwards until it reaches an active card. Note that the Home card is always active.
GotoNextCard	C	Goes to the next highest and active card.
CardNextFN	F	Returns the number of the next card. If the current card is the highest numbered card then 0 is returned. Put CardNextFN into cnum

CardPriorFN	F	Returns the number of the previous card. If the current card is card 1 then 0 is returned. Put CardPriorFN into cnum
DeleteCard	C	Deletes the target card and if the specified destination card is non zero then that card will be loaded. If the target card is the current card and no destination is given then the command will be ignored DeleteCard 12,0 tries to delete card 12 DeleteCard 12,5 tries to delete card 12 and load 5
DeactivateCard	C	Sets a card to inactive but does not delete it. This makes the card inaccessible until reactivated. DeactivateCard 12
ActivateCard	C	This activates the target card and if that card is non zero it may be loaded. @ simply activate a card ActivateCard 12,0 @ activate a card and go to it ActivateCard 12,12
TotalCardsFN	F	Returns the total number of cards in the current stack/application. Put TotalCardsFN into numcards
ActiveCardsFN	F	Returns the number of active cards in the current stack/application. Put ActiveCardsFN into cnum
CardExistsFN	F	Returns true if the target card exists otherwise returns false Put CardExistsFN(9) into res
CardActiveFN	F	Returns true if the target card is active otherwise returns false Put CardActiveFN(20) into x
CreateCardFN	F	Creates a new card from the target card. The new

		card is a duplicate of the target card, including controls, their handlers/procedures, and values. Put CreateCardFN(2) into cardnum GotoCard cardnum
CardIDFN	F	Returns the number of the current card. Put CardIDFN into field 1
CardNameFN	F	Returns the name of the current card. Put CardNameFN into field 1
CardEnableColor	C	This enables or disables the background colour of a card. CardEnableColor cid,value where value is 0 or 1
CardSetColor	C	This simply sets the background colour of the specified card. If the card colour is already disabled then the card will appear white. CardSetColor cid,red,green,blue
CardGetColor	C	Returns the background colour of the current card. CardGetColor red,green,blue
CardSetName	C	Sets the name and title of the specified card. CardSetName(cid,name)
CardSetWidth	C	Sets the width of the specified card. CardSetWidth(cid,number)
CardSetHeight	C	Sets the height of the specified card. CardSetHeight(cid,number)
CardSetLeft	C	Sets the left side of the specified card in pixels from the left hand side of the screen. CardSetLeft(cid,number)
CardSetTop	C	Sets the top of the specified card in pixels from the top of the screen.

		CardSetTop(cid,number)
CardSetCenter	C	Centers the specified card within the screen. If the card number specified is not the current card then the card will be centered when it is opened because the command simply changes the x and y coordinates of the card in order to center it. If the card number passed is zero then the current card will be centered. CardSetCenter(cnumber)
CardLoadImage	C	Loads the named local image file and attaches the image to the card. The image can be left at its normal size or scaled to fill the card using values of 0 or 1 respectively. CardLoadImage(cid,fname,scale) CardLoadImage(5,fname,1)
CardLoadImageAbs	C	Loads the named absolute image file and attaches the image to the card. The image can be left at its normal size or scaled to fill the card using values of 0 or 1 respectively. CardLoadImageAbs(cid,fname,scale) CardLoadImageAbs(5,fname,1)
CardLoadImageXP	C	Loads the named local cross-platform image file and attaches the image to the card. The image can be left at its normal size or scaled to fill the card using values of 0 or 1 respectively. CardLoadImageXP(cid,fname,scale) CardLoadImageXP(5,fname,1)
CardLoadImageXPAbs	C	Loads the named absolute cross-platform image file and attaches the image to the card. The image can be left at its normal size or scaled to fill the card using values of 0 or 1 respectively. CardLoadImageXPAbs(cid,fname,scale) CardLoadImageXPAbs(5,fname,1)
CardBankImage	C	Attaches a bank image to the specified card. This is useful when many cards use the same image so requiring just one copy of the image to be held in

		memory. Furthermore, when the image bank is changed then all cards using it will be automatically updated. CardBankImage(cid,bankid) CardBankImage(12,2)
CardRemoveImage	C	Remove the image from the specified card. If the image was attached to the card using a file then it will be deleted from memory. CardRemoveImage(cid) CardRemoveImage(12)
CardLeftFN	F	Returns the distance of the specified card from the lefthand side of the screen. Put CardLeftFN(cnum) into cleft
CardTopFN	F	Returns the distance of the specified card from the top of the screen. Put CardTopFN(cnum) into ctop
CardWidthFN	F	Returns the width of the specified card. Put CardWidthFN(cnum) into cleft
CardHeightFN	F	Returns the height of the specified card. Put CardHeightFN(cnum) into cleft

10 Control Types

There are currently fourteen types of controls available in the HyperNext Language, they are Button, Canvas, Field, Text, Movie, Check Box, Radio Button. Pop up Menu, Slider, ScrollBar, Progress Bar, ListBox, Sprite Surface and Timer. All controls, except Timer, can be visible to the user. Controls are created in using the Creator/Developer by simply creating one in the Controls window and then by placing them onto a Card after which their specific properties can be set. They can also be created at runtime when a card is created or duplicated.

HyperNext distinguishes sharply between the actions and uses of controls. For instance, buttons are used to initiate actions, canvases for displaying graphics/pictures, fields for holding editable/scrolling text, texts for displaying text, and movies for playing movies and sounds. Canvases though are highly flexible and can be programmed to act like buttons and text fields etc.

Buttons

Buttons allow the user to interact with the program and each button has its own handler. In addition to their size and placement, buttons have attributes for image, sound and goto card, as well as natural attributes such as caption etc.

A button can display an image which is often more interesting and effective than a simple line of text on a grey background.

Buttons can be set to play a predefined sound when pressed although any sounds must already be defined in the Sound Library.

Often a button is simply needed in order to navigate through the stack of cards and so HyperNext buttons have a Goto property. If a button has its Goto property set then this will override the attached user defined handler.

By default the action handler of a newly created button is empty but it is possible to create complex programs within each handler, and to create many procedures within each handler. To edit the action handler simply press the Script button in the Properties window when the relevant button is highlighted.

Button properties set at design time

Property	Description
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the button.
Height	The height of the button.
Text	The text that a user will see at runtime.
Font	The name of the font used to display the text.
Fontsize	The size of the text.
Bold	Sets the bold either on or off.
Italic	Set the italic either on or off.
Underline	Sets the underline either on or off.
Align	Text alignment – left, center or right
Picture	The picture that will be displayed instead of the text. The image ratio check box determines how the image will be displayed.
Sound	The sound played when the button is pressed.
Goto	The card to be loaded when the button is pressed. This will override any code in the action handler.

Name	Type	Description
ButtonIDFN	F	When used within the action handler of a button returns the numeric identity of that button. Put ButtonIDFN into bid
ButtonSetMode	C	Sets whether the button is enabled or not In the following the button identity is bid, and value determines whether the button will be enabled or disabled. The button is disabled when value is zero and when non zero is enabled. ButtonSetMode bid,value
ButtonSetView	C	Sets whether the button will be visible or not In the following bid is the button identity, and value determines whether the button will be in view or will be hidden. When value is zero the button is hidden and when non zero is shown. ButtonSetView bid,value
ButtonSetLeft	C	Sets the distance in pixels of the left side of the

		button from the left side of the card. ButtonSetLeft bid,value
ButtonSetTop	C	Sets the distance in pixels of the top of the button from the card top. ButtonSetTop bid,value
ButtonSetWidth	C	Sets the width of the button in pixels. ButtonSetWidth bid,value
ButtonSetHeight	C	Sets the height of the button in pixels. ButtonSetHeight bid,value
ButtonSetFont	C	Sets the font name used to display the text or caption within the button. ButtonSetFont bid,fname
ButtonSetSize	C	Sets the size of the text or caption within the button. ButtonSetSize bid,value
ButtonSetAlign	C	Sets the alignment of the text or caption within the button. ButtonSetAlign bid,value Values can be 1 - left 2 - center 3 - right
ButtonSetBold	C	Sets the bold attribute of the text or caption within the button to either on or off - 0 is off, 1 is on. ButtonSetBold bid,value

ButtonSetItalic	C	Sets the italic attribute of the text or caption within the button to either on or off - 0 is off, 1 is on. ButtonSetItalic bid,value
ButtonSetUnderline	C	Sets the underline attribute of the text or caption within the button to either on or off - 0 is off, 1 is on. ButtonSetUnderline bid,value
ButtonSetText	C	Sets the text or caption within the button. ButtonSetText bid,value
ButtonModeFN	F	The returned value indicates whether the specified button is enabled or disabled. A return value of non zero indicates the button is enabled otherwise it is disabled. Put ButtonModeFN(3) into bokay
ButtonViewFN	F	The returned value indicates whether the specified button is visible or hidden. A return value of non zero indicates the button is visible otherwise it is hidden. Put ButtonViewFN(3) into bokay
ButtonLeftFN	F	Returns the distance in pixels of the left side the button from the left side of the card. Put ButtonLeftFN(1) into num
ButtonTopFN	F	Returns the distance in pixels of the top of the button from the top of the card. Put ButtonTopFN(1) into num
ButtonWidthFN	F	Returns the width of the button.

		Put ButtonWidthFN(1) into num
ButtonHeightFN	F	Returns the height of the button. Put ButtonHeightFN(1) into num
ButtonFontFN	F	Returns the font name in which the button text is displayed. Put ButtonFontFN(1) into fname
ButtonSizeFN	F	Returns the size of the button text. Put ButtonSizeFN(1) into num
ButtonAlignFN	F	Returns the alignment of the button text. Values are: 1 - left 2 - center 3 - right 4 - default Put ButtonAlignFN(1) into num
ButtonBoldFN	F	Returns 1 if the button text bold is on otherwise it returns 0. Put ButtonBoldFN(1) into num
ButtonItalicFN	F	Returns 1 if the button text italic is on otherwise it returns 0. Put ButtonItalicFN(1) into num
ButtonUnderlineFN	F	Returns 1 if the button text underline is on otherwise it returns 0. Put ButtonUnderlineFN(1) into num
ButtonTextFN	F	Returns the text or caption displayed in the button. Put ButtonTextFN(1) into txt

ButtonSetBevel	C	Sets the type of bevel for the button. The value is in the range 0 to 7 as detailed below. ButtonSetBevel(bid,value) The values can be: 0 - Small bevel 1 - Normal bevel 2 - Large bevel 3 - Rounded bevel (OS X only) 4 - No bevel (Windows only) 5 - Round (OS X only) 6 - Large round (OS X only) 7 - Disclosure (OS X only) On other platforms the OS X only settings appear as the small bevel style.
ButtonSetType	C	Sets the type of button and how it responds to mouse presses. ButtonSetType(bid,value) The values can be: 0 - Remains in the down position until the mouse is released. 1 - Toggles, remains in the down position until clicked again. 2 - Sticky, remains in the down position when pressed.
ButtonSetValue	C	When set to 1 it makes the button appear pressed, and when 0 normal. ButtonSetValue(bid,value)
ButtonBevelFN	F	Returns the current bevel setting for the specified button. Put ButtonBevelFN(5) into bval
ButtonTypeFN	F	Returns the current type setting for the specified button. Put ButtonTypeFN(5) into bval
ButtonValueFN	F	Returns the current value setting for the specified button. Put ButtonValueFN(5) into bval

ButtonCallNumber	C	Calls the script in the button specified by the given card and button numbers. If the card number is zero then button is assumed to reside on the present card. ButtonCallNumber(cardnumber,buttonnumber) @ Button 6 on present card ButtonCallNumber(0,6) @ Button 12 on card 8 ButtonCallNumber(8,12)
ButtonCallName	C	Calls the script in the button specified by the given card number and button name. If the card number is zero then button is assumed to reside on the present card. ButtonCallName(cardnumber,buttonname) ButtonCallName(5,'SoundBeep')

Fields

A field control can both display text and receive text input via the keyboard or from the program.

At design time a field control must be set to one of the six types below

1 - single line.

2 - single line with dark border.

3 - multi line.

4 - multi line with dark border.

5 - multi line with vertical scroll bar.

6 - multi line with dark border and vertical scroll bar

Text can be placed into field 1 using the following command

```
Put x into Field 1
```

To access a field on an out of focus card use the field keywords FieldCardSet and FieldCardFN to set the field value and fetch the value respectively.

Fields can also have a structure in a similar manner to an array. In HyperNext an array is just a list of items separated by carriage returns(CRs). As lists are essential in most programs HyperNext has a set of commands capable of handling them. The Put command is the most versatile as shown

```
Put x into line 10 of field 1

Put x after line 10 of field 1

Put x before line 10 of field 1

Put x into word 10 of field 1

Put x into char 10 of field 1
```

Each field can have its own set of handlers that are activated when the Mouse Up event occurs within the field.

Name	Type	Description
FieldIDFN	F	When used within the action handler of a field returns the numeric identity of that field. Put FieldIDFN into cid
FieldSetMode	C	Sets whether the field is enabled or not In the following cid is the field control identity, and value determines whether the field will be enabled or disabled. When value is zero then the field is disabled and when non zero it is enabled. FieldSetMode cid,value
FieldSetView	C	Sets whether the field is visible or not In the following cid is the field control identity, and value determines whether the field will be in view or will be hidden. When value is zero then the field is hidden and when non zero it is visible. FieldSetView cid,value
FieldSetLeft	C	Sets the distance in pixels of the left side of the field from the left side of the card. FieldSetLeft bid,value
FieldSetTop	C	Sets the distance in pixels of the top of the field from the card top. FieldSetTop bid,value
FieldSetWidth	C	Sets the width of the field in pixels. FieldSetWidth bid,value
FieldSetHeight	C	Sets the height of the field in pixels. FieldSetHeight bid,value
FieldModeFN	F	The value returned indicates whether the specified field is enabled or disabled. A return value of non zero indicates the field is enabled otherwise it is disabled.

		Put FieldModeFN(3) into bokay
FieldViewFN	F	The value returned indicates whether the specified field is visible or hidden. A return value of non zero indicates the field is visible otherwise it is hidden. Put FieldViewFN(3) into bokay
FieldLeftFN	F	Returns the distance in pixels of the left side the field from the left side of the card. Put FieldLeftFN(1) into num
FieldTopFN	F	Returns the distance in pixels of the top of the field from the top of the card. Put FieldTopFN(1) into num
FieldWidthFN	F	Returns the width of the field control. Put FieldWidthFN(1) into num
FieldHeightFN	F	Returns the height of the field control. Put FieldHeightFN(1) into num
FieldReadOnly	F	Most fields are set to accept user input but their read-only mode can be changed with this command. When value is 0 the field will accept text, and when non zero it becomes read-only. FieldReadOnly id,value
FieldLineFN	F	When a field handler is activated this value will be set before any code in the handler is run. It will return the line number within the field where the Mouse Up event occurred. Put FieldLineFN into fline Put line fline of field 1 into fdata
FieldPosFN	F	When a field handler is activated this value will be

		set before any code in the handler is run. It will return the x position in pixels within the line where the Mouse Up event occurred. Put FieldPosFN into xpos
FieldCardSet	F	Sets the text of a field that resides on a card currently out of focus. At the current time it completely replaces the text already in the field. FieldCardSet cardid,fieldid,value
FieldCardFN	F	Returns the text from a field residing on a card currently out of focus. At the current time it can only fetch all of the text. FieldCardFN(cardnum,field index) Put FieldCardFN(21,7) into x
FieldTabs	C	This sets whether a field can receive tabs from the keyboard or whether the tab will move the focus to another control on the card. FieldTabs id,value
FieldPaperColor	C	Sets the paper colour of field. FieldPaperColor id,r,g,b
FieldTextColor	C	Sets the text colour of field. FieldTextColor id,r,g,b
FieldTextAlign	C	This sets the alignment of text for the entire field. 0 - default 1 - left 2 - center 3 - right FieldTextAlign id,value

FieldTextFont	C	Sets the name of the font in which the text will be displayed for the entire field. FieldTextFont id,value
FieldTextSize	C	Sets the size of the text for the entire field. FieldTextSize id,value
FieldTextBold	C	Sets whether the text is in bold or not for the entire field. FieldTextBold id,value
FieldTextItalic	C	Sets whether the text is in italic or not for the entire field. FieldTextItalic id,value
FieldTextUnderline	C	Sets whether the text is in underline or not for the entire field. FieldTextUnderline id,value
FieldSelectAlign	C	Sets the alignment for the selected text. FieldSelectAlign id,value
FieldSelectFont	C	Sets the font name for the selected text. FieldSelectFont id,value
FieldSelectSize	C	Sets the font size for the selected text. FieldSelectSize id,value
FieldSelectColor	C	Sets the color for the selected text. FieldSelectColor id,r,g,b
FieldSelectBold	C	Sets whether or not the selected text is in bold. FieldSelectBold id,value

FieldSelectItalic	C	Sets whether or not the selected text is in italic. FieldSelectItalic id,value
FieldSelectUnderline	C	Sets whether or not the selected text is in underline. FieldSelectUnderline id,value
FieldSelTextFN	F	Returns the text selected by the user double clicking in the specified field. For example, below specifies field 2. Put FieldSelTextFN(2) into txt

Field Events

In addition to the Mouse Down event a field can also receive several other events. However, in order to make programming easier for beginners most of these other events are disabled by default but can easily be enabled when a card (window) loads.

The event which triggered the field's handler can be found using the FieldEvent function as shown in the example below and the event numbers are listed in the following table:

Example

```
Put FieldEventFN(1) into evnum

If evnum=1 Then
   @ respond to Mouse Down
EndIf

If evnum=2 Then
   @ respond to Mouse Up
EndIf
```

Event	Description
1 - Mouse Down	The mouse button was pressed within the field.
2 - Mouse Up	The mouse button was released within the field.
3 - Mouse Drag	The mouse was dragged within the field.
4 – Mouse Move	The mouse moved within the field.
5 – Mouse Enter	The mouse entered the field.
6 – Mouse Exit	The mouse exited the field.
7 - Keydown	A key was pressed while the field had the focus.
8 – Got focus	The field received the focus.
9 – Lost focus	The field lost the focus.

Name	Type	Description
FieldEventFN	F	Returns an integer specifying the field event which caused the field handler to be called.

		Put FieldEventFN into evnum
FieldSetMouseUp	C	Disables or enables the field MouseUp event for the specified field using the values 0 or 1 respectively. FieldSetMouseUp(fid,value)
FieldSetMouseMove	C	Disables or enables the field MouseMove event for the specified field using the values 0 or 1 respectively. FieldSetMouseMove(fid,value)
FieldSetMouseEnter	C	Disables or enables the field MouseEnter event for the specified field using the values 0 or 1 respectively. FieldSetMouseEnter(fid,value)
FieldSetMouseExit	C	Disables or enables the field MouseExit event for the specified field using the values 0 or 1 respectively. FieldSetMouseExit(fid,value)
FieldSetGotFocus	C	Disables or enables the field Get Focus event for the specified field using the values 0 or 1 respectively. FieldSetGotFocus (fid,value)
FieldSetLostFocus	C	Disables or enables the field Lose Focus event for the specified field using the values 0 or 1 respectively. FieldSetLostFocus (fid,value)
FieldGiveFocus	C	Gives the focus to the specified field. FieldGetFocus(fid)
FieldRemoveFocus	C	Removes the focus from the specified field. FieldRemoveFocus(fid)
FieldKeyDownFN	F	Returns the character from the key press. Put FieldKeyDownFN into key

Canvases

Canvas are the most versatile control in HyperNext because they can display graphics, images, receive mouse events, and also receive images through drag and drop.

A canvas control has a graphics area that can be drawn into and assigned an image. As canvases can receive mouse down events they can be used to make custom controls.

At runtime a canvas can be assigned an image either by copying it from another canvas or loading it from a file . Image manipulation can also be performed on a canvas and the results saved to a file. All canvases and their associated images are stored in 32 bit colour.

Colours have three components - red, green and blue.
Their values range from 0 to 255.
Black equals 0,0,0 and white equals 255,255,255

Name	Type	Description
CanvasIDFN	F	When used within the action handler of a canvas returns the numeric identity of that canvas. Put CanvasIDFN into cid
CanvasSetMode	C	Sets whether the canvas is enabled or not In the following bid is the canvas identity, and value determines whether the canvas will be enabled or disabled. When value is zero the canvas is disabled and when non zero it is enabled. CanvasSetMode bid,value
CanvasSetView	C	Sets whether the canvas is visible or not In the following bid is the canvas identity, and value determines whether the canvas will be in view or will be hidden. When value is zero the canvas is hidden and when non zero it is visible. CanvasSetView bid,value
CanvasModeFN	F	The value returned indicates whether the specified canvas is enabled or disabled.

		A return value of non zero indicates the canvas is enabled otherwise it is disabled. Put CanvasModeFN(3) into bokay
CanvasViewFN	F	The value returned indicates whether the specified canvas is visible or hidden. A return value of non zero indicates the canvas is visible otherwise it is hidden. Put CanvasViewFN(3) into bokay
CanvasDoFill	C	Fills an area in the current colour starting at the given point. CanvasDoFill id,x,y
CanvasDrawLine	C	Draws a line between 2 points in the current colour. CanvasDrawLine id,x1,y1,x2,y2
CanvasDrawOval	C	Draws an empty oval in the current colour. CanvasDrawOval id,x,y,width,height
CanvasDrawRect	C	Draws an empty rectangle in the current colour. CanvasDrawRect id,x,y,width,height
CanvasFillRect	C	Draws a filled rectangle in the current colour. CanvasFillRect id,x,y,width,height
CanvasFillOval	C	Draws a filled oval in the current color. CanvasFillOval canvas,x,y,width,height @ The following example draws 1000 filled ovals at random positions. @ 1000 circles Local n,x,y,xw,yh Local red,green,blue put CanvasWidthFN(1) into xw put CanvasHeightFN(1) into yh for n=1 to 1000 put RndFN(0,255) into red

		put RndFN(0,255) into green put RndFN(0,255) into blue CanvasSetColor 1,red,green,blue put RndFN(1,xw) into x put RndFN(1,yh) into y FillOval 1,x,y,10,10 endfor
CanvasPlot	C	Plots a point in the current color. CanvasPlot id,x,y
CanvasSetColor	C	Set the pen to the given color. CanvasSetColor id,red,green,blue
CanvasGetColor	C	Gets the current pen color. CanvasGetColor id,red,green,blue
CanvasGetPixelColor	C	Gets the color of the pixel at the specified point. CanvasGetPixelColor id,x,y,red,green,blue
CanvasSetPenHeight	C	Sets the height of the pen CanvasSetPenHeight id,value
CanvasSetPenWidth	C	Sets the width of the pen CanvasSetPenWidth id,value
CanvasSetBold	C	Sets the bold text style to on or off CanvasSetBold id,value
CanvasSetFont	C	Sets the font such as “Courier”, “System” etc CanvasSetFont id,fontname
CanvasSetFontSize	C	Sets the size for the current font. CanvasSetFontSize id,size
CanvasSetItalic	C	Sets the italic text style to on or off CanvasSetItalic id,value
CanvasSetUnderline	C	Sets the underline text style to on or off. CanvasSetUnderline id,value

CanvasDrawWrap	C	Sets the wrap CanvasDrawWrap id,value,x,y,wrap
CanvasDrawText	C	Draws text at the specified coordinates CanvasDrawText id,value,x,y
CanvasSetLeft	C	Sets the distance in pixels of the left side of the canvas from the left side of the card. CanvasSetLeft bid,value
CanvasSetTop	C	Sets the distance in pixels of the top of the canvas from the card top. CanvasSetTop bid,value
CanvasSetWidth	C	Sets the width of the canvas in pixels. CanvasSetWidth (bid,value)
CanvasSetHeight	C	Sets the height of the canvas in pixels. CanvasSetHeight (bid,value)
CanvasLeftFN	F	Returns the distance in pixels of the left side the canvas from the left side of the card. Put CanvasLeftFN(1) into num
CanvasTopFN	F	Returns the distance in pixels of the top of the canvas from the top of the card. Put CanvasTopFN(id) into num
CanvasWidthFN	F	Returns the width of the specified canvas. Put CanvasWidthFN(id) into x
CanvasHeightFN	F	Returns the height of the specified canvas. Put CanvasHeightFN(id) into x
CanvasTextWidthFN	F	Returns the width in pixels of the given text. Put CanvasTextWidthFN(id,txt) into x
CanvasTextHeightFN	F	Returns the height in pixels of the given text.

		Put CanvasTextHeightFN(id,txt) into x
CanvasMouseXFN	F	Returns the x coordinate of where the mouse down event took place. Put CanvasMouseXFN into x
CanvasMouseYFN	F	Returns the y coordinate of where the mouse down event took place. Put CanvasMouseYFN into y
CanvasScroll	C	Scrolls an area of the specified canvas. CanvasScroll(cid,dx,dy,x,y,w,h) where cid - canvas identity dx - amount of x scroll (+vew or -ve). dy - amount of y scroll (+vew or -ve). x - x origin or scroll area. y - y origin of scroll area. w - width of area to be scrolled. h - height of area to be scrolled.
CanvasClear	C	Clears the canvas using the specified color. CanvasClear(cid,red,green,blue)
CanvasSetDrop	C	Disables or enables the specified canvas from receiving a dropped image file. CanvasSetDrop(cid,value) @ disable for canvas 5 CanvasSetDrop(5,0) @ enable for canvas 8 CanvasSetDrop(8,1)

Canvas Events

In addition to the Mouse Down event a canvas can also receive several other events. However, in order to make programming easier for beginners most of these other events are disabled by default but can easily be enabled when a card (window) loads.

The event which triggered the canvas's handler can be found using the CanvasEvent function as shown in the example below and the event numbers are listed in the following table:

Example

```
Put CanvasEventFN(1) into evnum

If evnum=1 Then
   @ respond to Mouse Down
EndIf

If evnum=2 Then
   @ respond to Mouse Up
EndIf

Etc .....
```

Event	Description
1 - Mouse Down	The mouse button was pressed within the canvas.
2 - Mouse Up	The mouse button was released within the canvas.
3 - Mouse Drag	The mouse was dragged within the canvas.
4 – Mouse Move	The mouse moved within the canvas.
5 – Mouse Enter	The mouse entered the canvas.
6 – Mouse Exit	The mouse exited the canvas.
7 - Keydown	A key was pressed while the canvas had the focus.
8 – Got focus	The canvas received the focus.
9 – Lost focus	The canvas lost the focus.

Name	Type	Description
CanvasEventFN	F	Returns an integer specifying the canvas event which caused the canvas handler to be called.

		Put CanvasEventFN into evnum
CanvasSetMouseUp	C	Disables or enables the canvas MouseUp event for the specified canvas using the values 0 or 1 respectively. CanvasSetMouseUp cid,value
CanvasSetMouseDrag	C	Disables or enables the canvas MouseDrag event for the specified canvas using the values 0 or 1 respectively. CanvasSetMouseDrag cid,value
CanvasSetMouseMove	C	Disables or enables the canvas MouseMove event for the specified canvas using the values 0 or 1 respectively. CanvasSetMouseMove cid,value
CanvasSetMouseEnter	C	Disables or enables the canvas MouseEnter event for the specified canvas using the values 0 or 1 respectively. CanvasSetMouseEnter cid,value
CanvasSetMouseExit	C	Disables or enables the canvas MouseExit event for the specified canvas using the values 0 or 1 respectively. CanvasSetMouseExit cid,value
CanvasSetKeyDown	C	Disables or enables the canvas MouseSetKeyDown event for the specified canvas using the values 0 or 1 respectively. CanvasSetKeyDown cid,value
CanvasSetGotFocus	C	Disables or enables the canvas Get Focus event for the specified canvas using the values 0 or 1 respectively. CanvasSetGotFocus cid,value
CanvasSetLostFocus	C	Disables or enables the canvas Lose Focus event for the specified canvas using the values 0 or 1 respectively. CanvasSetLostFocus cid,value

CanvasGetFocus	C	Gives the focus to the specified canvas. CanvasGetFocus cid
CanvasRemoveFocus	C	Removes the focus from the specified canvas. CanvasRemoveFocus cid
CanvasKeyDownFN	F	Returns the character from the key press. Put CanvasKeyDownFN into key

Texts

A text control is simply a static text that is placed onto a card at design time. However, during runtime its value and other attributes can be changed.

Typical uses might be as a heading, a counter or some other indicator.

Although text control is usually used to display a static text such as a header or other information it can though be used dynamically, for instance to indicate a time or a counter. If you need a dynamic text then it is usually better to use a Text control rather than a Field control because Texts operate much more quickly and with less visible flashing.

The properties of a text can be set both from within the Creator/Developer or at runtime, properties such as font, font size, bold, italic, underline, and text colour.

Text properties set at design time

Property	Description
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the text.
Height	The height of the text.
Text	The text that a user will see at runtime.
Font	The name of the font used to display the text.
Fontsize	The size of the text.
Bold	Sets the bold either on or off.
Italic	Set the italic either on or off.
Underline	Sets the underline either on or off.
Align	Text alignment – left, center or right

Name	Type	Description
TextSetValue	C	This set the current text value where tid is the text identifier and value is the new text value. TextSetValue tid,value
TextSetMode	C	Sets whether the text control is enabled or not

		In the following tid is the text control identity, and value determines whether the text will be enabled or disabled. When value is zero the text is disabled and when non zero it is enabled. TextSetMode tid,value
TextSetView	C	Sets whether the text control is visible or not In the following bid is the text control identity, and value determines whether the text will be in view or will be hidden. When value is zero the text is hidden and when non zero it is visible. TextSetView tid,value
TextSetLeft	C	Sets the distance in pixels of the left side of the text control from the left side of the card. TextSetLeft tid,value
TextSetTop	C	Sets the distance in pixels of the top of the text control from the card top. TextSetTop tid,value
TextSetWidth	C	Sets the width of the text control in pixels. TextSetWidth tid,value
TextSetHeight	C	Sets the height of the text control in pixels. TextSetHeight tid,value
TextSetFont	C	Sets the fontname used to display the text within the text control. TextSetFont tid,fname
TextSetSize	C	Sets the size of the text within the text control. TextSetSize tid,value
TextSetAlign	C	Sets the alignment of the text within the text control. TextSetAlign tid,value Values can be

		1 - left 2 - center 3 – right
TextSetBold	C	Sets the bold attribute of the text control to either on or off - 0 is off, 1 is on. TextSetBold tid,value
TextSetItalic	C	Sets the italic attribute of the text control to either on or off - 0 is off, 1 is on. TextSetItalic tid,value
TextSetUnderline	C	Sets the underline attribute of the text control to either on or off - 0 is off, 1 is on. TextSetUnderline tid,value
TextSetColor	C	Sets the colour of the specified text. TextSetColor tid,r,g,b
TextModeFN	F	The value returned indicates whether the specified text control is enabled or disabled. A return value of non zero indicates the text is enabled otherwise it is disabled. Put TextModeFN(3) into bokay
TextViewFN	F	The value returned indicates whether the specified text control is visible or hidden. A return value of non zero indicates the text is visible otherwise it is hidden. Put TextViewFN(3) into bokay
TextLeftFN	F	Returns the distance in pixels of the left side the text control from the left side of the card. Put TextLeftFN(1) into num
TextTopFN	F	Returns the distance in pixels of the top of the text control from the top of the card. Put TextTopFN(1) into num

TextWidthFN	F	Returns the width of the text control. Put TextWidthFN(1) into num
TextHeightFN	F	Returns the height of the text control. Put TextHeightFN(1) into num
TextFontFN	F	Returns the font name in which the text is displayed. Put TextFontFN(1) into fname
TextSizeFN	F	Returns the size of the text. Put TextSizeFN(1) into num
TextAlignFN	F	Returns the alignment of the text. Values are: 1 - left 2 - center 3 - right 4 - default Put TextAlignFN(1) into num
TextBoldFN	F	Returns 1 if the text bold is on otherwise it returns 0. Put TextBoldFN(1) into num
TextItalicFN	F	Returns 1 if the text italic is on otherwise it returns 0. Put TextItalicFN(1) into num
TextUnderlineFN	F	Returns 1 if the text underline is on otherwise it returns 0. Put TextUnderlineFN(1) into num
TextValueFN	F	Returns the text displayed in the text control. Put TextValueFN(1) into txt

Check Boxes

A check box has two states and allows the user to initiate an action or set a state by merely clicking the check box.

Check boxes each have their own handler and their attributes can be changed during runtime.

Check Box properties set at design time

Property	Description
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the check box.
Height	The height of the check box.
Text	The text that a user will see at runtime.
Font	The name of the font used to display the text.
Fontsize	The size of the text.
Bold	Sets the bold either on or off.
Italic	Set the italic either on or off.
Underline	Sets the underline either on or off.
State	Whether the box is checked or not, 1 - checked, 0 = unchecked.

Name	Type	Description
CheckBoxSetMode	C	Sets whether the check box is enabled or not In the following bid is the checkbooks identity, and value determines whether the check box will be enabled or disabled. When value is zero the check box is disabled and when non zero it is enabled. CheckBoxSetMode bid,value
CheckBoxSetView	C	Sets whether the check Box is visible or not In the following bid is the check box identity, and value determines whether the check box will be in view or will be hidden. When value is zero

		the check box is hidden and when non zero it is visible. CheckBoxSetView bid,value
CheckBoxSetState	C	Sets whether the check Box is checked or not In the following bid is the check box identity, and value determines whether the check box will be checked or unchecked. When value is zero the check box is unchecked and when non zero it is checked. CheckBoxSetState bid,value
CheckBoxSetLeft	C	Sets the distance in pixels of the left side of the check box from the left side of the card. CheckBoxSetLeft bid,value
CheckBoxSetTop	C	Sets the distance in pixels of the top of the check box from the card top. CheckBoxSetTop bid,value
CheckBoxSetWidth	C	Sets the width of the check box in pixels. CheckBoxSetWidth bid,value
CheckBoxSetHeight	C	Sets the height of the check box in pixels. CheckBoxSetHeight bid,value
CheckBoxSetFont	C	Sets the font name used to display the text or caption within the check box. CheckBoxSetFont bid,fname
CheckBoxSetSize	C	Sets the size of the text or caption within the check box. CheckBoxSetSize bid,value
CheckBoxSetBold	C	Sets the bold attribute of the text or caption within the check box to either on or off - 0 is off, 1 is on. CheckBoxSetBold bid,value
CheckBoxSetItalic	C	Sets the italic attribute of the text or caption within the check box to either on or off - 0 is

		off, 1 is on. CheckBoxSetItalic bid,value
CheckBoxSetUnderline	C	Sets the underline attribute of the text or caption within the check box to either on or off - 0 is off, 1 is on. CheckBoxSetUnderline bid,value
CheckBoxSetText	C	Sets the text or caption within the check box. CheckBoxSetText bid,value
CheckBoxModeFN	F	The value returned indicates whether the specified check box is enabled or disabled. A return value of non zero indicates the check box is enabled otherwise it is disabled. Put CheckBoxModeFN(3) into bokay
CheckBoxViewFN	F	The value returned indicates whether the specified check box is visible or hidden. A return value of non zero indicates the check box is visible otherwise it is hidden. Put CheckBoxViewFN(3) into bokay
CheckBoxStateFN	F	The value returned indicates whether the specified check box is checked or unchecked. A return value of non zero indicates the check box is checked otherwise it is checked. Put CheckBoxStateFN(3) into bokay
CheckBoxLeftFN	F	Returns the distance in pixels of the left side of the check box from the left side of the card. Put CheckBoxLeftFN(1) into num
CheckBoxTopFN	F	Returns the distance in pixels of the top of the check box from the top of the card. Put CheckBoxTopFN(1) into num
CheckBoxWidthFN	F	Returns the width of the check box. Put CheckBoxWidthFN(1) into num

CheckBoxHeightFN	F	Returns the height of the check box. Put CheckBoxHeightFN(1) into num
CheckBoxFontFN	F	Returns the font name in which the check box text is displayed. Put CheckBoxFontFN(1) into fname
CheckBoxSizeFN	F	Returns the size of the check box text. Put CheckBoxSizeFN(1) into num
CheckBoxBoldFN	F	Returns 1 if the check box text bold is on otherwise it returns 0. Put CheckBoxBoldFN(1) into num
CheckBoxItalicFN	F	Returns 1 if the check box text italic is on otherwise it returns 0. Put CheckBoxItalicFN(1) into num
CheckBoxUnderlineFN	F	Returns 1 if the check box text underline is on otherwise it returns 0. Put CheckBoxUnderlineFN(1) into num
CheckBoxTextFN	F	Returns the text or caption displayed in the check box. Put CheckBoxTextFN(1) into txt

Radio Buttons

Sometimes referred to as option buttons, radio buttons allow the user to select one item in a group and when an item is selected the other group items are automatically deselected.
Radio buttons each have their own handler and their attributes can be changed during runtime.

Note,
Radio buttons are assigned their group number at design time and currently this cannot be changed at runtime.

Radio Button properties set at design time

Property	Description
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the radio button.
Height	The height of the radio button.
Text	The text that a user will see at runtime.
Font	The name of the font used to display the text.
Fontsize	The size of the text.
Bold	Sets the bold either on or off.
Italic	Set the italic either on or off.
Underline	Sets the underline either on or off.
State	Whether the radio button is selected or not, 1 - selected, 0 = deselected.

Name	Type	Description
RadioSetMode	C	Sets whether the radio button is enabled or not. In the following example, bid holds the radio button identity, and value determines whether the radio button will be enabled or disabled. When value is zero the radio button is disabled and when non zero it is enabled.

		RadioSetMode bid,value
RadioSetView	C	Sets whether the radio button will be visible or not. When value is zero the radio button is hidden and when non zero it is visible. RadioSetView bid,value
RadioSetState	C	Sets whether the radio button is selected or deselected. When value is zero the radio button is deselected and when non zero it is selected. RadioSetState bid,value
RadioSetLeft	C	Sets the distance in pixels of the left side of the radio button from the left side of the card. RadioSetLeft bid,value
RadioSetTop	C	Sets the distance in pixels of the top of the radio button from the card top. RadioSetTop bid,value
RadioSetWidth	C	Sets the width of the radio button in pixels. RadioSetWidth bid,value
RadioSetHeight	C	Sets the height of the radio button in pixels. RadioSetHeight bid,value
RadioSetFont	C	Sets the name of the font used to display the text or caption within the radio button. RadioSetFont bid,fname
RadioSetSize	C	Sets the font size of the text within the radio button. RadioSetSize bid,value
RadioSetBold	C	Sets the bold attribute of the text within the radio button to either on or off - 0 is off, 1 is on. RadioSetBold bid,value

RadioSetItalic	C	Sets the italic attribute of the text within the radio button to either on or off - 0 is off, 1 is on. RadioSetItalic bid,value
RadioSetUnderline	C	Sets the underline attribute of the text within the radio button to either on or off - 0 is off, 1 is on. RadioSetUnderline bid,value
RadioSetText	C	Sets the text or caption within the radio button. RadioSetText bid,value
RadioStateFN	F	The returned value indicates whether the specified radio button is selected or deselected. A return value of 1 indicates the radio button is selected and 0 that it is deselected. Put RadioModeFN(3) into bstate
RadioViewFN	F	The returned value indicates whether the specified radio button is visible or hidden. A return value of 1 indicates the radio button is visible and 0 that it is hidden. Put RadioViewFN(3) into bview
RadioStateFN	F	The returned value indicates whether the specified radio button is selected or deselected. A return value of 1 indicates the radio button is selected and 0 that it is deselected. Put RadioStateFN(3) into bokay
RadioLeftFN	F	Returns the distance in pixels of the left side of the specified radio button from the left side of the card. Put RadioLeftFN(1) into xnum
RadioTopFN	F	Returns the distance in pixels of the top of the

		specified radio button from the top of the card. Put RadioTopFN(1) into ynum
RadioWidthFN	F	Returns the width of the specified radio button. Put RadioWidthFN(1) into wnum
RadioHeightFN	F	Returns the height of the specified radio button. Put RadioHeightFN(1) into hnum
RadioFontFN	F	Returns the font name in which the specified radio button text is displayed. Put RadioFontFN(1) into fname
RadioSizeFN	F	Returns the size of the text in the specified radio button. Put RadioSizeFN(1) into tsize
RadioBoldFN	F	Returns 1 if the radio button text is bold and 0 if not. Put RadioBoldFN(1) into num
RadioItalicFN	F	Returns 1 if the radio button text is italic and 0 if not. Put RadioItalicFN(1) into num
RadioUnderlineFN	F	Returns 1 if the radio button text is underline and 0 if not. Put RadioUnderlineFN(1) into num
RadioTextFN	F	Returns the text displayed in the radio button. Put RadioTextFN(1) into txt
RadioPressedFN	F	Returns the numeric identity of the radio button last pressed. Put RadioPressedFN into whichbutton
RadioFindGroupFN	F	Returns the numeric identity of the group containing the specified radio button.

		Put RadioFindGroupFN(1) into groupnum
RadioListGroupFN	F	Returns a list of radio buttons contained within the specified group. Put RadioListGroupFN(1) into grouplist

Popup Menus

Pop up menus allow the user to select one item from many and are especially useful when a list of items is large as for instance when selecting a font name.

Pop up menus each have their own handler and their attributes can be changed during runtime.

Popup Menu properties set at design time

Property	Description
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the popup menu.
Height	The height of the popup menu.
Text	The text that a user will see at runtime.
Font	The name of the font used to display the text.
Fontsize	The size of the text.
Bold	Sets the bold either on or off.
Italic	Set the italic either on or off.
Underline	Sets the underline either on or off.
Value	The rows of text making up the pop up menu.

Name	Type	Description
PopupSetMode	C	Sets whether the pop up menu is enabled or not When value is 1 then the pop up menu is disabled and when 1 it is enabled. PopupSetMode bid,value
PopupSetView	C	Sets whether the pop up menu is visible or not. When value is 0 then the pop up menu is hidden and when 1 it is visible. PopupSetView bid,value
PopupSetIndex	C	Sets which pop up menu item is displayed where the value 1 is the first item in the list.

		PopupSetIndex bid,value
PopupsetItem	C	Sets the row containing the specified text value. PopupSetItem bid,value
PopupSetLeft	C	Sets the distance in pixels of the left side of the pop up menu from the left side of the card. PopupSetLeft bid,value
PopupSetTop	C	Sets the distance in pixels of the top of the pop up menu from the card top. PopupSetTop bid,value
PopupSetWidth	C	Sets the width of the pop up menu in pixels. PopupSetWidth bid,value
PopupSetHeight	C	Sets the height of the pop up menu in pixels. PopupSetHeight bid,value
PopupSetFont	C	Sets the font used to display the text within the pop up menu. PopupSetFont bid,fname
PopupSetSize	C	Sets the size of the text within the pop up menu. PopupSetSize bid,value
PopupSetBold	C	Sets the bold attribute of the pop up menu text to either on or off - 0 is off, 1 is on. PopupSetBold bid,value
PopupSetItalic	C	Sets the italic attribute of the pop up menu text to either on or off - 0 is off, 1 is on. PopupSetItalic bid,value
PopupSetUnderline	C	Sets the underline attribute of the pop up menu text either on or off - 0 is off, 1 is on. PopupSetUnderline bid,value
PopupSetText	C	Sets the whole text of the pop up menu. The text should be in list form with each line representing one

		item in the pop up menu. PopupSetText bid,value
PopupModeFN	F	The returned value indicates whether the specified pop up menu is enabled or disabled. A return value of 1 indicates the pop up menu is enabled and 0 that it is disabled. Put PopupModeFN(3) into bokay
PopupViewFN	F	The returned value indicates whether the specified pop up menu is visible or hidden. A return value of 1 indicates the pop up menu is visible and 0 that it is hidden. Put PopupViewFN(3) into bokay
PopupIndexFN	F	The returned value indicates which pop up menu item is currently displayed. Put PopupIndexFN(3) into bokay
PopupLeftFN	F	Returns the distance in pixels of the left side of the pop up menu from the left side of the card. Put PopupLeftFN(1) into num
PopupTopFN	F	Returns the distance in pixels of the top of the pop up menu from the top of the card. Put PopupTopFN(1) into num
PopupWidthFN	F	Returns the width of the pop up menu. Put PopupWidthFN(1) into num
PopupHeightFN	F	Returns the height of the pop up menu. Put PopupHeightFN(1) into num
PopupFontFN	F	Returns the name of the font in which the pop up menu text is displayed. Put PopupFontFN(1) into fname
PopupSizeFN	F	Returns the size of the pop up menu text. Put PopupSizeFN(1) into num

PopupBoldFN	F	Returns 1 if the pop up menu text bold is on otherwise it returns 0. Put PopupBoldFN(1) into num
PopupItalicFN	F	Returns 1 if the pop up menu text is italicized otherwise it returns 0. Put PopupItalicFN(1) into num
PopupUnderlineFN	F	Returns 1 if the pop up menu text is underlined otherwise it returns 0. Put PopupUnderlineFN(1) into num
PopupTextFN	F	Returns as a list the complete text displayed in the pop up menu. Put PopupTextFN(1) into txt
PopupItemFN	F	Returns the item currently selected for the specified pop up menu. Put PopupItemFN(1) into txt

Scroll Bars

A text control is simply a static text that is placed onto a card at design time. However, during runtime its value and other attributes can be changed.

Typical uses might be as a heading, a counter or some other indicator.

Although text control is usually used to display a static text such as a header or other information it can though be used dynamically, for instance to indicate a time or a counter. If you need a dynamic text then it is usually better to use a Text control rather than a Field control because Texts operate much more quickly and with less visible flashing.

The properties of a text can be set both from within the Creator/Developer or at runtime, properties such as font, font size, bold, italic, underline, and text colour.

Scroll Bar properties set at design time

Property	Description
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the text.
Height	The height of the text.

Name	Type	Description
ScrollbarSetValue	C	This set the current text value where tid is the text identifier and value is the new text value. ScrollbarSetValue tid,value
ScrollbarSetMode	C	Sets whether the text control is enabled or not In the following tid is the text control identity, and value determines whether the text will be enabled or disabled. When value is zero the text is disabled and when non zero it is enabled. ScrollbarSetMode tid,value

ScrollbarSetView	C	Sets whether the text control is visible or not In the following bid is the text control identity, and value determines whether the text will be in view or will be hidden. When value is zero the text is hidden and when non zero it is visible. ScrollbarSetView tid,value
ScrollbarSetLeft	C	Sets the distance in pixels of the left side of the text control from the left side of the card. ScrollbarSetLeft tid,value
ScrollbarSetTop	C	Sets the distance in pixels of the top of the text control from the card top. ScrollbarSetTop tid,value
ScrollbarSetWidth	C	Sets the width of the text control in pixels. ScrollbarSetWidth tid,value
ScrollbarSetHeight	C	Sets the height of the text control in pixels. ScrollbarSetHeight tid,value
TextSetFont	C	Sets the fontname used to display the text within the text control. TextSetFont tid,fname
TextSetSize	C	Sets the size of the text within the text control. TextSetSize tid,value
TextSetAlign	C	Sets the alignment of the text within the text control. TextSetAlign tid,value Values can be 1 - left 2 - center 3 – right
TextSetBold	C	Sets the bold attribute of the text control to

		either on or off - 0 is off, 1 is on. TextSetBold tid,value
TextSetItalic	C	Sets the italic attribute of the text control to either on or off - 0 is off, 1 is on. TextSetItalic tid,value
TextSetUnderline	C	Sets the underline attribute of the text control to either on or off - 0 is off, 1 is on. TextSetUnderline tid,value
TextSetColor	C	Sets the colour of the specified text. TextSetColor tid,r,g,b
TextModeFN	F	The value returned indicates whether the specified text control is enabled or disabled. A return value of non zero indicates the text is enabled otherwise it is disabled. Put TextModeFN(3) into bokay
TextViewFN	F	The value returned indicates whether the specified text control is visible or hidden. A return value of non zero indicates the text is visible otherwise it is hidden. Put TextViewFN(3) into bokay
TextLeftFN	F	Returns the distance in pixels of the left side the text control from the left side of the card. Put TextLeftFN(1) into num
TextTopFN	F	Returns the distance in pixels of the top of the text control from the top of the card. Put TextTopFN(1) into num
TextWidthFN	F	Returns the width of the text control. Put TextWidthFN(1) into num
TextHeightFN	F	Returns the height of the text control. Put TextHeightFN(1) into num

TextFontFN	F	Returns the font name in which the text is displayed. Put TextFontFN(1) into fname
TextSizeFN	F	Returns the size of the text. Put TextSizeFN(1) into num
TextAlignFN	F	Returns the alignment of the text. Values are: 1 - left 2 - center 3 - right 4 - default Put TextAlignFN(1) into num
TextBoldFN	F	Returns 1 if the text bold is on otherwise it returns 0. Put TextBoldFN(1) into num
TextItalicFN	F	Returns 1 if the text italic is on otherwise it returns 0. Put TextItalicFN(1) into num
TextUnderlineFN	F	Returns 1 if the text underline is on otherwise it returns 0. Put TextUnderlineFN(1) into num
TextValueFN	F	Returns the text displayed in the text control. Put TextValueFN(1) into txt

Movies

A movie control can be placed onto a card and at design time assigned a movie from the library. Alternatively at runtime your users can simply drag and drop a movie onto the movie control or assign one from a file. Movie controls can play movies. MP3s and other sounds.

Currently only two movie controls can be assigned to each card.
One movie can be visible to users and can be used to play the movie, MP3 or other media. The second movie is often hidden and used to open media files and find their properties before passing on the media file to the visible movie control.

Movie properties set at design time

Property	Description
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the movie.
Height	The height of the movie.

Name	Type	Description
MovieSetMode	C	Sets whether the movie is enabled or not In the following bid is the movie control identity, and value determines whether the movie will be enabled or disabled. When value is zero the movie is disabled and when non zero it is enabled. MovieSetMode(bid,value)
MovieSetView	C	Sets whether the movie is visible or not In the following bid is the movie control identity, and value determines whether the movie will be in view or will be hidden. When value is zero the movie is hidden and when non zero it is visible. MovieSetView(bid,value)
MovieSetLeft	C	Sets the distance in pixels of the left side of the movie from the left side of the card.

		MovieSetLeft(bid,value)
MovieSetTop	C	Sets the distance in pixels of the top of the movie from the card top. MovieSetTop(bid,value)
MovieSetWidth	C	Sets the width of the movie in pixels. MovieSetWidth(bid,value)
MovieSetHeight	C	Sets the height of the movie in pixels. MovieSetHeight(bid,value)
MovieModeFN	F	The value returned indicates whether the specified movie is enabled or disabled. A return value of non zero indicates the movie is enabled otherwise it is disabled. Put MovieModeFN(3) into bokay
MovieViewFN	F	The value returned indicates whether the specified movie is visible or hidden. A return value of non zero indicates the movie is visible otherwise it is hidden. Put MovieViewFN(3) into bokay
MovieLeftFN	F	Returns the distance in pixels of the left side the movie from the left side of the card. Put MovieLeftFN(1) into num
MovieTopFN	F	Returns the distance in pixels of the top of the movie from the top of the card. Put MovieTopBFN(1) into num
MovieWidthFN	F	Returns the width of the movie control. Put MovieWidthFN(1) into num
MovieHeightFN	F	Returns the height of the movie control. Put MovieHeightFN(1) into num
MovieSetControls	C	Sets the type of controls shown on the movie

		controller. MovieSetControls(bid,value) Where value = 0 - no controls 1 - badge 2 - full controller
MovieSetSpeaker	C	Sets whether the movie volume control is visible or not. It takes effect the next time the movie control is assigned a movie or sound. MovieSetSpeaker(bid,value)
MovieSetFile	C	Tries to open a movie or sound file and attach it to the movie control. The function MovieOkayFN can be used to se if the command was successful or not. The filename refers to a local file or specifies the path to a local file. MovieSetFile(bid,fname)
MovieSetFileAbs	C	Tries to open a movie or sound file and attach it to the movie control. The function MovieOkayFN can be used to se if the command was successful or not. The filename specifies the absolute path to the file. MovieSetFileAbs(bid,fname)
MovieSetLooping	C	Sets whether the movie will loop or not. MovieSetLooping(bid,value)
MovieSetStep	C	Sets whether or not the movie has forward/reverse arrows. It takes effect the next time the movie control is assigned a movie or sound. MovieSetStep(bid,value)
MovieSetVolume	C	Sets the volume of the movie in the range 0 to 255. MovieSetVolume(bid,value)
MoviePlayReverse	C	Plays the movie in reverse. MoviePlayReverse(bid)

MoviePlayHalfSpeed	C	Plays the movie at half normal speed. MoviePlayHalfSpeed(bid)
MoviePlayNormalSpeed	C	Plays the movie at normal speed. MoviePlayNormalSpeed(bid)
MoviePlayDoubleSpeed	C	Plays the movie at twice normal speed. MoviePlayDoubleSpeed(bid)
MoviePause	C	Pauses(stops) the movie. MoviePause(bid)
MoviePoll	C	This polls the movie and is often needed on the Windows platform to ensure smooth playback. MoviePoll(bid)
MovieVolumeFN	F	Returns the volume of the movie. The volume ranges from 0 to 255. Put MovieVolumeFN(1) into num
MoviePositionFN	F	Returns the current position in seconds of the movie or sound. Put MoviePositionFN(1) into num
MovieLengthFN	F	Returns the length of the movie or sound in seconds. Put MovieLengthFN(1) into num
MovieIsPlayingFN	F	Returns whether the movie or sound is playing or has stopped (paused). Returns 1 when playing and 0 when stopped. Put MovieIsPlayingFN(1) into okay
MovieBaseWidthFN	F	Returns the actual width of the movie in pixels. Put MovieBaseWidthFN(1) into num
MovieBaseHeightFN	F	Returns the actual height of the movie in pixels. Put MovieBaseHeightFN(1) into num
MovieOkayFN	F	Returns whether the selected movie is a valid

		movie or media file. Put MovieOkayFN(1) into okay

Listboxes

Listboxes are used to display one or more columns of information. Their contents can be sorted and items can have checkboxes. Rows and cells are numbered from 1 upwards and not from 0 as in many other programming languages. Note, the maximum number of columns visible is 64.

Listboxes each have their own action handler and can respond to many events. See the function ListboxEventFN for more details.

Name	Type	Description
CanvasIDFN	F	When used within the action handler of a canvas returns the numeric identity of that canvas. Put CanvasIDFN into cid
ListboxSetMode	C	Sets whether the canvas is enabled or not In the following bid is the canvas identity, and value determines whether the canvas will be enabled or disabled. When value is zero the canvas is disabled and when non zero it is enabled. ListboxSetMode bid,value
ListboxSetView	C	Sets whether the canvas is visible or not In the following bid is the canvas identity, and value determines whether the canvas will be in view or will be hidden. When value is zero the canvas is hidden and when non zero it is visible. ListboxSetView bid,value
ListboxModeFN	F	The value returned indicates whether the specified canvas is enabled or disabled. A return value of non zero indicates the canvas is enabled otherwise it is disabled. Put ListboxModeFN(3) into bokay
ListboxViewFN	F	The value returned indicates whether the

		specified canvas is visible or hidden. A return value of non zero indicates the canvas is visible otherwise it is hidden. Put ListboxViewFN(3) into bokay
ListboxSetLeft	C	Sets the distance in pixels of the left side of the canvas from the left side of the card. ListboxSetLeft bid,value
ListboxSetTop	C	Sets the distance in pixels of the top of the canvas from the card top. ListboxSetTop bid,value
ListboxSetWidth	C	Sets the width of the canvas in pixels. ListboxSetWidth bid,value
ListboxSetHeight	C	Sets the height of the canvas in pixels. ListboxSetHeight bid,value
ListboxLeftFN	F	Returns the distance in pixels of the left side the canvas from the left side of the card. Put ListboxLeftFN(1) into num
ListboxTopFN	F	Returns the distance in pixels of the top of the canvas from the top of the card. Put ListboxTopFN(id) into num
ListboxWidthFN	F	Returns the width of the specified canvas. Put ListboxWidthFN(id) into x
ListboxHeightFN	F	Returns the height of the specified canvas. Put ListboxHeightFN(id) into x
ListboxSetTextFont	C	Sets the text font for the specified listbox. ListboxSetTextFont(lbox,fontname)
ListboxTextFontFN	F	Returns the text font name for the specified listbox. ListboxTextFontFN(lbox)

		Put ListboxTextFontFN(5) into fontname
ListboxSetTextSize	C	Sets the text size for the specified listbox. ListboxSetTextSize(lbox,textsize)
ListboxTextSizeFN	F	Returns the text size for the specified listbox. ListboxTextSizeFN(lbox) Put ListboxTextSizeFN(5) into tsize

Listbox Events

In addition to the Mouse Down event a canvas can also receive several other events. However, in order to make programming easier for beginners most of these other events are disabled by default but can easily be enabled when a card (window) loads.

The event which triggered the canvas's handler can be found using the CanvasEvent function as shown in the example below and the event numbers are listed in the following table:

Example

```
Put ListboxEventFN(1) into evnum

If evnum=6 Then
  @ respond to file drop
EndIf

If evnum=7 Then
  @ respond to double click
EndIf

Etc .....
```

Event	Description
1 – Cell action	A cell was edited or a cell checkbox was clicked
2 – Cell clicked	A cell received a mouse click.
3 – DragReorder rows	A row was dragged into another position.
4 – Drag row	A row was dragged out of the listbox.
5 – Drop drag	A text was dragged and dropped onto the listbox.
6 – Drop file	A file/folder or files were dropped onto the listbox.
7 – Double click	A mouse double click occured.
8 – Header Pressed	A column within the header was pressed.
9 – Cell keydown	A cell received a key down.
10 – Cell got focus	A cell received the focus.
11 – Cell text change	The text changed within a cell.
12 – Cell lost focus	A cell lost the focus.
13 – Sort column	A column was sorted

Name	Type	Description
ListboxBuild	C	Returns an integer specifying the canvas event which caused the canvas handler to be called. Put CanvasEventFN into evnum
ListboxSetColumns	C	Sets the number of columns in the listbox. ListboxSetColumns bid,ncols
ListboxSetRows	C	Sets the number of rows in the listbox. ListboxSetRows bid,nrows
ListboxDeleteAll	C	Deletes all the rows in columns in the listbox. ListboxDeleteAll bid
ListboxAddRow	C	Adds a row with the given value being placed in the first cell of the row. ListboxAddRow bid,value
ListboxInsertRow	C	Inserts a row before the given position and places the given value in the first cell of the row. ListboxInsertRow bid,row,value
ListboxDeleteRow	C	Deletes the specified row from the listbox. ListboxDeleteRow bid,row
ListboxSetWidths	C	Sets the widths of the columns. The value passed can have a number of formats with columns widths being separated by commas. The width values can be in pixels as in value = 125,56,92 for three columns In percentages as in value = 25%,35%,40% for three columns. If not enough columns widths are passed then the remaining columns are evenly spaced. ListboxSetWidths bid,value

ListboxSetRowHeight	C	Sets the default height of the rows in pixels. ListboxSetRowHeight bid,value
ListboxAllowHeading	C	Allows the listbox to have or not have a header. ListboxAllowHeading bid,value
ListboxSetHeading	C	Sets an individual column header to the text in value. ListboxSetHeading bid,col,value
ListboxPressHeader	C	Causes the specified column of the header to be pressed. ListboxPressHeader bid,col
ListboxRefresh	C	Refreshes or redraws the specified cell. ListboxRefresh bid,row,col
ListboxFill	C	Fills the listbox with the value given. This can also be used to clear the listbox. ListboxFill bid,value
ListboxSetCellValue	C	Sets the value of the specified cell. ListboxSetCellValue bid,row,col,value
ListboxSetCellBold	C	Sets the text format of the specified cell to bold or non-bold . ListboxSetCellBold bid,row,col,value
ListboxSetCellItalic	C	Sets the text format of the specified cell to italic or non-italic. ListboxSetCellItalic bid,row,col,value
ListboxSetCellUnderline	C	Sets the text format of the specified cell to underline or non-underline. ListboxSetCellUnderline bid,row,col,value
ListboxSetCellType	C	Sets the type of the specified cell.

		ListboxSetCellType bid,row,col,type type can be 0 - default 1 - normal 2 - checkbox 3 - editable.
ListboxSetCellAlign	C	Sets the text alignment of the specified cell. ListboxSetCellAlign bid,row,col,align align can be 0 - default 1 - left 2 - centre 3 - right 4 – decimal
ListboxSetCellOffset	C	Sets the decimal offset of the specified cell. The offset is measured in pixels from the right edge. ListboxSetCellOffset bid,row,col,offset
ListboxSetCellCheck	C	Sets the checkbox value of the specified cell. ListboxSetCellCheck bid,row,col,value
ListboxSetCellTag	C	Sets the tag value of the specified cell. A tag is a value not visible to the user that can be used for many purposes included categorising, counting and sorting. ListboxSetCellTag bid,row,col,value
ListboxSetCellEdit	C	Sets whether the specified cell is editable or not. ListboxSetCellEdit bid,row,col,value
ListboxSetCellBorderLeft	C	Sets the left border type of the specified cell. ListboxSetCellBorderLeft bid,row,col,value value can be 0 - default 1 - none

		2 - thin dotted 3 - thin solid 4 - thick solid 5 - double thin solid
ListboxSetCellBorderRight	C	Sets the right border type of the specified cell. ListboxSetCellBorderRight bid,row,col,value
ListboxSetCellBorderTop	C	Sets the top border type of the specified cell. ListboxSetCellBorderTop bid,row,col,value
ListboxSetCellBorderBottom	C	Sets the bottom border type of the specified cell. ListboxSetCellBorderBottom bid,row,col,value
ListboxSetColumnType	C	Sets the type of the specified column. ListboxSetColumnType bid,col,type type can be 0 - default 1 - normal 2 - checkbox 3 - editable.
ListboxSetColumnAlign	C	Sets the text alignment of the specified column. ListboxSetColumnAlign bid,col,align align can be 0 - default 1 - left 2 - centre 3 - right 4 – decimal
ListboxSetColumnOffset	C	Sets the decimal offset of the specified cell. The offset is measured in pixels from the right edge. ListboxSetColumnOffset bid,col,offset

ListboxSetScrollHorizontal	C	Sets whether or not the listbox has a horizontal scroll bar. ListboxSetScrollHorizontal bid,value
ListboxSetScrollVertical	C	Sets whether or not the listbox has a vertical scroll bar. ListboxSetScrollVertical bid,value
ListboxSetGridHorizontal	C	Sets the horizontal grid for the listbox. ListboxSetGridHorizontal bid,value value can be 0 - default 1 - none 2 - thin dotted 3 - thin solid 4 - thick solid 5 - double thin solid
ListboxSetGridVertical	C	Sets the vertical grid for the listbox where value has the same attributes as the horizontal grid. ListboxSetGridVertical bid,value
ListboxSetColumnSort	C	Sets sort direction of the specified column in the listbox. ListboxSetColumnSort bid,col,direct direct can be 0 - do not sort 1 - descending 2 – ascending
ListboxSort	C	Sort the listbox. ListboxSort bid
ListboxSetDragReOrder	C	Specifies whether or not the rows of the listbox can be re-ordered by dragging them. ListboxSetDragReOrder bid,value
ListboxSetDrag	C	Specifies whether or not the row of the listbox can be dragged out of it. ListboxSetDrag bid,value

ListboxSetFileDrop	C	Specifies whether or not the listbox can accept files dropped onto it. ListboxSetFileDrop bid,value
ListboxSetDragDrop	C	Specifies whether or not the listbox can accept text dropped onto it from other controls. ListboxSetDragDrop bid,value
ListboxHighlightRow	C	Highlights the specified row of the listbox. ListboxHighlightRow bid,row

ListboxEventFN	F	Returns the event that occurred in the listbox.
		
		Put ListboxEventFN(1) into num
		@ check for file dropped
		If num=6 Then
		DoSomething
		EndIf
		
		values are
		1 - Cell action either editable cell or checkbox clicked.
		2 - Cell clicked.
		3 - Drag reorded rows.
		4 - Drag row.
		5 - Drop drag.
		6 - Drop file.
		7 - Double click.
		8 - Header pressed
		9 - Cell key down.
		10 - Cell got focus.
		11 - Cell text change.
		12 - Cell lost focus.
		13 - Sort column.

ListboxColumnsFN	F	Returns the number of columns in the listbox. Put ListboxColumnsFN(1) into num
ListboxRowsFN	F	Returns the number of rows in the listbox. Put ListboxRowsFN(1) into num
ListboxLastIndexFN	F	Returns the number of the last row added

		or inserted. Put ListboxLastIndexFN(1) into num
ListboxIndexFN	F	Returns the number of the row currently selected. Put ListboxIndexFN(1) into num
ListboxCellValueFN	F	Returns the value of the specified cell. Put ListboxCellValueFN(1,row,col) into num
ListboxCellTypeFN	F	Returns the type of the specified cell. Put ListboxCellTypeFN(1,row,col) into num
ListboxCellTagFN	F	Returns the tag value of the specified cell. Put ListboxCellTagFN(1,row,col) into num $S ListboxCellCheckFN Returns the checked value of the specified cell.. Put ListboxCellCheckFN(1,row,col) into num
ListboxRowListFN	F	Returns the cell values of the specified row as a list. Put ListboxRowListFN(1,row) into num
ListboxColumnListFN	F	Returns the cell values of the specified column as a list. Put ListboxColumnListFN(1,col) into num
ListboxScrollPositionFN	F	Returns the row number of the top row visible. Put ListboxScrollPositionFN(1) into num
ListboxScrollXPositionFN	F	Returns the horizontal scroll thumb position for the listbox. Put ListboxScrollXPositionFN(1) into num

ListboxDragReOrderFN	F	Returns whether the listbox allows drag and reorder of rows. Put ListboxDragReOrderFN(1) into num
ListboxDragFN	F	Returns whether the listbox allows dragging of rows out of it. Put ListboxDragFN(1) into num
ListboxDroppedStateFN	F	Returns the state of the listbox drop. Put ListboxDroppedStateFN(1) into state where state can be 0 - nothing dropped. 1 - text dropped from another control. 2 - one or more files or folders dropped.
ListboxDroppedSourceTypeFN	F	Returns the source type of the drop. Put ListboxDroppedSourceTypeFN(1) into num
ListboxDroppedSourceIDFN	F	Returns the index of the dropped source object. Put ListboxDroppedSourceIDFN(1) into num
ListboxDroppedTextFN	F	Returns actual dropped text. Put ListboxDroppedTextFN(1) into num
ListboxDroppedCountFN	F	Returns the number of objects dropped on the listbox. Put ListboxDroppedCountFN(1) into num
ListboxDroppedRowFN	F	Returns the index of the row being dragged. Put ListboxDroppedRowFN(1) into num
ListboxFileCountFN	F	Returns the number of file or folders dropped onto the listbox. Put ListboxFileCountFN(1) into fcount
ListboxFilePathsFN	F	Returns a path list of all the files dropped onto the listbox .

		Put ListboxFilePathsFN(1) into plist
ListboxFileNamesFN	F	Returns a name list of all the files dropped onto the listbox . Put ListboxFileNamesFN(1) into flist
ListboxFileTypesFN	F	Returns a type list of all the files dropped onto the listbox. The type can be 0 for folder and 1 for file. Put ListboxFileTypesFN(1) into tlist
ListboxFileExtensionsFN	F	Returns an extension list of all the files dropped onto the listbox. This can be useful in filtering file types, for instance when deciding which files might be playable by a movie control. Put ListboxFileExtensionsFN(1) into elist
ListboxRowClickedFN	F	Returns the index number of the row clicked. Put ListboxRowClickedFN(1) into num
ListboxColumnClickedFN	F	Returns the index of the column clicked. Put ListboxColumnClickedFN(1) into num
ListboxKeyDownFN	F	Returns the keydown character. Put ListboxKeyDownFN(1) into num
ListboxLostFocusRowFN	F	Returns the number of the row which just lost focus. Put ListboxLostFocusRowFN(1) into num
ListboxLostFocusColumnFN	F	Returns the number of the column which just lost focus. Put ListboxLostFocusColumnFN(1) into num
ListboxColumnSortedFN	F	Returns the number of the column which was sorted. Put ListboxColumnSortedFN(1) into num

Sliders

Sliders are similar to scrollbars except they do not have end arrows and are usually just dragged to change their output value in a more abrupt fashion.

Sliders each have their own action handler and can have many of their attributes changed during runtime as listed below.

Slider properties set at design time

Property	Description
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the slider.
Height	The height of the slider.

Name	Type	Description
SliderSetMode	C	Sets whether the slider is enabled or not In the following bid is the slider control identity, and value determines whether the slider will be enabled or disabled. When value is zero the slider is disabled and when non zero it is enabled. SliderSetMode bid,value
SliderSetView	C	Sets whether the slider is visible or not In the following bid is the slider control identity, and value determines whether the slider will be in view or will be hidden. When value is zero the slider is hidden and when non zero it is visible. SliderSetView bid,value
SliderSetLeft	C	Sets the distance in pixels of the left side of the slider from the left side of the card. SliderSetLeft bid,value
SliderSetTop	C	Sets the distance in pixels of the top of the slider

		from the card top. SliderSetTop bid,value
SliderSetWidth	C	Sets the width of the slider in pixels. SliderSetWidth bid,value
SliderSetHeight	C	Sets the height of the slider in pixels. SliderSetHeight bid,value
SliderModeFN	F	The value returned indicates whether the specified slider is enabled or disabled. A return value of non zero indicates the slider is enabled otherwise it is disabled. Put SliderModeFN(3) into bokay
SliderViewFN	F	The value returned indicates whether the specified slider is visible or hidden. A return value of non zero indicates the slider is visible otherwise it is hidden. Put SliderViewFN(3) into bokay
SliderLeftFN	F	Returns the distance in pixels of the left side the slider from the left side of the card. Put SliderLeftFN(1) into num
SliderTopFN	F	Returns the distance in pixels of the top of the slider from the top of the card. Put SliderTopBFN(1) into num
SliderWidthFN	F	Returns the width of the slider control. Put SliderWidthFN(1) into num
SliderHeightFN	F	Returns the height of the slider control. Put SliderHeightFN(1) into num
SliderSetMinValue	C	Sets the minimum value which the slider can report or be set to. SliderSetMinValue bid,value

SliderSetMaxValue	C	Sets the maximum value which the slider can report or be set to. SliderSetMaxValue bid,value
SliderSetValue	C	Sets the slider position and value. SliderSetValue bid,value
SliderSetLineStep	C	On the Windows platform sets the step value by which the slider moves when the slider arrows are moved. On the Macintosh platform it does nothing. SliderSeLineStep bid,value
SliderSetPageStep	C	The amount by which the slider value will change when the slider track is clicked. SliderSetPageStep bid,value
SliderSetLiveScroll	C	Sets whether the slider will change its value in realtime when moved. SliderSetLiveScroll bid,value
SliderMinValueFN	F	Returns the minimum value setting of the slider. Put SliderMinValueFN(1) into num
SliderMaxValueFN	F	Returns the maximum value setting of the slider. Put SliderMaxValueFN(1) into num
SliderValueFN	F	Returns the position value setting of the slider. Put SliderValueFN(1) into num
SliderLineStepFN	F	Returns the line step value setting of the slider. Put SliderLineStepFN(1) into num
SliderPageStepFN	F	Returns the page step value setting of the slider. Put SliderPageStepFN(1) into num
SliderLiveScrollFN	F	Returns whether the slider is set to live scroll or not. Put SliderLiveScrollFN(1) into value

Scrollbars

Scrollbars allow the user to control the position or value of some other object such as when scrolling continuous text or a picture, or changing a numeric value.
HyperNext supports both horizontal and vertical scrollbars.
Scrollbars each have their own action handler and can have many of their attributes changed during runtime as listed below.

Scrollbar properties set at design time

Property	Description
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the scrollbar.
Height	The height of the scrollbar.

Name	Type	Description
ScrollbarSetMode	C	Sets whether the scrollbar is enabled or not In the following bid is the scrollbar control identity, and value determines whether the scrollbar will be enabled or disabled. When value is zero the scrollbar is disabled and when non zero it is enabled. ScrollbarSetMode bid,value
ScrollbarSetView	C	Sets whether the scrollbar is visible or not In the following bid is the scrollbar control identity, and value determines whether the scrollbar will be in view or will be hidden. When value is zero the scrollbar is hidden and when non zero it is visible. ScrollbarSetView bid,value
ScrollbarSetLeft	C	Sets the distance in pixels of the left side of the scrollbar from the left side of the card. ScrollbarSetLeft bid,value

ScrollbarSetTop	C	Sets the distance in pixels of the top of the scrollbar from the card top. ScrollbarSetTop bid,value
ScrollbarSetWidth	C	Sets the width of the scrollbar in pixels. ScrollbarSetWidth bid,value
ScrollbarSetHeight	C	Sets the height of the scrollbar in pixels. ScrollbarSetHeight bid,value
ScrollbarModeFN	F	The value returned indicates whether the specified scrollbar is enabled or disabled. A return value of non zero indicates the scrollbar is enabled otherwise it is disabled. Put ScrollbarModeFN(3) into bokay
ScrollbarViewFN	F	The value returned indicates whether the specified scrollbar is visible or hidden. A return value of non zero indicates the scrollbar is visible otherwise it is hidden. Put ScrollbarViewFN(3) into bokay
ScrollbarLeftFN	F	Returns the distance in pixels of the left side the scrollbar from the left side of the card. Put ScrollbarLeftFN(1) into num
ScrollbarTopFN	F	Returns the distance in pixels of the top of the scrollbar from the top of the card. Put ScrollbarTopBFN(1) into num
ScrollbarWidthFN	F	Returns the width of the scrollbar control. Put ScrollbarWidthFN(1) into num
ScrollbarHeightFN	F	Returns the height of the scrollbar control. Put ScrollbarHeightFN(1) into num
ScrollbarSetMinValue	C	Sets the minimum value which the scrollbar can report or be set to.

		ScrollbarSetMinValue bid,value
ScrollbarSetMaxValue	C	Sets the maximum value which the scrollbar can report or be set to. ScrollbarSetMaxValue bid,value
ScrollbarSetValue	C	Sets the scrollbar position and value. ScrollbarSetValue bid,value
ScrollbarSetLineStep	C	Sets the amount by which the scroll bar position will be incremented or decremented when the arrows are clicked. ScrollbarSetLineStep bid,value
ScrollbarSetPageStep	C	The amount by which the scrollbar value will change when the scrollbar track is clicked. ScrollbarSetPageStep bid,value
ScrollbarSetLiveScroll	C	Sets whether the scrollbar will change its value in realtime when moved. ScrollbarSetLiveScroll bid,value
ScrollbarMinValueFN	F	Returns the minimum value setting of the scrollbar. Put ScrollbarMinValueFN(1) into num
ScrollbarMaxValueFN	F	Returns the maximum value setting of the scrollbar. Put ScrollbarMaxValueFN(1) into num
ScrollbarValueFN	F	Returns the position value setting of the scrollbar. Put ScrollbarValueFN(1) into num
ScrollbarLineStepFN	F	Returns the line step value setting of the scrollbar. Put ScrollbarLineStepFN(1) into num
ScrollbarPageStepFN	F	Returns the page step value setting of the scrollbar. Put ScrollbarPageStepFN(1) into num
ScrollbarLiveScrollFN	F	Returns whether the scrollbar is set to live scroll or not.

		Put ScrollbarLiveScrollFN(1) into value

Progress Bars

Progress bars give a visual indication of a tasks progress and can also be clicked on to change their value.

Setting their maximum value to 0 causes the progress bar to display a Barber Pole and this is suitable when a task' s state is indeterminate

Progress bars each have their own action handler and can have many of their attributes changed during runtime as listed below.

Progress bar properties set at design time	
Property	**Description**
Name	Currently this is not used at runtime.
Left	The position in pixels from the left side of the card.
Top	The position in pixels from the top of the card.
Width	The width of the progress bar.
Height	The height of the progress bar.

Name	Type	Description
ProgressbarSetMode	C	Sets whether the progress bar is enabled or not In the following bid is the progress bar control identity, and value determines whether the progress bar will be enabled or disabled. When value is zero the progress bar is disabled and when non zero it is enabled. ProgressbarSetMode bid,value
ProgressbarSetView	C	Sets whether the progress bar is visible or not In the following bid is the progress bar control identity, and value determines whether the progress bar will be in view or will be hidden. When value is zero the progress bar is hidden and when non zero it is visible. ProgressbarSetView bid,value
ProgressbarSetLeft	C	Sets the distance in pixels of the left side of the

		progress bar from the left side of the card. ProgressbarSetLeft bid,value
ProgressbarSetTop	C	Sets the distance in pixels of the top of the progress bar from the card top. ProgressbarSetTop bid,value
ProgressbarSetWidth	C	Sets the width of the progress bar in pixels. ProgressbarSetWidth bid,value
ProgressbarSetHeight	C	Sets the height of the progress bar in pixels. ProgressbarSetHeight bid,value
ProgressbarModeFN	F	The value returned indicates whether the specified progress bar is enabled or disabled. A return value of non zero indicates the progress bar is enabled otherwise it is disabled. Put ProgressbarModeFN(3) into bokay
ProgressbarViewFN	F	The value returned indicates whether the specified progress bar is visible or hidden. A return value of non zero indicates the progress bar is visible otherwise it is hidden. Put ProgressbarViewFN(3) into bokay
ProgressbarLeftFN	F	Returns the distance in pixels of the left side the progress bar from the left side of the card. Put ProgressbarLeftFN(1) into num
ProgressbarTopFN	F	Returns the distance in pixels of the top of the progress bar from the top of the card. Put ProgressbarTopBFN(1) into num
ProgressbarWidthFN	F	Returns the width of the progress bar control. Put ProgressbarWidthFN(1) into num
ProgressbarHeightFN	F	Returns the height of the progress bar control. Put ProgressbarHeightFN(1) into num

ProgressbarSetMinValue	C	Sets the minimum value which the progress bar can report or be set to. ProgressbarSetMinValue bid,value
ProgressbarSetMaxValue	C	Sets the maximum value which the progress bar can report or be set to. ProgressbarSetMaxValue bid,value
ProgressbarSetValue	C	Sets the progress bar position and value. ProgressbarSetValue bid,value
ProgressbarMinValueFN	F	Returns the minimum value setting of the progress bar. Put ProgressbarMinValueFN(1) into num
ProgressbarMaxValueFN	F	Returns the maximum value setting of the progress bar. Put ProgressbarMaxValueFN(1) into num
ProgressbarValueFN	F	Returns the position value setting of the progress bar. Put ProgressbarValueFN(1) into num

Timers

A timer control resides on a Card and calls a handler/script at times determined by its period setting. This should not be confused with the Main Timer which is a global timer as detailed in the MainTimer section.

When the card on which the timer resides goes out of focus any associated timers will pause until their card comes back into focus again. Timers can only execute their handler when no other script is running. The timer script can be edited via the Script button on the Properties window. Timers are not yet supported in plug-ins. Although timers will work within the Developer they will not be saved or incorporated into the built plug-in.

The firing state of a timer cannot be saved and therefore when a stack is first loaded all timers are switched off with their periods set to a default value of 1000 ms. If a timer should be working then it needs to be set up when its card loads.

Timer Modes

Mode	Description
Off	The timer is effectively disabled and will play no part in the operation of the stack. Its mode can be changed at runtime.
Single	The timer counts down and when its count reaches zero its script executes. After execution the timer will enter Off mode.
Periodic	The timer will count down and when its count reaches zero its script executes. After execution the timer will remain in Periodic mode and the cycle will recommence.

Name	Type	Description
TimerSet	C	This sets the specified timer into one of three modes, off, single or multi. In single mode, the timer will fire just once after its countdown has reached zero and it will then switch into off mode. in multi mode it will fire and then start its countdown again. TimerSet tid,mode,period mode : 0 = off 1 = single 2 = multi period: measured in milliseconds, i.e 1220 ms = 1.220 seconds

TimerOff	C	This switches the specified timer into off mode. TimerOffB tid

11 Canvas Graphics

Most computer languages having a command for copying areas of image use a complex looking command requiring many parameters. In contrast, HyperNext tries to makes it easier for beginners by breaking the usual command into three separate commands. Firstly the image source must be specified, then the image target, followed by the command that actually copies the image area.

These commands are CanvasSource, CanvasTarget and CanvasCopy.

With these commands it is possible to rapidly copy and scaling areas of canvases. Canvases can also be saved - see the section on Files Graphics.

Examples

One off screen buffer is also available for holding and processing images. It is accessible using canvas identity 0.

Clearing a buffer

```
@ Buffer  --> white
CanvasClear 0,255,255,255

@ Buffer --> red
CanvasClear 0,255,0,0

@ Canvas 1 on the card in focus
CanvasClear 1,255,255,255
```

Copying

```
@ Copy canvas 1 to canvas 2, do not scale
CanvasSource  1,0,0,0,0
CanvasDest   2,0,0,0,0
CanvasCopyAll


@ Copy canvas 1 to canvas 2, scale to fit canvas 2
CanvasSource  1,0,0,0,0
CanvasDest   2,0,0,0,0
CanvasCopyScale
```

```
@ Copy canvas 1 to canvas 2, area 1 into area 2
CanvasSource  1,50,50,100,100
CanvasDest    2,200,200,80,80
CanvasCopyArea
```

Name	Type	Description
CanvasClear	C	Clears a canvas by filling it with the specified colour. CanvasClear 1,red,green,blue
CanvasSetBuffer	C	Sets the size of an off screen buffer and fills it with the colour white. CanvasSetBuffer width,height
CanvasSource	C	Defines the source image area, an area of a canvas or off screen buffer. cid = source identity, 0 - buffer, >0 - a canvas CanvasSource cid,x,y,width,height If some of the parameters are not needed, such as when copying whole images then simply replace the redundant parameters with 0 or whatever, they just become place holders for the compiler and do not slow down the runtime. For instance @ Copying all of canvas 12 CanvasSource 12,0,0,0,0
CanvasDest	C	Defines the destination image area, an area of a canvas or off screen buffer. cid = destination identity, 0 - buffer, >0 - a canvas CanvasDest cid,x,y,width,height
CanvasCopyArea	C	Copies the source area to the destination area with scaling. This is the most general purpose copying command and it can perform the operations of both the CanvasCopyAll and CanvasCopyScale commands although it needs all parameters specifying.

		CanvasCopyArea
CanvasCopyAll	C	Copies the whole source image to the destination without scaling or changing coordinates. If the source image is larger than the destination then some the source will go off the destination. CanvasCopyAll
CanvasCopyScale	C	Copies and scales the source image to fit the destination image. CanvasCopyScale

12 Playing Sounds & Music

HyperNext has commands for playing sound files and MP3s. On Windows machines QuickTime needs to be installed in order to play some sound formats.

There are five dedicated sound channels and any sound or music file must be allocated to a channel before it can be played.

Commands & Functions

Name	Type	Description
SoundPlay	C	Plays the given sound file once on the specified sound channel where the filename refers to a local file. The filename can also refer to the local pathname to the file. SoundPlay(channel,filename)
SoundLoop	C	Plays the given sound file repeatedly on the specified sound channel where the filename refers to a local file. The filename can also refer to the local pathname to the file. SoundLoop(channel,filename)
SoundPlayAbs	C	Plays the given sound file once on the specified sound channel where the filename refers to the absolute path to a file. SoundPlayAbs(channel,filename)
SoundLoopAbs	C	Plays the given sound file repeatedly on the specified sound channel where the filename refers to the absolute path to a file. SoundLoopAbs(channel,filename)
SoundVolume	C	Sets the volume for the specified sound channel. The value ranges between 0 to 256. SoundVolume(channel,value)
SoundStop	C	Immediately stops the sound file being played. SoundStop(channel)

SoundIsPlayingFN	F	Returns 1 if the specified sound channel is playing a sound otherwise it returns 0. Put SoundIsPlayingFN(sid) into value

13 Note Player

HyperNext supports note playing based upon QuickTime Musical instruments. On Windows machines note playing requires QuickTime to be installed. There are 128 instruments available and a list of their names can be accessed using a HyperNext function.

Three different approaches to playing notes are supported. The first two are useful for playing single notes whereas the third allows melodies or sequences of notes to be defined and played. At the present time HyperNext only supports one Note channel so if a melody is playing then any PlayPitch or PlayNote command will be ignored.

(1) Notes can be played individually using the PlayPitch command which specifies the instrument, pitch, velocity and duration. This method is useful for allowing the user to play a particular note, perhaps via a keyboard displayed on a card.

(2) Notes can be played individually using the PlayNote command which specifies the instrument, octave, note, velocity and duration. This method is useful for allowing the user to play a particular note, perhaps via a keyboard displayed on a card.

(3) Notes can be played as group using the MelodyPlay command. This is more powerful than the PlayPitch/PlayNote commands and gives greater control over the musical output.

Commands & Functions

Name	Type	Description
PlayPitch	C	This plays a note with pitch in the range 0 to 127 where middle C equals 60 and higher values give higher pitches. Changing the value by 1 changes the pitch by one half. The note can be stopped using PlayEnd. Velocity represents how hard the key is pressed and ranges from 0 to 127 where 0 signifies not being pressed and 127 is the hardest (loudest). PlayPitch instrument,pitch,velocity,duration instrument = 1 to 128 pitch = 0 to 127 velocity = 0 to 127 duration = milliseconds

PlayNote	C	This plays a single note where middle C is in the fifth octave. The note can be stopped using PlayEnd. Notes have the form of A B C D E F G. PlayNote instrument,octave,note,velocity,duration
IsPlayingFN	F	This returns 1 if a note is currently being played otherwise it returns 0.
PlayEnd	C	This immediately ends the note currently being played.
MelodySet	C	Specifies the melody to be played by the MelodyPlay command. For example MelodySet waltz The melody is a sequence of notes held in a variable and it can have a complex nature depending upon the required melody. Each note or descriptor is separated by a space and the following notes and descriptors are understood by the melody note player:- notes A B C D E F G # raises a note by half a step, eg A# - lowers a note by half a step. eg G- O+ raises octave. O- lowers octave, the melody is assumed to start off in the fifth octave. i num changes the emphasis on a note, higher numbers give louder notes. L num changes the length of notes. num determines the length, 1 is whole note, 4 is quarter note. optionally a . (dot) may be added to increase the length. p num rests (pauses) for num where 1 is a whole rest, 8 is an eight rest etc. T num changes the tempo where num specifies the number of quarter notes.
MelodyPlay	C	Plays the melody specified by the MelodySet command for the specified number of times. If the melody is already playing this command will be ignored. @ Play tune 3 times MelodyPlay 3

MelodyLoop	C	Sets the number of times the melody is to be played. This can be changed while the melody is playing and its value is decremented each time a melody is finished. Its current value can be found using MelodyLoopsFN.
MelodyLoopsFN	F	Returns the current loop value. If the initial loop was set at 1 and the melody is still playing then this value will be 1. It is only decremented at the end of a melody.
MelodyInterval	C	This sets the interval between one loop ending and the next loop starting. Its default value is 2000 milliseconds and it can be changed while a melody is playing. @ set to 1.5 seconds MelodyInterval 1500
MelodySpeed	C	Specifies the speed or tempo for the melody. The default speed is 1.0 but it can be made slower or faster. For instance, values of 0.5 and 2.0 set the speed to half speed and double speed respectively. This can be changed while the melody is playing.
MelodyPause	C	This pauses the melody currently being played.
MelodyContinue	C	If a melody is paused this continues its playing.
MelodyEnd	C	The melody playing is stopped, it cannot be continued although it can be restarted using MelodyPlay.
CurrentNoteFN	F	Returns the position in the melody of the current note being played.
MelodyLengthFN	F	Returns the length in notes of the current melody.
SetInstrumentName	C	Changes the current instrument to the named instrument. The default instrument is Grand Piano and it can be changed while the melody is playing.
SetInstrumentNumber	C	Sets the instrument number which ranges from 1 to 128. The instrument can be changed while a melody is playing.
InstrumentNowFN	F	Returns the number of the current instrument. The name of the current instrument can be found by using the InstrumentNameFN function.

		Local inum,iname Put InstrumentNowFN into inum Put InstrumentNameFN(inum) into iname
InstrumentListFN	F	Returns a list containing the names of all the 128 instruments.
InstrumentNumberFN	F	When given a name it returns the number of that instrument. If no such instrument exists then 0 is returned.
InstrumentNameFN	F	When given an identity in the range 1 to 128 it returns the corresponding instrument name otherwise it returns empty.
MaxInstrumentsFN	F	Returns the number of instruments available, currently it is 128.

Instrument List

Instruments	
Number	Name
1	Acoustic Grand Piano
2	Bright Acoustic Piano
3	Electric Grand Piano
4	Honkytonk Piano
5	Rhodes Piano
6	Chorused Piano
7	Harpsichord
8	Clavinet
9	Celesta
10	Glockenspiel
11	Music Box
12	Vibraphone
13	Marimba
14	Xylophone
15	Tubular Bells
16	Dulcimer
17	Hammond Organ
18	Percussive Organ
19	Rock Organ
20	Church Organ
21	Reed Organ
22	Accordion
23	Harmonica
24	Tango Accordion
25	Acoustic Nylon Guitar
26	Acoustic Steel Guitar
27	Electric Jazz Guitar
28	Electric Clean Guitar
29	Electric Guitar Muted
30	Overdriven Guitar
31	Distortion Guitar
32	Guitar Harmonics
33	Acoustic Fretless Bass
34	Electric Bass Fingered
35	Electric Bass Picked
36	Fretless Bass
37	Slap Bass 1
38	Slap Bass 2
39	Synth Bass 1
40	Synth Bass 2
41	Violin
42	Viola

43	Cello
44	Contrabass
45	Tremolo Strings
46	Pizzicato Strings
47	Orchestra Harp
48	Timpani
49	Acoustic String Ensemble
50	Acoustic String Ensemble 2
51	Synth Strings 1
52	Synth Strings 2
53	Aah Choir
54	Ooh Choir
55	SynthVox
56	Orchestra Hit
57	Trumpet
58	Trombone
59	Tuba
60	Muted Trumpet
61	French Horn
62	Brass Section
63	Synth Brass 1
64	Synth Brass 2
65	Soprano Sax
66	Alto Sax
67	Tenor Sax
68	Baritone Sax
69	Oboe
70	English Horn
71	Bassoon
72	Clarinet
73	Piccolo
74	Flute
75	Recorder
76	Pan Flute
77	Bottle Blow
78	Shakuhachi
79	Whistle
80	Ocarina
81	Square Wave
82	Saw Wave
83	Calliope
84	Chiffer
85	Charang
86	Solo Vox
87	5th Saw Wave
88	Bass & Lead
89	Fantasy
90	Warm
91	Polysynth
92	Choir

93	Bowed
94	Metal
95	Halo
96	Sweep
97	Ice Rain
98	Sound Tracks
99	Crystal
100	Atmosphere
101	Brightness
102	Goblins
103	Echoes
104	Space
105	Sitar
106	Banjo
107	Shamisen
108	Koto
109	Kalimba
110	Bag Pipe
111	Fiddle
112	Shannai
113	Tinkle Bell
114	Agogo
115	Steel Drums
116	Woodblock
117	Timpani
118	Melodic Tom
119	Synth Drum
120	Reverse Cymbal
121	Guitar Fret Noise
122	Breath Noise
123	Seashore
124	Bird Tweet
125	Telephone Ring
126	Helicopter
127	Applause
128	Gunshot

14 Printing

HypThis section details how to handle printing from within your program. The printer handler can send output to a real printer or PDF depending upon the Operating System printer options.

Printing generally occurs in two stages:-
Firstly, PageSetup is called allowing the user to specify the actual printer, paper size, and the layout, either portrait or landscape. PageSetup then returns printer settings describing the size of the paper and the size of the printable area. The program then uses these printer settings to guide the generation of graphics.
Secondly, OpenPrinter is called allowing the user to specify the number of copies, and page ranges to be printed. This actually connects the printer and program so that any text or graphics written to an offscreen buffer can be sent to the printer, one page at a time. The printer and program will remain connected until the printer connection is closed.

Canvas zero is used as an offscreen buffer. Using a canvas as a buffer allows graphics to be drawn directly to it, as well as images, text etc.

The margins are set to a standard value and cannot be made smaller. However, thay can be increased by using the PageSetup settings to direct the graphics to a specific area of the offscreen buffer.

All printer measurements are in pixels except for printer resolution which is in dots per inch(dpi).

NOTE,
Always close the printer otherwise HyperNext will hang as the printer will indefinitely wait for the next page to arrive.

Commands & Functions

Name	Type	Description
PrinterInitialisedFN	F	If the printer has been initialised it will return the value 1, otherwise it will be 0 . The printer can be initialised by calling the PrinterPageSetup command. Put PrinterInitialised FN into okay

PrinterPageSetup	C	This calls the Page Setup dialog box allowing the user to choose the printer, layout and paper sizes etc. The program does not wait at this instruction until the dialog box is closed so if you intend to use the printer setting immediately after the dialog box closes you should make a loop which periodically checks whether the dialog box has closed or not.
		
		PrinterPageSetup
		
		The following example shows how to respond to the page setup dialog box. This does not send any content to the printer but just sets it up and then assigns values to the page size, layout etc as used by the associated HyperNext functions for later use by the program.
		
		Global printerString
		Local pstatus,t,okay,s1
		
		Put 0 into pstatus
		PrinterPageSetup
		
		SetMouseCursor(1)
		For t=1 to 25
		Wait(1,1000)
		Put PrinterInitialisedFN into okay
		If okay=1 Then
		Put PrinterPageSetupFN into pstatus
		If pstatus>0 Then
		Put PrinterSettingsFN into printerString
		ExitFor
		EndIf
		EndIf
		EndFor
		SetMouseCursor(3)

PrinterPageSetupFN	F	Returns 0 if the PrinterPage dialog box is still open, 1 if the user cancelled the operation, and 2 if they accepted the settings. Put PrinterPageSetupFN into pstate
PrinterSettingsFN	F	Returns the current Printer Page settings. The setting string format is highly dependent upon the computer platform. On the Macintosh OS X platform they are a multi-line string and on Macintosh OS 9 and Windows platforms a more complex format. If the settings are to be saved to a file it is best to use a binary save option. Put PrinterSettingsFN into psettings1
PrinterPageRestore	C	Restores the Page Setup using previously stored printer settings. PrinterPageRestore(psettings1)
PrinterTest	C	Tests the printer by sending a page to it containing a border, a large circle and some text all in black. The Printertest command only works when a connection has been made to the printer device using OpenPage. PrinterTest
PrinterSendPage	C	Sends the current graphics to the printer and then creates a new page. The printer will not quit after this command but will wait for the close command. PrinterSendPage
PrinterClose	C	Sends any waiting graphics to the printer and then closes it. Closing the printer breaks the connection between the program and the printer so allowing the program to continue. If the printer is not closed then the program will hang indefinitely. PrinterClose
PrinterSetHorizontalRes	C	Sets the horizontal resolution of the printer. The default settings are 72dpi but can be changed up to the maximum supported by the printer. If you want the maximum and do not know its value simply pass -1 as the parameter. Note, changing the resolution does not change the print size of your graphics, it only changes the print quality. If

		the resolution is doubled then your graphics can be doubled in size to take advantage of the new resolution. PrinterSetHorizontalRes(hres)
PrinterSetVerticalRes	C	Sets the vertical resolution of the printer. The default settings are 72dpi but can be changed up to the maximum allowed by the printer. If you want the maximum allowed by the printer simply pass -1 as the parameter. PrinterSetVerticalRes(vres)
PrinterHorizontalResFN	F	Returns the current horizontal resolution measured in dpi. Put PrinterHorizontalResFN into hres
PrinterVerticalResFN	F	Returns the current vertical resolution measured in dpi. Put PrinterVerticalResFN into vres
PrinterHorizontalMaxFN	F	Returns the current maximum horizontal resolution measured in dpi. If the resolution is changed it will take effect after a PageSetup or OpenPrinter directive is called. Put PrinterHorizontalMaxFN into hres
PrinterVerticalMaxFN	F	Returns the current maximum vertical resolution measured in dpi. Put PrinterVerticalMaxFN into vres
PrinterPageLeftFN	F	This returns the left side of the printable area. At the present time this always returns 0 as the margins are set automatically. Put PrinterPageLeft FN into pleft
PrinterPageTopFN	F	This returns the top of the printable area. At the present time this always returns 0 as the margins are set automatically. Put PrinterPageTopFN into ptop
PrinterPageHeightFN	F	Returns the height of the paper as set by PageSetup.

		Put PrinterPageHeightFN into pheight
PrinterPageWidthFN	F	Returns the width of the paper as set by PageSetup. Put PrinterPageWidthFN into pwidth
PrinterHeightFN	F	Returns the height of the printable area, that is the graphics area. When using Canvas(0) as an offscreen buffer its height should be set to this value. Put PrinterHeightFN into height
PrinterWidthFN	F	Returns the width of the printable area, that is the graphics area. When using Canvas(0) as an offscreen buffer its width should be set to this value. Put PrinterWidthFN into width
PrinterOpenDefaultFN	F	Attempts to open a connection to the default printer without displaying an OpenPrinter dialog box and if successful it creates a new graphics page. The function returns 1 if successful and 0 if not. When the paramater ps is 0 the current Page Setup settings are used and when 1 it will display the Page Setup dialog box. Put PrinterOpenDefaultFN(ps) into pokay
PrinterOpenDialogFN	F	Opens a dialog box allowing the user to open a connection with a printer. It automatically creates a new graphics page. If the connection was made successfully then the function returns 1 otherwise it returns 0. When the paramater ps is 0 the current Page Setup settings are used and when 1 it will display the Page Setup dialog box. Put PrinterOpenDialogFN(ps) into pokay
PrinterCopiesFN	F	Returns the number of document copies requested by the user via the OpenPrinterDialog box. Put PrinterCopiesFN into ncopies
PrinterFirstPageFN	F	Returns the first page number of the document to be printed as requested by the user via the OpenPrinterDialog box.

		Put PrinterFirstPageFN into firstpage
PrinterLastPageFN	F	Returns the last page number of the document to be printed as requested by the user via the OpenPrinterDialog box. Put PrinterLastPageFN into lastpage

15 Files

This section details the commands available for finding out about folder and files, and for creating, copying and deleting them. Generally there are two ways of refering to a file, either local to the the project/stack/standalone folder of else absolute where a full pathname is specified. Most of the time it is best to use local files as they will be within your directory and unlikely to go astray or be interfered with.

These commands can also be used to set the attributes of the files, attributes such as Creator and Type.

Once a file has been either located or created it can then be operated upon using the binary/text file options given in both this section and in the Text Files section.

Files and folders are accessed via handles and if the value of a handle is zero then the file or folderitem does not exist.

General Files

Name	Type	Description
VolumeCountFN	F	Returns the number of volumes such as hard disks on the computer. Put VolumeCountFN into vcount
VolumeListFN	F	Returns a list containing the name of each volume on a separate line. Put VolumeListFN into vlist
VolumeNameFN	F	Returns the name of the numbered volume. Volume numbers start at 1 and if the number is out of range the function will return an empty value. Put VolumeNameFN(vnum) into vname
MyDirectoryFN	F	Returns the absolute address of the running project, stack or standalone project. Most file operation work locally, and refer to folders and files within the running project's folder. Put MyDirectoryFN into myfolder
MyFolderFN	F	Exactly the same as the MyDirectory function.

		Put MyFolderFN into myfolder
ApplicationFolderFN	F	Returns the absolute path to the folder containing either HyperNext Creator, Developer, Player or a Standalone depending upon which type of program the function is called from. Put ApplicationFolderFN into appfolder
FolderUserAppDataFN	F	Returns the absolute path to the folder where application preferences should be stored. Put FolderUserAppDataFN into prefsfolder for example with a user called John Windows platform C:\Documents and Settings\John\Application Data OS X platform OS Tiger:Users:John:Library:Preferences OS 9 platform OS 9.2.1:System Folder:Preferences
FolderUserMainFN	F	Returns the absolute address of the main folder for the currently logged in user. Put FolderUserMainFN into mainwhere for example with a user called John Windows platform C:\Documents and Settings\John OS X platform OS Tiger:Users:John OS 9 platform OS 9.2.1
FolderUserDocumentsFN	F	Returns the absolute address of the documents folder for the currently logged in user. Put FolderUserDocumentsFN into docswhere
FolderUserDesktopFN	F	Returns the absolute address of the desktop folder for the currently logged in user. Put FolderUserDesktopFN into deskwhere

FolderSeparatorFN	F	Returns the folder separator for the platform which the program is running on. For Windows it is a \ character and for Macintosh a : character Put FolderSeparatorFN into sep
FolderItemGet	C	When given the path name of a local folder or file it returns details about it. If the item does not exist then the returned details can be used to create the item and then access it. FolderItemGet(fname,fhandle,fdetails,ftypes,fpaths, fnames,fextens) where fname - path name of the folder. fhandle - an integer referring to the particular file or folder. fdetails - a list of folder attributes - see section Folder/File Details. ftypes - a list giving the types of item contained within the folder. 1 = folder, 0 = file. fpaths - a list giving the pathnames of folders/files contained within the folder. fnames - a list giving the names of folders/files contained within the folder. fextens - a list giving the extensions of folders/files contained within the folder.
FolderItemGetAbs	C	Similar to FolderItemGet except it uses an absolute filename/path. When given the path name of a folder or of a file it returns details about it. If the item does not exist then the returned details can be used to create the item and then access it. FolderItemGetAbs(fname,fhandle,fdetails,ftypes, fpaths,fnames,fextens)
FolderItemAsk	C	Presents a dialog box allowing the user to choose a folder. FolderItemAsk fname,fhandle,fdetails,ctypes,cnames
FolderItemNew	C	Presents a dialog box allowing the user to choose the name and location of where a new folder should be placed. This does not create the folder but returns a handle and details about the potential folder.

		FolderItemNew name,fhandle,fdetails,ctypes,cnames
FileGet	C	When given the local path name of a file it returns details about it. If the file does not exist then the returned details can be used to create the file and then access it. FolderGet(fname,fhandle,fdetails,ftypes,fpaths, fnames,fextens) where fname - path name of the folder. fhandle - an integer referring to the particular file or folder. fdetails - a list of folder attributes - see section Folder/File Details. ftypes - a list giving the types of item contained within the folder. 1 = folder, 0 = file. fpaths - a list giving the pathnames of folders/files contained within the folder. fnames - a list giving the names of folders/files contained within the folder. fextens - a list giving the extensions of folders/files contained within the folder.
FileGetAbs	C	When given the absolute path name of a file it returns details about it. If the file does not exist then the returned details can be used to create the file and then access it. FolderGetAbs(fname,fhandle,fdetails,ftypes,fpaths, fnames,fextens) where fname - path name of the folder. fhandle - an integer referring to the particular file or folder. fdetails - a list of folder attributes - see section Folder/File Details. ftypes - a list giving the types of item contained within the folder. 1 = folder, 0 = file. fpaths - a list giving the pathnames of folders/files contained within the folder. fnames - a list giving the names of folders/files contained within the folder.

		fextens - a list giving the extensions of folders/files contained within the folder.
FileAsk	C	Presents a dialog box allowing the user to choose a file which already exists. FileAsk fname,fhandle,fdetails
FileAskFilter	C	Presents a dialog box allowing the user to choose a file which already exists and has the specified file type. If the filter parameter is empty then all files are shown. FileAskFilter fname,filter,fhandle,fdetails Eg, filter might be = 'HNapp' etc
FileNew	C	Presents a dialog box allowing the user to choose the name and location of where a new file should be placed. This does not create the file but returns a handle and details about the potential file. FileNew fname,fhandle,fdetails
FolderItemFree	C	Frees the space used by a specified folder/file handle. FolderItemFree fhandle
FolderItemRename	C	Renames the folder specified by the folder handle. FolderItemRename fhandle,fname
FolderItemDelete	C	Deletes the folder specified by the folder handle. The folder must be empty otherwise the command has no effect. To delete a folder containing folders/files, one must first obtain details about that folder and then use them to delete each item in turn. FolderItemDelete fhandle
FolderItemDeleteAll	C	Deletes the folder specified by the folder handle. The specified folder need not be empty as this command will delete all subfolders and files within the folder. FolderItemDeleteAll(fhandle)

FolderItemMove	C	Moves the folder specified by the folder handle to the location specified by the destination handle. FolderItemMove fhandle,dhandle
FolderItemCopy	C	Copies the folder specified by the folder handle to the location specified by the destination handle. FolderItemCopy fhandle,dhandle
FolderItemLaunch	C	Launches the application specified by the given file handle. If value is zero then the application will run in the background, and if non zero will run in the foreground. FolderItemLaunch fhandle,value
FolderItemVisible	C	Sets the visibility of the folder/file specified by the folder item handle. If value is zero then it will be invisible, and if non zero will be visible. FolderItemVisible fhandle,value
FolderItemLocked	C	Lock or unlocks the folder/file specified by the folder item handle. If value is zero then it will be unlocked, and if non zero will be locked. FolderItemLocked fhandle,value
FolderItemMacType	C	Sets the 4 character Mac Type of the file specified by the folder item handle. FolderItemMacType fhandle,value
FolderItemMacCreator	C	Sets the 4 character Mac Creator of the file specified by the folder item handle. FolderItemMacCreator fhandle,value
CreateFolder	C	Creates a folder specified by the folder item handle and returns a folder index which can be used to actually access the folder and change its name. CreateFolder fhandle,findex

Folder/File Details

The details returned from accessing a file or folder are given as a list of attributes separated by carriage returns.

1 - Name of file or folder.
2 - 1 if it exists, 0 otherwise.
3 - 1 if it is a folder, 0 if a file.
4 - 1 if an alias.
5 - 1 if visible.
6 - 1 if locked.
7 - 1 if writeable.
8 - 1 if readable.
9 - the absolute path.
10 - the number of items in the folder.
11 - The Mac Type code.
12 - The Mac Creator code.
13 - the creation short date.
14 - the creation short time.
15 - the modification short date.
16 - the modification short time.
17 - the display name.
18 - 1 if the extension is visible.
19 - the length of the data fork.
20 - the length of the resource fork.
21 - the Macintosh Directory ID - an integer.
22 - the Macintosh VRefNum - an integer.
23 - Actual name including extension.
24 - filename extension.
25 - type (user created).
26 - Parent directory absolute pathname.

Binary Files

A binary file is a computer file whose characters can contain numeric values between 0 and 255. To the human eye they are generally not readable as they contain raw data. Binary files are used for data, multi-media and executable programs.

To operate on a binary file a file handle must first be obtained. This file handle can then be used to create a file variable so enabling operations to be carried out on that file.

Listed below are commands for reading and writing binary files. Some of the commands and functions can be used for both reading and writing so they are therefore listed under each mode.

Note, the current read/write position within the file is automatically updated whenever a read/write operation is carried out.

Name	Type	Description
CreateBFile	C	Creates a binary file specified by the folder item handle and returns a file index which can be used to actually access the file, change its name, read and write to it. CreateBFile fhandle,findex
OpenAsBFile	C	Opens an existing binary file specified by the folder item handle and returns a file index which can be used to actually access the file, change its name, read and write to it. OpenAsBFile fhandle,findex
FileSetBPosition	C	Sets the position in the binary file where the next read/write will take place. FileSetBPosition findex,fpos
CloseBFile	C	Close the specified binary file. CloseBFile findex
FileSetBLength	C	Sets the length of the binary file data fork. If the specified length is less than the actual length then the data fork will be truncated. FileSetBLength findex,flength

EndBFileFN	F	Returns a value indicating whether the position within the specified binary file has reached the end or not. If zero then the end has not been reached and if 1 then the end of the binary file has been reached. Put EndBFileFN(findex) into field 1
LengthBFileFN	F	Returns the length of the data fork of the specified binary file. Put LengthBFileFN(findex) into flen
PositionBFileFN	F	Returns the current read/write position within the specified binary file. Put PositionBFileFN(findex) into fpos
ReadBByte	C	Reads a single byte from the current position. The byte value ranges from 0 to 255. ReadBByte findex,byte
WriteBByte	C	Writes a single byte to the current position of the specified binary file. The byte value ranges from 0 to 255. WriteBByte findex,byte
ReadBVariable	C	Reads a variable from the current position. The variable has a string form and can be over 2GB in length. ReadBVariable findex,svar Put svar into field 1
WriteBVariable	C	Writes a variable at the current position of the specified binary file. Put field 1 into svar WriteBVariable findex,svar
ReadBBoolean	C	Reads a boolean value from the current position. The value can be either 0 or 1. ReadBBoolean(findex,var)
WriteBBoolean	C	Writes a boolean value to the current position of the specified binary file. The value can be either 0 or 1 but if the value is non zero then 1 will be written. WriteBoolean(findex,value)

ReadBShort	C	Reads a short integer from the current position. A short variable holds two bytes and its value ranges from -32768 to 32767. ReadBShort(findex,var)
WriteBShort	C	Writes a short integer to the current position of the specified binary file. WriteBShort(findex,value)
ReadBLong	C	Reads a long integer from the current position. A long variable holds four bytes and its value ranges from -2,147,483,648 to 2,147,483,647. ReadBLong(findex,var)
WriteBLong	C	Writes a long integer to the current position of the specified binary file. WriteBLong(findex,value)
ReadBDouble	C	Reads a double floating point value from the current position. A double variable holds eight bytes and its value ranges from about 2.2251 e-308 to 1.7977 e+308. ReadBDouble(findex,var)
WriteBDouble	C	Writes a double floating point value to the current position of the specified binary file. WriteBDouble(findex,value)
ReadBPString		Reads a Pascal string of text from the current position. Pascal strings can hold up to 255 characters. ReadBPString(findex,var)
WriteBPString		Writes a string of text having up to 255 characters to the current position of the specified binary file. If the string is longer than 255 characters then it will be truncated. WriteBPString(findex,value)
FileBSetEndian	C	Sets the endian value for the specified binary file. If set to 1 then the byte order is low byte, high byte. For Macintosh platform it is generally set to 0 and

		for the Windows platform to 1. This is important when reading in Short and Long values. FileBEndianSet(findex,1)
FileBEndianFN	F	Returns the endian setting for the specified binary file. Put FileBEndianFN(findex) into field 1

Binary Example

These show how to write and read to/from binary files. No checking for the file's existence or type is made using the file handle or returned details.

If a file handle is zero then the file does not exist and any attempt to access its properties will cause a runtime error.

```
@ Write to a named file
Local fname,fvar,fdets,ftypes,fpaths,fnames,fextens
Local findex,n
FileGet(fname,fvar,fdets,ftypes,fpaths,fnames,fextens)
CreateBFile(fvar,findex,n)
WriteBVariable(findex,'hello')
For n=1 to 10
   WriteBByte(findex,n)
EndFor
CloseBFile(findex)

@ Read from a named file, assumes format known
Local fname,fvar,fdets,ftypes,fpaths,fnames,fextens
Local findex,n
FileGet(fname,fvar,fdets,ftypes,fpaths,fnames,fextens)
OpenAsBFile(fvar,findex)
ReadBVariable(findex,mess)
For n=1 to 10
   ReadBByte(findex,num)
   Put num After mess
EndFor
CloseBFile(findex)
```

Text Files

As variables in HyperNext are based on strings a set of text file commands are needed for saving and loading them.

In order to identify each file for reading and writing, a file variable is created specifically for that file. This file variable is used to indicate which file the read or write operation should be directed to.

There are three sets of related file commands which treat the filename or path as either local, absolute or pertaining to files in the 'Data:Text' folder.

Name	Type	Description
CreateTWrite	C	Creates a text file specified by the folder item handle and returns a file index which can be used to actually write to the file. CreateTWrite fhandle,findex
OpenTReadAsk	C	Displays a dialog box asking the user which text file to open for reading from. OpenTReadAsk(fname,fvar)
OpenTRead	C	Opens a text file for reading with the specified name. The name refers to a file in the 'Data:Text' folder. OpenTRead(fname,fvar)
OpenTReadLoc	C	Opens a text file for reading with the specified name. The pathname refers to a file located in the project's folder or subfolder. OpenTReadLoc(fname,fvar)
OpenTReadAbs	C	Opens a text file for reading with the specified absolute pathname. OpenTReadAbs(fname,fvar)
OpenAsTRead		Opens an existing text file specified by the folder item handle and returns a file index which can be used to read from the file. OpenAsTRead fhandle,findex
ReadTLine	C	Reads a line of text from the specified file. ReadTLine(fvar,value)

CloseTRead	C	Closes the specified read file. CloseTRead(fvar)
OpenTWriteAsk	C	Displays a dialog box asking the user which text file to open for writing to. OpenTWriteAsk(fname,fvar)
OpenTWrite	C	Opens a text file for writing with the specified name. The name refers to a file in the 'Data:Text' folder. OpenTWrite(fname,fvar)
OpenTWriteLoc	C	Opens a text file for writing with the specified name. The pathname refers to a file located in the project's folder or subfolder . OpenTWriteLoc(fname,fvar)
OpenTWriteAbs	C	Opens a text file for writing with the specified absolute pathname. OpenTWriteAbs(fname,fvar)
OpenAsTWrite	C	Opens an existing text file specified by the folder item index and returns a file index which can be used to write to the file. OpenAsTWrite findex,findex
WriteTLine	C	Writes a line of text to the specified file. WriteTLine(fvar,value)
CloseTWrite	C	Closes the specified write file. CloseTWrite(fvar)
EndTFileFN	F	Returns true if the specified text file has reached the end, otherwise returns false. Put EndTFileFN(fileid) into filedone

Reading example

@ Read file into field 1

```
Local fend,fname,h1,sdata
Put 'testdata' into fname
OpenTRead(fname,h1)
Put EndTFileFN(h1) into fend
While fend=0
   ReadTLine(h1,sdata)
   Put sdata After field 1
   Put EndTFileFN(h1) into fend
EndWhile
CloseTRead(h1)
```

Writing example

```
@ Write 10 numbers to a file

Local n,fname,h1
Put 'testdata' into fname
OpenTWrite(fname,h1)
For n=1 to 10
   WriteTLine(h1,n)
EndFor
CloseTWrite(h1)
```

Graphics Files

In conjunction with the image manipulation commands a set of commands for loading and saving canvases is available.

Files
There are two variants of these commands that differ only in how the filename/pathname is interpreted. The filename can refer to either a file within the projects folder or else an absolute filename.

Scaling
When loading a canvas from a file, the type of image scaling must be specified. If the scale is non zero then the image will be scaled to fill the entire canvas otherwise its size will be unchanged.

Image format
The default formats are Pict and BMP for Macintosh and Windows respectively. A JPEG save option is also available.

Name	Type	Description
CanvasSaveAsk	C	Display a dialog box asking the user for the name of the file in which to save the specified canvas. CanvasSaveAsk(cid,fname)
SaveAskCanvas	C	The legacy name for CanvasSaveAsk SaveAskCanvas(cid,fname)
CanvasSave	C	Save the given canvas to the specified local file. CanvasSave(cid,fname)
SaveCanvas	C	The legacy name for CanvasSave SaveCanvas(cid,fname)
CanvasSaveAbs	C	Save the given canvas to the specified absolute file. CanvasSaveAbs(cid,fname)
CanvasLoadAsk	C	Display a dialog box asking the user which file to load into the specified canvas. CanvasLoadAsk(cid,fname,scale)
LoadAskCanvas	C	The legacy name for CanvasLoadAsk.

		LoadAskCanvas(cid,fname,scale)
CanvasLoad	C	Load the given local file into the specified canvas. CanvasLoad(cid,fname,scale)
LoadCanvas	C	The legacy name for CanvasLoad LoadCanvas(cid,fname,scale)
CanvasLoadAbs	C	Load the given absolute file into the specified canvas. CanvasLoadAbs(cid,fname,scale)
CanvasJPEGSave	C	Save the given canvas to the specified local file in JPEG format. The quality can range from about 25 to 100 percent. CanvasJPEGSave(cid,fname,quality) @ Save 80 percent CanvasJPEGSave(1,fname,80)
CanvasJPEGSaveAbs	C	Save the given canvas to the specified absolute file in JPEG format. CanvasJPEGSaveAbs(cid,fname,quality)
CanvasJPEGSaveAsk	C	Display a dialog box asking the user for the name of the file in which to save the specified canvas in JPEG format. CanvasJPEGSaveAsk(cid,fname)

16 Networks

Easy Networks

The HyperNext language has built-in network handling called Easy Networks for communicating across a network with other HyperNext stacks, applications and REALbasic applications. Easy networks is aimed at beginners and those wanting to implement some easy to set up communications. However, a conventional and more powerful network based on TCP/IP is under development for HyperNext.

Easy Networks has its own simple protocol where every message is prefixed by an integer header command that can be chosen by the software designer. For example, a simple message might be

25,loginid

In order to makes communications more secure HyperNext has no predefined message header commands and leaves it up to the designer to choose the command integers and any corresponding messages. Header commands can range from zero up to over 1000 million and the message body itself can be encrypted as needed.

Easy Network Sockets

Easy networks is based around conventional TCP/IP sockets and allows many sockets to be created and active at any one time although in practice this will be limited by the speed at which the HyperNext runtime engine can process events as all network events are placed on the internal interrupt queue which is still quite slow.

Plug-ins

Easy network can be used from within plug-ins so allowing network extensions to be built for HyperNext Creator.

Network Events

When certain network events occur they automatically trigger script handlers that can be defined at design time from the Easy Network menu items located in the Edit menu.

The socket related to the event can be found from the EasyIndexFN function.

Connected Event

This event handler is called when an Easy socket makes a connection.

Example

```
Local mess
Put 'Connection by socket ' into mess
Append EasyIndexFN onto mess
Message mess
```

Error Event

This event handler is called when an Easy socket error occurs. Note, when a socket is disconnected error number 102 is automatically generated to indicate that the socket was disconnected.

Example

```
Local socknum,err,mess
Put EasyIndexFN into socknum
Put EasyErrorFN(socknum) into err
If err=102 Then
   Message 'disconnectd'
Else
   Put 'Socket error occurred ' into mess
   Append err onto mess
   Message mess
EndIf
```

Received Event

This event called is called when a message has successfully been received.

Example

```
Local snum,cmd,sdata
Put EasyIndexFN into snum
Put EasyRxCommandFN(snum) into cmd
Put EasyRxDataFN(snum) into sdata
If cmd=1 Then
   Put sdata into field 1
Else
   If cmd=2 Then
      Put sdata into field 2
   Else
      Message 'Unknown command'
   EndIf
EndIf
```

Send Completed Event

This event called is called when a message has been sent.

Example

```
Local mess
Put 'Message sent by socket ' into mess
Append EasyIndexFN onto mess
Message mess
```

Send Progress Event

This event handler is called when progress has been made in sending a message. The number of bytes sent and remaining can then be accessed using the EasyBytesSentFN and EasyBytesLeftFN functions.

Example

```
Put EasyIndexFN into socknum
Put EasyBytesSentFN(socknum) into field 3
Put EasyBytesLeftFN(socknum) into field 4
```

Name	Type	Description
NetworkRefresh	C	Rescans the computer's network hardware and refreshes the status functions. NetworkRefresh
NetCardCountFN	F	Returns the number of network cards. Put NetCardCountFN into numcards
NetIPListFN	F	Returns a list containing the input address for each network card. Put NetIPListFN into addresses
NetMacListFN	F	Returns a list of Mac addresses for the user's computer. Put NetMacListFN into macaddresses
NetSubnetListFN	F	Returns a list of subnet masks for the user's computer. Put NetSubnetListFN into subnets
EasyIndexFN	F	Returns the number of the socket that triggered the event.

		EasyIndexFN Example (in Received event handler) Put EasyIndexFN into socknum Put EasyRxDataFN(socknum) into field 3
EasyCountFN	F	Returns the number of Easy sockets currently in existence. EasyCountFN Example Put EasyCountFN into numsockets

EasyCreateFN	F	Creates a new Easy socket and returns the corresponding socket number.
		
		EasyCreateFN
		
		Example
		
		Put EasyCreateFN into snum
		If snum>0 Then
		EasyConnect snum,addr,port,ti
		EndIf

EasyListen	C	Sets the specified Easy socket into listening mode on the given port. In order to make a network connection between two applications, one of them must be listening. EasyListen socket,port
EasyConnect	C	Attempts to make a connection via the specified Easy socket to the target address. The address can be specified by the IP number such as 123.171.32.84 or if the target is on the local host by specifying localhost. The timeout parameter is currently is not used. EasyConnect socket,address,port,timeout Example Put '123.171.32.84' into addr EasyConnect 5,addr,4000,0 Put 'localhost' into addr EasyConnect 5,addr,4000,0

EasySendMessage	C	Sends a message using the specified Easy socket. The message comprises a command header which is simply an integer plus the actual data. EasySendMessage socket,command,data Example, target understands command 23. Global eSockNum Local cmd,sdata Put 23 into cmd Put field 5 into sdata EasySendMessage eSockNum,cmd,sdata
EasyDisconnect	C	Disconnects the specified Easy socket. When the socket is actually disconnected then the sockets Disconnect event handler will be triggered. EasyDisconnect socket
EasyDelete	C	Deletes the specified Easy socket. If the specified is the last in the socket list then it is deleted and the list count is decremented by one but otherwise the socket is simply disconnected. EasyDelete socketnum
EasyConnectedFN	F	Returns the connection status for the specified Easy socket. States 0 - disconnected 1 - trying to connect 2 - connected Put EasyConnectedFN(socketnum) into cstate
EasyAddressFN	F	Returns the IP address of the remote machine for the specified Easy Socket. Put EasyAddressFN(socketnum) into remoteIP
EasyErrorFN	F	Returns the error status for the specified Easy socket. Usually this is used from within the Easy Network Error Event handler. Put EasyErrorFN(socketnum) into err Error codes 0 - no error 100 - OpenDriver Error.

		102 - Lost connection (also occurs during disconnect). 103 - Name resolution error. 105 - Address in use error. 106 - Invalid state error. 107 - Invalid port error. 108 - Out of memory error.
EasyRxStateFN	F	Returns the reception state for the specified Easy socket. States 0 - idle 1 - receiving 2 - data ready Put EasyRxStateFN(socketnum) into rxstate
EasyRxCommandFN	F	Returns the command portion of the received message. Put EasyRxCommandFN(socketnum) into scmd
EasyRxDataFN	F	Returns the data portion of the received message. Put EasyRxDataFN(socketnum) into sdata
EasyTxStateFN	F	Returns the transmission state for the specified Easy socket. 0 - idle 1 - sending 2 - aborted 3 - completed Put EasyTxStateFN(socketnum) into txstate
EasyTxBytesSentFN	F	Returns the number of bytes sent of the current message. The value is only accurate when used within the Send Progress event handler. Put EasyTxBytesSentFN(socketnum) into numsent
EasyTxBytesLeftFN	F	Returns the number of bytes remaining to be sent for the current message. The value is only accurate when used within the Send Progress event handler. Put EasyTxBytesLeftFN(socketnum) into numleft
URLexistsFN	F	Returns true if the given web page exists otherwise false. The filename specifies a local address. The yield

		flag when true allows background events such as progress bars and timers to work while setting yield to false freezes most background activity. The timeout value cause the URLexistsFN function to abort if success has not occured within that time. Put URLexistsFN(addr,yield,timeout,fname) into okay example addr = http://www.tigabyte.com/exists.txt yield - true / false timeout - time in seconds fname - file in which to save the download.
URLexistsAbsFN	F	The same as URLexistsFN except the filename specifies an absolute address.
HTTPSetHeaderName	C	This sets the HTTP header name to be passed to the web server via the URLexistsFN or URLexistsAbsFN functions. HTTPSetHeaderName(name) HTTPSetHeaderName('User-Agent')
HTTPSetHeaderValue	C	This sets the HTTP header value to be passed to the web server via the URLexistsFN or URLexistsAbsFN functions. HTTPSetHeaderValue(value) HTTPSetHeaderValue('My Easy Web Browser')

HTTP Example

This example finds your IP address using the website 'www.whatismyip.org'. If successful it puts the IP address number into field 1.

```
Local webaddr,yield,timeout,filename,okay

@ Set params for URLexists function
HTTPSetHeaderName('User-Agent')
HTTPSetHeaderValue('My Web Browser')
Put 'http://www.whatismyip.org/' into webaddr
Put 1 into yield
Put 30 into timeout
Put 'webtext.txt' into filename

Put URLexistsFN(webaddr,yield,timeout,filename) into okay

@ If okay then write IP to field 1
If okay=1 Then
   Local h1,sdata
   OpenTReadLoc(filename,h1)
      ReadTLine(h1,sdata)
      Put sdata After field 1
   CloseTRead(h1)
Else
   Put  'unsuccessful' into field 1
EndIf
```

Web Server

By using just a few commands HyperNext can provide a very basic web server capability which allows a HyperNext application, stack or a project running within the Creator/Developer to serve web pages. These pages can be text graphics or multimedia as requested by a web browser.

Note, the server speed is very low and does not support multiple connections, so serving only one page at a time. Furthermore, the present configuration does not support logging accesses etc .

The usual port for serving web pages is port 80 but this can be changed when starting up the server.

The default directory is the project/stack/application folder and the web pages and web directories must reside here.

Name	Type	Description
WebServerStart	C	Sets up and starts the web server to listen on the specified port. WebServerStart port
WebServerStop	C	Stops the webserver. WebServerStop
WebServerPortFN	F	Returns the port number which the server is using. Put WebServerPortFN into portnum
WebServerAddressFN	F	Returns the full address including the port number as in 80.229.165.175:80 Put WebServerAddressFN into fulladdress
WebServerIPFN	F	Returns the address without port number as in 80.229.165.175 Put WebServerIPFN into ipnumber

17 RBscript

Introduction

HyperNext Creator, Developer and Player are all written in REALbasic and therefore allow the programmer to access to RBscript, a typed object orientated language similar to modern BASICs.

RBscript runs much faster than the native HyperNext language and so is highly useful where speed is important. Although HyperNext is more than fast enough for many operations, certain complex operations such as neural networks, graphics, image or heavy text processing require the speed of Rbscript.

Example 1

This example uses the PUT command to place the RBscript source code into the source valuable. The following code sets the context of RBscript to canvas 1 and draws 10000 points.

```
Local src
RBsSetCanvas 1
Put 'Dim n,x,y as integer' into src
Put 'Dim clr as color' after src
Put 'for n=1 to 10000' after src
Put 'clr=RGB(Rnd*255,Rnd*255,Rnd*255)' after src
Put 'SetForeColor(clr)' after src
Put 'x=Rnd*300' after src
Put 'y=rnd*300' after src
Put 'PlotPoint x,y' after src
Put 'next' after src
RBsSetSource src
RBsTest
RBsRun
```

Example 2

This example uses the RBsAdd command to place the RBscript source code into the source variable. The RBscript code draws 10000 points.
This method is easier to use as the RBscript system itself keeps track of the RBscript source code.

```
Local src
RBsSetCanvas 1
RBsClear
```

```
RBsAdd 'Dim n,x,y as integer'
RBsAdd 'Dim clr as color'
RBsAdd 'for n=1 to 10000'
RBsAdd 'clr=RGB(Rnd*255,Rnd*255,Rnd*255)'
RBsAdd 'SetForeColor(clr)'
RBsAdd 'x=Rnd*300'
RBsAdd 'y=rnd*300'
RBsAdd 'PlotPoint x,y'
RBsAdd 'next'
RBsTest
RBsRun
```

Name	Type	Description
RBsSetSource	C	Sets the source or code to be used by the RBscript engine. RBsSetSource src
RBsSourceFN	F	Returns the current source code. Put RbsSourceFN into src
RBsClear	C	Clears the source code from the Rbscript internal source text. RbsClear
RBsAdd	C	Adds the specified text to the Rbscript internal source text. RbsAdd 'Dim x,y as Integer'
RBsTest	C	The RBscript engine tests the source code to see if it has any errors. Afterwards RBsTestErrorFN can be used to see if the test was successful or not. RBsTest
RBsRun	C	Executes the given source code. RBsRun
RBsInput	C	Accepts input from the keyboard or other specified source.
RBsSetEncoding	C	Sets the text encoding to be used.

RBsSetCanvas	C	Specifies the canvas to be used by RBscript. RBsSetCanvas cid
RBsStateFN	F	Returns the state of the RBscript engine, whether it is running etc. 0 - Ready 1 - Running 2 - Complete 3 - Aborted Put RBsStateFN into res
RBsTestErrorFN	F	Returns a value specifying whether an error occurred during the testing of the source code. A value of zero means no error occurred other the value represents the line number in the source where the error occurred. Put RbsTestErrorFN into res
RBsTestErrorNumFN	F	Returns the error number from the Rbscript compiler. Put RbsTestErrorNumFN into errnum
RBsTestErrorMsgFN	F	Returns the error message from the Rbscript compiler. Put RbsTestErrorMsgFN into errmess
RBsTestSrcLineFN	F	Returns the source line where the error occured. Put RbsTestSrcLineFN into src
RBsRunErrorFN	F	Returns a value specifying whether an error occurred during the execution of the source code. A value of zero means no error occurred Put RBsRunErrorFN into res
RBsRunErrorNumFN	F	Returns the error number from the aborted Rbscript run. Put RbsRunErrorNumFN into errnum
RBsRunErrorMsgF	F	Returns the error message from the aborted Rbscript run. Put RbsRunErrorMsgFN into errmess

RBsRunSrcLineFN	F	Returns the source line where the error occurred during the aborted Rbscript run. Put RbsRunSrcLineFN into src
RBsOutputFN	F	Returns text output from the running RBscript. Put RBsOutputFN into field 1

RBscript Graphics

A current limitation of RBscript is that it can work only with one canvas at a time. The canvas is passed to it using the RBsSetCanvas command as detailed above.

Once the RBscript has been assigned a canvas then the commands listed here can be used within RBscript and a predefined set of variables can be used to allow interaction with the RBscript.

Name	Type	Description
ClearRect	C	Clears an area of the canvas to the colour of the parent window background.. ClearRect(x,y,width,height)
DrawCautionIcon	C	Draws the caution icon at the specified coordinates. DrawCautionIcon(x,y)
DrawLine	C	Draws a line from x1,y1 to x2,y2 in the current colour. DrawLine(x1,y1,x2,y2)
DrawNoteIcon	C	Draws the note icon at the specified coordinates. DrawNoteIcon(x,y)
DrawOval	C	Draws the outline of an oval in the current colour. DrawOval(x,y,width,height)
DrawRect	C	Draws the outline of a rectangle in the current colour. DrawRect(x,y,width,height)
DrawRoundRect	C	Draws the outline of a rounded rectangle in the current colour. DrawRoundRect(x,y,width,height,ovalwidth)
DrawStopIcon	C	Draws the stop icon at the specified coordinates. DrawStopIcon(x,y)
DrawString	C	Draws the text at the specified coordinates. DrawString(string,x,y)

DrawStringWrap	C	Draws the text at the specified coordinates using wrap defined by the wrap width. DrawStringWrap(string,x,y,wrapwidth)
FillOval	C	Draws an oval filled with the current colour. FillOval(x,y,width,height)
FillRect	C	Draws a rectangle filled with the current colour. FillRect(x,y,width,height)
FillRoundRect	C	Draws a rounded rectangle filled with the current colour. FillRoundRect(x,y,width,height,ovalwidth)
GetHeightFN	F	Returns the height of the canvas. Put GetHeightFN into hgt
GetPixelFN	F	Returns the colour of the specified pixel. Put GetPixelFN(x,y) into colr
GetTextFontFN	F	Returns the name of the font being used. Put GetFontFN into fname
GetPixelFN	F	Returns the colour of the specified pixel. Put GetPixelFN(x,y) into colr
GetTextSizeFN	F	Returns the size of the text. Put GetTextSizeFN(x,y) into tsize
GetWidthFN	F	Returns the width of the canvas. Put GetWidthFN into hgt
PlotPoint	C	Plots a point at the specified coordinates in the current colour. PlotPoint(x,y)
RefreshCanvas	C	Forces OS X to refresh the screen. On Macintosh OS 9

		and Windows it has no effect. RefreshCanvas
RndDbFN	F	Returns a random double value in the range 0 <= r < 1 Put RndDbFN into field 2
RndRangeFN	F	Returns a random integer in the range min <= r <= max Put RndRangeFN(-13,123) into field 2
SeedFN	F	Returns the value of the seed for the random number generator. When RBscript first runs the seed and random number generator are unset. Put SeedFN into field 2
SetBold	C	Sets the text bold mode either on or off. SetBold(boolean)
SetForeColor	C	Sets the forecolour. SetForeColor(colr)
SetItalic	C	Sets the text italic mode either on or off. SetItalic(boolean)
SetOldRenderer	C	Sets the graphic rendering mode to either old(Classic) or new(Quartz) as on OS X. Using the old render on OS X is much faster. This has no effect on Mac Classic or Windows. SetOldRenderer(boolean)
SetPenHeight	C	Sets the height of the pen. SetPenHeight(psize)
SetPenWidth	C	Sets the width of the pen. SetPenWidth(psize)
SetPixel	C	Sets the colour of the specified pixel

		SetPixel(x,y,colr)
SetSeed	C	Seeds the random number generator using the specified integer value. If the value is 0 then the random number generator is seeded with the current time. SetSeed(2561)
SetTextFont	C	Sets the text font. SetTextFont(fname)
SetTextSize	C	Sets the size of the text. SetTextSize(tsize)
SetUnderline	C	Sets the text underline mode either on or off. SetUnderline(boolean)
StringHeightFN	F	Returns the height of the specified text using the wrap value in pixels. Put StringHeightFN(string,wrapval) into hgt
StringWidth	C	Returns the width of the specified text. Put StringWidthFN(string) into width

Neural Network Interface

Procedures & Functions

These procedures and functions provide a way of passing information to and from an RBscript. They are very general in nature and are intended to be wrapped in developer plug-in defined procedure and functions.

Most neural network systems need input, output, training and testing parameters. They also store their weights, log intermediate training results and final results.

This interface provides a way of encapsulating functionality and data, so making it easier to develop neural networks, both from within plug-ins and from within HyperNext Creator using HyperNext script.

Name	Type	Description
RBsSetGraphSpec	C	Sets the graph or chart specification variable for the specified canvas using the given text. RbsSetGraphSpec(canvasid,txt) String variable - gbGraphSpec
RBsSetNeuralSpec	C	Sets the neural network specification variable for the specified canvas using the given text. RbsSetNeuralSpec(canvasid,txt) String variable - gbNeuralSpec
RBsSetTrainSpec	C	Sets the training specification variable for the specified canvas using the given text. RbsSetTrainSpec(canvasid,txt) String variable - gbTrainSpec
RBsSetQuerySpec	C	Sets the query specification variable for the specified canvas using the given text. RbsSetQuerySpec(canvasid,txt) String variable - gbQuerySpec
RBsSetLearnSpec	C	Sets the learning specification variable for the

		specified canvas using the given text. RbsSetLearnSpec(canvasid,txt) String variable - gbLearnSpec
RBsSetTrainingData	C	Sets the training data variable for the specified canvas using the given text. RbsSetTrainingData(canvasid,txt) String variable – gbTrainingData
RBsSetValidationData	C	Sets the validation data variable for the specified canvas using the given text. RbsSetValidationData(canvasid,txt) String variable – gbValidationData
RBsSetAnalysisData	C	Sets the analysis data variable for the specified canvas using the given text. RbsSetAnalysisData(canvasid,txt) String variable – gbAnalysisData
RBsSetWeights	C	Sets the weights data variable for the specified canvas using the given text. RbsSetTrainingWeights(canvasid,txt) String variable – gbWeights
RBsSetCommand	C	Sets the command variable for the specified canvas using the given text. RbsSetCommand(canvasid,txt) String variable – gbCommand
RBsWeightsFN	F	Returns the weights data for the specified canvas . Put RbsWeightsFN(canvasid) into wstate String variable – gbWeights
RBsResultsFN	F	Returns the results data for the specified canvas .

		Put RbsResultsFN(canvasid) into results String variable – gbResults
RBsLogFN	F	Returns the log data for the specified canvas . Put RbsLogFN(canvasid) into runlog String variable – gbLog

Variables

These variables are only directly accessible from within an Rbscript. From HyperNext they can only be set or interrogated using one of the procedures and functions listed above.

Name	Rbscript Data Type	Description
gbAnalysisData	string	Holds the analysis data.
gbCommand	string	Holds the command data.
gbEscape	boolean	Set to true when the Escape key has been pressed AND gbYield is set to true, otherwise it remains unchanged.
gbGraphSpec	string	Holds the graph specification.
gbLearnSpec	string	Holds the learning specification.
gbLog	string	Holds the log data resulting from a run.
gbNeuralSpec	string	Holds the neural network specification.
gbQuerySpec	string	Holds the query specification.
gbResults	string	Holds the results output by the neural network.
gbState	string	Holds the general state.
gbTrain	string	Holds the train command.
gbTrainingData	string	Holds the training data.
gbTrainSpec	string	Holds the training specification.

gbValidationData	string	Holds the validation data.
gbWeights	string	Holds the neural network weights.
gbYield	boolean	When set to true allows the Escape key to be checked.

18 Image Banks

HyperNext has a set of image banks allowing images to be stored, retrieved and loaded/saved to and from files. The images can also be used in sprite animations and copied to/from canvases. Each bank holds one image and the number of banks allowed is dicated by the memory available.

Files used by image banks can be either local or absolute. Usually it is best to use local files.

There are also two sets of commands for saving/loading images. One is for cross-platform use and the other for standard image formats such as jpegs, picts etc.

Commands & Functions

Name	Type	Description
ImageBankReset	C	Resets the number of image banks to zero and clears all images. ImageBankReset
ImageBankReserve	C	Reserves a specified number of image banks in advance. ImageBankReserve(10)
ImageBankCountFN	F	Returns the number of currently allocated image banks. Put ImageBankCount FN into npics
ImageCanvasToBank	C	Copies the specified canvas to the specified image bank. When the scale value is nonzero then the image will be scaled to fit the image bank otherwise its size will remain unchanged. ImageCanvasToBank(cid,bid,scale) ImageCanvasToBank(5,2,1)
ImageCanvasToBankArea	C	Copies an area of the specified canvas to an area of the specified image bank.

		ImageCanvasToBankArea(cid,bid,x1s,y1s,ws,hs,x1d,y1d,wd,hd) where cid - canvas number bid - image bank number x1s - source x coord y1s - source y coord ws - source width hs - source height x1d - destination x coord y1d - destination y coord wd - destination width hd - destination height ImageCanvasToBankArea(5,2,10,10,50,50,90,90,50,50)
ImageBankToCanvas	C	Copies the specified image bank to the specified canvas. When the scale value is nonzero then the image will be scaled to fit the canvas otherwise its size will remain unchanged. ImageBankToCanvas(bid,cid,scale) ImageBankToCanvas(2,5,1)
ImageBankToCanvasArea	C	Copies an area of the specified image bank to an area of the specified canvas. ImageBankToCanvasArea(bid,cid,x1s,y1s,ws,hs,x1d,y1d,wd,hd) ImageBankToCanvasArea(2,5,10,10,50,50,90,90,50,50)
ImageBankLoad	C	Loads an image from the specifed local file into an image bank. When the scale value is nonzero then the image will be scaled to fit the bank otherwise its size will remain unchanged. ImageBankLoad(bid,fname,scale) ImageBankLoad(1,filename,0)
ImageBankLoadAbs		Loads an image from the specifed absolute file

		into an image bank. When the scale value is nonzero then the image will be scaled to fit the bank otherwise its size will remain unchanged. ImageBankLoadAbs(bid,fname,scale)
ImageBankLoadAsk	C	Loads an image using the file-open dialog box into an image bank. When the scale value is nonzero then the image will be scaled to fit the bank otherwise its size will remain unchanged. ImageBankLoadAsk(bid,scale) ImageBankLoadAsk(1,0)
ImageBankSave	C	Saves an image from an image bank to the specifed local file. ImageBankSave(bid,fname) ImageBankSave(1,filename)
ImageBankSaveAbs	C	Saves an image from an image bank to the specifed absolute file. ImageBankSaveAbs(bid,fname) ImageBankSaveAbs(1,filename)
ImageBankSaveAsk	C	Saves an image from an image bank to the file selected using the file dialog box. ImageBankSaveAsk(bid) ImageBankSaveAsk(1)
ImageBankLoadXP	C	Loads an image from the specifed local file into an image bank. When the scale value is nonzero then the image will be scaled to fit the bank otherwise its size will remain unchanged. The image must be in the HyperNext cross-platform format. ImageBankLoadXP(bid,fname,scale) ImageBankLoadXP(1,filename,0)
ImageBankLoadXPAbs	C	Loads an image from the specifed absolute file into an image bank. When the scale value is nonzero then the image will be scaled to fit the bank otherwise its size will remain unchanged. The image must be in the HyperNext cross-

		platform format. ImageBankLoadXPAbs(bid,fname,scale) ImageBankLoadXPAbs(1,filename,0)
ImageBankLoadXPAsk	C	Loads an image using the file-open dialog box into an image bank. When the scale value is nonzero then the image will be scaled to fit the bank otherwise its size will remain unchanged. The image must be in the HyperNext cross-platform format. ImageBankLoadXPAsk(bid,scale) ImageBankLoadXPAsk(1,0)
ImageBankSaveXP	C	Saves an image from an image bank to the specifed local file using the HyperNext cross-platform format. ImageBankSaveXP(bid,fname) ImageBankSaveXP(1,filename)
ImageBankSaveXPAbs	C	Saves an image from an image bank to the specified absolute file using the HyperNext cross-platform format. ImageBankSaveXPAbs(bid,fname)
ImageBankSaveXPAsk	C	Saves an image from an image bank to the file selected using the file dialog box. The image is in the HyperNext cross-platform format. ImageBankSaveXPAsk(bid) ImageBankSaveXPAsk(1)
ImageBankValidFN	F	Returns the value 1 if the specified image bank holds an image. Put ImageBankValidFN(bid) into okay
ImageBankSetSizeFN	F	Returns 1 if the specified image bank was successfully set to the given size. ImageBankSetSizeFN(bid,width,height) Put ImageBankSetSize(bid,800,600) into okay

ImageBankWidthFN	F	Returns the width of the specified image bank. Put ImageBankWidthFN(bid) into width
ImageBankHeightFN	F	Returns the height of the specified image bank. Put ImageBankHeightFN(bid) into height
ImageBankFileNameFN	F	Returns the filename of the file last accessed using any of the image bank file commands. Put ImageBankFileNameFN into fname
ImageBankPathNameFN	F	Returns the pathname of the file last accessed using any of the image bank file commands. Put ImageBankPathNameFN into fname
ImageBankRotateLeft	C	Looked at from the top of the image, the whole image is rotated left by 90 degrees. The image will be resized to match the rotation. ImageBankRotateLeft(bid)
ImageBankRotateRight	C	Looked at from the top of the image, the whole image is rotated right by 90 degrees. The image will be resized to match the rotation. ImageBankRotateRight(bid)
ImageBankRotateDown	C	The whole image is rotated left by 180 degrees. ImageBankRotateDown(bid)
ImageBankFlipH	C	Flips the image horizontally. ImageBankFlipH(bid)
ImageBankFlipV	C	Flips the image vertically. ImageBankFlipV(bid)
ImageBankStringFN	F	This functions takes the specified image bank and returns it in string or text form. It is useful when an image needs to be sent across a network or when a set of images needs to be stored in one file. Put ImageBankStringFN(bid) into var

		@ Store image bank 5 in variable idata Put ImageBankStringFN(5) into idata
ImageBankFromString	C	This takes a string and converts it into an image placing it in the specified image bank. ImageBankFromString(bid,var) @ Restore image bank 2 from string svar ImageBankFromString(2,svar)
ImageBankClear	C	Clears the image from the specifed image bank so freeing up memory. This also sets the image bank status to invalid. ImageBankClear(bid)

19 Sprite Animation

HyperNext programs can have one animation area where sprite objects can be placed, animated and moved. The drawing of sprite objects within the animation area is controlled automatically by the animation area itself. An animation area can also have a backdrop image attached.

A user simply creates a sprite, attaches it to the animation surface and then controls the sprite coordinates. Smooth movement can be achieved as the animation surface itself handles redrawing, collision detection and sprite priorities.

Commands & Functions

Name	Type	Description
AnimationSetDepth	C	Sets the color depth of the animation area. Valid values are 0, 8, 16 or 32. If 0 is selected then the animation area automatically sets its color depth to match that of the screen depth. AnimationSetDepth(32)
AnimationShow	C	Makes the animation area visible on the card or window. AnimationShow
AnimationHide	C	Hides the animation area on the card or window. AnimationHide
AnimationEnable	C	Enables the animation area so that sprites and other animation can take place. AnimationEnable
AnimationDisable	C	Disables the animation area so that sprite movement and other animation cannot occur. AnimationDisable
AnimationSetCoords	C	Sets the x and y coordinates of the animation area. AnimationSetCoords(x,y)

AnimationSetSize	C	Sets the width and height of the animation area. AnimationSetSize(width,height)
AnimationSetPeriod	C	Sets how often in milliseconds the animation area will be updated. AnimationSetPeriod(period)
AnimationClear	C	Clears the animation area removing all sprites but leaving the backdrop image showing. AnimationClear
AnimationBackdrop	C	Sets the specified bank image as the backdrop for the animation. AnimationBackdrop(bid)
AnimationRun	C	Starts the animation running continuously. AnimationRun
AnimationRunOnce	C	Runs the animation just once. AnimationRunOnce
AnimationUpdate	C	Cause the animation to update now and does not wait for the animation timer. AnimationUpdate
AnimationStop	C	Stops or pauses the animation. AnimationStop
AnimationDepthFN	F	Returns the current color depth setting of the animation. Put AnimationDepthFN into depth
AnimationLeftFN	F	Returns the left hand side or x coordinate of the animation area. Put AnimationLeftFN into x
AnimationTopFN	F	Returns the top side or y coordinate of the animation area. Put AnimationTopFN into y
AnimationWidthFN	F	Returns the width of the animation area. Put AnimationWidthFN into width

AnimationHeightFN	F	Returns the height of the animation area. Put AnimationHeightFN into height
AnimationMouseFN	F	Returns a list comprising which sprite objects were under the mouse pointer when it was clicked. Put AnimationMouseFN into slist

20 Sprite Objects

Sprites are objects comprising an image and other attributes such as coordinates, size and direction that can be automatically moved around by the animation area.

Commands & Functions

Name	Type	Description
SpriteCreateFN	F	Creates one sprite and automatically attaches it to the animation area. If the sprite was successfully created it will return the sprite identity otherwise it will return zero. SpriteCreateFN(bimage,x,y,priority,group,direct) Put SpriteCreateFN(1,50,59,1,1,1) into sid where bimage - the bank holding the required image. x - the x coordinate where the sprite will be placed. y - the y coordinate where the sprite will be placed. priority - the order in which sprites are refreshed. group - the group which the sprite belongs to as used by the collisiion handler. if negative - the sprite can collide with any other sprite except those in group 0. if positive - the sprite can collide with any other sprite except those in group 0 and its own group. direct - a value defined by the user indicating the sprites movement direction usually 8 directions are used as in main compass points.
SpriteReCreateFN	F	Similar to the SpriteCreateFN function except it updates the properties of the given sprite. This can be used to animate a sprite with different images.

		SpriteReCreateFN(sid,bid,x,y,priority,group,dir) Put SpriteReCreateFN(1,1,50,59,1,1,1) into sid
SpriteClose	C	Detaches the sprite from the animation area. SpriteClose(sid)
SpriteStateFN	F	Returns the state of the specified sprite. Values can be 0 - the sprite does not exist 1 - the sprites exists and is active (attached) 2 - the sprite exista and is inactive (unattached) Put SpriteStateFN(sid) into state
SpriteSetCoords	C	Sets the coordinates of the specified sprite. SpriteSetCoords(sid,x,y)
SpriteSetPriority	C	Sets the priority of the specified sprite. SpriteSetPriority(sid,priority)
SpriteSetGroup	C	Sets the group of the specified sprite. SpriteSetGroup(sid,group)
SpriteSetDirection	C	Sets the direction of the specified sprite. SpriteSetDirection(sid,direct)
SpriteCountFN	F	Returns the number of existing sprites. Put SpriteCountFN into nsprites
SpriteFetchBank	C	Assigns the specified bank image to the sprite. SpriteFetchBank(sid,bid)
SpriteXFN	F	Returns the x coordinate of the specified sprite. Put SpriteXFN(sid) into x
SpriteYFN	F	Returns the y coordinate of the specified sprite. Put SpriteYFN(sid) into y

SpriteWidthFN	F	Returns the width of the specified sprite. Put SpriteWidthFN(sid) into width
SpriteHeightFN	F	Returns the height of the specified sprite. Put SpriteHeightFN(sid) into height
SpritePriorityFN	F	Returns the priority of the specified sprite. Put SpritePriorityFN(sid) into priority
SpriteGroupFN	F	Returns the group of the specified sprite. Put SpriteGroupFN(sid) into group
SpriteDirectionFN	F	Returns the direction of the specified sprite. Put SpriteDirectionFN(sid) into direct
SpriteCollisionFN	F	Returns a list of any sprites which have collided. If the list is empty then no collisions occurred. Put SpriteCollisionFN into clist
SpriteMouseHitFN	F	Returns a list of any sprites under the mouse pointer when it was clicked. If the list is empty then sprites were hit. Put SpriteMouseHitFN into clist
SpriteKeyTestFN	F	Returns whether the specified key was pressed or not. If equal to 1 then the key was pressed else returns zero. Put SpriteKeyTestFN(key) into kval

21 AppleScript

AppleScript is a scripting language that is understood by all modern Macintosh computers and allows you to control the computer without using the keyboard or mouse. There are times when HyperNext might not have the required functionality and perhaps Applescript might provide an alternative solution.

Many applications also understand AppleScript and some can be controlled from your HyperNext stack or standalone.

At the present time only one AppleScript can be active at any one time. So if you have several AppleScripts then each must be executed before the others. For instance

```
AScriptExecute(script1)

AScriptExecute(script2)
```

NOTE
The commands AScriptCompile and AScriptRun are now obsolete and have been replaced by the single command AScriptExecute.

Example

The first text is an AppleScript that tells the Finder(Mac OS) to Beep.

```
Tell application “Finder”
   Beep
End Tell
```

Assuming the above AppleScript is in field 1 then the HyperNext code might be

```
Local src1, errflag

Put field 1 into src1

AScriptExecute(src1)
Put AScriptErrLastFN into errflag
If errflag=0 Then
  Put AScriptResultFN into field 2
Else
   Beep
EndIf
```

Name	Type	Description
AScriptExecute	C	This takes an AppleScript, tries to compile it and if successful then executes it. Any compilation or runtime error will be flagged and can be checked using AScriptErrLastFN. The function AScriptErrWhereFN indicates where any error occurred, either in the compilation or execution phase. AScriptExecute(src1)
AScriptErrLastFN	F	This function can be used to check if there has been an error during compilation or during execution of the compiled AppleScript. If it is non zero than an error occurred. Put AScriptLastErrFN into errflag
AScriptErrWhereFN	F	Returns where any error occurred. When 1, the error occurred in the compilation phase, and when 2, in the execution phase. A zero value indicates that no error occurred. Put AScriptErrWhereFN into where
AScriptResultFN	F	Returns the result, if any, after running the AppleScript. If the result is textual then it will be surrounded by quotation marks. If numeric, it will just be a text string. Put AScriptResultFN into answer
AScriptErrKnownFN	F	Assuming that AScriptErrLastFN gives an error then this function will return 1 if the error is known otherwise it returns 0. Note, if the return value is 0 there may still be an error so AScriptErrLastFN should be checked first. Put AScriptErrKnownFN into errknown
AScriptErrNumberFN	F	The number or code of the error. Put AScriptErrNumberFN into errnum
AScriptErrMessageFN	F	A description of the error.

		Put AScriptErrMessageFN into errmess
AScriptPropCountFN	F	Returns the number of properties in the compiled AppleScript Put AScriptPropCountFN into nvars

AScriptPropNameFN	F	Returns the name of the property with the given index. Indexes range from 1 to the number of AppleScript properties.
		
		The following example builds a list of property names contained in a compiled AppleScript.
		
		Local cnt, nprops,namelist,n
		put '' into namelist
		put AScriptPropCountFN into cnt
		If cnt>0 Then
		For n=1 To cnt
		Put AScriptPropNameFN(n) after namelist
		EndFor
		EndIf

AScriptPropValueFN	F	Returns the value of the property when given a property name. Put AScriptPropValueFN(pname) into answer
AScriptSetPropValue	C	This sets the value of the named property. AScriptSetPropValue pname,value

22 Apple Events

Apple Events are a way of communicating with the Operating System or other applications.

A current limitation of HyperNext is that it cannot yet handle the folder items returned by some Apple Events.

Example - Eudora

The following example checks the In box of the Eudora e-mail application.

```
Local obj1,obj2,obj3,u,i,res
Put AEGetNamedObjDescFN('euMB',0,'In') into obj1
NewAppleEvent('core','cnte','CSOm')
AESetObjSpecParam('----',obj1)
AESetMacTypeParam('kocl','euMS')
Put AESendFN into res

If res=1 Then
   Put AEReplyIntegerFN into u
   NewAppleEvent('core','getd','CSOm')
   for i=1 to u
      Put AEGetIndexedObjDescFN('euMS',obj1,i) into obj2
      Put AEGetPropertyObjDescFN(obj2,'euSu') into obj3
      AESetObjSpecParam('----',obj3)
      Put AESendFN into res
      If res=1 Then
         Put AEReplyStringFN after field 1
      EndIf
   EndFor
EndIf
```

Example - String Tx

```
@ Apple Event Tx 0- sending string

Local obj1,obj2,obj3
Local mess,u,i,res,creatorcode
Put field 1 into mess
Put 'ttxt' into creatorcode
NewAppleEvent('user','user',creatorcode)
AESendString mess
put AESendFN into res
If res=1 Then
    put AEReplyStringFN into field 2
    Beep
EndIf
```

23 Serial Ports

The following commands and functions allow a HyperNext program to control the serial ports on Windows, Macintosh OS X and OS 9 platforms. This allows a HyperNext program to receive data from a wide range of serial devices such as Global Positioning Systems and laboratory devices like digital multi-meters. It also allows control of devices such as DC electric motors, stepper motors and any other device employing an RS-232 serial controller.

The commands have been tested on the Windows platform with both the standard D9 serial connector and Keyspan USB to serial converter module. They have also been tested with the Keyspan on both Macintosh OS X and OS 9 platforms.

Each serial port can trigger two types of event, an error event and a data ready event. When either event is triggered then the Serial handler will be called. The serial handler script is accessible via the Edit Menu: Serial Ports option. Once triggered, the associated serial port can be found using the SerialIdentityFN or the SerialErrorFN functions.

Name	Type	Description
SerialSetPort	C	Assigns the numbered physical port to the specified serial port. SerialSetPort(snum,pnum)
SerialSetBaud	C	Sets the rate at which data will be sent and received through the specified port. The speed is measured in baud and a list of possible values is given below. SerialSetBaud(snum,value) Possible baud rates:- 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 14400, 19200, 28800, 38400, 57600, 115200, 230400
SerialSetDataBits	C	Sets the number of data bits to be used by the specified port. Possible values are 5, 6, 7 or 8 data bits. SerialSetDataBits(snum,value)
SerialSetParity	C	Sets the parity of the specified paort. Possible values are shown below.

		SerialSetParity(snum,value) Possible parity values:- 0 - No parity 1 - Odd parity 2 - Even parity
SerialSetStopBits	C	Sets the number of stop bits for the specified port. Possible values are 1, 1.5, and 2 data bits. SerialSetStopBits(snum,value)
SerialEnableCTS	C	Enables or disables CTS flow control for the specified port. A value of 1 enables and 0 disables it. SerialEnableCTS(snum,value)
SerialSetDTR	C	Sets the state of the Data Terminal Ready line for the specified port to either 0 or 1. SerialSetDTR(snum,value)
SerialEnableDTR	C	Enables or disables DTR flow control for the specified port. A value of 1 enables and 0 disables it. SerialEnableDTR(snum,value)
SerialSetRTS	C	Sets the state of the Request To Send line for the specified port to either 0 or 1. SerialSetRTS(snum,value)
SerialEnableXON	C	Enables or disables XON flow control for the specified port. A value of 1 enables and 0 disables it. SerialEnableXON(snum,value)
SerialClose	C	Closes the specified serial port. SerialClose(snum)
SerialClearBreak	C	Immediately clears the break signal on the specified port without the need to call SerialReset. SerialClearBreak(snum)

SerialFlushBuffer	C	Clears all of the data from the buffer of the specified serial port. SerialFlushBuffer(snum)
SerialLeaveDTROnClose	C	Tells the serial port controller not to negate Data Terminal Ready on closing it. This only has an effect when the serial port is open. SerialLeaveDTROnClose(snum)
SerialPoll	C	Updates the properties of the specified serial port and if any data is available then the serial port data event will be triggered. SerialPoll(snum)
SerialReset	C	Resets the byte format and baud rate of the specified port assuming that it is already open. SerialReset(snum)
SerialSetBreak	C	Immediately sets the break signal on the specified serial port without the need to call SerialReset. SerialSetBreak(snum)
SerialWrite	C	Writes the given text to the specified serial port. SerialWrite(snum,text)
SerialTxWait	C	Waits until all data written to the specified serial port by the SerialWrite command has been sent. SerialTxWait(snum)
SerialIdentityFN	F	Returns the identity of the serial port which triggered the serial port event. After interrogating this function its value is reset to zero. Put SerialIdentityFN into snum
SerialErrorFN	F	Returns a non-zero value if a serial port error occured. The non-zero value indicates which port flagged an error event. After interrogating this function its value is reset to zero. Put SerialErrorFN into snum

SerialPortCountFN	F	Returns the number of serial ports available on the computer. Put SerialPortCountFN into nports
SerialPortNamesFN	F	Returns a list of serial port names on the computer with each entry being on a separate line. The list can then be searched for a particular port name and if found then its line number can be passed to the SerialSetPort command prior to the port being opened. Put SerialPortNamesFN into portnames
SerialPortMaxSpeedsFN	F	Returns a list of the rated baud speeds of all serial ports on the computer with each entry being on a separate line. Put SerialPortMaxSpeedsFN into maxspeeds
SerialPortRatedSpeedsFN	F	Returns a list of the rated baud speeds of all serial ports on the computer with each entry being on a separate line. Put SerialPortRatedSpeedsFN into rspeeds
SerialPortInputDriversFN	F	Returns a list of Input driver names on the computer with each name being on a separate line. Put SerialPortInputDriversFN into dinNames
SerialPortOutputDriversFN	F	Returns a list of Output driver names on the computer with each name being on a separate line. Put SerialPortOutputDriversFN into doutNames
SerialOpenPortFN	F	Tries to open the specified serial port. If the port was successfully opened then 1 is returned otherwise 0. Put SerialOpenPortFN(snum) into okay
SerialPortStateFN	F	Returns whether the specified port is open or closed. A value of 0 means closed, and 1 is open. Put SerialPortStateFN(snum) into popen

SerialLookAheadFN	F	Returns all the characters from the buffer of the specified port without deleting them. Put SerialLookAheadFN(snum) into stext
SerialReadBytesFN	F	Returns the specified number of characters from the buffer of the specified port and then deletes those characters from the buffer. Put SerialReadBytesFN(snum,nchars) into stext
SerialReadAllFN	F	Returns all the data in the buffer for the specified port and then deletes all characters from the buffer. Put SerialReadAllFN(snum) into stext
SerialCTSFN	F	Returns the state of the Clear To Send line for the specified port. Put SerialCTSFN(snum) into ctsVal
SerialDCDFN	F	Returns the state of the Data Carrier Detect line for the specified port. Put SerialDCDFN(snum) into dcdVal
SerialDSRFN	F	Returns the state of the Data Set Ready line for the specified port. Put SerialDSRFN(snum) into dsrVal
SerialRIFN	F	Returns the state of the Ring Indicator for the specified port. Put SerialRIFN(snum) into rival
SerialLastErrorFN	F	Returns the error state for the specified port. If non-zero then an error occurred. Put SerialLastErrorFN(snum) into enum
SerialMacInRefFN	F	Returns the Macintosh In Driver reference number for the specified port. Put SerialMacInRefFN(snum) into minnum
SerialMacOutRefFN	F	Returns the Macintosh Out Driver reference number for the specified port. Put SerialMacOutRefFN(snum) into moutnum

SerialBaudFN	F	Returns the baud rate setting for the specified port. Put SerialBaudFN(snum)
SerialDataBitsFN	F	Returns the number of data bits used for the specified port. Put SerialDataBitsFN(snum) in to dbits
SerialParityFN	F	Returns the parity setting used for the specified port. Put SerialParityFN(snum) into parity
SerialStopBitsFN	F	Returns the number of stop bits used for the specified port. Put SerialStopBitsFN(snum) into sbits

24 Menus

HyperNext enables the programmer to create and modify their own menus. In Design mode menus can be created and modifed using the Menu Designer which is available via the Windows menu.

The menu designer in HyperNext Creator allows new menu titles and their associated items to be created and edited. Each menu item has its own script and can also have a command key associated with it.

At the current time each menubar has a default set of menu items. If they are not required or are not suitable for the application or stack then they can be disabled at runtime. When disabled a menu item is greyed out and cannot be selected by the user.

Menu items can be referred to either by their name or else by both their menu title number and menu item index. If referred to by name, HyperNext will search through the menu bar titles and associated items until it finds the first matching name and will ignore any later items having the same name. If referred to by index then they will always be found, assuming that they exist.

Menu indexes start at zero, with the System menu having title index zero, the File menu title index 1, and the Edit menu title index 2 etc.
For example, usually the first item on the File menu is the New item. To refer to the New item use menu title index 2 and menu item 1.

When using the menu commands and functions, if the specified menu item does not exist then no action will take place and the command will fail silently.

Name	Type	Description
MenuCallName	C	Calls the script associated with the specified menu item name. MenuCallName(mname)
MenuCallNumber	C	Calls the script specified by both the menu title number and menu item index. MenuCallNumber(mnumber,mindex)
MenuDisableName	C	Disables the specified menu item having the specified name. MenuDisableName(mname)
MenuDisableNumber	C	Disables the menu item specified by both the menu title number and menu item index.

		MenuDisableNumber(mnumber,mindex)
MenuEnableName	C	Enables the specified menu item having the specified name. MenuEnableName(mname)
MenuEnableNumber	C	Enables the menu item specified by both the menu title number and menu item index. MenuEnableNumber(mnumber,mindex)
MenuChangeName	C	Changes the name of the named menu item. MenuChangeName(oldname,newname)
MenuChangeNumber	C	Changes the name of the menu item specified by both the menu title number and menu item index. MenuChangeName(mnumber,mindex,newname)
MenuTickName	C	Ticks the name of the named menu item. MenuTickName(name)
MenuTickNumber	C	Ticks the name of the menu item specified by both the menu title number and menu item index. MenuTickNumber(mnumber,mindex)
MenuUnTickName	C	Unticks the name of the named menu item. MenuUnTickName(name)
MenuUnTickNumber	C	Unticks the name of the menu item specified by both the menu title number and menu item index. MenuUnTickNumber(mnumber,mindex)
MenuChangeTitle	C	Changes the title of the menu specified by the menu title number. MenuChangeTitle(mnumber,title)
MenuOverride	C	Allows the programmer to override actions for items on the Edit menu and their associated actions such as Copy, Cut, Paste. When set to 1 the actions will not take place and when 0 they can happen. For instance, when 0 the Edit menu can copy text from the clipboard into a field but

		when overriden the text copy will not take place. MenuOverride(boolean)
MenuIDFN	F	Returns the title index of the menu item which has just been triggered. Put MenuIDFN into field 1
MenuItemFN	F	Returns the item index of the menu item which has just been triggered. Put MenuItemFN into field 1
MenuTitleCountFN	F	Returns the number of menu titles on the menu bar. Put MenuTitleCountFN into field 1
MenuItemCountFN	F	Returns the number of menu items on the menu specified by the title index. Put MenuItemCountFN(mnumber) into field 1
MenuTitleListFN	F	Returns a list of the titles on the menubar. Put MenuTitleListFN into field 1
MenuItemListFN	F	Returns a list of the menu item names located on the menu specified by the title index. Put MenuItemListFN(mnumber) into field 1
MenuNameCountFN	F	Returns the number of menu items having the specified name. Put MenuNameCountFN(name) into field 1
MenuNameExistsFN	F	Returns 1 if a menu item having the specified name exists, otherwise it returns 0. Put MenuNameExistsFN(name) into field 1
MenuNumberExistsFN	F	Returns 1 if a menu item specified by the given title index and item index exists, otherwise it returns 0. Put MenuNumberExistsFN(mnumber,mitem) into field 1
MenuNameEnabledFN	F	Returns 1 if a menu item having the specified

		name is enabled, otherwise it returns 0. Put MenuNameEnabledFN(name) into field 1
MenuNumberEnabledFN	F	Returns 1 if a menu item specified by the given title index and item index is enabled, otherwise it returns 0. Put MenuNumberEnabledFN(mnumber,mitem) into field 1
MenuNameTickedFN	F	Returns 1 if a menu item having the specified name is ticked, otherwise it returns 0. Put MenuNameTickedFN(name) into field 1
MenuNumberTickedFN	F	Returns 1 if a menu item specified by the given title index and item index is ticked, otherwise it returns 0. Put MenuNumberTickedFN(mnumber,mitem) into field 1

25 Registration

These Registration commands and functions make it easier for a HyperNext programmer to create their own user registration scheme. When the HyperNext stack or program is run then a user clicking the Register menu option will be offered a dialog box where they can enter their registration information into the appropriate fields.

The register dialog box holds ten fields where a user can enter their registration details. These fields are name, address1, address2, address3, date and five serial code fields. If the user presses the registration button then the data from these fields will be placed into relevant variables and a registration script will be called so that the data can be processed. However, if the user presses the cancel button then no further actions will be taken.

In a simple registration scheme the following fields might be paired:

Name - code 1
Address 1 - code 2
Address 2 - code 3
Address 3 - code 4
Date - code 5

where the Name data could be used to generate a serial for code 1 and if the user enters the correct name and matching code then they will be accepted as valid.

The user data can be accessed using several HyperNext functions but for security reasons once a function is called then its data is cleared so making it difficult for anyone to access the data by peeking into memory.

The registration dialog box can display details of the software developer including their name, web and email addresses. The background color of the dialog box can also be specified. It is also possible to display a text value holding the time before expiry or any other value specified by the program creator.

Name	Type	Description
RegSetWho	C	Sets the name of the software creator for display in the registration dialog box. RegSetWho(value)
RegSetWeb	C	Sets the web address of the software creator for display in the registration dialog box. RegSetWeb(value)

RegSetEmail	C	Sets the email address of the software creator for display in the registration dialog box. RegSetEmail(value)
RegSetTime	C	Sets the remaining time for display in the registration dialog box. RegSetTime(value)
RegSetColor	C	Sets the background color of the registration dialog box. The color values are in the range 0 to 255, where zero is darkest and 255 is lightest. RegSetColor(red,green,blue)
RegSetUser	C	Sets the user name to be displayed in the program after the registration is complete. It can have any text value. RegSetUser(value)
RegNameFN	F	Returns the text that the user entered into the name field of the registration dialog. Put RegNameFN into rname
RegAddressFN	F	Returns the text that the user entered into the specified address field of the registration dialog. There are three address fields. Put RegAddressFN(1) into addr1 Put RegAddressFN(2) into addr2 Put RegAddressFN(3) into addr3
RegDateFN	F	Returns the text that the user entered into the date field of the registration dialog. Put RegDateFN into dnum
RegCodeFN	F	Returns the text that the user entered into the specified code field of the registration dialog. There are five code fields. Put RegCodeFN(1) into code1 Put RegCodeFN(2) into code2 Put RegCodeFN(3) into code3 Put RegCodeFN(4) into code4 Put RegCodeFN(5) into code5

RegClearData	C	Clears all of the user data which was entered into the fields and function memory via the registration dialog. RegClearData

26 Encryption

These Blowfish keywords provide some basic encryption functionality enabling messages to be securely sent over the internet. Note, the Blowfish functions operate on blocks of text 8 bytes long therefore a text longer than this needs to be broken into blocks.

The MD5 hash functions allow a user to verify if a received text or file has been tampered with.

Name	Type	Description
BlowfishSetKey	C	Sets the key to be used by for encryption and decryption. BlowfishSetKey(keytext)
BlowfishEncryptFN	F	Returns the passed text message in an encrypted form. Put BlowfishEncryptFN(mess) into codedtext
BlowfishDecryptFN	F	Returns the passed encrypted text as readable text. Put BlowfishDecryptFN(codedtext) into mess
MD5binFN	F	Returns the passed text in a binary(byte) form. Note, the result should not be placed in a field as the field formatting may corrupt the value. Put MD5binFN(mess) into hashtext
MD5hexFN	F	Returns the passed text in a hexadecimal form. Note, the result can be placed in a field as the result has a text format. Put MD5hexFN(mess) into hashtext

Blowfish example

```
@ Encrypt
BlowfishSetKey('1234567890')
Put 'Hello' into mess
Put BlowfishEncryptFN(mess) into codedmess

@ Decrypt
BlowfishSetKey('1234567890')
Put BlowfishDecryptFN(codedmess) into newmess
Message newmess
```

27 Receiving Emails (POP3)

This set of commands and functions allows a HyperNext program to communicate with a POP3 mail server and allows emails plus their attachments to be received.

Note, several of these POP functions wait until either a response from the mail server arrives or until timeout occurs. The timeout period is measured in 60ths of a second.

The general procedure for communicating with a POP3 mail server is

1 - Login
2 - Get email count
3 - Receive any headers
4 - Receive any emails and delete from server if required.
5 - Logout

The headers specify who an email is from, its subject line, its sent date and its total size including attachments.

Attachments can be saved using standard HyperNext file commands. Most image attachments will be in Base64 format in which case they will need decoding before being displayed or saved to a file.

Name	Type	Description
PopLoginFN	F	This function logs in to the POP3 mail server and effectively locks the server until logout is performed. It returns 1 if login was successful otherwise it returns 0. PopLoginFN(port,addr,uname,pword,encrypt,timeout) where port - server port, usually 110 addr - server address uname - user name pword - user password encrypt - 1 for login encryption, 0 for no encryption twait - time to wait for server response before giving up @ Example Login to server Put

		PopLoginFN(port,addr,uname,pword,encrypt,twait) into res
PopCheckServerFN	F	This function returns 1 if the server is active and awaiting commands, otherwise it returns 0. PopCheckServerFN(timeout) Put PopCheckServerFN(200) into res
PopCountFN	F	Asks the server for the number of emails it is holding. PopCountFN(timeout) Put PopCountFN(200) into emailCount
PopRollbackFN	F	Asks the server to reset its state to that immediately after login. This command is useful when emails have been deleted and need restoring. If the rollback was successful then the function will return 1 otherwise it will return 0. PopRollbackFN(timeout) Put PopRollbackFN(200) into res
PopDisconnectFN	F	Asks the server to disconnect or logout. If the logout response is received from the server then the function will return 1 otherwise it will return 0. Note, Before quitting, HyperNext always closes the connection with the mail server so as to prevent the server being locked. PopDisconnectFN(timeout) Put PopDisconnectFN(200) into res
PopErrorCodeFN	F	This can be used to see if the mail server responded with an error. A response of 0 indicates no error. Related functions are PopErrorMessageFN and PopErrorMessageIDFN. This function is reset to 0 after being called. Put PopErrorCodeFN into res
PopErrorMessageFN	F	Gives the error message sent by the mail server. This function is cleared after being called. Put PopErrorMessageFN into field 1

PopErrorMessageIDFN	F	Gives the error message number sent by the mail server. This function is cleared after being called. Put PopErrorMessageIDFN into field 2
PopSendCommandFN	F	This function allows special commands to be sent to the mail server. If successful then the function will return 1 otherwise it will return 0. PopSendCommandFN(cmd,timeout) Put PopSendCommandFN(cmd,600) into res
PopCommandReplyFN	F	Returns any reply from the mail server in respone to the PopSendCommandFN function. This function is cleared after being called. Put PopCommandReplyFN into reply
PopCommandDataFN	F	Returns any data from the mail server in response to the PopSendCommandFN function. This function is cleared after being called. Put PopCommandDataFN into data
PopGetHeadersFN	F	This function asks the mail server to send any headers. If successful the function returns 1 otherwise it returns 0. There is one header for each email and the headers help decide whether an email should be retrieved or just deleted. The header information can be accessed using the following functions:- PopFromListFN, PopSubjectListFN, PopDateListFN and PopSizeListFN. Note, the timeout depends upon the expected maximum time needed to retrieve the emails. PopGetHeadersFN(timeout) Put PopGetHeadersFN(600) into headers
PopFromListFN	F	Returns in list form the FROM addresses of the emails. Line 1 is for email number 1 and so on. PopFromListFN Put PopFromListFN into fromList
PopSubjectListFN	F	Returns in list form the SUBJECT lines of the emails. Line 1 is for email number 1 and so on.

		PopSubjectListFN Put PopSubjectListFN into subjectList
PopDateListFN	F	Returns in list form the DATES of the emails. Line 1 is for email number 1 and so on. PopDateListFN Put PopDateListFN into dateList
PopSizeListFN	F	Returns in list form the size of each email. Line 1 is for email number 1 and so on. The size is in bytes and indicates whether the email has an attachment or not. The only way of knowing whether an email has an attachment is to retreive it. PopSizeListFN Put PopSizeListFN into sizeList
PopGetEmailFN	F	This fetches the email with the specified identity from the mail server. If successful it returns 1 else it returns 0. PopGetEmailFN(id,timeout) Put PopGetEmailFN(3,600) into res
PopDeleteEmailFN	F	This deletes the email with the specified identity from the mail server. If successful it returns the identity of the deleted email else it returns 0. PopDeleteEmailFN(id,timeout) Put PopGetEmailFN(3,600) into res
PopBodyPlainFN	F	This returns the plain text body of the last retrieved email. if it returns empty then the email had no plain text body although it might have a rich text or HTML body. PopBodyPlainFN Put PopBodyPlainFN into txt
PopBodyRichFN	F	This returns the rich text body of the last retrieved email. if it returns empty then the email had no rich text body.

		PopBodyRichFN Put PopBodyRichFN into txt
PopBodyHTMLFN	F	This returns the HTML body of the last retrieved email. if it returns empty then the email had no HTML body. PopBodyHTMLFN Put PopBodyHTMLFN into txt
PopBodySourceFN	F	This returns the complete source contents of the last retrieved email. The contents can include headers, body texts and attachments all joined together. PopBodySourceFN Put PopBodySourceFN into txt
PopAttachmentCountFN	F	This returns the number of attachments for the last retrieved email. Each attachment can be accessed using an integer identity starting at 1. PopAttachmentCountFN Put PopAttachmentCountFN into numAtts
PopAttachmentNameFN	F	This returns the name of the specified attachment for the last retrieved email. PopAttachmentNameFN(aid) Put PopAttachmentNameFN(1) into name
PopAttachmentEncodingFN	F	This returns the encoding of the specified attachment for the last retrieved email. If the attachment is an image it is usually encoded with Base64 encoding. PopAttachmentEncodingFN(aid) Put PopAttachmentEncodingFN(1) into encoding
PopAttachmentDataFN	F	This returns the data of the specified attachment for the last retrieved email. This data may be an encoded image, sound, movie or perhaps just a text. This data can be processed and saved in a file using the relevant HyperNext file commands. PopAttachmentDataFN(aid)

		Put PopAttachmentDataFN(1) into data
PopAttachmentMacTypeFN	F	This returns the Macintosh Type of the specified attachment for the last retrieved email. PopAttachmentMacTypeFN(aid) Put PopAttachmentMacTypeFN(1) into mtype
PopAttachmentMacCreatorFN	F	This returns the Macintosh Creator of the specified attachment for the last retrieved email. PopAttachmentMacCreatorFN(aid) Put PopAttachmentMacCreatorFN(1) into mcret
PopAttachmentMimeFN	F	This returns the Mime type of the specified attachment for the last retrieved email. PopAttachmentMimeFN(aid) Put PopAttachmentMimeFN(1) into mimetype
PopOneFromFN	F	Returns the FROM address of the specified email. PopOneFromFN(mid) Put PopOneFromFN(6) into fromaddr
PopOneSubjectFN	F	Returns the SUBJECT line of the specified email. PopOneSubjectFN(mid) Put PopOneSubjectFN(6) into subject
PopOneDateFN	F	Returns the DATE of the specified email. This date can be quite a complex string. PopOneDateFN(mid) Put PopOneDateFN(6) into datetxt
PopOneSizeFN	F	Returns the SIZE of the specified email in bytes. PopOneSizeFN(mid) Put PopOneSizeFN(6) into size
PopClearEmail	C	This clears the last email from memory and frees up space used by its bodies and attachments.

		PopClearEmail

28 Sending Emails (SMTP)

Managing the sending of emails is much easier than that of receiving them. Emails are sent using the HyperNext SMTP commands and functions. The program just needs to create an email, connect to the server, send the email and then disconnect.

Server login is not usually necessary because most email servers are provided by the user host Internet Service Provider. Note, that the SMTP and POP3 mail servers are unrelated.

Name	Type	Description
EmailSetServer	C	This command does not make the connection to the mail server it just provides the necessary details so that the EmailSendFN function can itself connect to the server. EmailSetServer(port,addr,uname,pword) where port - server port, usually 25 addr - server address uname - user name, not usually needed pword - user password, not usually needed
EmailDisconnect	C	This disconnects from the mail server. EmailDisconnect
EmailCancelSend	C	This command cancels any pending email transmission. EmailCancelSend
EmailCreateNew	C	Creates a new blank email with no attachments. EmailCreateNew
EmailSetBodyPlain	C	This sets the plain text body of the email. EmailSetBodyPlain(text)
EmailSetBodyRich	C	This sets the rich text body of the email. EmailSetBodyRich(text)
EmailSetBodyHTML	C	This sets the HTML text body of the email.

		EmailSetBodyHTML(text)
EmailSetFrom	C	This sets the FROM address of the email. EmailSetFrom(text)
EmailSetSubject	C	This sets the SUBJECT line of the email. EmailSetSubject(text)
EmailSetHeader	C	This sets the header of the email. A header is often used to identify the email client and has two part:- the header name and the header value. EmailSetHeader(hname,hvalue)
EmailSetTo	C	This sets the TO address of the email. If more than one address is required then the addresses should be given in standard HyperNext list form. EmailSetTo(textlist)
EmailSetToCC	C	This sets the TO CC addresses of the email. If more than one address is required then the addresses should be given in standard HyperNext list form. Using ToCC lets the recipients see who else received the email. EmailSetToCC(textlist)
EmailSetToBCC	C	This sets the TO BCC addresses of the email. If more than one address is required then the addresses should be given in standard HyperNext list form. Using ToBCC prevents the recipients seeing who else received the email. EmailSetToCC(textlist)
EmailAttach	C	This attaches a local file such as text or graphic to the email. Note, if an image is to be sent then its file should be encoded with Base64 before its filename/path is given otherwise the file will probably be corrupted during tranmission. EmailAttach(fname,name,encoding,MacT,MacC,Mime) where fname - the file to send name - the name the recipient sees

		encoding - any encoding used such as Base64 MacT - the MacType of the attachment, if any MacC - the MacCreator of the attachment, if any Mime - the MIME or multimedia type. If a parameter is not used just pass an empty string in its place.
EmailAttachAbs	C	This attaches a file specified by an absolute file path. Note, if an image is to be sent then its file should be encoded with Base64 before its filename/path is given otherwise the file will probably be corrupted during tranmission. EmailAttachAbs(fpath,name,encoding,MacT,MacC,Mime) where fpath - the path to the file name - the name the recipient sees encoding - any encoding used such as Base64 MacT - the MacType of the attachment, if any MacC - the MacCreator of the attachment, if any Mime - the MIME or multimedia type. If a parameter is not used just pass an empty string in its place.
EmailSendFN	F	This function connects to the server and sends the built email waiting either until the server responds or until a timeout occurs. It returns 1 if successful otherwise 0. If a very large email is to be sent then a longer timeout should be specified because the function might timeout and report failure even though the email is later sent successfully. EmailSendFN(timeout) Put EmailSendFN(600) into res
EmailServerResponseFN	F	This function returns the message sent by the server when the connection was first made. EmailServerResponseFN Put EmailServerResponseFN into mess
EmailConnectedFN	F	This function returns 1 if the server connection still exists otherwise it returns 0.

		EmailConnectedFN Put EmailConnectedFN into res
EmailSocketErrorFN	F	This function 0 if no connection error occurred otherwise it returns 1. A connection error will occur if the given server name does not exist. EmailSocketErrorFN Put EmailSocketErrorFN into res
EmailErrorCodeFN	F	This function returns the error code as reported by the server . If no error occured it returns 0. The error code is cleared after the value is returned. EmailErrorCodeFN Put EmailErrorCodeFN into res
EmailErrorMessageFN	F	This function returns the error message as reported by the server . If no error occured it returns an empty string. The error message is cleared after the value is returned. EmailErrorMessageFN Put EmailErrorMessageFN into errmess
EmailFindAddressesFN	F	This takes a block of text and attempts to return email addresses in list form. It uses the specified start and end separators in searching for the addresses. Email addresses in the FROM section of an email are ofter separated by '<' and '>' as in '<help@apple.com>'. This is useful in finding the FROM address when replying to an email. EmailFindAddressesFN(txt,sep1,sep2) Put EmailFindAddressesFN(txt,sep1,sep2) into addrlist

29 USB HID Comms

These commands and function give some basic support for USB communications with HIDs (Human Interface Devices). HIDs are devices such as mice, joysticks and certain electronics units have a maximum data rate of 64KB/sec.

USB HIDs are identified to software by a number of attributes. They should provide at least their Product and Vendor numbers. Other attributes are Manufacturer, Product Information, Version and Serial number. HyperNext provides these attributes in the form of lists so that the relevant connected devices can be found. HyperNext also tries to produce a unique hash code for each device that is useful in rapidly searching the lists.

To interact with these devices follow these steps:-

1. Find all connected devices --> a list of attributes
2. Select device using Product Vendor ID attributes
3. Connect to device
4. Communicate with device
5. Disconnect from device.

Note, USB supports hot plugging whereby devices can be plugged in and out while the computer is running. HyperNext caters for this. Difficulties can arise though if several devices having identical identifier attributes are plugged and unplugged because HyperNext has no way of knowing which device is which.

Note, Vendor and Product IDs are expected to be in decimal although in the literature they are often specified in Hexadecimal. Just use the HexFN function if conversion is necessary.

Name	Type	Description
USBResetFN	F	This function clears all connections including the USB data lists and then returns the number of devices that it disconnected. USBResetFN Put USBResetFN into num
USBFindAllFN	F	This function scans the USB ports for all devices and builds up the lists of device attributes. It returns the number of devices found. USBFindAllFN

		Put USBFindAllFN into dcount
USBCountFN	F	Simply returns the number of devices attached to the ports. It is much faster than the USBFindAllFN function because it does not build any lists. USBCountFN Put USBCountFN into dcount
USBUniquesFN	F	This function uses the device hashes to return the number of unique device types. If there are ten devices attached to the ports but 6 have identical attributes then this function will return the value 5. USBUniquesFN Put USBUniquesFN into num
USBGetSingleDeviceFNFN	F	This function returns the device number for a single device matching the passed criteria. It should be used when only one device matching the criteria is likely to be attached to the ports. If no matching device is found then the function returns 0. The device number returned can then be used to connect and communicate with the device. If some criteria are not needed then an empty string can be passed in their place. Note, when other devices are plugged or unplugged from the ports the scan list positions can change but the device number provided by this functions preserves a link to the required device. USBGetSingleDeviceFN(vend,prod,man,prodinfo,version,serial,hash) where vend - Vendor ID prod - Product ID man - Manufacture string prodinfo - Product Info version - Version number serial - device serial number hash - device hash string Put USBGetSingleDeviceFN(vend,prod,man,prodinfo,version,serial,hash) into devID
USBGetMatchingDevicesFN	F	This function returns a list of all devices matching the passed criteria. It should be used when several similar

		devices are expected to be attached to the ports. If no matching device is found then the function returns an empty list. The device numbers returned can then be used to connect and communicate with the device. If some criteria are not needed then an empty string can be passed in their place. USBGetMatchingDevicesFN(vend,prod,man,prodinfo,version,serial,hash) where vend - Vendor ID prod - Product ID man - Manufacture string prodinfo - Product Info version - Version number serial - device serial number hash - device hash string Put USBGetMatchingDevicesFN(vend,prod,man,prodinfo,version,serial,hash) into devList
USBValidFN	F	This function returns 1 if the passed USB device ID number is valid and 0 otherwise. USBValidFN(uid) Put USBValidFN(5) into okay
USBConnectFN	F	This function tries to connect to the specified USB device and returns 1 if successful and 0 otherwise. If the device does not exist it returns an empty value. USBConnectFN(uid) Put USBConnectFN(5) into okay
USBConnectedFN	F	This function returns 1 if the specified USB device is connected and 0 if not connected. If the device does not exist it returns an empty value. USBConnectedFN(uid) Put USBConnectedFN(5) into res
USBDisconnectFN	F	This function tries to disconnect the specified USB device and returns 1 if successful and 0 otherwise. If the device does not exist it returns an empty value.

		USBDisconnectFN(uid) Put USBDisconnectFN(5) into res
USBLastErrorFN	F	This function returns the last error for the specified USB device. If no error occured it returns 0. If the device does not exist it returns an empty value. Error numbers are Operating System specific. The error value is reset to zero after being read. USBLastErrorFN(uid) Put USBLastErrorFN(5) into enum
USBVendorFN	F	This function returns the Vendor ID for the specified USB device number. If the device does not exist it returns an empty value. USBVendorFN(uid) Put USBVendorFN(5) into vid
USBProductFN	F	This function returns the Product ID for the specified USB device number. If the device does not exist it returns an empty value. USBProductFN(uid) Put USBProductFN(5) into pid
USBManufacturerFN	F	This function returns any Manufacturer string for the specified USB device number. If the device does not exist it returns an empty value. USBManufacturerFN(uid) Put USBManufacturerFN(5) into manstr
USBProductInfoFN	F	This function returns any Product Information string for the specified USB device number. If the device does not exist it returns an empty value. USBProductInfoFN(uid) Put USBProductInfoFN(5) into pinfo
USBVersionFN	F	This function returns any Version string for the specified USB device number. If the device does not exist it returns an empty value. USBVersionFN(uid)

		Put USBVersionFN(5) into vers
USBSerialFN	F	This function returns any Serial Number string for the specified USB device number. If the device does not exist it returns an empty value. Note, the serial number is made up from byte values in the range 0 to 255 and so must be decoded using HyperNext byte functions if it is to display correctly. USBSerialFN(uid) Put USBSerialFN(5) into sernum
USBHashFN	F	This function returns the hash string for the specified USB device number. If the device does not exist it returns an empty value. USBHashFN(uid) Put USBHashFN(5) into hash
USBVendorListFN	F	This function returns the list of Vendor IDs for the attached devices. USBVendorListFN Put USBVendorListFN into vlist
USBProductListFN	F	This function returns the list of Product IDs for the attached devices. USBProductListFN Put USBProductListFN into plist
USBManufacturerListFN	F	This function returns the list of Manufacturer strings for the attached devices. USBManufacturerListFN Put USBManufacturerListFN into manlist
USBProductInfoListFN	F	This function returns the list of Product Information strings for the attached devices. USBProductInfoListFN Put USBProductInfoListFN into pinfolist
USBVersionListFN	F	This function returns the list of Version strings for the attached devices.

		USBVersionListFN Put USBVersionListFN into verlist
USBSerialListFN	F	This function returns the list of Serial Number strings for the attached devices. USBSerialListFN Put USBSerialListFN into serist
USBHashListFN	F	This function returns the list of Hash strings for the attached devices. USBHashListFN Put USBHashListFN into pinfolist
USBReadFN	F	This function tries to read a message string of the requested length from the specified USB device. It returns the message, if any. USBReadFN(uid,length) Put USBReadFN(5,9) into mess
USBWriteFN	F	This function tries to send a message to the specified USB device. It returns the number of bytes actually sent. USBWriteFN(uid,message) Put USBWriteFN(5,mess) into num

30 LightStone Bio-feedback

The LightStone is a USB HID biofeedback device that monitors heart rate and skin resistance. It is comes with the Wild Divine meditation and games software.

A HyperNext program can read and display the data simultaneously from several LightStone devices using both inbuilt generic USB functions or some HyperNext LightStone functions. The LightStone functions are optimised and much faster than HyperNext script. Speed is important because the LightStone sample rate is 30HZ.

These LightStone functions also allow some warnings for heart rates exceeding predefined ranges and for adjusting the heart rate peak detection algorithm.

The steps for using a LightStone device are:-

1. Find the device using Vendor Product IDs
2. Connect to it.
3. Reset the device data structures
4. Retrieve and make a data sample
5. Read a heart value and heart rate
6. Read a skin value
7. Goto step 4
8. When finished disconnect device

Name	Type	Description
LStoneResetFN	F	This function resets the channel and associated data structures for a specific LightStone device. If successful it returns 1 otherwise 0. If the device does not exist it returns an empty value. LStoneResetFN(id) Put LStoneResetFN(2) into res
LStoneHeartSetLeakFN	F	This function set the leakage parameter of the heart rate detection algorithm for the specified LightStone device. If successful it returns 1. If the device does not exist it returns an empty value. A leakage parameter is necessary because the DC level of the heart signal from a LightStone device can change so making peak detection less reliable. The leakage value starts at a default value of 0.01 and the effect of changing it can be monitored using the LStoneHeartTrackFunction. If the leakage is too low then peaks can be missed and if too high

		false peaks can be registered. LStoneHeartSetLeakFN(id,value) Put LStoneHeartSetLeakFN(1,0.02) into res
LStoneHeartSetRangeFN	F	This function set the acceptable ranges the heart rate detection monitor for the specified LightStone device. If successful it returns 1 but if the values are out of range such as -ve it returns 0. If the device does not exist it returns an empty value. The default range is 40 to 240 beats per minute. LStoneHeartSetRangeFN(id,low,high) Put LStoneHeartSetRangeFN(1,60,90) into res
LStoneHeartResetFN	F	This function resets the heart peak detection algorithm for the specified LightStone device. If successful the function returns 1 otherwise 0. If the device does not exist it returns an empty value. LStoneHeartResetFN(id) Put LStoneHeartResetFN(1) into res
LStoneMakeSampleFN	F	This function reads in a data sample from the device and produces readings for heart value, heart rate, heart peak and skin resistance value. These values can then be read by the relevant functions. If successful the function returns 1 otherwise 0. If the device does not exist it returns an empty value. LStoneMakeSampleFN(id) Put LStoneMakeSampleFN(id) into res
LStoneSkinValueFN	F	This function returns the skin resistance value reading for the last sample. If the device does not exist it returns an empty value. The range is roughly between 190 to 500 although it depends upon the person being monitored and their physical condition such as having poor circulation or being cold. In terms of relaxation, lower values mean more relaxed. LStoneSkinValueFN(id)

		Put LStoneSkinValueFN(1) into skinval
LStoneHeartValueFN	F	This function returns the heart value reading for the last sample. If the device does not exist it returns an empty value. The range is roughly between 1200 to 3200 although it depends upon the person being monitored. This value is the basis for the heart rate measurement and peak detection. LStoneHeartValueFN(id) Put LStoneHeartValueFN(1) into heartval
LStoneHeartRateFN	F	This function returns the heart rate reading in beats per minute for the last sample. If the device does not exist it returns an empty value. LStoneHeartRateFN(id) Put LStoneHeartRateFN(1) into heartrate Note, this function works in conjunction with the LStoneHeartSetRangeFN function as follows:- = 0 - no heart rate = -1 - heart rate out of range too low = -2 - heart rate out of range too high = otherwise the in range heart rate
LStoneHeartTrackFN	F	This function returns a track value releated to the heart value reading and leakage for the last sample. If the device does not exist it returns an empty value. It can be plotted to see if peaks are being missed or false peaks being recognised. LStoneHeartTrackFN(id) Put LStoneHeartTrackFN(1) into htrack
LStoneHeartPeakFN	F	This function returns 1 when a heart peak occured in the last sample otherwise it returns 0. If the device does not exist it returns an empty value. This can be used to sound a beep or trigger some other code when a heart peak occurs. LStoneHeartPeakFN(id) Put LStoneHeartPeakFN(1) into hpeak

LStoneSetVersionFN	F	This function sets the sampling version to be used for the specified LightStone device. It returns 1 when successful or 0 otherwise. If the device does not exist it returns an empty value. Currently there are two versions, namely 1 and 2. However, as different LightStone versions become available then other sampling versions may be required. Version 1 is default and gives out a wider range of skin values than version 2. LStoneSetVersionFN(id,version) Put LStoneSetVersionFN(5,2) into res

End of Guide