Support Group Application Note
Number: 061
Issue: 2
Author:

Acorn

Creating 6502 Applications using !65Host

Applicable Hardware :	Archimedes range	Related Application Notes:	Archimedes 6502 Emulation 6502 to ARM Application Note

Support Group
Acorn Computers Limited
Acorn House
Vision Park
Histon
Cambridge CB4 4AE

OBTAINING !65HOST

The !65Host BBC Emulator supplied on the RISC OS Extras disc has many useful features above and beyond that of the version supplied on Applications disc 2. The RISC OS Extras disc is available from Acorn dealers.

!65Host's name comes from the 6502 microprocessor, which is at the heart of the BBC Model B. The 'Host' part is derived from the fact that the program emulates a stand alone machine rather than a second processor add on. It is possible using !65Host to take BBC B, B+, Master 128 or Master Compact software, and to package it up conveniently as a single RISC OS application. That is a single icon can be created, that when double clicked on, will start the BBC emulator and enter it running a particular BBC program. What is more, such applications once created, may be moved freely between ADFS hard discs, ADFS floppy discs, Econet or SCSIFS.

A LITTLE BACKGROUND INFORMATION

In order to fully understand how to create "6502 application" it will be necessary to discuss a little of how the RISC OS desktop works.

To create an application, first create a directory but begin its name with the ! (pronounced pling) character. Instead of the light blue folder icon, it will be given the default application icon (an Archimedes logo with the word APP in its top left corner).

When you double click on a directory, a window called a directory viewer will open to show its contents. In order to show an applications contents the shift key should be held down when double clicking, and the application will open just as a directory does. If you try this on the application supplied with RISC OS such as **!Paint, !Edit, !Draw** and **!65Host** you will see they contain several files. Sometimes you will see that an application has subdirectories inside itself.

When you first open a directory viewer RISC OS "sees" the applications it contains. If the application has not been "seen" before then RISC OS looks inside the application for a file called !Boot and if it finds one it executes it. The job of a !Boot file is to let RISC OS know where the application is and what it can do. For example !Draw's boot file lets RISC OS know that there is an application, !Draw, that can be run and passed a Draw file if it is double clicked on.

!65Host's !Boot file is very simple and just sets up a system variable that is the full path name of !65Host. A system variable is just like a string variable in BASIC. To use the contents of a system variable refer to it by its name enclosed in angle brackets. !65Host's !Boot file sets up a variable called **<65Host$Dir>**. For instance you can use this variable to view the contents of !65Host by typing ***CAT <65Host$Dir>**.

When you double click on an application RISC OS looks for a file inside the application called !Run. If one is not found you will actually receive an error message. For this reason alone all applications should contain a !Run file. The job of the !Run file is to kick off the actual process that leads to the application running. This might involve loading modules into memory, declaring how much memory is needed, setting system variables and so on. It could, however, be as simple as a single * command.

The !Run file of an application is almost always a special form of command file called an Obey file. Obey files may be created from !Edit by menuing on the !Edit icon on the icon bar and selecting **Create->New Obey File**. Obey files when run, pass each of their lines to the * prompt just as if you had typed the line at the * prompt. They also have the advantage of knowing where they are. That is the system variable **<Obey$Dir>**, is the full path name of the obey file currently executing. Comments may be added to obey files by placing a " | " character at the start of the line, and it is commonplace to see such comments.

If you now open up !65Host and drag its !Boot file onto !Edit you should see how it works. The final line has a command that we haven't discussed yet, we will deal with its implications later.

CREATING YOUR OWN 6502 APPLICATION

We now have enough information to create a "6502 application". The program in question is given below and is a standard "**Hello World**" style program merely designed to boost our confidence. It prints a read "Hello World" banner and catalogues the current directory.

Format a 3.5 inch disc and copy !65Host onto it. Double click on !65Host to start the emulator and enter the following:

```
*MOUNT 0
10 MODE 7
20 PRINT CHR$ (129); CHR$ (157); CHR$ (131); CHR$ (141); "Hello World"
30 PRINT CHR$ (129); CHR$ (157); CHR$ (131); CHR$ (141); "Hello World"
40 *CAT
SAVE "HelloW"
*BUILD !Boot
0001 CHAIN "HelloW"
[Escape]
*QUIT
```

You should now have left the emulator and be back in the RISC OS desktop. The directory viewer for your disc will now have icons for the two files you just created in the emulator. Next create the directory **!HelloW** in the root of the disc next to !65Host. Hold down shift and open !HelloW then create the following obey file and save it into !HelloW as !Run.

```
Dir <Obey$Dir>.Program
Run <65Host$Dir>.!Run &B03 !Boot
```

Then create a subdirectory Program inside !HelloW, and move the two files you created under the emulator into the Program directory.

To recap you should now have...

- **!65Host (Application)**
  - **!Run (Obey File)**
    - **!Boot (Command File)**
- **!HelloW (Application)**
  - **Program (Directory)**
    - **HelloW (BASIC File)**

The moment of truth. Double click on !HelloW. The BASIC program HelloW will be run under 6502 emulation.

The bottom level of directories corresponds to the "Shift-Breakable" BBC disc, and this is where other "6502 applications" that you may create, would hold their actual programs.

!65HOST BOOT OPTIONS

The **ReadMe** file inside !65Host gives the technical details of the boot option that we used in the above !Run file. In our case we used **&B03**, as a !Boot option, and it is worth looking at this in a little more depth. The hex digit 3 (bits 0-1) instructed the BBC Emulator to use the same boot action as ***OPT4,3** would confer on a floppy disc, that is **EXEC**. The hex digit B instructs the emulator to start up in a generic filing system called ARFS.

As we created !HelloW on an ADFS floppy disc we could have use &003 as the boot option and entered the default filing system, ADFS. Or &803 could have been used, specifying ADFS directly. However, this would mean that the !HelloW application, if copied to a network directory, would no longer function. It is for this reason that we specified ARFS. As RISC OS makes its filing systems appear very similar we can specify ARFS and in effect say 'I don't care which filing system we enter'. This confers filing system portability on any "6502 application" created in this manner.

GIVING YOUR APPLICATION ITS OWN ICON

Now we have created our portable application, we will most probably want to give it its own unique icon, so it may be distinguished. This is easily done from !Paint.

Start !Paint and click on its icon to create a new sprite file. Menu on the empty browser window and select "create". The name of the sprite should be the same as that of our application "!HelloW". The sprite should also have width of 34, a height of 17 and be in mode 12. These settings are standard that must be adhered to. Now use !Paint to design your sprite and when you have finished save it into !HelloW as a file called "!Sprites".

Now when you application is 'seen' on the Desktop your sprite will be used for its icon. RISC OS does this automatically, however, to be absolutely correct we should create a !Boot file for our application that contains the line:

```
Iconsprites <Obey$Dir>.!Sprites
```

SOME FURTHER IDEAS

If you intend to use your 6502 applications on a network then it is probably a good idea to create the applications as indicated so far. This means that each user should have a copy of !65Host in their directory. Alternatively, a single copy of !65Host may be referenced by the !Armboot file in each users directory. For example if !65Host were placed in the directory $.Apps you may use a line such as this:

```
Run $.Apps.!65Host.!Boot
```

You must ensure that all the files that comprise !65Host are given public read access. This approach could potentially save 100K per user.

Another approach that may be attractive is to 'hide' !65Host inside your 6502 Application, next to your !Run file. This has two benefits:

a) Everything needed by the application is contained within it, and it may be easily copied from disc to disc.

b) It may be less confusing for some users, as they need only to see a single icon.

You may like to add a function key definition that quits the 6502 emulator and return the user to the desktop. For example we could have added an extra line to our !HelloW application's !Boot file to set key F1 to quit:

```
*KEY1 *QUIT |M
CHAIN "HelloW"
```