HAC help

Introduction

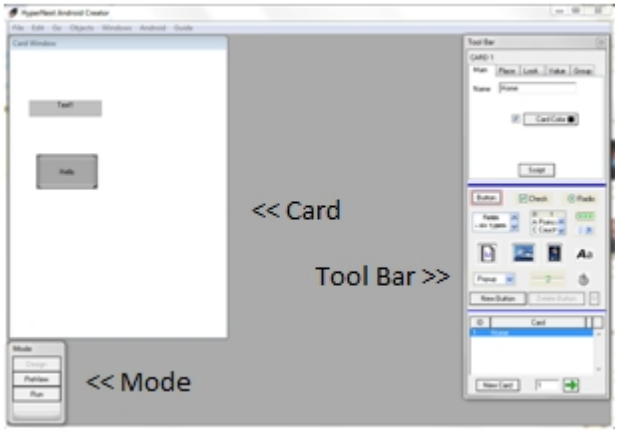
HAC v1.16

Welcome

HyperNext Android Creator (from now on referred to as HAC) is an easy to use software creation system that allows almost anyone to quickly start building their own Android applications. HAC currently only works under Windows (XP, Vista or Windows 7) but a Mac OS X version is in development.

The HAC interface has just one design window and simple toolbar plus three modes: Design, Preview and Run. Controls such as buttons can be rapidly placed and their English-like scripts edited with its easy to use Script Editor.

With HAC there is no need to learn Java and it is much easier to use than Eclipse or Netbeans.



HAC has online help that makes exploring the HyperNext language easy, lots of example programs can be downloaded from http://www.hypernextandroid.com/index.html, and internet forums where you can discuss HAC with other developers and users, http://www.hypernextandroid.com/forums/phpBB3/index.php. HAC is not only great for beginners but is powerful enough to produce more complex software with its flexible HyperNext scripting language.

HAC v1.16

Installing HAC

In order to develop with HAC you MUST have the Java JDK, the Android SDK, and Apple Quicktime installed on the computer on which you are using HAC. Java JDK, Android SDK and Apple Quicktime are free to download and to use.

The Java JDK can be downloaded from http://www.oracle.com/technetwork/java/javase/downloads/index.html
You want "Java Platform Standard Edition". Click on the download JDK button, select Windows in the list box and agree to to the license agreement, click the "Continue" button and download the file to your computer. If you are using 64 Bit Windows you still need the normal (32 Bit) Windows option not the Windows X64 one. When downloaded install JDK in the default location.

The Android SDK can be downloaded from http://developer.android.com/sdk/index.html
Click on the recommended installer .exe version for Windows and it will download to your computer. When downloaded install it to the following location, C:\android-sdk-windows, do not install in the default Program Files folder as the space in the filename can cause build problems.

The Android SDK will not install unless you have already installed the Java JDK, occasionally it will fail to see a successfully installed JDK, if this happens simply click the "Back" button and then "Next" and this usually fixes the problem.

Once installed the Android SDK window will open and allow you to choose which updates to install,
a: press the "Update all" button and a new window will open
b: check the "Accept all" radio button and press "Install", this can take anything up to a couple of hours depending on the speed of your internet connection and your computer.
c: when the updates are complete close the SDK window.

Apple Quicktime is used by HAC files running on your Windows computer, it can be downloaded from
http://www.apple.com/quicktime/download/
When downloaded install it to the default location.

Reinstalling/Uninstalling HAC

You can uninstall HAC from your computer in one of three ways. By selecting the Uninstall option from the HAC folder in the Windows Start Menu, by using the Programs/Uninstall section on the Windows Control Panel, and by running the HAC setup program and choosing Remove.

HAC does not delete the My Android or the My Android Projects folders in the Windows Documents folder, if you no longer need these you should delete them yourself.

Before reinstalling HAC you must first uninstall HAC, the quickest way to do this is simply to run HAC setup and choose Remove, then run HAC setup again to reinstall it. This also applies if you want to update HAC to a newer version.

WARNING - The HAC installer overwrites the demo projects in the My Android Projects folder, so if you have modified any of the demo projects and you want to keep the new versions, then you must either rename the project folders or backup and restore the projects after reinstalling HAC.

HAC paths

When HAC runs for the first time it tries to locate the required additional programs on your computer, if it fails to find any of these it will give an error message. To tell HAC where all the necessary files are use the "Android Paths" option found in the "Android" menu. For full functionality all paths should have a green check mark next to them, if it is a red cross click on the button on the left and browse to the correct folder.

Debuggable

Within HAC's "Android" menu there is the "Your Application" window, on here there is a "Debuggable" checkbox that sets a debuggable flag in your application's apk file. When checked the flag is set to TRUE and when unchecked the flag is set to FALSE.

If the debuggable flag is TRUE then the installing apk will always overwrite any existing apk on the sdcard regardless of their version numbers. When debuggable is set to FALSE then the installing apk will only overwrite an existing apk on the sdcard if the installing apk has a higher version number. Therefore when developing your app always have debuggable set to TRUE and set it to FALSE once your app is ready for release.

HAC for beginners

HAC

HyperNext Android Creator is a self contained and easy to use software creation program. It has an easy to use interface and builds standalone applications. HAC 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. HyperNext is the name of the programming language used in HAC.

The main aspects of the HAC 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 HAC

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.

Once HAC has loaded it will display two main 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.

Android emulator

To test your HAC programs on an Android emulator you must first create one, these emulators are known as Android Virtual Devices (AVD). To create an AVD,

1. Using the "Android" menu select the "Android SDK" option

2. The Android SDK and AVD Manager will open, click on the Virtual Devices entry in the list box on the left

3. Click the "New" button and enter the following details
Name - Android 22
Target - Android 2.2 - API Level 8
Size - 32
Skin - click the Resolution radio button and type the numbers 320 and 480 into the 2 boxes

4. Click the "Create AVD" button. If all goes well you should have a new AVD showing in the AVD list

5. Close the Android SDK and AVD Manager window by clicking on the red close box in the top right corner

6. To run the new AVD select "Launch Emulator" from the "Go" menu.

Hello World

Traditionally the program written by programmers when they use a development system for the first time is a "Hello World" program, the most simple program that does nothing more than print "Hello World" on the screen.

Lets produce a "Hello World" program in HAC

1. Run HAC and start a new project by clicking the "New" item in the "File" menu. When the "Save As" dialogue opens call your new project "Hello World".

2. We will set the card size to match a typical Android phone, 320 X 480 pixels

a. on the Tool Bar on the right select the "Place" tab
b. type 320 into the "Width" box
c. type 480 into the "Height" box.

3. We will put a button on the screen that when clicked will print "Hello World"
a. on the Tool Bar click on the button marked "Button"
b. on the Tool Bar click on the button marked "New Button"
c. click on the Card Window where you want the button to be placed.

4. We will now ad the "Hello World" script to the button
a. click on the button on the card and it will be highlighted
b. select the "Main" tab on the Tool Bar
c. click the "Script" button and the Script Editor will open
d. click in the large edit area on the right
e. on a new line type Message 'Hello World'
f. close the Script Editor.

5. Test your program within HAC for bugs
a. on the "Mode" window click the "Run" button
b. a Splash screen should appear, click on it to dismiss it
c. your card containing the button should appear
d. click the button and a dialogue box should appear displaying "Hello World"
e. click the "OK" button to close the dialogue box, to return to Design mode click the "Design" button on the "Mode" window.

6. Test your program on an Android emulator
a. Using the "Go" menu select "Launch Emulator"
b. The Android emulator (AVD) will take a minute or so to load, when loaded click "Menu" or drag the padlock across the screen to unlock the AVD
c. Using the "Go" menu select "ADB Start" to start the ADB debugger, because sometimes ADB does not start when an AVD does
d. Using the "Go" menu select "Run Android"
e. A log window will open showing the steps as the program is compiled and sent to the emulator, and then your program will appear on the emulator.

HyperNext Language

Introduction

The HyperNext programming language is similar to Hypertalk as used by Hypercard on Apple Macintosh computers. 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.

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.

For a full list of all the HyperNext keywords see the Glossary.

App startup sequence

When a HAC developed app is run inside HAC or Android it initially follows a set sequence of executing certain scripts, until a card has loaded it is not possible to refer to it or execute its scripts.

This is the Startup script sequence:-

1 - Load the Main code
2 - Constants section
3 - Variables section
4 - Startup section
5 - Load the Home Card
6 - Execute Home card's opening script

NOTES

Main Code Startup

In each of its three startup sections it is possible to execute code and not just declare variables. However if the code takes too long to execute the apps Home Card will be delayed from loading

Card Opening Script

Whenever a new card is loaded it's script is always executed before the card is displayed allowing the card's script to initialize controls etc. However if a card's startup script takes too long the card may appear to hang giving the user a poor perception of the app.

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. Examples 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 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 application will become sluggish in responding to button presses and timers etc.

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
* No 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.

HAC and Android

Android Devices

Generally most Android devices are connected to the development PC by a USB cable but some devices such as tablets do not yet have a USB debug driver available so a wireless or ethernet connection must be used to work with them. If your PC has a wireless capability HAC has an option to connect with such a device - "Wireless Connect", found in the "GO" menu.

HAC has a window called "Manage Devices" for managing and selecting devices. When the "Manage Devices" window opens it automatically scans for both emulators and attached devices. It has two lists:-
1) Android Virtual Devices - a list of all available emulators.
2) Android Runnable - a list of running emulators and attached devices.

To select an emulator or device it must be present in the "Android Runnable" list. If your device isn't attached yet just plug it in, wait a moment and then press the "Refresh All" button until the device appears in the "Android Runnable" list.

HAC Applications

The Android OS was designed to run on Mobile phones and to take care of the life-cycle of an app so that several apps could simultaneously co-exist in memory together, with only the front app being fully active. Furthermore the Android OS is supposed to be responsible for quitting apps when it deems they are no longer needed so the Android app guidelines discourage the use of a Quit button. However many users want a Quit button and so HAC apps have one by default.

A HAC application runs in memory all the time and even if the user presses the Back Button or another app takes control the HAC app will remain in the background and active, so it can play music, communicate with the web or number crunch etc. All HAC apps show an icon in the Status Bar so users know their HAC built app is still running.

Although Android OS is supposed to control the life-cycle of an app many app developers know that there are times when the app user really does want to quit an app and therefore by default all HAC apps have a Quit button on the menu. If your apps don't need a Quit button then you can easily remove it.

Although background apps are not supposed to take resources its clear that they do and having several apps in the background can noticeably slow down the system response to such an extent that the user must restart their device. As many apps have no way to quit them "App Killer" programs have become some of the most popular downloads.

With HAC apps the background task is set to idle unless the code is actually doing something so a typical HAC app should have little resource overhead. If sent to the background it can easily be made foreground again by pressing and holding the Home button down, then choosing the HAC app icon.

Android Keys

APPLICATION KEYS

In order to run your app on either the Android Emulator or on an Android Device it must be signed using a key. The key uniquely identifies the app as your creation and allows only yourself to update your app on the device and on Android Market. Usually when developing your app a Debug key is used for simplicity. However, Android Market will not allow apps to be uploaded that were signed using a Debug Key.

Before uploading your app to the Android Market it must be signed using a Private key. A Private key is created using a password and other identifying information about yourself and your business. The Private key cannot be recreated and if

lost you lose the ability to update any apps that were signed by it. Private Keys must expire after 2033.

In HAC keys can be created and assigned to apps using the Android Key Window that can be opened using the "Android Key" option on the "Android" menu. It has options for using a Debug Key, creating a new Private Key, and assigning an already existing Private Key to an app.

(1) Debug Key
By default HAC uses a Debug Key stored in the "My Android" folder which is in the "My Documents" folder. Therefore while developing your app you don't need to bother creating a Private Key or to remember passwords as HAC knows the passwords for the Debug Key. The Debug Key has an expiry period of about 38 years so will not need to be recreated unlike the usual Android SDK Debug Key which expires after 365 days.

(2) Assigning a Private Key
If you already have made a Private Key it can be loaded using this option. Use the "Private Key Path" button to locate the keystore. For security reasons the password for this key has to be entered. HAC does not store your passwords and so when it quits or loads another Private Key the passwords are cleared from its memory.

(3) Creating a Private Key
HAC greatly simplifies the procedure for creating a Private Key although it may still seem daunting the first time that you create one. IMPORTANT - always record your password somewhere safe and make backup copies of your Private Keys. If you lose either then the app that was signed with that Private Key will not be able to be updated on Android Market.

To create a Private Key several fields need filling in:-

Key Name:
This is the name of the Private Key and the resulting Private Key file has the format name.keystore. By default it is placed in your "My Android" folder.

Password:
This is needed by HAC to make changes to the app using your key. For security reasons every time you switch to another key HAC will ask for your password.

Certificate Settings:-
These are the identifying features in your key but you do not have to fill them all in.
a) First and Last names
b) Organization Unit
c) Organization
d) City or Locality
e) State or Province
f) Country Code

For security reasons there is a Clear button to empty the fields.

Icons

LAUNCHER ICON

The Launcher Icon is the icon displayed on the Home Screen list of applications and should help distinguish your application from others. The maximum size of the Launcher icon is 72 pixels and the icon format set by Google is .png (however the Android OS currently also works with .jpg or .jpeg icons).

In HAC the Launcher Icon is set via the "Your Application" window that is opened using the "Your Application" item on the "Android" menu.

There are two ways to add an icon;
1) Drop your icon on the icon area below the "Launcher" button.
2) Load your icon from disk by clicking the "Launcher" button.

The official Goggle Android website on designing Launcher icons that also covers color schemes and sizes etc can be found at,
http://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher.html

STATUS BAR ICON

The Status Bar Icon is the icon displayed at the top of the screen in the Status Bar, and appears when your application is running. Google recently decreed that all apps running a background service must display a Status Bar Icon so that users can see which background services are running. The current size is 25x25 pixels and the icon format set by Google is .png (however the Android OS currently also works with .jpg or .jpeg icons).

In HAC the Status Bar Icon is set via the "Your Application" window that is opened using the "Your Application" item on the "Android" menu.

There are two ways to select the icon;
1) Drop your icon on the icon area below the "Status Bar" button.
2) Load your icon from disk by clicking the "Status Bar" button.

The official Goggle Android website on designing Status Bar icons that also covers color schemes and sizes etc can be found at,
http://developer.android.com/guide/practices/ui_guidelines/icon_design_status_bar.html

Gestures

HAC currently just supports swipe gestures, allowing list boxes, fields and canvases to be scrolled, or making it possible to swipe across a card (screen) to cause another card to appear. Swipe gestures are only recognized when made in horizontal and vertical directions, and are automatically scaled by HAC for the device's screen size and orientation

The gesture settings affect how the screen responds to touch and can greatly affect the user's experience. There are two types of touch screen on Android devices, capacitive and resistive. Capacitive screens are much more sensitive to touch making it easier to scroll controls without the touch being interpreted as a press event. Resistive screens generally need a firmer and faster touch for a gesture to be interpreted as one and not be mistaken for a press event.

Gestures have their own event handler, so when a gesture occurs the program can handle it and perform an action such as moving to the next card or running a script. Use the "Key Down" handler inside the "Special" code section of the Script Editor to specify your code to handle gestures. Gestures for cards are disabled by default but can easily be enabled.

Note
Gestures only work inside your application when running on an AVD or Android hardware device, inside HAC only simulated gestures can be used. An application running within HAC cannot detect gestures, although the gesture event can be triggered inside HAC via a button or mouse press etc.

Keyboards

Android devices can have several keyboards, the soft keyboard, in-built physical keyboard as in the G1, and external USB keyboards.

The soft keyboard is the keyboard that appears at the bottom of the screen when the user touches an EditText or the system thinks the user needs to input some text. This keyboard cannot be moved so you need to design your screens carefully so the input box or EditText will not be hidden by the keyboard.

For many users the soft keyboard is not very intuitive and they can have problems trying to make it appear or disappear. HAC allows the user to dismiss the soft keyboard by simply touching part of the card/window that is not a control. The soft keyboard automatically appears when a user touches an EditField so that they can enter some text into it. When the user dismisses the keyboard the EditText will retain focus and cursor position.

With a HAC application when the soft keyboard is visible it can also be dismissed by pressing the "Back Key". This

overrides the normal behaviour of the "Back Key" causing the application to be deactivated.

Hard Keyboards are things such as the physical keyboard on the G1 and plugged in external USB keyboards. When hard keyboards are present HAC doesn't display the soft keyboard so users have full access to the screen.

HAC currently only works reliably with English keyboards as at this time the Android OS is still not fully compatible with other character sets.

Location GPS

HAC built Android apps support GPS location functionality that can run in the background and continuously log readings. Furthermore as HAC apps can be fully active in the background this allows the program to process the GPS information and produce graphs, logs etc.

Most Android phones have GPS but many budget tablets do not. Detecting whether an Android device has GPS hardware is not straightforward because in some budget tablets that lack GPS hardware the detection function always returns true and the system menu might even have a working GPS toggle checkbox. The only way to know whether GPS is working is by checking if it has been active and returned a location.

There are basically two types of location providers, mobile network and GPS hardware:-

(1) Mobile network is fast but its accuracy is usually only to a few hundred metres and it may cost money to access it.

(2) GPS is slower but much more accurate, perhaps down to 10 metres although it might not work indoors.

NOTE
The first time GPS is used it can take the device's hardware several minutes to get a fix and if there are insufficient satellites with a strong signal the fix can fail. It takes 3 satellites to give a location fix, and 4 satellites to give a fix with altitude.

Screen

Screen Refreshing

The frequency with which an app refreshes its screen can have a major impact on how responsive the app appears to the user. If an app does not use graphics such as fast plotting or sprite animation then the screen refresh could be set at 20 per second or even lower. If your app uses rapidly changing graphics then in order for the user not to see flickering the speed needs to be above 25 refreshes per second. If your app is a fast moving game then perhaps 40 to 60 refreshes per second might be needed.

Screen Scaling

It is likely your Android app will run on a variety of Android devices ranging from smart phones having screen resolutions of 320x480 pixels or less, to tablets having resolutions of 800x600 or more.

HAC gives your application 3 options of how to scale the screens:-

(1) None
Your app screen will not be scaled. This is most useful when your app is designed for a small screen but must still run on a tablet. If your app runs on a smaller screen than designed for the bottom and right hand side of the Card will be cut off. When running on a larger screen your app will be surrounded by black background.

(2) To Fit
Your app will be stretched to fit the screen but its proportions will remain unchanged, allowing your app to be scaled either up or down depending on the Android device. When running on a larger screen your app may be surrounded by black background.

(3) Stretch

Your app will be stretched to completely fit the screen so possibly resulting in it looking distorted. This is the most flexible as your app will never be surrounded by a black background.

Screen Size

In HAC cards (screens) can have varying sizes. However this is not yet supported on the Android OS so to simplify things the default screen size in a HAC application is determined by the size of the Home Card (also called card 1). This size is then used within your Android app for scaling, orientations etc.

HAC also has the potential to offset your cards from the top left of the screen but this is currently disabled. In the near future this will be enabled as it is especially useful on tablet devices that have much larger screen sizes than smart phones.

A HAC application should also take into account the height of the Status Bar. The height of the Status Bar can be different depending upon the device - for smart phones it is generally 25 pixels high and for some tablets 35 pixels high, but according to Android documentation it could eventually be up to to 48 pixels high.

A rough design is to assume the minimum status bar height of 25 pixels and therefore set the height of your Home Card to be 25 pixels smaller. For example if designing for a smart phone screen of height 480 pixels, then 480-25 = 455.

Note, your app will still be scaled to fit but by allowing for Status Bar height this will minimize any vertical distortion.

StatusBar hide show

The status bar (often called the menu bar), is the screen area at the top of the screen where notifications appear.

The status bar commands **StatusBarHide** and **StatusBarShow** allow the app to hide or show the status bar and so use the full area of the screen. Note, on some Android devices such as the Kindle full screen display might not be possible.

Android Strings

Strings on Android devices can be much more complex than might appear as they can come from many sources and have many formats. There are two classes of string functions, character based and byte based.

The character based string function allows you to to find how many characters a string contains and operate upon the characters within it.

The byte based string functions depend upon the representation used, this allows you to to find how many bytes a string contains and to operate upon the bytes within the string.

The two values returned can be quite different for example:-

Assume a string contains Chinese idiograms and is equal to "\uF93D\uF936\uF949\uF942"

The function LenFN counts the number of characters in the string and returns 4
The function LenBFN counts the number of bytes in the string and for UTF-32 returns 16

If bytes are required then the user can just use the default platform encoding or specify an encoding using the command StringSetEncoding

StringSetEncoding 'UTF-32'

To find the encoding use the function StringEncodingFN

The default string encode is UTF-8, other values are UTF-16, UTF-32 and those described here,
http://download.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html

Note, the StringSetEncoding command does not allow the encoding to be set to an empty string. Also due to the large number of alternate encodings it does not check if the encoding is a valid one.

Android Application Permissions

In order to protect the user from malware programs etc Google has given the Android OS a permission system that allows users to decide whether a program requiring certain permissions should be installed or not. Example permissions are internet access, write to external storage, access camera etc.

As the application creator you have to enable these permissions from within HAC, if you forget to enable the permission then your program will not run correctly on the Android device. Your program will still run but when it tries to access a function requiring a denied permission it will fail. Each application should include within it a list of required permissions and the Android app installer will inform the user what permissions the app needs and whether the user will allow it to be installed.

You can set the permissions for your App using the Android Permissions Window which is available via the Android menu - option Android Permissions. HAC currently just supports the following 5 permissions but there are literally hundreds available within the Android OS.

1) Write External Storage
This is always enabled as your application has to be installed to the SD card or external storage.

2) Internet
Used when your app needs to access the internet such as when sending an email or requesting a web page.

3) Camera
Used when your app needs to access the camera.

4) Location - Coarse
Used to access the coarse location such as determining the cell on which your device is connected.

5) Location - Fine
Used to access GPS to return position coordinates etc.

Android Market

To publish an app on Android Market you must have an Android Market account for developers which costs $25, and before opening an Android Market account you need a Google account. If you do not have a Google account, you can get a free account at www.google.com/accounts.

In order to put your app on Android Market it must be in the form of an Android package known as an APK file. HAC automatically creates an APK file every time you select "Build Android" or "Run Android" and the procedure is successful . You can see the file in your project folder, it has the extension .apk. The APK file contains all the resources necessary for your app to run on an Android device.

Apps on Android Market (as well as on Android devices) are recognized by their package name and neither Market or any device will allow two apps to have the same name. You set the package name of your app in the Android/Your Application menu, it is normal to include the name of your website in your app's package name, eg com.mywebsite.myappname.

You will also need some screenshots of your application so that potential customers can view the product before buying or downloading from Android Market. To get a screenshot, open the project in HAC and choose "Launch emulator" then "Run Android". When your app is running on the AVD click the emulator so that it is the active window, then press Alt+PrtScr at the same time. This places a screenshot of the AVD into your computer's memory which you can paste into a graphics/drawing program by selecting Paste from the Edit menu or by pressing Ctrl+V in the graphics program.

Menu commands

File

New

Starts a new HAC project. Opens a dialogue box so the user can choose the name of the new project and where it will be saved.

Shortcut - Ctrl+N

Open

Opens a previously saved HAC project.

Shortcut - Ctrl+O

Save

Saves the current project.

Shortcut - Ctrl+S

Save as

Saves the current project, this option allows you to change the project name and where it is saved.

Close

Closes the current project.

Shortcut - Ctrl+W

Print

Shortcut - Ctrl+P

Page setup

Quit

Closes HAC.

Shortcut - Ctrl+Q

Edit

Cut

Cuts the selected text in the Script Editor.

Shortcut - Ctrl+X

Copy

Copies the text selected in the Script Editor to the clipboard.

Shortcut - Ctrl+C

Paste

Pastes the text currently held on the clipboard to the position of the cursor in the Script Editor.

Shortcut - Ctrl+V

Clear

Select all

Selects all the text on the Script Editor editing pane.

Shortcut - Ctrl+A

Select none

Selects none of the text on the Script Editor editing pane.

Find

Finds a specified piece of text on the Script Editor editing pane. It can also be used to replace a specified piece of text with a second piece of text.

Shortcut Ctrl+F

Find again

Continues a search for a previously sought piece of text.

Shortcut Ctrl+G

Delete Card

Deletes the selected card. The Home Card can not be deleted.

Edit Script

Opens the Script Editor.

Preferences

Opens a dialogue so you can set various options for the Script Editor.

Go

Compile

Compilation checks the whole project for errors and if any are found the Script 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.

Shortcut - Ctrl+K

Run

The following sequence occurs when a project is run within HAC

1. The project is automatically saved
2. An attempt is made to compile the project
3. The Home card is then loaded and run.

Shortcut - Ctrl+R

It is possible to run your program within HAC but the HAC Runtime Engine does not have the same functionality as an Android Emulator or Device. You can test your program with the core HyperNext keywords and functionality such as memory, loops, data processing,
canvas graphics, changing card and some controls like buttons. However, GPS, string encoding and other device specific functions may not give the same result within HAC as they would on an Android Device.

Launch Emulator

The currently selected AVD is launched.

ADB Start

Starts the ADB debugger.

ADB Kill

Stops the ADB debugger.

Build Android

Compiles and tries to build an Android program but does not run it.

Run Android

The following sequence occurs when a project is run within HAC by selecting "Run Android"

1. The project is automatically saved
2. An attempt is made to compile the project
3. If an AVD is running the program is sent to the AVD and the Home card is loaded and run.

Show Log

Opens the Log window showing various debug messages.

Debugger DDMS

Starts the Android DDMS debugger.

Wireless Connect

Opens a dialogue that will create a wireless connection between the computer HAC is running on and an Android hardware device such as a phone or a tablet. Type the IP number of the Android device into the "IP Address" box and click the "Connect" button.

To see if a connection is successfully made go to the "Android" menu and select "Manage Devices". A dialogue box will open listing all the virtual (AVD) and real (hardware) Android devices on or connected to your HAC computer.

To find the IP Address on most Android devices go to "Settings", "Wireless & networks", "Wi-Fi settings", and select a network to which the device is connected to (the IP number will look something like 192.168.1.72).

Objects

Project Info

Displays a dialogue box showing the number of objects in the current project.

Windows

Show Card

Makes the currently selected card visible.

Show Toolbar

Makes the Tool Bar visible.

Sound Library

Displays the Sound Library dialogue.

Image Library

Displays the Image Library dialogue.

Movie Library

Displays the Movie Library dialogue.

About Box

Presents a dialogue box on which to create an About Box for your HAC program.

Splash Screen

Presents a dialogue box on which to create a Splash Screen for your HAC program.

Menu Designer

Displays the dialogue box you will use to design the menu system for your program.

Preview Card

Shows what the Home Card of your program will look like when run on an Android device.

Shortcut - Ctrl+D

Android

Your Application

This is where you set many important features of your program, such as name, icons, and version number.

The **Package Name** of your application is important. Apps on Android devices and on Android Market are recognized by their package name and neither Market or any device will allow two apps to have the same name. It is standard practice to include the name of your website in your app's package name, eg
com.mywebsite.myappname
org.myotherwebsite.myappname

The **App Name** is the name of your application as it is shown in the file system.

The **Display Title** of your app is shown when loading on a device and on the Application page of the device.

For more on icons see the section HAC and Android.

When your application is finished and ready to be released you should untick the "Debuggable" box as this will slightly improve the performance of your app.

(Also see "**Debuggable**" in "**Introduction**".)

Android Paths

Displays a dialogue that tells HAC where all the necessary files are on your computer.

Android Key

This is where you set the Android Key for your application.

Android Permissions

This is where you set the permissions for your Android App.

1) Write External Storage
This is always enabled as your application has to be installed to the SD card or external storage.

2) Internet
Used when your app needs to access the internet such as when sending an email or requesting a web page.

3) Camera
Used when your app needs to access the camera.

4) Location - Coarse
Used to access the coarse location such as determining the cell on which your device is connected.

5) Location - Fine
Used to access GPS to return position coordinates etc.

Android SDK

Opens the "Android SDK and AVD Manager".

Manage Devices

Opens the "Android Device Manager".

Guide

Overview

Gives a brief introduction to HAC.

About HAC

Shows the HAC About Box.

Register HAC

Only on unregistered copies of HAC, will take the user to the TigaByte Software site to purchase a full copy of HAC.

HAC Tools

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

An image can only be displayed if it has previously 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.

Image Library

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

HAC currently supports the following image formats and their extensions:

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

Movie Library

Movies can be made available in the Tool Bar 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 movie types are:

Quicktime, mov, avi, mpg, mpeg and swf

Script Editor

The Script Editor is used for editing the MainCode, 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 on the "Main" tab of the Tool Bar. The Script Editor is also automatically invoked when an error is found during compilation.

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 Editor - saves any changes and closes 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.

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.

HAC currently supports the following sound formats and extensions:

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

Splash Screen

This allows either text or an image to be displayed in the Splash Screen which is the first window visible when your application starts, if no image is present then only the specified text will be displayed.

An image can only be displayed if it has previously 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.

Menu Designer

Menu Designer

The Menu Designer allows a user defined menu to be built and handlers defined for each menu item. HAC has a set of commands for manipulating menu items and calling their associated handlers. Each menu item has its own script.

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.

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.

Designing Menus

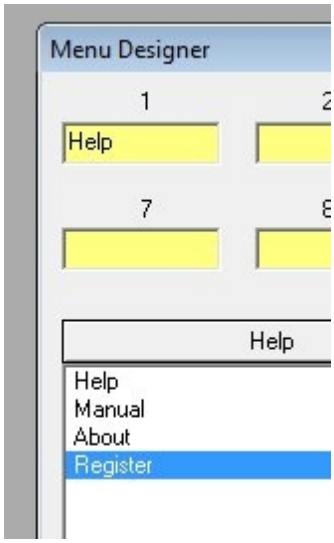
A HAC application can have up to 12 top level menu items, and each of these can have up to 12 sub-menu items. In HAC there are two default menus, "About" and "Quit", but your application does not need to have any menus.

On an Android phone only the first 6 top level menus are shown, if there are more than 6 then only the first 5 are shown along with a "More" option. Choosing "More" reveals the remaining menus in list form.

The yellow fields hold the names of the 12 top level menus. It is generally easier to work from left to right although HAC removes empty menus.

To create a new menu type the name for your menu item into an empty yellow field. This will be the top level name for the menu. To add sub-menu items click the "New Item" button, and edit the name in the name box. To add an image to a menu click the "Icon" button, images must be of a .png .jpg or .jpeg picture format.

Because of the way HAC creates menus the first item of each menu is ignored, so you should put a dummy value as the first item in each menu. The simplest thing is to use the menu title as the first menu item. For example if you wanted to create a menu called Help that would have three menu items called Manual, About and Register your menu would look like this in the Menu Designer.



The "Help" menu item will not be shown in the menus created in HAC or Android, "Manual" will be the first menu item.

Program Structure

Statements

In computer programming a statement is the smallest element of a programming language, and a program is formed by a sequence of one or more statements. A statement will have internal components such as identifiers (for example variable names) and expressions (for example mathematical assignments).

In HyperNext all statements are simple as opposed to compound or complex, and there can be only one statement per line. Some examples of HyperNext statements are;

Put 3 into num
Add 2 to bnum
Call MyProcedure

If you need to perform more than simple calculations (such as adding 3 numbers together) then you will have to break the calculation down in to a series of simple statements.

Procedures and 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 or button. 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.

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 MyBeepFN into num

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

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

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.

MainCode

This is where global procedures are declared and defined. Your global variables can also be declared here and even given starting values, although 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 large as it may have many global variables.

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

Cards

Within HyperNext language cards are the main organizational unit. Only one card may be visible at a time but 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 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 HAC 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.

The first card is recognized as the Home card by HyperNext and it cannot be deleted. Whenever a HAC application starts up it will always load the Home card first, however it can 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
```

For Card commands see the **Cards** entry in the **Glossary**.

Parameters and Brackets

HyperNext may seem confusing as to when the parameters for functions and procedures need brackets (parentheses to Americans) around them. There are three types of procedure/function in HyperNext, HyperNext defined procedures, HyperNext defined functions, and user defined procedures.

HyperNext defined procedures do not need brackets (and cannot have brackets) around their parameters, but HyperNext defined functions and user defined procedures do. To put it another way, if the procedure/function statement includes the keywords "Put" or "Call" then any parameters present will be enclosed in brackets.

Note there must be no space between the user defined procedure name or the function name and the brackets.

HyperNext Procedures
ArrayCopy aname1,aname2
DeleteLineVar data,5
ButtonSetAlign button_id,0

HyperNext Functions
Put ArrayCreateFN(aname1,nrows,ncolumns) into ok
Put AscFN(string1) into res
Put ButtonFontFN(1) into fname

User Defined Procedures
Call MyProcedure(val1,val2)
Call OtherProcedure(val1)
Call ThisProcedure(1,2,3)

Animation Overview

The animation area is a control that allows sprites to be drawn, moved and checked for colliding with any other sprite. It also allows the cursor keys/trackerball state to be tested. The animation can have a backdrop image and can also have canvas commands directed to draw on this image.

Sprites are objects that can move within the animation area and they have properties of their image, coordinates, size and priority of being drawn. Their size or bounding rectangle is used to check for collisions with other sprites.

There are five aspects to using an animation area:

1) The backdrop image and sprite images that are usually placed inside the 'Local' folder and loaded into image banks.
2) Creating and placing the sprites using images from existing image banks.
3) The animation refresh speed, usually a minimum of 25 refreshes per second or a period of 40mS is needed for smooth animation. If the sprites never move then a much lower refresh period can be used.
4) The animation movement script is located inside the 'Specials' 'Animation' section of the script editor. This script is used to check for trackerball and cursor key presses and to change the sprite images and their coordinates prior to the animation graphics being refreshed. (NOTE - after the sprites have been moved then the AnimationEndFrame command must be called so the the graphics can be refreshed and the collision detector triggered.)
5) The collision handler script is located inside the animation handler and is only called when two sprites have collided.

A typical animation frame is as follows:-

1) Inside the animation script
- check for trackerball or key presses
- update any sprite images
- move the sprites
- call AnimationEndFrame (refresh graphics and collision check)

2) Inside the collision script:-
- use the SpriteCollisionFN function to find which two sprites collided.
- update the sprites

Variables

Defining Variables

Variables are used to store numbers, text, lists of names etc. In HAC, 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 or as global, or the compiler will flag an error. However, variables can be used before they are declared, something that can make a procedure or algorithm easier to follow.

A variable name must start with a letter but it does not matter whether lower or upper case letters are used. name, name1 and name2 are all different and valid variable names, name7 and Name7 are both valid variable names but they would both point to the same variable.

Variables must be either Global or Local. Global variables can be accessed from any handler/procedure whereas Local variables can only be accessed from within the handler/procedure in which they are declared.

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

```
Local x1,y1
Global name,job
Local z,w
```

Simple Variables

A simple variable contains a piece of text with no <Carriage Return> characters. Here, a Carriage Return is denoted by CR. For example a simple variable might contain a number, a set of numbers separated by commas, or a sentence.

Some simple variable assignments are,

```
Local number, letter, string
Put 7 into number
Put 'a' into letter
Put 'Qwerty' into string
```

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 a HyperNext predefined procedure/function are expected to be Boolean, any non zero value will be interpreted as true by that procedure/function.

Complex Variables

All HyperNext variables are strings that can be interpreted in different ways. As well as being either treated as text or numeric HyperNext can treat them as complex. Complex means they have some underlying structure such as a string of words or a list of sentences. HyperNext has several commands for handling complex variables.

Complex variables can have 3 basic units line, chunk or char

```
Put line 10 of list into field 1
Put chunk 10 of list into field 1
Put char 10  of list into field 1
```

In HyperNext a variable is considered complex when it contains one or more CR (carriage return) 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.

Controls

Control Types

There are currently 10 types of control available in HAC, they are Button, Canvas, Field, Text, Movie, Check Box, Radio Button, List Box, Sprite and Timer. All controls, except Timer, can be visible to the user. Controls are created in HAC by simply clicking the desired one in the Tool Bar 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.

For a list of Control related commands and functions see the Glossary.

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, 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, any image stored in the Image Library can be used.

Buttons can be set to play a predefined sound file when pressed, any sound 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 Tool Bar window when the relevant button is highlighted.

Canvases

A canvas is the most versatile control in HyperNext because it can display shapes, text, images, receive mouse events, and also have images placed on it through drag and drop. A canvas control has a graphics area that can be drawn on and assigned an image, and 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
White equals 255,255,255
Red equals 255,0,0
Green equals 0,255,0
Blue equals 0,0,255

Check Boxes

A Check Box has only two states, it is either checked or unchecked. A Check Box 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.

Fields

A field control can both display text and receive text input via the keyboard or from the program. Touching a field on a device without a physical keyboard will cause the soft keyboard to appear, to dismiss the soft keyboard just touch the card. Fields can be read-only in which case the soft keyboard will not appear when the field is touched

Text can be placed into field 1 using the following command

Put xstr into Field 1

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 of which Put is the most versatile.

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

A limitation of current text fields is that they can only use a mono font. If you use a non-mono font then they will not be displayed correctly.

Changing the content of a field is computationally very slow as after the field contents are changed the field is redrawn. If you need to update a field it is usually quicker to copy a field to a variable, modify the variable then copy the variable back to the field.

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

In addition to the Mouse Down event a field can also receive several other events. In order to make programming easier for beginners most of these events are disabled by default but can easily be enabled when a card (window) loads. The event which triggered the field handler can be found using the FieldEvent function.

List Boxes

List boxes are used to display one or more columns of information, their cells can hold text, images and check boxes and their contents can be sorted. Rows and cells are numbered from 1 upwards and not from 0 as in many programming languages. The current maximum number of cells allowed is 2000.

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

Note - while running inside HAC list boxes do not have the look and feel of list boxes displayed on an Android device and some of the Android functionality, such as cell color, is not available. Therefore always test your list boxes on an emulator or a real Android device.

List boxes can have the following properties set at design time.

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 list box.
Height - the height of the list box.

Movies

A movie control can be placed onto a card at design time and assigned a movie from the Movie 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.

Progress Bars

Progress bars give a visual indication of a task's 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 for 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.

Progress Bars will be added to HAC in a future upgrade.

Radio Buttons

Sometimes referred to as option buttons, Radio Buttons allow the user to select one item from a group. When a Radio Button is clicked it is selected and all the other Radio Buttons in that group are automatically deselected.

Radio Buttons each have their own handler and their attributes can be changed during runtime, however Radio Buttons must be assigned their group number at design time and currently this cannot be changed at runtime.

Scroll Bars

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.

Scroll Bars will be added to HAC in a future upgrade.

Sprites

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. They can have their image and size changed at any time.

HAC apps 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. The backdrop image is initially transferred from an image bank and held as a refresh copy in the animation area's back picture. Currently an animation area has a limit of 100 sprites.

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.

Animation Events Code that must be executed when the animation event fires should be placed in the Specials section of the Script Editor. Such code might check for sprites collision etc.

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 a Text control is usually used to display a static text such as a header or other information it can also 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 Tool Bar or at runtime, properties such as font, font size, bold, italic, underline, and text colour.

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. Timers can be placed on a card just like any other control except they are not visible. Use the toolbar to create and set their properties.

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. If a timer should be working then it needs to be set up when its card loads.

qwert

Key Shortcuts

Some frequently used shortcuts employing the Control Key (Ctrl) plus another key are:

Ctrl+N – New

Ctrl+O – Open

Ctrl+S – Save

Ctrl+W – Close

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

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

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

Ctrl+B - Build Android.

Ctrl+J - Run Android.

Ctrl+L - Open Log.

Ctrl+Q - either Quits HAC or ends a run and returns to Creator.

Ctrl+E – opens the Script Editor

On Mac OS X use the Command Key instead of the Control Key.

Key words

A-H

A

Add

Adds one variable to another. Values can be doubles, singles or integers.

Add x to y

After

See "Put"

And

See "If"

Append

Appends the first text to the end of the second text.

Append s1 onto s2

AscBFN

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

AscFN

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

B

Base64DecodeFN

Base 64 encoding is a useful way of converting binary data into a textual form that can be reliably transmitted over a network.

This function decodes a Base 64 source.

Base64DecodeFN(message)

Put Base64DecodeFN(src) into field 1

Base64EncodeFN

Base 64 encoding is a useful way of converting binary data into a textual form that can be reliably transmitted over a network.

Base64EncodeFN(message)

Put Base64EncodeFN(mess) into field 1

Beep

Simply plays the currently designated OS system beep.

Before

See "Put"

C

Call

Call is used to jump to a Procedure or Handler.

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

Clear

Clears a target variable or field. Variables can be global or local.

Clear x
Clear field 1

Char

See "Put"

ChrFN

Returns the character value of the given number.

@ Put a space into s1
Put ChrFN(32) into s1

ChrBFN

Returns the byte string value of the given number.

@ Put a space into s1
Put ChrBFN(32) into s1

D

Decrement

Decrements a variable or field by one. Values can be doubles, singles or integers.

Decrement x
Decrement field 12

DeleteLineVar

Deletes the specified line in the target variable. All higher lines are moved down by one line.

DeleteLineVar data,5

Divide

Divides one variable by another. Values can be doubles, singles or integers.

Divide x by y

E

Else

See "If"

EndFor

See "For"

EndIf

See "If"

EndWhile

See "While"

ExitFor

See "For"

ExitWhile

See "While"

F

For

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 a procedure or a function parameter as one of the limit variables.

From

See "Subtract"

G

Global

This is used to declare global variables, which are visible anywhere within the program. The declaration needs to be made in every procedure/handler in which the variables are going to be accessed.

```
Global x,y
```

GotoLabel

This causes the program to jump to the labeled line. A Goto is local and cannot jump outside the current handler/procedure.

```
If x=2 Then
   GotoLabel 1
Else
   Beep
End If

Label 1
@ x=2
```

GotoCard

Goto the specified card, either a number or a name.

```
Goto 14
Goto finances
```

GotoHome

Goes to the first card. In HyperNext the first card called is always called Home and cannot be deleted.

```
GotoHome
```

I-P

keyl

If

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 x=2 Or Y=2 Then
   Message ok
Else
   Message notok
End If
```

If Then can be nested

```
If no2<10 then
   Append '  ' onto numstr
   Else
   If no2<100 then
      Append ' ' onto numstr
   Endif
Endif
```

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
@ code
EndIf

@ an invalid if then
IF (x=2) AND (y=2) OR (z=2) AND (w=2) Then
@ code
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.

Increment

Increments a variable or field by one. Values can be doubles, singles or integers.

```
Increment x
Increment field 12
```

IntegerTryFN

When given a number it tries to return the numeric value without any trailing '0'

IntegerTryFN(value)

Example:-

```
Put 6 into x
Put 2 into y

@ x = 3.0
Divide x by y

@ res = 3
Put IntegerTryFN(x) into res
```

Into

See "Put"

L

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

LeftFN

Returns the left n characters of the given text.

Put LeftFN(s1,n) into s2

LenBFN

Returns the number of bytes in the given text.

Put LenBFN(s1) into slen

LenFN

Returns the number of characters in the given text.

Put LenFN(s1) into slen

Line

See "Put"

LowerFN

Returns the lower case version of the given text.

Put LowerFN(s1) into s2

LtrimFN

Returns the given text with all left leading spaces removed.

Put LtrimFN(s1) into s2

M

Message

Displays a dialogue box with the specified message.

Message ‘hello'
Message value

MiddleFN

Returns a chunk of text comprising num characters from position pst of the given text.

Put MiddleFN(string1,pst,num) into chunkstr

Multiply

Multiplies one variable by another. Values can be doubles, singles or integer.

Multiply x by y

N

NthChunkFN

Returns the nth chunk from the given text using the given separator. A chunk can be a single character, a word or a line.

Put NthChunkFN(sourcestr,sep,nthword) into strbit

NStringsFN

Returns the specified number of strings concatenated together.

Put NStringsFN(source,count) into multi_str

O

Of

See "Put"

Or

See "If"

P

Put

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 four 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

(4) Put chunk of x into y (Puts a chunk of text into the target variable. Chunk can be a single character, a word or a line)
Put char 3 of xstr into char 7 of field 1
Put word 3 of xstr into word 7 of field 1
Put line 3 of xstr 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.

Q-Z

Q

Quit

This command will cause the program to quit, before quitting the state of the program will be saved.

Quit

QuitSave

This command will cause a program to quit but will only save if the value is true. Use this command to disable automatic saving when quitting.

QuitSave value

R

ReplaceAll

Replaces all occurrences of the specified word in the given text with the second specified word.

ReplaceAll word1 with word2 in source

word1 and word2 do not have to be single words, they can be any text.

ReplaceOne

Replaces the first specified word in the given text with the second specified word.

ReplaceOne word1 with word2 in source

word1 and word2 do not have to be single words, they can be any text.

Reset

Sets the variable or field to 0.

Reset x
Reset field 1

RightFN

Returns the n rightmost characters from the given text.

Put RightFN(s1,n) into s2

RtrimFN

Returns the given text with all right trailing spaces removed.

Put RtrimFN(s1) into s2

S

Set

Sets the variable or field to 1.

```
Set  x
Set  field 1
```

Step

See "For"

StringEncodingFN

This function returns the current string encoding.

```
Put StringEncodingFN into strcode
```

StringSetEncoding

This command sets the string encoding using an international standard string value.

```
StringSetEncoding 'UTF-32'
```

Encoding values can be "UTF-8", "UTF-16", "UTF-32" and those described here:
http://download.oracle.com/javase/1.5.0/docs/guide/intl/encoding.doc.html

Subtract

Subtracts one variable from another. Values can be doubles, singles or integers.

```
Subtract x from y
```

SwapLinesVar

Swaps two lines in a target variable.

```
SwapLinesVar  data,line1,line2
```

T

Then

See "If"

To

See "Add", "For"

TrimFN

Returns the given text with both left and right spaces removed.

```
Put TrimFN(s1) into s1
```

U

UpperFN

Returns the uppercase version of the given text.

Put UpperFN(s1) into s1

V

ValFN

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

W

While

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
```

Word

See "Put"

Arrays

Array Commands

ArrayCopy

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

ArrayDeleteAll

This deletes all arrays from memory and reclaims their storage space.

ArrayDeleteAll

ArrayFill

This fills the entire arrays with the specified value.

ArrayFill aname,value
ArrayFill name,25

ArrayFillColumn

This fills one column of the array with the specified value.

ArrayFillColumn aname,column,value
ArrayFillColumn aname,10,25

ArrayFillRow

This fills one row of the array with the specified value.

ArrayFillRow aname,row,value
ArrayFillRow named,10,25

ArrayPutColumn

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,column,list,sep
ArrayPutColumn aname,25,list,sep

ArrayPutRow

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

ArrayPutValue

This puts the specified value into the named array location designated by the row and column.

ArrayPutValue aname,row,column,value
ArrayPutValue aname,5,10,value

ArrayPutWhole

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

Array Functions

ArrayBytesStoredFN

Returns the number of bytes used by the named array.

Put ArrayBytesStoredFN(aname) into num

Before being called the required string representation should be set using the function StringSetEncoding.

ArrayColumnCountFN

Returns the number of columns in the named array.

Put ArrayColumnCountFN(aname) into num

ArrayColumnListFN

Returns the specified column from the array as a list.

Put ArrayColumnListFN(aname,column) into list
Put ArrayColumnListFN(aname,10) into list

ArrayCountFN

Returns the number of arrays in existence.

Put ArrayCountFN into num

ArrayCreateFN

Creates an array having the specified number of rows and columns. The array must be given a name before the ArrayCreateFN function is called.

```
Global aname1, aname2
Local nrows, ncolumns, ok
Put 'my_array' into aname1
Put 'array_1' into aname2
Put 5 into nrows
Put 8 into ncolumns
Put ArrayCreateFN(aname1,nrows,ncolumns) into ok
Put ArrayCreateFN(aname2,7,5) into ok
```

Function return numbers -
0 - successfully created.
1 - ** not used **
2 - name already exists
3 - ** not used **
4 - insufficient memory.

ArrayExistsFN

Returns 1 if the named array exists otherwise it returns 0.

Put ArrayExistsFN(aname) into num

ArrayFindAllColumnsFN

Searches the given row and returns the indexes in list form of all the columns that matched the given text. If the result is 0 then either no columns contained that value or else the parameters were out of range.

ArrayFindAllColumnsFN(aname,rownum,case,value)

Put ArrayFindAllColumnFN('test',3,1,'oranges') into col_list

Note, inside HAC the search does not check for case so 'Hello' equals 'HELLO' equals 'heLLo'

ArrayFindAllRowsFN

Searches the given column and returns the indexes in list form of all the rows that matched the given text. If the result is 0 then either no rows contained that value or else the parameters were out of range.

ArrayFindAllRowsFN(aname,colnum,case,value)

Put ArrayFindAllRowFN('test',3,1,'oranges') into row_list

Note, inside HAC the search does not check for case so 'Hello' equals 'HELLO' equals 'heLLo'

ArrayFindColumnFN

Searches the given row and returns the index of the first column that matches the given text. If the result is 0 then either no columns contained that value or else the parameters were out of range.

ArrayFindColumnFN(aname,rownum,case,value)

Put ArrayFindColumnFN('test',3,1,'oranges') into col_num

Note, inside HAC the search does not check for case so 'Hello' equals 'HELLO' equals 'heLLo'

ArrayFindRowFN

Searches the given column and returns the index of the first row that matches the given text. If the result is 0 then either no rows contained that value or else the parameters were out of range.

ArrayFindRowFN(aname,colnum,case,value)

Put ArrayFindRowFN('test',3,0,'oranges') into row_num

Note, inside HAC the search does not check for case so 'Hello' equals 'HELLO' equals 'heLLo'

ArrayResizeFN

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.

Put ArrayResizeFN(aname,nrows,ncolumns) into ok

Function return numbers -
0 - successfully resized.
1 - name not found
2 - ** not used **
3 - ** not used **
4 - insufficient memory.

ArrayRowCountFN

Returns the number of rows in the named array.

Put ArrayRowCountFN(aname) into num

ArrayRowListFN

Returns the specified row from the array as a list.

Put ArrayRowListFN(aname,row) into list
Put ArrayRowListFN(aname,10) into list

ArrayStatsFN

Returns a list detailing the arrays in existence. 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.

Put ArrayStatsFN into slist

ArrayValueFN

This returns the value from the named array location designated by the row and column.

Put ArrayValueFN(aname,row,column) into value

ArrayWholeListFN

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.

Put ArrayWholeListFN(aname,sep) into list

Cards

CardActiveFN

This function returns 1 if the target card is active otherwise it returns 0.

Put CardActiveFN(20) into x

CardBankImage

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 card_id,bank_id

CardBankImage 12,2

CardEnableColor

This enables or disables the background colour of a card.

CardEnableColor card_id,value

where value is 0 or 1.

CardExistsFN

This function returns 1 if the target card exists otherwise it returns 0.

Put CardExistsFN(9) into result

CardIDFN

This function returns the number of the current card.

Put CardIDFN into field 1

CardLoadImage

Loads the named local image file and attaches the image to the specified card. The image can be left at its normal size or scaled to fill the card using values of 0 or 1 respectively.

CardLoadImage card_id,fname,scale

CardLoadImage 5,fname,1

CardLoadImageAbs

Loads the named absolute image file and attaches the image to the specified card. The image can be left at its normal size or scaled to fill the card using values of 0 or 1 respectively.

CardLoadImageAbs card_id,fname,scale

CardLoadImageAbs 5,fname,1

CardNameFN

This function returns the name of the current card.

Put CardNameFN into field 1

CardNextFN

This function returns the number of the next card, if the current card is the highest numbered card then 0 is returned.

Put CardNextFN into cnumCardPriorFN

CardPriorFN

This function returns the number of the previous card, if the current card is card 1 then 0 is returned.

Put CardPriorFN into cnum

CardRemoveImage

Removes the image from the specified card. If the image was attached to the card using a file then it will be deleted from memory but if it was from an Image Bank it will remain in the bank.

CardRemoveImage card_id

CardRemoveImage 12

CardSetColor

This simply sets the background colour of the specified card. If the card colour is disabled then the card will appear white.

CardSetColor card_id,red,green,blue

CardsActiveFN

This function returns the number of active cards in the application.

Put CardsActiveFN into cnum

CardsTotalFN

This function returns the total number of cards in the application.

Put CardsTotalFN into numcards

Controls

Buttons

Button Commands

ButtonCallName

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'

ButtonCallNumber

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

ButtonLoadLibImage

Loads the given image pathname into the button. The image is scaled to fill the button. The image pathname can be allocated form the Image Library or from a file command.

ButtonLoadLibImage button_id,fname

ButtonLoadLibSound

Loads the given sound pathname into the button. The sound pathname can be allocated from the Sound Library or from a file command.

ButtonLoadLibSound button_id,fname

ButtonSetAlign

Sets the alignment of the text or caption within the button.

ButtonSetAlign button_id,value

Values can be
1 - left
2 - center
3 - right

ButtonSetBevel

Sets the type of bevel for the button.

ButtonSetBevel button_id,value

ButtonSetBold

Sets the bold attribute of the text or caption within the button to either on or off, 0 is off 1 is on.

ButtonSetBold button_id,value

ButtonSetFont

Sets the font name used to display the text or caption within the button.

ButtonSetFont button_id,fname

ButtonSetHeight

Sets the height of the button in pixels.

ButtonSetHeight button_id,value

ButtonSetItalic

Sets the italic attribute of the text or caption within the button to either on or off, 0 is off 1 is on.

ButtonSetItalic button_id,value

ButtonSetLeft

Sets the distance in pixels of the left side of the button from the left side of the card.

ButtonSetLeft button_id,value

ButtonSetMode

Sets whether the button is enabled or not using the values 1 or 0 respectively.

ButtonSetMode button_id,value

ButtonSetSize

Sets the size of the text or caption within the button.

ButtonSetSize button_id,value

ButtonSetText

Sets the text or caption within the button.

ButtonSetText button_id,value

ButtonSetTop

Sets the distance in pixels of the top of the button from the card top.

ButtonSetTop button_id,value

ButtonSetType

Sets the type of button and how it responds to clicks.

ButtonSetType button_id,value

ButtonSetUnderline

Sets the underline attribute of the text or caption within the button to either on or off, 0 is off 1 is on.

ButtonSetUnderline button_id,value

ButtonSetValue

When set to 1 it makes the button appear pressed, and when 0 normal.

ButtonSetValue button_id,value

ButtonSetView

Sets whether the button will be visible or not using the values 1 or 0 respectively.

ButtonSetView button_id,value

ButtonSetWidth

Sets the width of the button in pixels.

ButtonSetWidth button_id,value

Button Functions

ButtonAlignFN

Returns the alignment of the button text.

Put ButtonAlignFN(1) into value

Values are:
1 - left
2 - center
3 - right
4 - default

ButtonBevelFN

Returns the current bevel setting for the specified button.

Put ButtonBevelFN(button_id) into bval

ButtonBoldFN

Returns 1 if the button text bold is on otherwise it returns 0.

Put ButtonBoldFN(1) into num

ButtonFontFN

Returns the font name in which the button text is displayed.

Put ButtonFontFN(1) into fname

ButtonHeightFN

Returns the height of the button.

Put ButtonHeightFN(1) into num

ButtonIDFN

When used within the action handler of a button returns the numeric identity of that button.

Put ButtonIDFN into bid

ButtonItalicFN

Returns 1 if the button text italic is on otherwise it returns 0.

Put ButtonItalicFN(1) into num

ButtonLeftFN

Returns the distance in pixels of the left side the button from the left side of the card.

Put ButtonLeftFN(1) into num

ButtonModeFN

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

ButtonSizeFN

Returns the size of the button text.

Put ButtonSizeFN(1) into num

ButtonTextFN

Returns the text or caption displayed in the button.

Put ButtonTextFN(1) into txt

ButtonTopFN

Returns the distance in pixels of the top of the button from the top of the card.

Put ButtonTopFN(1) into num

ButtonTypeFN

Returns the current type setting for the specified button.

Put ButtonTypeFN(button_id) into bval

ButtonUnderlineFN

Returns 1 if the button text underline is on otherwise it returns 0.

Put ButtonUnderlineFN(1) into num

ButtonValueFN

Returns the current value setting for the specified button.

Put ButtonValueFN(button_id) into bval

ButtonViewFN

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

ButtonWidthFN

Returns the width of the button.

Put ButtonWidthFN(1) into num

Canvas

Copying Canvases

Most computer languages with a command for copying areas of image use a complex looking command having many parameters. HyperNext tries to make it easier for beginners by breaking the single complicated command into three simpler commands.

First the image source must be specified, then the image destination, followed by the command that actually copies the image area. These commands are all related to either user created canvases or an off-screen buffer and are **CanvasSource**, **CanvasDest** and three CanvasCopy commands (**CanvasCopyAll**, **CanvasCopyScale** and **CanvasCopyArea**). With these commands it is possible to easily copy and scale areas of canvases, and canvases can also be saved - see the section on Files/Graphics.

```
@ 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
```

One off-screen buffer for holding and processing images is available, this buffer is accessible just like other canvases except its identity number is 0. To create this off screen buffer use

CanvasSetBuffer width,height

Canvas Commands

CanvasCopyAll

Copies the whole source image to the destination without scaling or changing coordinates. If the source image is larger than the destination then some of the source will go off the destination canvas.

```
CanvasSource  canvas_id1,x,y,w,h
CanvasDest   canvas_id2,x1,y1,w1,h1
```

CanvasCopyAll

See "Copying Canvases"

CanvasCopyArea

Copies the canvas source area to the canvas destination area with scaling. This is the most general purpose copying command and can perform the operations of both the CanvasCopyAll and CanvasCopyScale commands, although it needs all parameters to be set in the CanvasSource and CanvasDest commands.

```
CanvasSource  canvas_id1,x,y,w,h
CanvasDest    canvas_id2,x1,y1,w1,h1
CanvasCopyArea
```

See "Copying Canvases"

CanvasCopyScale

Copies and scales the source canvas to fit the destination canvas.

```
CanvasSource  canvas_id1,x,y,w,h
CanvasDest    canvas_id2,x1,y1,w1,h1
CanvasCopyScale
```

See "Copying Canvases"

CanvasClear

Clears the canvas using the specified color.

```
CanvasClear  canvas_id,red,green,blue
```

CanvasDest

Defines the destination image area for canvas copying.

```
CanvasDest canvas_id,x,y,width,height
```

canvas_id is the destination canvas, 0 is the buffer, >0 is a user created canvas

See "Copying Canvases"

CanvasDrawLine

Draws a line between 2 points in the current colour.

```
CanvasDrawLine   canvas_id,x1,y1,x2,y2
```

CanvasDrawOval

Draws an empty oval in the current colour.

```
CanvasDrawOval  canvas_id,x,y,width,height
```

CanvasDrawRect

Draws an empty rectangle in the current colour.

CanvasDrawRect canvas_id,x,y,width,height

CanvasFillOval

Draws a filled oval in the current colour.

CanvasFillOval canvas_id,x,y,width,height

CanvasFillRect

Draws a filled rectangle in the current colour.

CanvasFillRect canvas_id,x,y,width,height

CanvasGetColor

Gets the current pen color.

CanvasGetColor canvas_id,red,green,blue

CanvasGetPixelColor

Gets the color of the pixel at the specified point.

CanvasGetPixelColor canvas_id,x,y,red,green,blue

CanvasLoadLibImage

Loads the given image pathname into the canvas. The image can be scaled to fill the canvas or left to fit proportionally. The image pathname can be allocated form the Image Library or from a file command.

CanvasLoadLibImage canvas_id,fname,scale

CanvasPlot

Plots a point in the current color.

CanvasPlot canvas_id,x,y

CanvasSetBold

Sets the bold text style to on or off, 1 is on 0 is off.

CanvasSetBold canvas_id,value

CanvasSetBuffer

Creates an off-screen image buffer, whose Canvas_id number is 0.

CanvasSetBuffer width,height

CanvasSetColor

Set the pen to the given color.

CanvasSetColor canvas_id,red,green,blue

CanvasSetFontSize

Sets the size for the current font.

CanvasSetFontSize canvas_id,size

CanvasSetHeight

Sets the height of the canvas in pixels.

CanvasSetHeight canvas_id,value

CanvasSetItalic

Sets the italic text style to on or off, 1 is on 0 is off.

CanvasSetItalic canvas_id,value

CanvasSetLeft

Sets the distance in pixels of the left side of the canvas from the left side of the card.

CanvasSetLeft canvas_id,value

CanvasSetMode

Sets whether the canvas is disabled or enabled using the values 0 or 1 respectively.

CanvasSetMode canvas_id,value

CanvasSetPenHeight

Sets the height of the pen

CanvasSetPenHeight canvas_id,value

Android Pen Height = width

CanvasSetUnderline

Sets the underline text style to on or off, 1 is on 0 is off.

CanvasSetUnderline canvas_id,value

CanvasSetPenWidth

Sets the width of the pen

CanvasSetPenWidth canvas_id,value

CanvasSetTop

Sets the distance in pixels of the top of the canvas from the card top.

CanvasSetTop canvas_id,value

CanvasSetView

Sets whether the canvas is invisible or visible using the values 0 or 1 respectively.

CanvasSetView canvas_id,value

CanvasSetWidth

Sets the width of the canvas in pixels.

CanvasSetWidth canvas_id,value

CanvasSource

Defines the source image area for canvas copying.

CanvasSource canvas_id,x,y,width,height

canvas_id = source canvas, 0 is the buffer, >0 - is a user created canvas

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.

@ Copying all of canvas 12
CanvasSource 12,0,0,0,0

See "Copying Canvases"

Canvas Draw

CanvasHeightFN

Returns the height of the specified canvas.

Put CanvasHeightFN(canvas_id) into x

CanvasIDFN

When used within the action handler of a canvas returns the numeric identity of that canvas.

Put CanvasIDFN into canvas_id

CanvasLeftFN

Returns the distance in pixels of the left side of the canvas from the left side of the card.

Put CanvasLeftFN(canvas_id) into num

CanvasModeFN

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

CanvasTopFN

Returns the distance in pixels of the top of the canvas from the top of the card.

Put CanvasTopFN(canvas_id) into num

CanvasViewFN

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

CanvasWidthFN

Returns the width of the specified canvas.

Put CanvasWidthFN(canvas_id) into x

Graphics Direction

Canvas drawing commands can be directed either to a canvas or to an animation surface, by default all canvas drawing takes place on the specified canvas.

It can be useful to draw on an animation surface and this can be achieved by directing any canvas draws to the animation surface. For example in a space game the animation surface may need an instrument panel which updates as the game progresses.

WARNING, not all canvas commands have been updated to work with the animation surface so after using graphics direction it is safest to reset it back to zero.

These canvas commands currently work with graphics direction:-
CanvasClear, CanvasPlot
CanvasDrawLine, CanvasDrawOval, CanvasDrawRect, CanvasDrawText
CanvasFillOval, CanvasFillRect,
CanvasSetBold, CanvasSetUnderline, CanvasSetFontSize
CanvasSetColor, CanvasSetPenHeight

GraphicsDirectionFN

This function indicates where the canvas drawing is taking place. A 0 result indicates normal canvas drawing and 1 indicates drawing on the animation surface.

Put GraphicsDirectionFN into direct

GraphicsDirectionSet

This command directs canvas drawing to either a canvas or to an animation surface, by default drawing is directed to the specified canvas. If the graphics direction is set to draw on an animation surface then the canvas identity specified in a canvas command will be ignored and the canvas command will simply draw on the animation surface.

@ set to draw on canvas
GraphicsDirectionSet 0

@ set to draw on animation surface
GraphicsDirectionSet 1

NOTE, this command works only on an Android device or AVD and will not work inside HAC because currently its animation surface is protected and cannot be written on.

Check Boxes

Check Box Commands

CheckBoxSetBold

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 cb_id,value

CheckboxSetColor

Sets the color of the text in the specified check box.

CheckboxSetColor cb_id,red,green,blue

@ check box 5, color = white
CheckboxSetColor 5,255,255,255

NOTE, inside HAC the text color of a check box is always black so if it is used on a black or very dark card then it will not be visible. If you need a black/dark colored card then use a Static Text next to the check box for viewing inside HAC.

CheckBoxSetFont

Sets the font name used to display the text or caption within the check box.

CheckBoxSetFont cb_id,fname

CheckBoxSetHeight

Sets the height of the check box in pixels.

CheckBoxSetHeight cb_id,value

CheckBoxSetItalic

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 cb_id,value

CheckBoxSetLeft

Sets the distance in pixels of the left side of the check box from the left side of the card.

CheckBoxSetLeft cb_id,value

CheckBoxSetMode

Sets whether the check box is enabled or not, when value is zero the check box is disabled and when non zero it is enabled.

CheckBoxSetMode cb_id,value

CheckBoxSetSize

Sets the size of the text or caption within the check box.

CheckBoxSetSize cb_id,value

CheckBoxSetState

Sets whether the check Box is checked or not, when value is zero the check box is unchecked and when non zero it is checked.

CheckBoxSetState cb_id,value

CheckBoxSetText

Sets the text or caption within the check box.

CheckBoxSetText cb_id,value

CheckBoxSetTop

Sets the distance in pixels of the top of the check box from the card top.

CheckBoxSetTop cb_id,value

CheckBoxSetUnderline

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 cb_id,value

CheckBoxSetView

Sets whether the check Box is visible or not, when value is zero the check box is hidden and when non zero it is visible.

CheckBoxSetView cb_id,value

CheckBoxSetWidth

Sets the width of the check box in pixels.

CheckBoxSetWidth cb_id,value

Check Box Functions

CheckBoxBoldFN

Returns 1 if the check box text bold is on otherwise it returns 0.

Put CheckBoxBoldFN(cb_id) into num

CheckBoxFontFN

Returns the font name in which the check box text is displayed.

Put CheckBoxFontFN(cb_id) into fname

CheckBoxHeightFN

Returns the height of the check box.

Put CheckBoxHeightFN(cb_id) into num

CheckBoxItalicFN

Returns 1 if the check box text italic is on otherwise it returns 0.

Put CheckBoxItalicFN(cb_id) into num

CheckBoxLeftFN

Returns the distance in pixels of the left side of the check box from the left side of the card.

Put CheckBoxLeftFN(cb_id) into num

CheckBoxModeFN

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(cb_id) into bokay

CheckBoxSizeFN

Returns the size of the check box text.

Put CheckBoxSizeFN(cb_id) into num

CheckBoxStateFN

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 unchecked.

Put CheckBoxStateFN(cb_id) into bokay

CheckBoxTextFN

Returns the text or caption displayed in the check box.

Put CheckBoxTextFN (cb_id) into txt

CheckBoxTopFN

Returns the distance in pixels of the top of the check box from the top of the card.

Put CheckBoxTopFN(cb_id) into num

CheckBoxUnderlineFN

Returns 1 if the check box text underline is on otherwise it returns 0.

Put CheckBoxUnderlineFN(cb_id) into num

CheckBoxViewFN

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(cb_id) into bokay

CheckBoxWidthFN

Returns the width of the check box.

Put CheckBoxWidthFN(cb_id) into num

Fields

FieldCommands

FieldCardSet

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 card_id,field_id,valuestr

FieldGiveFocus

Gives the focus to the specified field.

FieldGiveFocus field_id

FieldPaperColor

Sets the paper colour of the field.

FieldPaperColor field_id,r,g,b

FieldReadOnly

Sets a field's read-only state. When value is 0 the field will accept text, and when non zero it becomes read-only.

FieldReadOnly field_id,value

FieldRemoveFocus

Removes the focus from the specified field.

FieldRemoveFocus field_id

FieldSetGotFocus

Disables or enables the field Get Focus event for the specified field using the values 0 or 1 respectively.

FieldSetGotFocus field_id,value

FieldSetHeight

Sets the height of the field in pixels.

FieldSetHeight field_id,value

FieldSetLeft

Sets the distance in pixels of the left side of the field from the left side of the card.

FieldSetLeft field_id,value

FieldSetLostFocus

Disables or enables the field Lose Focus event for the specified field using the values 0 or 1 respectively.

FieldSetLostFocus field_id,value

FieldSetMode

Sets whether the field is disabled or enabled or not using the values 0 or 1 respectively.

FieldSetMode field_id,value

FieldSetMouseExit

Disables or enables the field MouseExit event for the specified field using the values 0 or 1 respectively.

FieldSetMouseExit field_id,value

FieldSetMouseMove

Disables or enables the field MouseMove event for the specified field using the values 0 or 1 respectively.

FieldSetMouseMove field_id,value

FieldSetMouseUp

Disables or enables the field MouseUp event for the specified field using the values 0 or 1 respectively.

FieldSetMouseUp field_id,value

FieldSetMouseEnter

Disables or enables the field MouseEnter event for the specified field using the values 0 or 1 respectively.

FieldSetMouseEnter field_id,value

FieldSetTop

Sets the distance in pixels of the top of the field from the card top.

FieldSetTop field_id,value

FieldSetTransparent

Sets the transparency of the specified field on the specified card. When transparent mode is on, the card's background/color is visible through the field and when off, the field's own color shows. Transparenency is set to off by default.

Note, if the field is on the active card then the field will be immediately redrawn/refreshed.

FieldSetTransparent card,field,mode

mode
 0 = off (default)
 1 = transparent

@ set transparent to on for field 6 on card 2
FieldSetTransparent 2,6,1

FieldSetView

Sets whether the field is invisible or visible using the values 0 or 1 respectively.

FieldSetView field_id,value

FieldSetWidth

Sets the width of the field in pixels.

FieldSetWidth field_id,value

FieldSetWordWrap

Controls how the text is formatted when displayed inside the field. If enabled then words are not split across lines otherwise lines are split at the line length irrespective of the character there. (0 word wrap is off, 1 word wrap is on)

FieldSetWordWrap field_id,value

This command has no affect inside HAC as text in fields is always word-wrapped in HAC.

FieldTextBold

Sets whether the text is in bold or not for the entire field using the values 1 or 0 respectively.

FieldTextBold field_id,value

FieldTextColor

Sets the text colour of the field.

FieldTextColor field_id,r,g,b

FieldTextFont

Sets the name of the font in which the text will be displayed for the entire field.

FieldTextFont field_id,value

FieldTextSize

Sets the size of the text for the entire field.

FieldTextSize field_id,value

FieldTextUnderline

Sets whether the text is in underline or not for the entire field using the values 1 or 0 respectively.

FieldTextUnderline field_id,value

Field Functions

FieldCardFN

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.

Put FieldCardFN(card_id,field_index) into xstr
Put FieldCardFN(21,7) into xstr

FieldEventFN

Returns a number specifying the field event which caused the field handler to be called.

Put FieldEventFN(field_id) into evnum

The values returned are;
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.

FieldHeightFN

Returns the height of the field control.

Put FieldHeightFN(field_id) into num

FieldIDFN

When used within the action handler of a field returns the numeric identity of that field.

Put FieldIDFN into field_id

FieldKeyDownFN

Returns the character from the key press.

Put FieldKeyDownFN into key

FieldLeftFN

Returns the distance in pixels of the left side the field from the left side of the card.

Put FieldLeftFN(field_id) into num

FieldModeFN

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(field_id) into bokay

FieldPosFN

This returns the position of the cursor within the field text and uses the absolute position within the text.

Put FieldPosFN into xpos

FieldTopFN

Returns the distance in pixels of the top of the field from the top of the card.

Put FieldTopFN(field_id) into num

FieldViewFN

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(field_id) into bokay

FieldWidthFN

Returns the width of the field control.

Put FieldWidthFN(field_id) into num

FieldWordWrapFN

Returns the state of word-wrap for the specified field.

Put FieldWordWrapFN(field_id) into wrap

When run inside HAC this always returns 1 as HAC fields are always wrapped.

List Boxes

List Box Commands

ListboxAddRow

Adds a row with the given value being placed in the first cell of the row.

ListboxAddRow lb_id,value

ListboxBuild

Builds a list box having the specified number of rows and columns.

ListboxBuild lb_id,nrows,ncols

ListboxDeleteAll

Deletes all the rows and columns in the list box and resets its events to zero.

ListboxDeleteAll lb_id

ListboxDeleteRow

Deletes the specified row from the list box.

ListboxDeleteRow lb_id,row

ListboxFill

Fills the list box with the value given. This can also be used to clear the list box.

ListboxFill lb_id,value

ListboxHighlightRow

Highlights the specified row of the list box.

ListboxHighlightRow lb_id,row

ListboxInsertRow

Inserts a row before the given position and places the given value in the first cell of the row.

ListboxInsertRow lb_id,row,value

ListboxRefresh

Refreshes or redraws the specified cell.

ListboxRefresh lb_id,row,column

ListboxSetCellBackColor

Sets the background color of the specified cell. The default color is black - (0,0,0)

ListboxSetCellBackColor lb_id,row,column,red,green,blue

ListboxSetCellCheck

Sets the checkbox value of the specified cell, 0 is unchecked 1 is checked.

ListboxSetCellCheck lb_id,row,column,cbvalue

ListboxSetCellImageFile

Sets the image for the cell where the image is specified using a file path. The image will be scaled and centralized to fit the cell.

ListboxSetCellImage lb_id,row,column,filepath

ListboxSetCellLibImage

Loads the given image pathname into the specified list box cell. The image can be scaled to fill the cell or left to fit proportionally. The image pathname can be allocated form the Image Library or from a file command.

ListboxSetCellLibImage lb_id,row,column,fname,scale

ListboxSetCellTag

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 including categorizing, counting and sorting.

ListboxSetCellTag lb_id,row,column,value

ListboxSetCellTextColor

Sets the text color of the specified cell. The default color is white - (255,255,255)

ListboxSetCellTextColor lb_id,row,column,red,green,blue

ListboxSetCellType

Sets the type of the specified cell.

ListboxSetCellType lb_id,row,column,type

type can be
0 - text
1 - text
2 - checkbox
3 - reserved for editable
4 - image

ListboxSetCellValue

Sets the value of the specified cell.

ListboxSetCellValue lb_id,row,column,value

ListboxSetGridHorizontal

Sets the horizontal grid for the list box.

ListboxSetGridHorizontal lb_id,value

value can be 0 for none or 1 for solid.

ListboxSetGridVertical

Sets the vertical grid for the list box.

ListboxSetGridVertical lb_id,value

value can be 0 for none or 1 for solid.

ListboxSetHeight

Sets the height of the list box in pixels.

ListboxSetHeight lb_id,value

ListboxSetHighlightCell

Sets the highlight value of the specified cell.

ListboxHighlightCell lb_id,row,column,value

value = 0 highlight is off
value = 1 highlight is on

ListboxSetHighlightColor

This sets the highlight color as used by the **ListboxSetHighlightCell** command. This has no affect inside HAC.

ListboxSetHighlightColor(lb_id,row,column,red,green,blue)

@ list box 5, row = 2, column = 3, color = Blue
ListboxSetHighlightColor(5,2,3,0,0,255)

ListboxSetLeft

Sets the distance in pixels of the left side of the list box from the left side of the card.

ListboxSetLeft lb_id,value

ListboxSetMode

Sets whether the List Box is enabled or not, when value is zero the List Box is disabled and when 1 it is enabled.

ListboxSetMode lb_id,value

ListboxSetRowHeight

Sets the default height of the rows in pixels.

ListboxSetRowHeight lb_id,value

ListboxSetTextFont

Sets the text font for the specified list box.

ListboxSetTextFont lb_id,fontname

ListboxSetTextSize

Sets the text size for the specified list box.

ListboxSetTextSize lb_id,textsize

ListboxSetTop

Sets the distance in pixels of the top of the list box from the top of the card.

ListboxSetTop lb_id,value

ListboxSetView

Sets whether the List Box is visible or not, when value is zero the List Box is hidden and when 1 it is visible.

ListboxSetView lb_id,value

ListboxSetWidth

Sets the width of the list box in pixels.

ListboxSetWidth lb_id,value

ListboxSetWidths

Sets the widths of the columns. The value passed can have a number of formats with column widths being separated by commas.

The width values can be in pixels as in value = 125,56,92 for three columns, or in percentages as in value = 25%,35%,40% for three columns.

If not enough column widths are passed then the remaining columns are evenly spaced.

If the list box does not have its column widths set then each column is given a default width of 50 pixels.

ListboxSetWidths lb_id,value

List Box Functions

ListBoxCellCheckFN

Returns the checked value of the specified cell.

Put ListboxCellCheckFN(lb_id,row,column) into num

ListBoxCellTagFN

Returns the tag value of the specified cell.

Put ListboxCellTagFN(lb_id,row,column) into num

ListBoxCellTypeFN

Returns the type of the specified cell.

Put ListboxCellTypeFN(lb_id,row,column) into num

ListboxCellValueFN

Returns the value of the specified cell.

Put ListboxCellValueFN(lb_id,row,column) into num

ListboxColumnClickedFN

Returns the index of the column clicked.

Put ListboxColumnClickedFN(lb_id) into num

ListboxColumnListFN

Returns the cell values of the specified column as a list.

Put ListboxColumnListFN(lb_id,column) into numlist

ListboxColumnsFN

Returns the number of columns in the list box.

Put ListboxColumnsFN(lb_id) into num

ListboxEventFN

Returns the event that occurred in the list box.

Put ListboxEventFN(lb_id) into num

values of num are
1 - Cell checkbox clicked.
2 - Cell pressed.
10 - Cell got focus.
12 - Cell lost focus.

ListboxHeightFN

Returns the height of the specified list box.

Put ListboxHeightFN(lb_id) into x

ListboxIndexFN

Returns the number of the row currently selected.

Put ListboxIndexFN(lb_id) into num

ListboxLastIndexFN

Returns the number of the last row added or inserted.

Put ListboxLastIndexFN(lb_id) into num

ListboxLeftFN

Returns the distance in pixels of the left side the list box from the left side of the card.

Put ListboxLeftFN(lb_id) into num

ListboxLostFocusColumnFN

Returns the number of the column which just lost focus.

Put ListboxLostFocusColumnFN(lb_id) into num

ListboxLostFocusRowFN

Returns the number of the row which just lost focus.

Put ListboxLostFocusRowFN(lb_id) into num

ListboxModeFN

The value returned indicates whether the specified list box is enabled. A return value of 1 indicates the list box is enabled 0 means it is disabled.

Put ListboxModeFN(lb_id) into bokay

ListboxRowClickedFN

Returns the index number of the row clicked.

Put ListboxRowClickedFN(lb_id) into num

ListboxRowListFN

Returns the cell values of the specified row as a list.

Put ListboxRowListFN(lb_id,row) into numlist

ListboxRowsFN

Returns the number of rows in the list box.

Put ListboxRowsFN(lb_id) into num

ListboxScrollPositionFN

Returns the row number of the top row visible.

Put ListboxScrollPositionFN(lb_id) into num

ListboxScrollXFN

Returns the horizontal scroll thumb position for the list box.

Put ListboxScrollXPositionFN(lb_id) into num

ListboxTextFontFN

Returns the text font name for the specified list box.

Put ListboxTextFontFN(lb_id) into fontname

ListboxTextSizeFN

Returns the text size for the specified list box.

Put ListboxTextSizeFN(lb_id) into tsize

ListboxTopFN

Returns the distance in pixels of the top of the list box from the top of the card.

Put ListboxTopFN(lb_id) into num

ListboxViewFN

The value returned indicates whether the specified list box is visible. A return value of 1 indicates the list box is visible 0 means it is hidden.

Put ListboxViewFN(lb_id) into bokay

ListboxWidthFN

Returns the width of the specified list box.

Put ListboxWidthFN(lb_id) into x

Movies

MovieHeightFN

Returns the height of the movie control.

Put MovieHeightFN(movie_id) into num

MovieLeftFN

Returns the distance in pixels of the left side the movie from the left side of the card.

Put MovieLeftFN(movie_id) into num

MovieLoadLibVideo

Loads the given video pathname into a Movie Player. The video pathname can be allocated form the Movie Library or from a file command.

MovieLoadLibVideo mp_id,fname

MovieModeFN

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(movie_id) into bokay

MovieSetHeight

Sets the height of the movie in pixels.

MovieSetHeight movie_id,value

MovieSetLeft

Sets the distance in pixels of the left side of the movie from the left side of the card.

MovieSetLeft movie_id,value

MovieSetMode

Sets whether the movie is enabled or not, 0 is off 1 is on.

MovieSetMode movie_id,value

MovieSetTop

Sets the distance in pixels of the top of the movie from the card top.

MovieSetTop movie_id,value

MovieSetView

Sets whether the movie is visible or not, when value is zero the movie is hidden and when non zero it is visible.

MovieSetView movie_id,value

MovieSetWidth

Sets the width of the movie in pixels.

MovieSetWidth movie_id,value

MovieTopFN

Returns the distance in pixels of the top of the movie from the top of the card.

Put MovieTopFN(movie_id) into num

MovieViewFN

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(movie_id) into bokay

MovieWidthFN

Returns the width of the movie control.

Put MovieWidthFN(movie_id) into num

Progress Bar

Progress Bars will added to HAC in a future upgrade.

Progress Bar Comands

ProgressBarSetHeight

Sets the height of the progress bar in pixels.

ProgressBarSetHeight pb_id,value

ProgressBarSetLeft

Sets the distance in pixels of the left side of the progress bar from the left side of the card.

ProgressBarSetLeft pb_id,value

ProgressBarSetMaxValue

Sets the maximum value which the progress bar can report or be set to.

ProgressBarSetMaxValue pb_id,value

ProgressBarSetMode

Sets whether the progress bar is enabled or not, when value is zero the progress bar is disabled and when non zero it is enabled.

ProgressBarSetMode pb_id,value

ProgressBarSetTop

Sets the distance in pixels of the top of the progress bar from the card top.

ProgressBarSetTop pb_id,value

ProgressBarSetValue

Sets the progress bar position and value.

ProgressBarSetValue pb_id,value

ProgressBarSetView

Sets whether the progress bar is visible or not, when value is zero the progress bar is hidden and when non zero it is visible.

ProgressBarSetView pb_id,value

ProgressBarSetWidth

Sets the width of the progress bar in pixels.

ProgressBarSetWidth pb_id,value

Progress Bar Functions

ProgressBarHeightFN

Returns the height of the progress bar control.

Put ProgressBarHeightFN(pb_id) into num

ProgressBarLeftFN

Returns the distance in pixels of the left side the progress bar from the left side of the card.

Put ProgressBarLeftFN(pb_id) into num

ProgressBarMaximumFN

Returns the maximum value setting of the progress bar.

Put ProgressBarMaximumFN(pb_id) into num

ProgressBarModeFN

The value returned indicates whether the specified progress bar is enabled. A return value of non zero indicates the progress bar is enabled otherwise it is disabled.

Put ProgressBarModeFN(pb_id) into bokay

ProgressBarMouseXFN

ProgressBarMouseYFN

ProgressBarTopFN

Returns the distance in pixels of the top of the progress bar from the top of the card.

Put ProgressBarTopBFN(pb_id) into num

ProgressBarValueFN

Returns the position value setting of the progress bar.

Put ProgressBarValueFN(pb_id) into num

ProgressBarViewFN

The value returned indicates whether the specified progress bar is visible. A return value of non zero indicates the progress bar is visible otherwise it is hidden.

Put ProgressBarViewFN(pb_id) into bokay

ProgressBarWidthFN

Returns the width of the progress bar control.

Put ProgressBarWidthFN(pb_id) into num

Radio Buttons

Radio Button Commands

RadioSetBold

Sets the bold attribute of the text within the radio button to either on or off, 0 is off 1 is on.

RadioSetBold rb_id,value

RadiobuttonSetColor

Sets the color of the text in the specified radio button.

RadioSetColor rb_id,red,green,blue

@ radio button 7, color = green
RadioSetColor 7,0,255,0

NOTE, inside HAC the text color of a radio button is always black so if it is used on a black or very dark card then it will not show up very well. If you need this setup then consider using a Static Text next to the radio button for viewing inside HAC.

RadioSetFont

Sets the name of the font used to display the text or caption within the radio button.

RadioSetFont rb_id,fname

RadioSetHeight

Sets the height of the radio button in pixels.

RadioSetHeight rb_id,value

RadioSetItalic

Sets the italic attribute of the text within the radio button to either on or off, 0 is off 1 is on.

RadioSetItalic rb_id,value

RadioSetLeft

Sets the distance in pixels of the left side of the radio button from the left side of the card.

RadioSetLeft rb_id,value

RadioSetMode

Sets whether the radio button is enabled or not, when value is zero the radio button is disabled and when non zero it is enabled.

RadioSetMode rb_id,value

RadioSetSize

Sets the font size of the text within the radio button.

RadioSetSize rb_id,value

RadioSetState

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 rb_id,value

RadioSetText

Sets the text or caption within the radio button.

RadioSetText rb_id,value

RadioSetTop

Sets the distance in pixels of the top of the radio button from the card top.

RadioSetTop rb_id,value

RadioSetUnderline

Sets the underline attribute of the text within the radio button to either on or off, 0 is off 1 is on.

RadioSetUnderline rb_id,value

RadioSetView

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 rb_id,value

RadioSetWidth

Sets the width of the radio button in pixels.

RadioSetWidth rb_id,value

Radio Button Functions

RadioBoldFN

Returns 1 if the radio button text is bold and 0 if not.

Put RadioBoldFN(rb_id) into num

RadioFindGroupFN

Returns the numeric identity of the group containing the specified radio button.

Put RadioFindGroupFN(rb_id) into groupnum

RadioFontFN

Returns the font name in which the specified radio button text is displayed.

Put RadioFontFN(rb_id) into fname

RadioHeightFN

Returns the height of the specified radio button.

Put RadioHeightFN(rb_id) into hnum

RadioItalicFN

Returns 1 if the radio button text is italic and 0 if not.

Put RadioItalicFN(rb_id) into num

RadioLeftFN

Returns the distance in pixels of the left side of the specified radio button from the left side of the card.

Put RadioLeftFN(rb_id) into xnum

RadioListGroupFN

Returns a list of radio buttons contained within the specified group.

Put RadioListGroupFN(rb_id) into grouplist

RadioModeFN

The returned value indicates whether the specified radio button is enabled. A return value of 1 indicates the radio button is enabled and 0 that it is disabled.

Put RadioModeFN(rb_id) into bstate

RadioPressedFN

Returns the numeric identity of the radio button last pressed.

Put RadioPressedFN into whichbutton

RadioSizeFN

Returns the size of the text in the specified radio button.

Put RadioSizeFN(rb_id) into tsize

RadioStateFN

The returned value indicates whether the specified radio button is selected. A return value of 1 indicates the radio button is selected and 0 that it is deselected.

Put RadioStateFN(rb_id) into bokay

RadioTextFN

Returns the text displayed in the radio button.

Put RadioTextFN(rb_id) into txt

RadioTopFN

Returns the distance in pixels of the top of the specified radio button from the top of the card.

Put RadioTopFN(rb_id) into ynum

RadioUnderlineFN

Returns 1 if the radio button text is underline and 0 if not.

Put RadioUnderlineFN(rb_id) into num

RadioViewFN

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(rb_id) into bview

RadioWidthFN

Returns the width of the specified radio button.

Put RadioWidthFN(rb_id) into wnum

Sprites

Sprite Commands

AnimationBackdrop

Sets the specified bank image as the backdrop for the animation. Inside Android the image will be stretched to fill the backdrop but inside HAC the image will be tiled if it is smaller that the animation surface.

AnimationBackdrop bi_id

AnimationClear

Clears the animation area removing all sprites but leaving the backdrop image showing.

AnimationClear

AnimationDisable

Disables the animation area so that sprite movement and other animation cannot occur.

AnimationDisable

AnimationEnable

Enables the animation area so that sprites and other animation can take place.

AnimationEnable

AnimationEndFrame

This command has two effects and in the following order:-
1) Causes the sprite animation to be refreshed
2) Check for sprite collisions and if found calls the Animation controls handler.

AnimationEndFrame

AnimationHide

Hides the animation area on the card or window.

AnimationHide

AnimationReorder

This recalculates the order for drawing sprites as specified by a sprite's priority value. Sprites with lower value priorities are drawn first and so can appear behind sprites with higher priorities.

AnimationReorder

When setting up the animation your app will run faster if the call AnimationReorder is made after all the sprites have loaded rather than after each single load. The same reasoning applies to setting a group of sprite priorities.

AnimationReorder should be called after (preferably a group)-
SpriteCreateFN
SpriteRecreateFN
SpriteSetPriority

AnimationRun

Starts the animation running continuously.

AnimationRun

AnimationRunOnce

Runs the animation just once.

AnimationRunOnce

AnimationSetCoords

Sets the x and y coordinates of the animation area.

AnimationSetCoords x,y

AnimationSetPeriod

Sets how often in milliseconds the animation area will be updated.

AnimationSetPeriod period

AnimationSetSize

Sets the width and height of the animation area.

AnimationSetSize width,height

AnimationShow

Makes the animation area visible on the card or window.

AnimationShow

AnimationStop

Stops or pauses the animation.

AnimationStop

AnimationUpdate

Causes the animation to update and does not wait for the animation timer to refresh the animation surface. It does not run the animation so sprites do not change their positions and the collision checker is not called.

AnimationUpdate

SpriteClose

Detaches the sprite from the animation area.

SpriteClose sp_id

SpriteFetchBank

Assigns the specified bank image to the sprite.

SpriteFetchBank sp_id,bi_id

SpriteSetCoords

Sets the coordinates of the specified sprite.

SpriteSetCoords sp_id,x,y

SpriteSetPriority

Sets the priority of the specified sprite.

SpriteSetPriority sp_id,priority

Note - This command can change the order in which sprites are displayed, also called the z-plane order.

SpriteSetGroup

Sets the group of the specified sprite.

SpriteSetGroup sp_id,group

SpriteSetDirection

Sets the direction of the specified sprite.

SpriteSetDirection sp_id,direct

Sprite Functions

AnimationHeightFN

Returns the height of the animation area.

Put AnimationHeightFN into height

AnimationLeftFN

Returns the left hand side or x coordinate of the animation area.

Put AnimationLeftFN into x

AnimationMouseFN

Returns a list comprising which sprite objects were under the mouse pointer when it was clicked.

Put AnimationMouseFN into slist

AnimationTopFN

Returns the top side or y coordinate of the animation area.

Put AnimationTopFN into y

AnimationWidthFN

Returns the width of the animation area.

Put AnimationWidthFN into width

SpriteCollisionFN

Returns a list of any sprites which have collided. If the list is empty then no collisions occurred. It only makes sense to

use this function within the animation controls handler as this is called after a collision occurs.

Put SpriteCollisionFN into clist

SpriteCountFN

Returns the number of existing sprites.

Put SpriteCountFN into nsprites

SpriteCreateFN

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.

Put SpriteCreateFN(bimage,x,y,w,h,priority,group,direct) into sp_id

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
w - width that is independent from the bank image
h - height that is independent from the bank image
priority - the order in which sprites are refreshed
group - the group which the sprite belongs to as used by the collision handler. If negative the sprite can collide 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 the main compass points.

Put SpriteCreateFN(1,50,59,20,20,1,1,1) into sp_id

Note - This command can change the order in which sprites are displayed, also called the z-plane order.

Note - setting w and h (width and height) has no effect when running inside HAC.

SpriteDirectionFN

Returns the direction of the specified sprite.

Put SpriteDirectionFN(sp_id) into direct

SpriteGroupFN

Returns the group of the specified sprite.

Put SpriteGroupFN(sp_id) into group

SpriteHeightFN

Returns the height of the specified sprite.

Put SpriteHeightFN(sp_id) into height

SpriteKeyTestFN

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

The key values are:-
19 - up
20 - down
21 - left
22 - right
23 - pressed

On HAC this can test for any key but in an Android app it currently only checks for Trackerball movement and presses as they are equivalent to cursor keys inside HAC.

SpriteMouseHitFN

Returns a list of any sprites under the mouse pointer when it was clicked. If the list is empty then no sprites were hit.

Put SpriteMouseHitFN into clist

SpritePriorityFN

Returns the priority of the specified sprite.

Put SpritePriorityFN(sp_id) into priority

SpriteReCreateFN

Same as the SpriteCreateFN function except it updates the properties of the given sprite. This can be used to animate a sprite with different image and size.

Put SpriteReCreateFN(sp_id,bimage,x,y,w,h,priority,group,direct) into res

Put SpriteReCreateFN(1,1,50,59,10,10,1,1,1) into res

Note - This command can change the order in which sprites are displayed, also called the z-plane order.

SpriteStateFN

Returns the state of the specified sprite.

Put SpriteStateFN(sp_id) into value

Value can be -
0 - the sprite does not exist
1 - the sprite exists and is active (attached)
2 - the sprite exists and is inactive (unattached)

SpriteWidthFN

Returns the width of the specified sprite.

Put SpriteWidthFN(sp_id) into width

SpriteXFN

Returns the x coordinate of the specified sprite.

Put SpriteXFN(sp_id) into x

SpriteYFN

Returns the y coordinate of the specified sprite.

Put SpriteYFN(sp_id) into y

Animation Events

Animation Events code that must be executed when the animation event fires should be placed in the Specials section of the Script Editor. Such code might check for sprites colliding etc.

Graphics Direction

For information about canvas drawing commands being directed either to a canvas or an animation surface see Graphics Direction in the Controls/Canvas section.

Texts

Text Commands

TextGetColor

TextSetAlign

Sets the alignment of the text within the text control.

TextSetAlign text_id,value

Values can be
1 - left
2 - center
3 – right

TextSetBold

Sets the bold attribute of the text control to either on or off, 0 is off 1 is on.

TextSetBold text_id,value

TextSetColor

Sets the colour of the specified text.

TextSetColor text_id,r,g,b

TextSetFont

Sets the font name used to display the text within the text control.

TextSetFont text_id,fname

TextSetHeight

Sets the height of the text control in pixels.

TextSetHeight text_id,value

TextSetItalic

Sets the italic attribute of the text control to either on or off, 0 is off 1 is on.

TextSetItalic text_id,value

TextSetLeft

Sets the distance in pixels of the left side of the text control from the left side of the card.

TextSetLeft text_id,value

TextSetMode

Sets whether the text control is enabled or not using the values 1 or 0 respectively.

TextSetMode text_id,value

TextSetSize

Sets the size of the text within the text control.

TextSetSize text_id,value

TextSetTop

Sets the distance in pixels of the top of the text control from the card top.

TextSetTop text_id,value

TextSetUnderline

Sets the underline attribute of the text control to either on or off, 0 is off 1 is on.

TextSetUnderline text_id,value

TextSetValue

This sets the current text value where text_id is the text identifier and value is the new text value.

TextSetValue text_id,value

TextSetView

Sets whether the text control is visible or not using the values 1 or 0 respectively.

TextSetView text_id,value

TextSetWidth

Sets the width of the text control in pixels.

TextSetWidth text_id,value

Text Functions

TextAlignFN

Returns the alignment of the text.

Put TextAlignFN(text_id) into value

Values are:
1 - left
2 - center
3 - right
4 - default

TextBoldFN

Returns 1 if the text bold is on otherwise it returns 0.

Put TextBoldFN(text_id) into num

TextFontFN

Returns the font name in which the text is displayed.

Put TextFontFN(text_id) into fname

TextHeightFN

Returns the height of the text control.

Put TextHeightFN(text_id) into num

TextItalicFN

Returns 1 if the text italic is on otherwise it returns 0.

Put TextItalicFN(text_id) into num

TextLeftFN

Returns the distance in pixels of the left side the text control from the left side of the card.

Put TextLeftFN(text_id) into num

TextModeFN

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(text_id) into bokay

TextSizeFN

Returns the size of the text.

Put TextSizeFN(text_id) into num

TextTopFN

Returns the distance in pixels of the top of the text control from the top of the card.

Put TextTopFN(text_id) into num

TextValueFN

Returns the text displayed in the text control.

Put TextValueFN(text_id) into txt

TextViewFN

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(text_id) into bokay

TextUnderlineFN

Returns 1 if the text underline is on otherwise it returns 0.

Put TextUnderlineFN(text_id) into num

TextWidthFN

Returns the width of the text control.

Put TextWidthFN(text_id) into num

Debug Commands

DebugInternal

This enables/disables message logging to the DDMS debugger from the HAC engine within your application.

DebugInternal value

@ turn on message logging
DebugInternal 1

@ turn off message logging
DebugInternal 0

DebugSet

This enables/disables message logging to the DDMS debugger from the HAC engine within your application. This can be used to globally switch off debug messages for when your application is released to the public.

DebugSet value

```
@ turn on message logging
DebugSet  1

@ turn off message logging
DebugSet  0
```

DebugMessage

This sends a message using the specified tag to the DDMS debugger. The tag is used to distinguish your messages from other messages originating from the Android OS and other running applications.
Note, the tag for the Internal Debugger is 'Debug'

```
DebugMessage  tag,msg

DebugMessage  'blue','Test message'

Put 'hello world' into mess
DebugMessage  'red',mess
```

Database

A database is an organized collection of data arranged in one or more tables. Each table has its own structure as defined by its record structure. Each record in the table is stored in a row and all rows within the table have the same format as defined by the number of columns and the column data types.

Android uses SQLite database type and so can understand other SQlite databases, however it does not support full SQLite functionality.

HAC apps can work with databases either internal to an app or on the sdcard, they can also simultaneously open and work with multiple databases.

SQLite supports 3 data formats although the format is not enforced and the data is treated as strings.
TEXT - string
INTEGER - 8 bytes
REAL - Double

There are 4 main areas of using databases:-
- Create or Open
- Insert data
- Retrieve data
- Upgrade

When performing a database operation the success or failure can be ascertained by examining the database error message using the **DbErrorMessageFN** function.

NOTE:- most of the database related commands and functions are identified by specifying its full file path.

NOTE Android OS bugs - special characters in queries, eg %, can cause problems so will need to be wrapped.

DatabasePathFN

Returns the full file path to the app's internal database area. It does not have a trailing '/'.

```
DatabasePathFN
```

Example -

```
Put DatabasePathFN into dbpath
Append '/' onto dbpath
```

DbClose

Close the database specified by the full file path and release any associated memory.

DbClose(filepath)

DbClose('/data/data/org.database/databases/info.db')

DbCreateFN

Tries to create or open a database specified by the full file path. If the database does not already exist then this function creates one. It returns 0 if the command failed else an identity number for the database.

DbCreateFN(fpath,version,table,structure)

The structure defines the arrangement of columns and their types and is best held in a variable and built up incrementally.

Example:-

```
Put '(' into structure
Append '_id integer primary key autoincrement' onto structure
Append 'title text not null' onto structure
Append 'author text not null' onto structure
Append 'comment text not null' onto structure

Put DbCreateDatabase(path,1,'books',structure) into res
```

0 = error
>0 = db identity

DbErrorMessageFN

Returns any error message for the last database operated on. If it is empty then no error occurred

DbErrorMessageFN(path)

```
Put DbErrorMessageFN(filepath) into error
```

DbExecCommandFN

This executes a raw command on the database in which the database is expected to return no data back to the app. If this function was successfully carried out by the database then the function will return 1 but if the database could not execute the command or it failed then 0 will be returned.

DbExecCommandFN(path,cmd)

This function can do virtually any command but because both SQLite and HAC use single quotes in their commands there can be conflicts. To avoid these single quote conflicts either of the following approaches can be used although the first approach is much easier:-

For example with the command 'INSERT INTO table1 (field1, field2) VALUES ('" + value_1 + "', '" + value2 + "')'

(1) When building the command string use some special character in place of the single quotes. Then before submitting the command replace that special character with a single quotes using HAC's **ReplaceAll** command:-

Put ChrFN(39) into c39

Put 'INSERT INTO table1 (field1, field2) VALUES (@" + value_1 + "@, @" + value2 + "@)' into cmd

ReplaceAll '@' with c39 in cmd

(2) Use ChrFN(39) in place of a single quote with the command string but this can be very messy.

Put ChrFN(39) into c39

```
Put 'INSERT INTO table1 (field1, field2)  VALUES (' into cmd
Append c39 onto cmd
Append "' + value_1 + "' onto cmd
Append c39 onto cmd
Appnd ', ' onto cmd
Append c39 onto cmd
Append "' + value2 + "' onto cmd
Append c39 onto cmd
Append ')' onto cmd
```

DbExistsFN

Checks whether a file is a valid database and can be opened. It returns 1 if the database is valid and could be opened otherwise it returns 0. The full file path specifies which database to check. Note, it actually tries to open the database because Android has problems opening some SQLite databases.

DbExistsFN(fpath)

DbExportFN

Copies a database file from the app's internal database area to an external location such as the SD card. The external path includes the name of the file and the internal database filename must be specified.

DbExportFN(extpath,dbname)

Example

Put DbExportFN('mtn/sdcard/packagename/Local/test.db','new.db') into res

DbImportFN

Copies a database file from an external location such as SD card to the application's internal database area. The external path includes the name of the file and the internal database filename must be specified.

DbImportFN(extpath,dbname)

Example

Put DbImportFN('mtn/sdcard/packagename/Local/test.db','new.db') into res

DbIsOpenFN

Returns 1 if the database is open otherwise it returns 0. The database is specified by the full file-path to it.

DbIsOpenFN(filepath)

Put DbIsOpenFN('/data/data/org.database/databases/info.db') into res

DbOpenFN

Tries to open the specified database in either read/write mode or just read only mode. If the database does not exist then it will NOT be created. It returns 1 if the database was opened successfully else returns 0.

DbOpenFN(path,mode)

where mode = 0 for read only and 1 for read/write

Queries

There are 3 stages to querying an already open database.

1 - Build the query and execute it.
2 - Operate on any retrieved results.
3 - Close the query to save memory.

DbExecQueryFN

Executes a raw query that is expected to return data such as rows and columns. If successful it returns 1 otherwise it returns 0.

DbExecQueryFN(path,query)

Example for returning all records and ordering by title

Put 'SELECT _id,title FROM books ORDER BY title' into query

Example for query using like and single quotes

Put 'SELECT _id,title FROM books WHERE title LIKE' into query
Append ChrFN(39) onto query
Append '%Android%' onto query
Append ChrFN(39) onto query

DbQueryClose

Close the current query and clear any associated data from memory. Note, it is important to close queries especially if the query returned a large number of results.

DbQueryClose path

DbQueryCount

Returns the row count as returned by the last query.

DbQueryCount(path)

Put DbQueryCount(path) into numrecordsfound

DbQueryResultFN

Used to extract a data item (field) from the last query. The data item is specified by the row number and column name. Rows start at 1.

DbQueryResultFN(path,row,colname)

@ get title from 5th result in query
Put DbQueryResultFN(path,5,'title') into title

Inserting data

There are 3 stages for inserting data into a database:-

1 - Create the insert record for the specified database and table
2 - Build the insert record by adding a field to it.
3 - Do the actual insert so placing the data into the database.

Note, at any time the DbErrorMessageFN function can be checked to see if an error occurred.

DbInsertByName

Builds a record that can be inserted into the database. Each record is built one column (field) at a time. Columns are specified by their name. Note, if the column data includes single quotes then it must be placed into a variable for submission.

DbInsertByName(path,colname,data)

Example a record with 3 fields:-
DbInsertByName path,'Title','Android development'
DbInsertByName path,'Author','Smith and Co'
DbInsertByName path,'Comment','covers db with examples'

DbInsertDoFN

This takes the assembled record and inserts it into the specified database, it returns 1 if successful otherwise it returns 0.

DbInsertDoFN(path)

Put DbInsertDoFN(fpath) into res

DbInsertInit

Initializes insert operation for specified database and table.

DbInsertInit path,table

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 dictated 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.

Image Bank Commands

ImageBankClear

Clears the image from the specified image bank so freeing up memory. This also sets the image bank status to invalid.

ImageBankClear b_id

ImageBankFlipH

Flips the image horizontally.

ImageBankFlipH b_id

ImageBankFlipV

Flips the image vertically.

ImageBankFlipV b_id

ImageBankLoad

Loads an image from the specified local file into an image bank. When the scale value is non-zero then the image will be scaled to fit the bank otherwise its size will remain unchanged.

ImageBankLoad b_id,fname,scale

ImageBankLoadAbs

Loads an image from the specified absolute file into an image bank. When the scale value is non-zero then the image will be scaled to fit the bank otherwise its size will remain unchanged.

ImageBankLoadAbs b_id,fname,scale

ImageBankReserve

Reserves a specified number of image banks in advance.

ImageBankReserve 10

ImageBankReset

Resets the number of image banks to zero and clears all images.

ImageBankReset

ImageBankRotateDown

The whole image is rotated left by 180 degrees.

ImageBankRotateDown b_id

ImageBankRotateLeft

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 b_id

ImageBankRotateRight

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 b_id

ImageBankSave

Saves an image from an image bank to the specified local file.

ImageBankSave b_id,fname

ImageBankSaveAbs

Saves an image from an image bank to the specified absolute file.

ImageBankSaveAbs b_id,fname

ImageBankToCanvas

Copies the specified image bank to the specified canvas. When the scale value is non-zero then the image will be scaled to fit the canvas otherwise its size will remain unchanged.

ImageBankToCanvas b_id,c_id,scale

ImageBankToCanvas 2,5,1

ImageBankToCanvasArea

Copies an area of the specified image bank to an area of the specified canvas.

ImageBankToCanvasArea b_id,c_id,x1s,y1s,ws,hs,x1d,y1d,wd,hd

b_id - image bank number
c_id - canvas number
x1s - source x coordinate
y1s - source y coordinate
ws - source width
hs - source height
x1d - destination x coordinate
y1d - destination y coordinate
wd - destination width
hd - destination height

ImageBankToCanvasArea 2,5,10,10,50,50,90,90 ,50,50

ImageCanvasToBank

Copies the specified canvas to the specified image bank. When the scale value is non-zero then the image will be scaled to fit the image bank otherwise its size will remain unchanged.

ImageCanvasToBank c_id,b_id,scale

ImageCanvasToBank 5,2,1

ImageCanvasToBankArea

Copies an area of the specified canvas to an area of the specified image bank.

ImageCanvasToBankArea c_id,b_id,x1s,y1s,ws,hs,x1d,y1d,wd,hd

c_id - canvas number
b_id - image bank number
x1s - source x coordinate
y1s - source y coordinate
ws - source width
hs - source height
x1d - destination x coordinate
y1d - destination y coordinate
wd - destination width
hd - destination height

ImageCanvasToBankArea 5,2,10,10,50,50,90,90 ,50,50

Image Bank Functions

ImageBankCountFN

Returns the number of currently allocated image banks.

Put ImageBankCountFN into npics

ImageBankFileNameFN

Returns the filename of the file last accessed using any of the image bank file commands.

Put ImageBankFileNameFN into fname

ImageBankHeightFN

Returns the height of the specified image bank.

Put ImageBankHeightFN(b_id) into height

ImageBankPathNameFN

Returns the pathname of the file last accessed using any of the image bank file commands.

Put ImageBankPathNameFN into fname

ImageBankSetSizeFN

Returns 1 if the specified image bank was successfully set to the given size.

Put ImageBankSetSizeFN(b_id,width,height) into result

Put ImageBankSetSizeFN(1,800,600) into okay

ImageBankValidFN

Returns the value 1 if the specified image bank holds an image.

Put ImageBankValidFN(b_id) into okay

ImageBankWidthFN

Returns the width of the specified image bank.

Put ImageBankWidthFN(b_id) into width

Library Commands

Resource Libraries

The resource libraries hold images, movies and sounds that can be loaded into some controls at design time and at run time. Loading a resource at runtime will permanently override any resource added to a control at design time.

ImageLibrary

LibImageCountFN

Returns a count of how many images are in the Image Library.

Put LibImageCountFN into numimages

LibImageFileFN

When given the name of an image file in the Image Library it returns the full path to that image. The path can be passed to other commands such as ButtonLoadLibImage().

This example loads the second image in the Image Library into a button.

```
Local ilist,fname,pathname

Put LibImageListFN into ilist

@ 2nd image
Put line 2 of ilist into fname
```

Put LibImageFileFN(fname) into pathname
ButtonLoadLibImage pathname

LibImageListFN

Returns in list format the names of the images held in the Image Library.

Put LibImageListFN into field 1

Movie Library

LibVideoCountFN

Returns a count of how many videos are in the Movie Library.

Put LibVideoCountFN into numvideos

LibVideoFileFN

When given the name of a video file in the Movie Library it returns the full path to that video. The path can be passed to controls such as the MoviePlayer.

Put LibVideoFileFN(fname) into pathname

LibVideoListFN

Returns in list format the names of the videos held in the Movie Library.

Put LibVideoListFN into field 1

Sound Library

LibSoundCountFN

Returns a count of how many sound files are in the Sound Library.

Put LibSoundsCountFN into numsounds

LibSoundFileFN

When given the name of a sound file in the Sound Library it returns the full path to that sound. The path can be passed to controls such as the SoundPlayer.

Put LibSoundFileFN(fname) into pathname

LibSoundListFN

Returns in list format the names of the sound files held in the Sound Library.

Put LibSoundListFN into field 1

Controls Using Libraries

ButtonLoadLibImage

Loads the given image pathname into the button. The image is scaled to fill the button. The image pathname can be allocated form the Image Library or from a file command.

ButtonLoadLibImage button_id,fname

ButtonLibLoadSound

Loads the given sound pathname into the button. The sound pathname can be allocated from the Sound Library or from a file command.

ButtonLoadLibSound button_id,fname

CanvasLoadLibImage

Loads the given image pathname into the canvas. The image can be scaled to fill the canvas or left to fit proportionally. The image pathname can be allocated form the Image Library or from a file command.

CanvasLoadLibImage canvas_id,fname,scale

ListboxSetCellLibImage

Loads the given image pathname into the specified list box cell. The image can be scaled to fill the cell or left to fit proportionally. The image pathname can be allocated form the Image Library or from a file command.

ListboxSetCellLibImage lb_id,row,column,fname,scale

MovieLoadLibVideo

Loads the given video pathname into a Movie Player. The video pathname can be allocated form the Movie Library or from a file command.

MovieLoadLibVideo mp_id,fname

Maths

AbsFN

Returns the absolute value of the given number.

Put AbsFN(num) into res

AcosFN

Returns the arccosine of the given number with the result being in radians.

Put AcosFN(num) into res

AsinFN

Returns the arcsine of the given number with the result being in radians.

Put AsinFN(num) into res

AtanFN

Returns the arctan of the given number with the result being in radians.

Put AtanFN(num) into res

Atan2FN

Returns the arctangent of two points x and y with the result being in radians.

Put Atan2FN(x,y) into res

BinFN

This function returns the binary version of the given value.

BinFN(value)

Put BinFN(val) into res

BitAndFN

This function bitwise ANDs two values and returns the result.

BitAndFN(value1,value2)

Put BitAndFN(val1,val2) into res

BitOrFN

This function bitwise ORs two values and returns the result.

BitOrFN(value1,value2)

Put BitOrFN(val1,val2) into res

BitXorFN

This function bitwise XORs two values and returns the result.

BitXorFN(value1,value2)

Put BitXorFN(val1,val2) into res

CeilingFN

Returns the given number rounded up to the nearest integer.

Put CeilingFN(num) into res

CosFN

Returns the cosine of the given number with the given number being in radians.

Put CosFN(num) into res

DivFN

Returns the result of integer division between two numbers.

Put DivFN(num1,num2) into res

ExpFN

Returns the exponential of the given number. That is e raised to the power of the given number.

Put ExpFN(num) into res

FloorFN

Returns the the given number rounded down to the nearest integer.

Put FloorFN(num) into res

HexFN

This function returns the hexadecimal version of the given value.

HexFN(value)

Out HexFN(val) into res

LogFN

Returns the natural logarithm of the given number.

Put LogFN(num) into res

MaxFN

Returns the maximum of two given numbers.

Put MaxFN(num1,num2) into res

MinFN

Returns the minimum of two given numbers.

Put MinFN(num1,num2) into res

ModFN

Returns the remainder of the division between the two given numbers.

Put ModFN(num1,num2) into res

OctalFN

This function returns the octal version of the given value.

OctalFN(value)

Put OctalFN(val) into res

PiFN

Returns the value Pi.

Put PiFN into res

PowerFN

Returns the given number raised to the given power.

Put PowerFN(num,pnum) into res

RndDbFN

Returns a double random number:

0 <= r < 1

Put RndDbFN into res

RndFN

Returns an integer random number within the range specified by the two numbers:

num1 <= r <= num2

Put RndFN(num1,num2) into res

RoundFN

Rounds the given number to the nearest integer.

Put RoundFN(num) into res

SignFN

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

Returns the sine of the given number with the given number being in radians.

Put SinFN(num) into res

SqrtFN

Returns the square root of the given number.

Put SqrtFN(num) into res

TanFN

Returns the tangent of the given number with the given number being in radians.

Put TanFN(num) into res

TruncFN

Truncates the fractional part of the given number.

Put TruncFN(num) into res

System & Events

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

To create a file handle use the **FolderItemGet** command.

FolderItemGet filepath,fhandle,fdets,ftypes,fpaths,fnames,fextens

filepath - the path or name of the file
fhandle - the file handle used by **CreateBFile**/**OpenAsBFile** commands
fdets, **ftypes**, **fpaths**, **fnames**, **fextens** - variables containing information on the file.

Binary File Commands

CloseBFile

Close the specified binary file.

CloseBFile findex

CreateBFile

Creates a binary file specified by the folder item handle and returns a file index which can be used to access the file, change its name, or read and write to it.

CreateBFile fhandle,findex

FileBSetEndian

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

FileSetBLength

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

FileSetBPosition

Sets the position in the binary file where the next read/write will take place.

FileSetBPosition findex,fpos

OpenAsBFile

Opens an existing binary file specified by the folder item handle and returns a file index which can be used to access the file, change its name, or read and write to it.

OpenAsBFile fhandle,findex

ReadBBoolean

Reads a boolean value from the current position, the value can be 0 or 1.

ReadBBoolean findex,var

ReadBByte

Reads a single byte from the current position. The byte value ranges from 0 to 255.

ReadBByte findex,byte

ReadBCharUTF16

Reads a two byte UTF16 char from the current position within the binary file.

ReadBCharUTF16 findex,value

ReadBDouble

Reads a double floating point value from the current position. A double floating point variable holds eight bytes and its value ranges from about 2.2251 e-308 to 1.7977 e+308.

ReadBDouble findex,var

ReadBInteger

Reads a four byte integer from the current position, a four byte integer's value ranges from -2,147,483,648 to 2,147,483,647.

ReadBInteger findex,var

ReadBLong

Reads a long integer from the current position. A long integer holds eight bytes. Note, this command only works in an app and not within HAC itself.

ReadBLong findex,var

ReadBShort

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

ReadBPString

Reads a Pascal string of text from the current position. Pascal strings can hold up to 255 characters.

ReadBPString findex,var

ReadBStringLine

Reads a line of text from the current position. The string is terminated by a char 13, char 10, combined char 13 plus char 10 or the end of file. Note this does not support the full Unicode set as the high order byte is set to zero and only the lower order byte is used

ReadBStringLine findex,var

ReadBVariable

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

WriteBBoolean

Writes a boolean value to the current position of the specified binary file. The value can be 0 or 1 but if the value is non zero then 1 will be written.

WriteBoolean findex,value

WriteBByte

Writes a single byte to the current position of the specified binary file. The byte value ranges from 0 to 255.

WriteBByte findex,byte

WriteBCharUTF16

Writes a 2 byte UTF16 char to the current position within the binary file. If the string is longer than 2 chars then the output is truncated.

WriteBCharUTF16 findex,value

WriteBDouble

Writes a double floating point value to the current position of the specified binary file.

WriteBDouble findex,value

WriteBInteger

Writes a four byte integer to the current position of the specified binary file.

WriteBInteger findex,value

WriteBLong

Writes a long integer to the current position of the specified binary file. Note, this command only works in an app and not within HAC itself.

WriteBLong findex,value

WriteBShort

Writes a short integer to the current position of the specified binary file.

WriteBShort findex,value

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

WriteBStringLine

Writes a line of text plus a carriage return (char 13) to the current position of the specified binary file.

WriteBPString findex,value

WriteBVariable

Writes a variable at the current position of the specified binary file.

Put field 1 into svar
WriteBVariable findex,svar

Binary File Functions

EndBFileFN

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

FileBEndianFN

Returns the endian setting for the specified binary file.

Put FileBEndianFN(findex) into field 1

LengthBFileFN

Returns the length of the specified binary file.

Put LengthBFileFN(findex) into flen

PositionBFileFN

Returns the current read/write position within the specified binary file.

Put PositionBFileFN(findex) into fpos

Camera

These camera operations do not work inside HAC, they just return an empty value.

CameraCountFN

This function returns the number of cameras available on a device.

Put CameraCountFN into numcameras

Note, the function does not check whether any camera is working or not.

RecordedImagePathFN

This function returns the path to the image just taken by the camera, if no image was taken then the result will be empty.

Put RecordedImagePathFN into picpath

RecordedVideoPathFN

This function returns the path to the video just taken by the camera, if no video was taken then the result will be empty.

Put RecordedVideoPathFN into vidpath

RecordImage

This command uses a device's camera to take a single photo at full resolution. Any recorded image will be automatically placed in the app's **REC_IMAGE** folder using the name passed to the camera. The path to the image can be found using the **RecordedImagePathFN** function. If the user cancelled the camera then the path will be empty.

The command takes two parameters, the camera to use and the name of the image file. If the device has only one camera then the camera parameter will be ignored and the available camera used. There is no need to add the suffix .jpg as it is added automatically to the file name.

RecordImage cameranumber,filename
RecordImage 1,'testphoto'

Cameras are:-
1 - rear facing
2 - front facing

NOTE
The usual camera behaviour is to pass any taken image path and control back to the app when the OK button is pressed.

However, some Android devices do not relinquish control when the camera OK button is pressed. To get around this problem if the Back button is pressed after a picture has been taken it will pass control and the image path back to the app.

RecordVideo

This command allows a device's camera to take a video. Any recorded video will be automatically placed in the app's **REC_VIDEO** folder. The path to the video can be found using the **RecordedVideoPathFN** function. If the user cancelled the camera then the path will be empty.

The command takes two parameters, the camera to use and the name of the video file. If the device has only one camera then the camera parameter will be ignored and the available camera used.

RecordVideo cameranumber,filename
RecordVideo 1,'testvideo'

Cameras are:-
1 - rear facing camera
2 - front facing

Canvas Events

CanvasEventFN

Returns an integer specifying the canvas event which caused the canvas handler to be called.

Put CanvasEventFN into evnum

CanvasKeyDownFN

Returns the character from the key press.

Put CanvasKeyDownFN into key

CanvasSetGotFocus

Disables or enables the canvas Get Focus event for the specified canvas using the values 0 or 1 respectively.

CanvasSetGotFocus canvas_id,value

CanvasSetKeyDown

Disables or enables the canvas MouseSetKeyDown event for the specified canvas using the values 0 or 1 respectively.

CanvasSetKeyDown canvas_id,value

CanvasSetLostFocus

Disables or enables the canvas Lose Focus event for the specified canvas using the values 0 or 1 respectively.

CanvasSetLostFocus canvas_id,value

CanvasSetMouseDrag

Disables or enables the canvas MouseDrag event for the specified canvas using the values 0 or 1 respectively.

CanvasSetMouseDrag canvas_id,value

CanvasSetMouseEnter

Disables or enables the canvas MouseEnter event for the specified canvas using the values 0 or 1 respectively.

CanvasSetMouseEnter canvas_id,value

CanvasSetMouseExit

Disables or enables the canvas MouseExit event for the specified canvas using the values 0 or 1 respectively.

CanvasSetMouseExit canvas_id,value

CanvasSetMouseMove

Disables or enables the canvas MouseMove event for the specified canvas using the values 0 or 1 respectively.

CanvasSetMouseMove canvas_id,value

CanvasSetMouseUp

Disables or enables the canvas MouseUp event for the specified canvas using the values 0 or 1 respectively.

CanvasSetMouseUp canvas_id,value

CanvasMouseXFN

Returns the x coordinate of where the mouse down event took place.

Put CanvasMouseXFN into xcrd

CanvasMouseYFN

Returns the y coordinate of where the mouse down event took place.

Put CanvasMouseYFN into ycrd

Cypher

CypherAesEncryptFN

This function returns the given text in AES 128 encrypted form using the specified password.

CypherAesEncryptFN(password,text)

Put CypherAesEncryptFN('audk57slf','Hello there') into field 1

CypherAesDecryptFN

This function returns the decrypted text using the specified password and AES 128 encrypted source text.

CypherAesDecryptFN(password,encryptedtext)

Put CypherAesDecryptFN('audk57slf845',source) into field 1

Date and Time

DateDayFN

A function that takes a variable holding the total number of seconds and returns the day portion of that date. Its range is 1 to 31.

Put DateDayFN(tsecs) into adate

DateDayOfWeekFN

A function that 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

A function that takes a variable holding the total number of seconds and returns the corresponding day of the year in integer form, for example day 285. It takes into account leap years and ranges from 1 to 366.

Put DateDayOfYearFN(tsecs) into adate

DateHourFN

A function that takes a variable holding the total number of seconds and returns the hour portion of that date. Its range is 0 to 23.

Put DateHourFN(tsecs) into adate

DateMinuteFN

A function that takes a variable holding the total number of seconds and returns the minute portion of that date. Its range is 0 to 59.

Put DateMinuteFN(tsecs) into adate

DateMonthFN

A function that takes a variable holding the total number of seconds and returns the month portion of that date. Its range is 1 to 12.

Put DateMonthFN(tsecs) into adate

DateNowFN

A function that creates a date value for the current date and time.

Put DateNowFN into totSecs1

DateSecondFN

A function that takes a variable holding the total number of seconds and returns the second portion of that date. It takes into account the extra seconds added to a year and its range is therefore 0 to 61.

Put DateSecondsFN(tsecs) into adate

DateTotalSecsFN

A function that 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 interrogated and modified.

Put DateTotalSecsFN(dvar) into date1

Valid format for dvar are:-

inside Android:- yyyy-MM-dd HH:mm:ss
eg 2011-07-30 15:35:26

inside HAC:- 15/10/2004, 15/10/04, 15Oct2004,

DateWeekOfYearFN

A function that takes a variable holding the total number of seconds and returns an integer for the week that the date falls on, for instance day 3 falls inside week 1. It takes into account leap years and its range is 1 to 53.

Put DateWeekOfYearFN(tsecs) into adate

DateYearFN

A function that takes a variable holding the total number of seconds and returns the year portion of that date.

Put DateYearFN(tsecs) into adate

MicroSecondsFN

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

Note, inside HAC this is accurate to microseconds but inside an Android device it is only accurate to the nearest millisecond.

TicksFN

Returns the number of ticks since the computer was started. Each tick is 1/60th of a second.

Put TicksFN into tnow

Email

This set of commands and functions allows a HAC program to communicate with a POP3 mail server and allows emails plus their attachments to be received. It works with the standard POP3 that comes with web hosting plus web based services such as GMail, Hotmail and Yahoo.

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

A header specifies who an email is from, its subject line, its sent date and its total size including attachments.

Managing the sending of emails is much easier than that of receiving them. Emails are sent using the built-in HyperNext SMTP commands and functions that are independent from the Android SDK mail functions. Sending a basic email is fairly straightforward. The program just needs to create a new email, fill in the email's details, connect to the server, send the email and then disconnect.

Recieving Email

PopAttachmentCountFN

This returns the number of attachments for the last retrieved email. Each attachment can be accessed using an integer identity starting at 1.

Put PopAttachmentCountFN into numAtts

PopAttachmentEncodingFN

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.

Put PopAttachmentEncodingFN(att_id) into enc_num

Put PopAttachmentEncodingFN(1) into encoding

PopAttachmentMimeFN

This returns the Mime type of the specified attachment for the last retrieved email. A Mime is a description such as text/plain, image/jpg etc

Put PopAttachmentMimeFN(att_id) into mm_tp

Put PopAttachmentMimeFN(1) into mimetype

PopAttachmentNameFN

This returns the name of the specified attachment for the last retrieved email. The attached file will be placed in the temporary inbox folder.

Put PopAttachmentNameFN(att_id) into att_name

Put PopAttachmentNameFN(1) into name

PopBodyEncodingFN

This returns the text encoding of the email. It has a format such as UTF-8, ISO-8859-1 etc.

PopBodyEncodingFN into encode

It does nothing inside HAC.

PopBodyHTMLFN

This returns the HTML body of the last retrieved email. If it returns empty then the email had no HTML body.

Put PopBodyHTMLFN into txt

PopBodyPlainFN

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.

Put PopBodyPlainFN into txt

PopBodyRichFN

This returns the rich text body of the last retrieved email. If it returns empty then the email had no rich text body.

Put PopBodyRichFN into txt

PopCheckServerFN

This function returns 1 if the server is active and awaiting commands, otherwise it returns 0.

Put PopCheckServerFN(timeout) into done

Put PopCheckServerFN(200) into res

PopClearEmail

This clears the last email from memory and frees up space used by its bodies and attachments.

PopClearEmail

PopCountFN

Asks the server for the number of emails it is holding. Note, some webmail services allow the user to specify the dates for which emails can be accessed by non web based access.

Put PopCountFN(timeout) into num

Put PopCountFN(200) into emailCount

PopDateListFN

Returns in list form the DATES of the emails. Line 1 is for email number 1 and so on.

Put PopDateListFN into dateList

PopDeleteEmailFN

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.

Put PopDeleteEmailFN(email_id,timeout) into done

Put PopGetEmailFN(3,600) into res

PopDisconnectFN

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, HAC always closes the connection with the mail server so as to prevent the server being locked.

Put PopDisconnectFN(timeout) into done

Put PopDisconnectFN(200) into res

PopErrorMessageFN

Gives the last error message generated by the POP mail handling. If the value is empty then no error occurred during the current mail operation. This function is cleared after being called.

Put PopErrorMessageFN into field 1

PopGetEmailFN

This fetches the email with the specified identity from the mail server, if successful it returns 1 else it returns 0. This will fill the values for the From, Date, Subject, Size, body and attachments. Any attachments will be placed in the temporary inbox folder.

Put PopGetEmailFN(email_id,timeout) into done

Put PopGetEmailFN(3,600) into res

This downloads one full email and the parts of that email can be accessed by these functions:-
PopOneFromFN, PopOneSubjectFN, PopOneDateFN, PopOneSizeFN, PopBodyEncodingFN, PopBodyPlainFN, PopBodyRichFN, PopBodyHTMLFN, PopAttachmentCountFN

NOTE, only one email can be in memory at any one time, although all the email headers can be in memory together.

There can be several attachments and this function tells how many there are:- **PopAttachmentCountFN.**

The attachment details are held in lists that can be accessed by these functions:-
PopAttachmentNameFN, PopAttachmentEncodingFN, PopAttachmentMimeFN

To find the file name of an existing 2nd attachment do:-
Put PopAttachmentNameFN(2) into fname

NOTE, HAC apps treat inline images, videos, audio as attachments.

PopGetFolderFN

This function returns the name of the current InBox folder.

Put PopGetFolderFN into boxfolder

PopGetHeadersFN

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.

Put PopGetHeadersFN(timeout) into done

Put PopGetHeadersFN(600) into headers

If the function is called successfully the header information can be accessed using the following functions:- **PopFromListFN, PopSubjectListFN, PopDateListFN, PopSizeListFN**.

For example, if there are 5 emails on the server, the function would download 5 headers, one per email.To find details of the 3rd email using its headers:-

Put PopFromListFN into templist
Put line 3 of templist into from

Put PopSubjectListFN into templist
Put line 3 of templist into subject

Put PopDateListFN into templist
Put line 3 of templist into date

Put PutSizeListFN into templist
Put line 3 of templist into size

By the way, the headers are downloaded first so that some decision can be made about the actual emails and whether the emails are worth downloading.

Note, the timeout depends upon the expected maximum time needed to retrieve the emails.

PopFromListFN

Returns in list form the FROM addresses of the emails. Line 1 is for email number 1 and so on.

Put PopFromListFN into fromList

PopLoginFN

This function logs into the POP3 mail server and effectively locks the server until logout is performed. It returns 1 if login was successful otherwise it returns 0.

PutPopLoginFN(port,addr,un,pw,encrypt,tw) into res

port - server port (often 110)
addr - server address
un - user name
pw - user password
encrypt - 1 for login encryption, 0 for no encryption
tw - time to wait for server response before giving up

PopOneDateFN

Returns the DATE of the specified email. This date can be quite a complex string.

Put PopOneDateFN(email_id) into date

Put PopOneDateFN(6) into datetxt

PopOneFromFN

Returns the FROM address of the specified email. Depending upon the server it might include the name of the sender plus their email address in brackets.

Put PopOneFromFN(email_mid) into address

Put PopOneFromFN(6) into fromaddr

PopOneSizeFN

Returns the SIZE of the specified email in bytes. Note, this does not work with GMail as that service does not provide access to email sizes.

Put PopOneSizeFN(email_id) into sz

Put PopOneSizeFN(6) into size

PopOneSubjectFN

Returns the SUBJECT line of the specified email.

Put PopOneSubjectFN(email_id) into sub

Put PopOneSubjectFN(6) into subject

PopReset

This completely resets the POP3 email. It closes any connection, clears lists and frees up space used by its bodies and attachments.

PopReset

PopSetFolder

This command sets the temporary folder where received emails will be stored. By default it is set to INBOX. If the folder does not exist then it will be created. Note, this folder is only intended as a temporary folder for processing one email and after the email is received it can be checked and moved to a permanent folder.

PopSetFolder foldername

PopSetFolder MyInBox

PopSizeListFN

Returns in list form the size of each email in bytes. Line 1 is for email number 1 and so on.

Put PopSizeListFN into sizeList

PopSubjectListFN

Returns in list form the SUBJECT lines of the emails. Line 1 is for email number 1 and so on.

Put PopSubjectListFN into subjectList

Sending Email

EmailAttachAbs

This attaches a file specified by an absolute file path.

EmailAttachAbs(fpath,name,Mime)

where:-
fpath - the path to the file
name - the name the recipient sees
Mime - the MIME or multimedia type

EmailCreateNew

Creates a new blank email with no attachments.

EmailCreateNew

EmailErrorCodeFN

This function returns the error code as reported by the server, if no error occurred it returns 0. The error code is cleared after the value is returned.

Put EmailErrorCodeFN into res

EmailErrorMessageFN

This function returns the error message as reported by the server. If no error occurred it returns an empty string. The error message is cleared after the value is returned.

Put EmailErrorMessageFN into errmess

EmailFindAddressesFN

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 a received email are often separated by '<' and '>' as in '<info@tigabye.com>'. This is useful in finding the FROM address when replying to an email.

Put EmailFindAddressesFN(txt,sep1,sep2) into list1

Put EmailFindAddressesFN(txt,sep1,sep2) into addrlist

EmailSendFN

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.

Put EmailSendFN(timeout) into done

Put EmailSendFN(600) into res

EmailSetBodyHTML

This sets the HTML text body part of the email.

EmailSetBodyHTML text

EmailSetBodyPlain

This sets the plain text body part of the email.

EmailSetBodyPlain text

EmailSetBodyRich

This sets the rich text body part of the email.

EmailSetBodyRich text

EmailSetFrom

This sets the FROM address of the email.

EmailSetFrom text

EmailSetServer

This command provides the necessary details so that the **EmailSendFN** function can connect to the server.

EmailSetServer addr,port,secure,uname,pword

where
addr - server address
port - server port,
secure - encrypted = 1, not encrypted = 0
uname - user name,
pword - user password

EmailSetSubject

This sets the SUBJECT line of the email.

EmailSetSubject text

EmailSetTo

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

EmailSetTo text

EmailSocketErrorFN

This function returns 0 if no connection error occurred otherwise it returns 1. A connection error will occur if the given server name does not exist.

Put EmailSocketErrorFN into res

Files

File Locations

Files can have pathnames either local to the app or absolute. When local to the app they can be referenced simply by their file name such as "data.txt" or "data.bin" etc. They will be stored with the apps "Local" folder on the SD card or default external storage.

HAC now uses a local folder called "Local" to hold pictures, video, text or any other files that must be bundled with your app. The contents of this folder will be installed with your app and appear in a folder called "Local" inside your app's root folder. The local folder is useful as it is writeable so can take files at runtime.When using HAC the local file must be placed within the root folder of the project.

Absolute pathname can refer to any location on the device by using a full pathname such as /mnt//disk/data.txt. However, the Android OS protects some areas of the storage system and does not allow access to them.

Android File Functions

These functions only work inside your app when running on an Android device or emulator, they do not work inside HAC as it runs on a different file system.

AppPathFN

This returns the path to your app folder, such as "/mnt/disk/org.myweb.test". If this is used as part of a path then the "/" path separator must be added to it.

Put AppPathFN into path

FileCopyFN

This function copies a file to another location by taking a source path and copying the file to a destination path. It returns 0 if successful and non-zero if an error occurred.

FileCopyFN(src,dest)

Put FileCopyFN('/sdcard/test1.jpg','sdcard/MyFolder/new.jpg') into result

Result:
0 - successful
1 - invalid source
2 - invalid destination
3 - could not read source
4 - could not write destination

FileCreateFN

This function creates the file specified by the given path and returns 0 if successful or 1 if it fails.

Put FileCreateFN(path) into res

Put FileCreateFN('/sdcard/test1.jpg') into res

FileDeleteFN

This function deletes the file specified by the given path, it returns 0 if successful and 1 if unsuccessful.

FileDeleteFN(path)

Put FileDeleteFN('/sdcard/test1.jpj') into res

FileMoveFN

This function moves a file to another location or it can be used to rename a file. It takes a source path and moves (or renames) the file to a destination path. It returns 0 if successful and non-zero if an error occurred.

FileMoveFN(src,dest)

Put FileMoveFN('/sdcard/a1.jpg','sdcard/MyFolder/new.jpg') into result

Result:
0 - successful
1 - invalid source
2 - invalid destination
3 - could not read source
4 - could not write destination

FolderCopyFN

This function copies a folder and its contents to another location. If the destination folder already exits it will return an error. It takes a source path and copies the folder to a destination path. It returns 0 if successful and non-zero if an error occurred.

FolderCopyFN(src,dest)

Put FolderCopyFN('/sdcard/MyFolder','sdcard/MyBackup') into result

Result:
0 - successful
1 - invalid source
2 - invalid destination
3 - could not read source
4 - could not write destination
5 - destination already exists

FolderCreateFN

This function creates the folder specified by the given path. It returns 0 if successful and 1 if unsuccessful.

FolderCreateFN(path)

Put FolderCreateFN('/sdcard/MyFolder') into result

FolderDeleteFN

This function deletes a folder and its contents specified by the given path. It returns 0 if successful and 1 if unsuccessful.

FolderDeleteFN(path)

Put FileDeleteFN('/sdcard/MyFolder') into result

FolderMoveFN

This function moves a folder and its contents to another location or it can be used to rename a folder. If the destination folder already exits it will return an error. It takes a source path and moves the folder to a destination path. It returns 0 if successful and non-zero if an error occurred.

FolderMoveFN(src,dest)

Put FolderMoveFN('/sdcard/MyFolder','sdcard/MyBackup') into result

Result:
0 - successful
1 - invalid source
2 - invalid destination
3 - could not read source
4 - could not write destination
5 - destination already exists

LocalPathFN

This function returns the path to the local folder such as '/mnt/disk/org.myweb.test/Local'

Put LocalPathFN into localpath

RootPathFN

This returns the path to the root folder on which your app is located such as "/mnt/disk". If this is used as part of a path then the "/" path separator must be added to it.

Put RootPathFN into path

File Commands

CloseTRead

Closes a text file using a file handle. It can work with both local and absolute text files.

CloseTRead fh_id

CloseTWrite

To close a file that has been written to, it can work with both local and absolute text files.

CloseTWrite fh_id

EndFileFN

This function checks wether the end of file has been reached, if not it places the line of text into a buffer where it can be read by the command ReadTLine. It requires a file handle and if the end of file has been reached the function returns 1 otherwise 0. It can work with both local and absolute text files.

Put EndFileFN(fh_id) into done

OpenTReadAbs

Opens a file specified by a full pathname for reading.

OpenTReadAbs fpath,fh_id

- fpath is the full file path
- fh_id is the file handle as used by the read and close commands.

OpenTReadLoc

Opens a local file for reading.

OpenTReadLoc fname,fh_id

- fname is the file name
- fh_id is the file handle as used by the read and close commands.

OpenTWriteAbs

Creates an absolute file or uses an existing absolute file.

OpenTWriteAbs fpath,fh_id

OpenTWriteAbs '/mnt/disk/data.txt',fh_id

OpenTWriteLoc

To create a local file or use an existing local file.

OpenTWriteLoc fname,fh_id

OpenTWriteLoc 'data.txt',fh_id

ReadTLine

This reads the next line, it can work with both local and absolute text files.

ReadTLIne fh_id,strdat

WriteTLine

Writes a line of text to a file, it can work with both local and absolute text files.

WriteTLine fh_id,sdata

WriteTLine fh_id,'hello there'

Media Folders

The media folders hold images, videos and sounds taken by the Android device. The images, videos and sounds produced by your device are held inside the following folders -

Images - **REC_IMAGE**
Videos - **REC_VIDEO**
Sounds - **REC_SOUND**

If you need to transfer media from these folders or delete media then use the HAC Android file commands such as **AppPathFN** etc.

The HAC functions for handling media files do nothing inside HAC and work only on an Android device or emulator.

MediaImageCountFN

This function returns a count of how many images are in the **REC_IMAGE** folder.

Put MediaImageCountFN into pics_total

MediaImageInfoFN

This function returns in list form the sizes of those image files held in the **REC_IMAGE** folder. Each line of the list holds the size of a file.

Put MediaImageInfoFN into picsz_list

MediaImageListFN

This function returns in list form the names of those images held in the **REC_IMAGE**, folder. Each line of the list holds the name of a file.

Put MediaImageListFN into pics_list

MediaSoundCountFN

This function returns a count of how many sound files are in the **REC_SOUND** folder.

Put MediaSoundCountFN into sounds_total

MediaSoundInfoFN

This function returns in list form the sizes of the sound files held in the **REC_SOUND** folder. Each line of the list holds the size of a file.

Put MediaSoundInfoFN into soundsz_list

MediaSoundListFN

This function returns in list form the names of those sound files held in the **REC_SOUND** folder. Each line of the list holds the name of a file.

Put MediaSoundListFN into sounds_list

MediaVideoCountFN

This function returns a count of how many videos are in the **REC_VIDEO** folder.

Put MediaVideoCountFN into vids_total

MediaVideoListFN

This function returns in list form the names of those videos held in the **REC_VIDEO** folder. Each line of the list holds the name of a file.

Put MediaVideoListFN into vids_list

MediaVideoInfoFN

This function returns in list form the sizes of the video files held in the **REC_VIDEO** folder. Each line of the list holds the size of a file.

Put MediaVideoInfoFN into vidsz_list

Gestures

Gesture Commands

CardGestureDo

Simulates a card gesture for the current card triggering the gesture event handler. (This also works inside HAC)

@ simulate upwards gesture
CardGestureDo 3

CardGestureSet

This enables/disables the gesture for the specified card, 1 is enabled 0 disabled.

CardGestureSet card_id,mode

GestureSetTime

The minimum time a touch event should exist before it can be interpreted as gesture. If the touch event lasts less than this time it will be interpreted as a press event. The default setting is 300 milliseconds.

GestureSetTime 150

SwipeSetMaxOffPath

Sets the maximum distance in pixels that the touch event must stay within for it to be interpreted as a gesture, the default setting is 250 pixels.

SwipeSetMaxOffPath 200

SwipeSetMinDistance

Sets the minimum distance in pixels that the touch event must cover for it to be interpreted as a gesture. The default setting is 120 pixels.

SwipeSetMinDistance 130

SwipeSetMinVelocity

Sets the minimum velocity for the touch event for it to be interpreted as a gesture, the default setting is 200.

SwipeSetMinVelocity 210

Gesture Functions

CardGestureModeFN

Returns whether the specified card has its gestures enabled/disabled.

Put CardGestureModeFN(card_id) into gmode

CardGestureValueFN

Returns the gesture just executed for the current card. After using this function a Card's gesture value will be set to 0.

Put CardGestureValueFN into gval

Gesture values are
0 - no gesture
1 - left
2 - right
3 - up
4 - down

GestureTimeFN

Returns the current Gesture Time.

Put GestureTimeFN into gtime

SwipeMaxOffPathFN

Returns the current value for the swipe maximum of path distance.

Put SwipeMaxOffPathFN into dist

SwipeMinDistanceFN

Returns the current value for the swipe minimum distance.

Put SwipeMinDistanceFN into dist

SwipeMinVelocityFN

Returns the current value for the swipe minimum velocity.

Put SwipeMinVelocityFN into minvel

HTTP

HTTPSetHeaderName

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

Currently this only works within HAC and not on Android. 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')

LaunchExternalURL

Launches the specified web URL in the default web browser. The specified address must be full, i.e have 'http://' in front of it.

LaunchExternalURL address

LaunchExternalURL 'http://www.google.com'

UrlFetchDataFN

Fetches the text of the web page from the given URL and returns it as a string.

UrlFetchDataFN

Put UrlFetchDataFN('http://yahoo.co.uk') into data

URLexistsAbsFN

Returns true (1) if the given web page exists otherwise false (0). The file-name specifies an absolute address. The yield flag when true (1) allows background events such as timers to work while setting yield to false (0) freezes most background activity. The timeout value causes the URLexistsFN function to abort if success has not occured in that time.

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

webadddr - http://www.tigabyte.com/exists.txt
yield - 1 / 0 (true / false)
timeout - time in seconds
filename - absolute file in which to save the download.

Note, the yield and timeout parameters are not currently used on Android.

URLexistsFN

Returns 1 if the given web page exists otherwise it returns 0, the file-name specifies a local address. The yield flag when true (1) allows background events such as timers to work while setting yield to false (0) freezes most background activity. The timeout value causes the URLexistsFN function to abort if success has not occurred in that time.

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

webadddr - http://www.tigabyte.com/exists.txt
yield - 1 / 0 (true / false)
timeout - time in seconds
filename - local file in which to save the download.

Note, the yield and timeout parameters are not currently used on Android.

HTTP Example

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
Local h1,sdata

@ 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

   OpenTReadLoc(filename,h1)

   ReadTLine(h1,sdata)

   Put sdata After field 1

   CloseTRead(h1)

Else

      Put  'unsuccessful' into field 1

EndIf
```

GPS

Using GPS

There are 3 steps in using GPS:-

1) Choose your provider(s) using the the command
GpsEnableProviderNetwork or **GpsEnableProviderGPS**

2) Start GPS using the command
GpsStart

3) In the GPS event check for the First Fix or location/time changed events then get the location data using the functions like

GpsLatitudeNumericFN, **GpsLongitudeNumericFN** etc

GPS Events

The GPS event numbers are;
1 started
2 stopped
3 Sat status - not used
4 first fix
5 location changed
6 time changed

When checking the event value always assume in future more event types will be added.

Note, the GPS Event code is accessed from the editor via the Specials button

GPS Commands

GpsDistanceBetweenCalc

This command calculates the distance and bearing between two locations but does not return the results. The results it generates can be accessed using the functions
GpsDistanceBetweenFN
GpsDistanceBearingInitialFN
GpsDistanceBearingFinalFN

GpsDistanceBetweenCalc lat1,long1,lat2,long2

GpsEnableProviderGPS

Sets whether or not the location should be acquired from a GPS satellite.

@ use GPS satellites
GpsEnableProviderGPS 1

@ do not use GPS satellites
GpsEnableProviderGPS 0

GpsEnableProviderNetwork

Sets whether or not the location should be acquired from the mobile network.

@ use mobile network
GpsEnableProviderNetwork 1

@ do not use mobile network
GpsEnableProviderNetwork 0

GpsStart

Starts the GPS location reporting and requests it sends a new fix either from mobile network or satellite when the fix change is greater than the minimum specified time or distance. Time is in milliseconds and Distance is in metres.

GpsStart time,distance

@ 10 seconds, 5 metres
GpsStart 10000,5

Note, an instruction to follow the time duration is not always carried out and on some Android devices it can be very erratic

GpsStop

Stops the reporting of location date from both the satellite and mobile network.

GpsStop

GPS Functions

GpsAccuracyFN

Returns the last accuracy reading in metres.

Put GpsAccuracyFN into acc

GpsAltitudeFN

Returns the last altitude reading in metres.

Put GpsAltitudeFN into alt

GpsBearingFN

Returns the direction of travel in degrees East of true North. If no bearing is available then 0 is returned.

Put GpsBearingFN into bearing

GpsDistanceBearingFinalFN

This function returns the final bearing between two locations as calculated by the **GpsDistanceBetweenCalc** command.

Put GpsDistanceBearingFinalFN into bearing2

GpsDistanceBearingInitialFN

This function returns the initial bearing between two locations as calculated by the **GpsDistanceBetweenCalc** command.

Put GpsDistanceBearingInitialFN into bearing1

GpsDistanceBetweenFN

This function returns the distance in metres between two locations as calculated by the **GpsDistanceBetweenCalc** command.

Put GpsDistanceBetweenFN into distance

GpsHasAccuracyFN

Returns 1 if the provider can report accuracy information and 0 otherwise.

Put GpsHasAccuracyFN into accOkay

GpsHasAltitudeFN

Returns 1 if the provider can report altitude information and 0 otherwise.

Put GpsHasAltitudeFN into altOkay

GpsHasBearingFN

Returns 1 if the provider can report bearing information and 0 otherwise.

Put GpsHasBearingFN into bearingOkay

GpsHasSpeedFN

Returns 1 if the provider can report speed information and 0 otherwise.

Put GpsHasSpeedFN into speedOkay

GpsLatitudeDegreesFN

Returns the last latitude reading in degrees form.

Put GpsLatitudeDegreesFN into lat

GpsLatitudeMinutesFN

Returns the last latitude reading in minutes form.

Put GpsLatitudeMinutesFN into lat

GpsLatitudeNumericFN

Returns the last latitude reading in numeric form.

Put GpsLatitudeNumericFN into lat

GpsLatitudeSecondsFN

Returns the last latitude reading in seconds form.

Put GpsLatitudeSecondsFN into lat

GpsLocationFN

Returns the attributes for the last location as a list of attributes.

Put GpsLocationFN into inf_list

1 - time
2 - string
3 - provider
4 - has accuracy
5 - has altitude
6 - has bearing
7 - has speed
8 - attitude
9 - longitude
10 - altitude
11 - bearing
12 - speed
13 - accuracy

GpsLocationsActiveFN

Returns 1 when the GPS hardware has been active else 0.

GpsLocationsActiveFN

Put GpsLocationsActiveFN into isactive

GpsLocationProviderFN

Who provided the location fix, either 1 for network or 2 for GPS satellite. This can also be used to check which mode the event is related to for example when a location stops or starts.

Put GpsLocationProviderFN into provider

GpsLongitudeDegreesFN

Returns the last longitude reading in degrees form.

Put GpsLongitudeDegreesFN into lon

GpsLongitudeMinutesFN

Returns the last longitude reading in minutes form.

Put GpsLongitudeMinutesFN into lon

GpsLongitudeNumericFN

Returns the last longitude reading in numeric form.

Put GpsLongitudeNumericFN into lon

GpsLongitudeSecondsFN

Returns the last longitude reading in seconds form.

Put GpsLongitudeSecondsFN into lon

GpsOnBoardFN

This function should return 1 if the Android device has GPS hardware and zero if not. However some devices, notably some low cost tablets lacking GPS hardware, will still report 1 with this function. Some of these tables still have the GPS menu preferences active even though they have no GPS hardware.

Put GpsOnBoardFN into res

GpsProviderGpsFN

Returns 1 when the GPS mode is enabled for acquiring location information, and 0 when not enabled.

GpsProviderGpsFN

Put GpsProviderGpsFN into gpsOkay

GpsProviderNetworkFN

Returns 1 when the mobile network is enabled for acquiring location information, and 0 when not enabled.

GpsProviderNetworkFN

Put GpsProviderNetworkFN into networkOkay

GpsSatelliteCountFN

Shows the number of satellites in view.

GpsSatelliteCount

Put GpsSatelliteCount into numsats

GpsSatelliteListFN

Returns a list of satellite data with each line in the list having the following format and with attributes separated by a single space:

1 - used in fix, either 1 or 0
2 - signal to noise ratio - floating point number
3 - azimuth in degrees - range 0 to 360
4 - elevation in degrees - range 0 to 90
5 - pseudo-random number - integer
6 - almanac - return 1 if GPS engine has almanac data for this satellite
7 - ephemeris - return 1 if GPS engine has ephemeris data for this satellite

GpsSatelliteListFN

Put GpsSatelliteListFN into inf_list

GpsSpeedFN

Returns the last speed reading in metres per second.

Put GpsSpeedFN into speed

GpsStringFN

Returns the data string reading from GPS.

Put GpsStringFN into data

GpsTimeFN

Returns the UTC time of the last fix, in milliseconds since January 1, 1970.

Put GpsTimeFN into time

GPS Conversions

These functions allow conversions to and from numeric and degree formats. Numeric format is simply a floating point or integer value.

Degree format is [+-]DDD.DDDDD where D indicates degrees.

Minute format is [+-]DDD:MM.MMMMM where D indicates degrees and M indicates minutes of arc (1 minute = 1/60th of a degree).

Second format is DDD:MM:SS.SSSSS where D indicates degrees, M indicates minutes of arc, and S indicates seconds of arc (1 minute = 1/60th of a degree, 1 second = 1/3600th of a degree).

GpsDegreesToNumericFN

This function converts the given value expressed in degree format into a numeric value. An example degree format is 1.54 as only degrees are shown and not minutes or seconds.

GpsDegreesToNumericFN(d)

Put GpsDegreesToNumericFN(degrees) into num

GpsMinutesToNumericFN

This function converts the given value expressed in minute format into a numeric value. An example minute format is 1:32.2 where degrees and minutes are shown but not seconds.

GpsMinutesToNumericFN(m)

Put GpsMinutesToNumericFN(minutes) into num

GpsNumericToDegreesFN

This function converts the given numeric value into a degree formatted value.

GpsNumericToDegreesFN(n)

Put GpsNumericToDegreesFN(num) into degrees

GpsNumericToMinutesFN

This function converts the given numeric value into a minute formatted value.

GpsNumericToMinutesFN(n)

Put GpsNumericToMinutesFN(num) into minutes

GpsNumericToSecondsFN

This function converts the given numeric value into a second formatted value.

GpsNumericToSecondsFN(n)

Put GpsNumericToSecondsFN(num) into seconds

GpsSecondsToNumericFN

This function converts the given value expressed in second format into a numeric value. An example second format is 1:32:15 where degrees, minutes and seconds are shown.

GpsSecondsToNumericFN(s)

Put GpsSecondsToNumericFN(seconds) into num

Misc

AndroidVersionFN

This function returns the numeric value of the Android OS or API, a return value of 4 means the app is running on Android 1.6. It can be used to override the Android installer settings and ensure that the app only runs on certain Android OS versions or else quits.

Put AndroidVersionFN into os_no

current values are:-

API	Platform	Code name
04	1.6	Donut
05	2.0	Eclaire
06	2.01	Eclaire 0 1
07	2.1x	Eclaire MR1
08	2.2x	Froyo
09	2.3 - 2.32	Gingerbread
10	2.33 - 2.34	Gingerbread MR1
11	3.0.x	Honeycomb
12	3.1.x	Honeycomb MR1
13	3.2	Honeycomb MR2

EventsCountFN

This function returns the number of unprocessed events in the event queue.

Put EventsCountFN into field 1

FlushEvents

This disposes of all events in the event queue.

FlushEvents

InputAddressFN

This returns the IP address currently used by the device independent of 3G or WiFi, it returns an empty string if no network is available. This function works both inside HAC and on an Android device.

Put InputAddressFN into field 1

Note, it does not access the network so the user does not incur any network charges.

InternalPathFN

Returns the full path to an apps internal database directory. It does not have the trailing '/'.

InternalPathFN

Put InternalPathFN into dbpath

PackageNameFN

Returns the package name of the app. This is useful as the Android system often refers to the app's resources using its package name.

Put PackageNameFN into field 1

SDcardFreeFN

Returns the size in kilobytes of the avaiable space on the SD card. It returns 0 inside HAC.

Put SDcardFreeFN into cardfree

SDcardSizeFN

Returns the size in kilobytes of the SD card. It returns 0 inside HAC.

Put SDcardSizeFN into cardsize

Toast Message

A toast message is a short lived message that appears in the lower center of the screen. It can be used to tell the user about the status of something they are doing or as a reminder.

ToastMessage text,duration

duration
0 := short - about 3 seconds
1 := long - about 5 seconds

ToastMessage 'Hello there',1

Notifications

Notifications are messages that appear in the status bar and are attached to the apps icon so they can be distinguished from other app and system notifications.

Touching a notification icon on the status bar will open up the notification panel so that the individual notifications and their texts can be seen. Each notification has both a title and a text body. Depending upon the notification settings, touching it could bring its owner app to the front or cancel the notification.

HAC apps can display up to 20 notifications at once and when a HAC built app quits it automatically cancels its notifications.

Currently HAC app notifications can play sounds but cannot access vibrate or flash the LEDS.

There are 4 stages to using a notification:-
1 - create the notification specifying its identity integer
2 - set any desired optional attributes
3 - send the notification
4 - cancel the notification if it is not set to auto-cancel.

NotifyCancel

This cancels the notification using the given integer identifier.

NotifyCancel nid

NotifyCancelAll

This cancels all the app's notifications.

NotifyCancelAll

NotifyCreateFN

This creates a new notification using the given integer identifier, identifiers must be greater than 0. It returns 1 if the notification was created successfully otherwise it returns 0.

NotifyCreateFN(nid,title,body)

Put NotifyCreateFN(1,'Announcement','New info available') into res

NotifySend

This sends the notification using the given integer identifier.

NotifySend nid

NotifySetAlertOnce

This sets the sound to play once.

NotifySetAlertOnce nid

NotifySetAutoCancel

This sets a notification to automatically cancel after being selected by the user. The default is to remain after being selected.

NotifySetAutoCancel nid

NotifySetNoClear

This sets the notification so that it will not be cleared when the clear button in the notification panel is pressed.

NotifySetNoClear nid

NotifySetOngoing

This flags the notification as being ongoing.

NotifySetOngoing nid

NotifySetSoundDefault

When a notification is sent there is the option to play a sound. This sets the sound to that of the default device sound. By default, no sounds are played.

NotifySetSoundDefault nid

NotifySetSoundInsistent

This sets the sound to play continuously.

NotifySetSoundInsistent nid

NotifySetSoundPath

When a notification is sent there is the option to play a sound. This sets the sound to that specified by the full file path, for instance an MP3 sound. By default no sounds are played.

NotifySetSoundPath nid,path

NotifySetTickerText

This sets the text that appears in the status bar after a title has appeared.

NotifySetTickerText nid,text

NotifyUpdateFN

This updates an existing notification using the given integer identifier. It returns 1 if the notification was updated successfully otherwise it returns 0.

NotifyUpdateFN(nid,title,body)

Put NotifyUpdateFN(1,'Announcement','New email arrived') into res

RSS

RSS are news feeds (URLs) delivering articles comprising both text and links to media. The most commonly used formats are RSS followed by Atom. HAC apps can process RSS/Atom sources from either a URL or a file containing data.

As some news feeds use custom formats, HAC tries to be flexible and processes as much of each article as possible. It does this by saving every field it can parse, and by creating a set of field names for each article so that the programmer/user can decide which fields to use. HAC creates fields in the order in which they were found.

As some RSS feeds can be huge, any attempt to download/process the entire feed could cause a memory exception. Therefore HAC allows feeds to be interrogated for their size and has options to process a specified range of articles. For instance, if there were 1000 articles in the source, then the user might like to just look at articles in the range 100 to 150 without downloading and parsing the entire source.

There are 3 steps in using RSS

1 - Get feed from URL or a file
2 - Call Parse feed
3 - Use results

Note, the parser just looks for top level elements/fields and does not look for sub-elements. For instance if there is an image within the description field it will be up to you to parse it out. However, if the feed consists of fields and has images using the image/log tag then they will be parsed and found.

RssArticleCountFN

This function returns the number of articles found by the previous function calls of RssParseCountFN and RssParseSourceFN.

Put RssArticleCountFN into count

RssClose

Closes the RSS and clears all articles plus their data.

RssErrorFN

Returns the error status for the last RSS command/function.

RssFieldByNameFN

For the specified article this function returns the content of the named field. Note, if more than one field has the specified name then the first field with that name will be accessed.

RssFieldByNameFN(article,name)

@ article 5, description field
Put RssFieldByNameFN(5,'description') into data

RssFieldByNumberFN

For the specified article this function returns the contents of the numbered field. This is useful when some fields share the same name, eg some RSS feeds have several description fields.

RssFieldByNumberFN(article,number)

@ article 5, field 9
Put RssFieldByNumberFN(5,9) into data

RssFieldCountFN

This function returns the number of field names for the specified article.

RssFieldCountFN(article)

@ article 5
Put RssFieldCountFN(5) into count

RssFieldListFN

This function returns in list form the names/attributes as found in the WHOLE source feed. For example, the title, author, image etc. Note, the attributes are unique and there are no duplicates. Some of the actual articles might not have all these attributes

Put RssFieldListFN into itemlist

RssFieldNamesFN

This function returns a list of field names found for the specified article. Note, more than one field may have the same name but contain different data.

RssFieldNamesFN(article)

@ article 5
Put RssFieldNamesFN(5) into fieldList

RssParseCountFN

This function returns how many articles are in the specified source. It parses the whole source but does not store articles. A limit can be placed by setting the max parameter. It returns the number of articles found otherwise it returns -1 if an error occurred.

Use this when you think that the feed may be huge and could affect stability.

RssParseCountFN(srcmode,source,max)

srcmode - 1 = file, 2 = URL
source - either full URL or full path to source file
max - number of articles

RssParseSourceFN

This function parses the whole source and stores the required article fields for later access. It returns the number of articles found otherwise it returns -1 if an error occurred. The range of articles stored can be set by specifying the first

article and the maximum number.

RssParseSourceFN(srcmode,source,start,max)

srcmode - 1 = file, 2 = URL
source - either full URL or full path to source file
start - first article (usually 1)
max - number of articles

Screen

There are two different commands for changing how fast the screen can be refreshed - **ScreenSpeedSet** and **AnimationSetPeriod**.

The command ScreenSpeedSet sets the maximum refresh rate and the sprite animation rate cannot exceed this. ScreenSpeedSet determines how fast the back buffer can be drawn to the screen and as the animation is drawn into the back buffer exceeding the screen refresh rate makes no sense and actually slows down the device.

The ScreenSpeedSet value should be at least as fast as the animation rate but if other graphics are being drawn, such as plots to a canvas, then it can be set higher. For instance usually 25 frames per second is the minimum for moving sprites, so set AnimationSetPeriod to 40mS and set ScreenSetSpeed at the default 30 frames per second.

Note - setting the ScreenSpeedSet value to higher than the device can handle will cause the app to lag. This is especially true of AVDs running on older PCs, 2GHz computers often have difficulties refreshing the screen at 5 frames per second. Modern tablets can perform much faster than older smart phones such as those running Android 1.6. The screen refresh rate does not mean a device will always have it's screen refreshed, the screen is only refreshed when something new is drawn on it or the program forces a refresh.

ScreenHeightFN

Returns the height of the first screen.

Put ScreenHeightFN into height

ScreenRefreshNow

This command forces the screen to be refreshed.

ScreenRefreshNow

ScreenSpeedFN

Returns the screen refresh speed in redraws per second.

Put ScreenSpeedFn into scrnspd

ScreenSpeedSet

Sets the screen refresh rate, the default speed is 30 refreshes per second.

ScreenSpeedSet 40

Due to Android hardware device limits the maximum number of screen refreshes is 60 per second. The emulator graphics speed is very much lower than that of a physical Android device. When testing on an AVD the app's performance will be

much better if the screen speed is set to 10 or lower, it also depends on the emulator's screen size. Once your app is ready for release the screen refresh can be set to the desired speed.

ScreenVisibleFN

This function indicates whether the app is foremost so that users can see its screen.

Put ScreenVisibleFN into visflag

HAC apps always update their screen when graphics operations take place, even when the screen is not visible. However in some circumstances a large speed improvement can be obtained by not updating the screen, such as when the app is in the background.

Note, Android is not a Real Time Operating System so sometimes it can take a few seconds for an app to be sent to the back.

ScreenWidthFN

Returns the width of the first screen.

Put ScreenWidthFN into width

StatusBarHide

Hides the status bar so allowing the app to use all of the screen.

StatusBarHide

StatusBarShow

Shows the status bar so returning the display to default mode.

StatusBarShow

Timer Commands

TimerSet

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 tim_id,mode,period

mode
0 = off
1 = single
2 = multi

period: measured in milliseconds, i.e 1220 = 1.220 seconds

TimerOff

This switches the specified timer into off mode.

TimerOff tim_id

UDP Messaging

UDP Messaging

UDP stands for User Datagram Protocol and is a fast way to send messages because unlike TCP it does not expect an acknowledgment from the receiver that the message was indeed received. The message is sent to the target as a datagram which has just two parts, the sender's address and the message content. It is useful when the loss of a single message is not crucial, although on local networks it is unlikely that messages will be lost.

UDP's advantages are:-
- fire and forget
- fast

Its disadvantages are:-
- does not work on some phone networks as they block ports
- if you and your target are behind a router then it might not work on the internet

Using UDP

1 - Create a UDP socket with the UdpCreateFN function
2 - Connect and bind the socket to a UDP port using the UdpConnect command that starts listening on that port.
3 - Send a message using the UdpSend command.
4 - Respond to an incoming message using the UdpEvent section.

Note, on some platforms the UDP listener will hear its own messages but they can be filtered out by checking if the source address of the datagram matches that of the device address. The maximum size of a datagram message is 65,508 bytes although some systems limit this to 8,192 bytes. Depending upon the network, if the message size exceeds the network limit then the message will be split into multiple datagram packets.

UDP Socket Events

When a UDP socket creates an event then the UDP event script will be called, this script is located inside the Specials section of the Editor. The UDP socket index and event type can be found using the UdpIndexFN and UdpEventFN functions respectively.

UDP Commands

UdpClose

This closes the UPD port as specifed by the given UDP socket index and stops listening for messages. If it it necessary to start listening again then just use the UdpConnect command for this socket.

UdpClose index

UdpClose 5

UdpConnect

This command makes a connection with the specified port using the given UDP socket index and starts listening for incoming messages. The time rest parameter specifies how much rest in milliseconds the listening thread should have. The buffersize sets the maximum size for the received message.

UdpConnect index,port,timerest,buffersize

UdpConnect 5,5042,1000,1024

UdpDelete

This deletes the specified UPD socket and clears any references from memory.

UdpDelete index

UdpDelete 5

UdpSend

This sends a datagram packet using the specified UDP socket using the given destination address, message and message encoding format. The UDP socket is already bound to a port using the previously ran UdpConnect command.

UdpSend index,address,data,encoding

UdpSend 5,'192.168.0.56','Hello','UTF8'

UDP Functions

UdpConnectedFN

Returns 1 if the specified UDP connection is used otherwise it returns 0.

UdpConnectedFN(index)

Put UdpConnectedFN(5) into used

UdpCountFN

This functions returns the number of created UDP sockets.

UdpCountFN

Put UdpCountFN into used

UdpCreateFN

This tries to create a UDP socket and returns the socket index if successful otherwise it returns 0.

UdpCreateFN

Put UdpCreateFN into idx

UdpErrorFN

This returns the error message from the last UDP socket which encountered an error. It is best used inside the UDP event script in conjunction with the function UdpEventFN to determine which error occurred.

UdpErrorFN

Put UdpErrorFN into errormessage

*** Example inside event script

```
Local idx,evt

Put UdpEventFN into evt
If evt=3 then
   Put UdpIndexFN into field 1
   Put UdpErrorFN into field 2
EndIf
```

UdpEventFN

This returns the event number that the UDP socket created, it is best called inside the UDP event script.

There are three event possibilities:-
1 Data received
2 Data sent
3 Error

UdpEventFN

Put UdpEventFN into event

*** Example inside event script

```
Local evt

Put UdpEventFN into evt
If evt=1 then
   Put 'Data received' into field 1
else
   If evt=2 then
     Put 'Message sent' into field 1
   Else
     Put 'Error' into field 2
   EndIf
EndIf
```

UdpIndexFN

This returns the index of the last UDP socket to create an event. It is best used inside the UDP event script to determine which UDP Socket called the script.

UdpIndexFN

Put UdpIndexFN into index

UdpListFN

This function returns a list of the currently used UDP socket numbers.

UdpListFN

Put UdpListFN into list

UdpPortFN

This returns the port number of the specified UDP socket.

UdpPortFN(idx)

Put UdpPortFN(2) into port

UdpRxAddressFN

Returns the input address of the sender of the datagram. It us best used inside the UDP event script

UdpRxAdressFN

Put UdpRxAddress into sender

UdpRxDataFN

Returns the data portion of the received datagram. It is assumed that the receiver already knows the encoding format of the message. It is best used inside the UDP event script.

UdpRxDataFN

Put UdpRxDataFN into message

UdpStateFN

Returns 1 if the specified UDP socket is enabled otherwise it returns 0.

UdpState(index)

Put UdpStateFN(5) into used