

Black Support For Network-Only Machines Functional Specification

Drawing No	1303/017/FS/3P95
Issue	1/3P95
Date	03/10/94
Author	Martin Clemoes, Martin NS, Jonathan Coxhead
Sheets	
Last Issue	I

1. History

3/10/94	MC	Issue 1 prepared for AMR (no change since Issue I).
20/3/95	MC	Updated for Developer pack.

2. Outstanding Issues

None.

3. Product Overview

RISC OS 3 version 3.5 was designed assuming the presence of a hard disc. We have taken the opportunity to widen the horizons to support networked machines where local mass storage may not be present. RISC OS Black takes into account the possibility of being set up to boot from either a hard disc or network, but not from floppy. A machine set up to boot from net would not be fully functional without a network connection, although it would at least display a screen to, for example, indicate what any net fault is.

This document specifies the changes to be made to RISC OS 3.5 and its accompanying disc image to support network booting. These changes consist of new network booting applications, entitled ArmBoot and ShareBoot supplied on floppy disc, plus some extensions to the OS ROM. Such network booting software would be used by a privileged system user to set up fileservers from which to boot network only machines.

4. Concepts And Definitions

4.1 Problems To Solve

RISC OS 3.5, the starting point for the RISC OS Black development, was designed on the assumption of the presence of a hard disc in each machine. Widening the specification to allow network booting introduces the following problems to solve:

a) Network storage areas user specific, not machine specific - a user may log on on different machines, with the same URD ... not possible with an ADFS hard disc alone.

b) Network connections may fail, so the machine has to be more self-sufficient without mass storage.

c) Security is more important on networks than hard discs.

d) Networks may contain mixtures of machines with various releases of RISC OS.

e) Applications may not be placed in standard places - a greater flexibility is required in locating applications to be displayed in the icon bar 'Apps' viewer.

f) Networks have lower and variable data transfer rates compared with hard discs.

g) More than one type of network must be supported - see 4.2 below.

h) Network fileservers are usually set up on site by IT coordinators, not pre-set on manufacture - introducing more user complexity and room for error.

4.2 Design Principles

The problems to solve listed above lead to a combination of technical solutions and design principles:

- Complexity is kept to an absolute minimum (for ease of use for client and IT Coordinator).

- User familiarity is retained as much as possible - user interface differences from hard disc machines minimised.

- No files need to be stored on each client machine (eg. on floppy) - all files are loaded, etc. from the network.

- Configuration is split between machine-specific details stored in central system privilege, world read only fileserver areas indexed by machine-specific address, and user-specific details stored in users' URD's.

- The RISC OS Black ArmBoot application remains compatible with all versions of RISC OS from 2.0 onwards, merely running with version-dependent features disabled as appropriate.

- The mechanism must work on Acorn Access and Level 4 networks (including Econet only, Ethernet only, and mixed AUN nets). No mechanism exists for NFS, and so is not specified here.

4.3 Boot-up Sequence

On a Level 4 network:

1) The client machine is configured to FileSystem Net, Boot, so when it is powered up or reset with CTRL/BREAK or CTRL/RESET, it initialises its ROM modules (system ROM plus network card ROM) and logs on as user 'Boot' to the network.

2) The client machine then runs the ArmBoot APPLICATION that is stored in

the read only Boot account (ususally the fileserver exported root directory).

3) The ArmBoot application may overwrite the client CMOS configuration memory with a CMOS image stored within the ArmBoot application on the net with a file/path name indexed by the client machine address, as discussed in §5.5.

4) The ArmBoot application contains read-only resources such as System, Fonts etc. These are booted.

5) The ArmBoot application executes a *Net:Bye to log the client off (as 'Boot').

6) The client machine is now ready for use by an end user. This user now logs onto the net using their own ID and password, displaying their URD.

7) An ArmBoot FILE is run automatically from the URD. This is of type Desktop.

8) The ArmBoot file filer_boot's: read/write resources such as Scrap; non-ROM applications placed in central read-only directories.

9) The ArmBoot file runs chosen applications etc. as a Desktop save file normally does.

10) The client machine is now fully available for work, being set up with both machine specific and user specific configuration.

On an Acorn Access network:

1) The client machine is configured to FileSystem Share, Boot, so that when it is powered up or reset with CTRL/BREAK or CTRL/RESET, it initialises its ROM modules and mounts the configured directory over the net.

2) The client machine then runs the ShareBoot APPLICATION that it finds in the mounted directory.

3) The ShareBoot application may overwrite the client CMOS configuration memory with a CMOS image stored within the ShareBoot application on the net with a file/path name indexed by the client machine address (as discussed in §5.5), with read only access permission.

4) The ShareBoot application contains read-only resources such as System, Fonts etc., plus the read/write resource Scrap. These are booted.

5) The client machine is now ready for use by an end user. This user now menus on the mounted disc icon on the icon bar, chooses 'Show discs' to display then mount other discs containing their working files.

6) If the user wishes, he/she double clicks on a Desktop save file they have stored earlier, eg. with a filename based on their own name.

7) The Desktop save file filer_boot's non-ROM applications placed in central read-only directories.

8) The Desktop save file runs chosen applications etc.

9) The client machine is now fully available for work, being set up with both machine specific and user specific configuration.

4.4 Objects New Or Changed

a) A new application 'ArmBoot' will be created for Level4 booting. This will be created by modifying the Boot application supplied with RISC OS 3.5, which will be supplied largely unaltered with hard disc RISC OS Black systems.

b) A new application 'ShareBoot' will be created for Acorn Access booting. This is the sister application of the almost identical 'ArmBoot' application referred to above, and will also be created by modifying the Boot application.

c) A new application called 'MchConfig' will be created to assist IT coordinators saving configuration within ArmBoot or ShareBoot ready for loading at end user boot time.

d) A new relocatable module 'BootCommands' will be created to implement some of the command line utilities such as *AddApp contained in !Boot.Library in RISC OS 3.5. This will guarantee the availability of these commands without requiring repeated reloading from the file server.

f) The Boot application will be modified (leaving the user interface unchanged) to take advantage of changes to ROM contents, eg. no need for some command line utilities as a result of d) above.

g) A cut-down version of Boot for emergency use (floppy disc boot) when net boot has failed. This will contain Scrap, but have several other resources removed to leave at least 1MB free on a 1.6MB floppy disc for Scrap to expand into.

h) A new version of Configure is needed. See §5.9 below to see more details.

i) The Desktop module will be modified to check for successful system boot when the desktop is started. If the configured boot-app could not be run, for any reason, then a message will be displayed saying that the boot has failed, giving the reason for failure and providing the options 'Retry', 'Cancel' and 'Boot from floppy.' The latter will simply execute the command '/ADFS::0.!Boot'; it will not honour the Opt 4 setting of the disc.

4.5 Interaction With Lock

The Lock configuration facility, as implemented for RISC OS 3.5, can lock CMOS from user alteration either in software alone, or as controlled by a hardware link. If CMOS is locked, ArmBoot and ShareBoot will not attempt to over-write CMOS during network boot. Thus the IT coordinator has a choice of two ways of protecting CMOS of network booted machines from user tampering - either overloading with definitive CMOS images from the network at boot time, or locking CMOS. Both cannot be used together.

The decision as to which of these (if any) is appropriate will probably call for some discussion in the manual, as it is a difficult one since the CMOS contains a mixture of local (machine-specific) and global (network-specific) state, together with user preferences.

4.6 Multiple Boot Applications

Users of machines with hard discs may execute the !Boot application on their hard disc when they power up, then use a network and encounter !ArmBoot or

ShareBoot applications. Any attempt to execute such a second boot application will have no effect.

If two of the boot applications, eg. ArmBoot and ShareBoot, are placed next to each other in the same directory - eg. the root exported directory of a fileserver - then a client machine may 'see' both applications at the same time. Depending on the filing system, one will be run, the other just booted. Use of environment variables will ensure that the booting of the second boot application will have no effect.

5. User Interface

5.1 UI Road Map

5.1.1 End-User Boot-Up

The end-user boot-up sequence of events and actions is as described in §4.3 above.

5.1.2 Level4 Fileserver Setup

This is done by the IT coordinator, either using the fileserver disc directly from ADFS on the host machine, or logging on as system user. He/she:

- Ensures an appropriate area of the disc is exported over the network as a Level4 fileserver.
- Copies ArmBoot from the Network Setup Application Suite discs to the root of the exported area, removing Configure from inside ArmBoot to a safe place if he/she so wishes.
- Places a copy of Scrap in each user's URD as is usual for Level4.
- Copies the rest of the Network Setup Application Suite to a subdirectory 'Apps' of the exported root directory so that ArmBoot ensures they are automatically displayed in the Apps icon viewer.

5.1.3 Acorn Access Host Setup

This is done by the IT coordinator, using the exported disc directly from ADFS on the host machine. He/she:

- Exports an appropriate area of the disc as a shared network resource.
- Copies ShareBoot from the Network Setup Application Suite discs to the exported area, removing Configure from inside ShareBoot to a safe place if he/she so wishes.
- Ensures that all of ShareBoot is exported read-only except for Scrap inside it which is exported read/write.
- Copies the rest of the Network Setup Application Suite to a directory 'Apps' placed next to ShareBoot, making sure this area is exported read-only.

5.1.4 Client Machine Setup

This is done by the IT coordinator, working from the client machine in question. He/she:

- Gains write access to ArmBoot/ShareBoot by logging on as system user (Level4) or adjusting access details temporarily to export read/write using the host machine while no users are around (Acorn Access).
- Uses MchConfig to set up a machine-specific directory within ArmBoot/ShareBoot.
- On the client machine runs Configure and uses the command line to set the client CMOS as required (eg. FileSystem Share, Boot).
- Uses MchConfig to save either the machine's CMOS, or the machine's configuration files, or both, to a definitive copy stored within ArmBoot or ShareBoot within a directory named after the unique address of the client machine (ie. specific to the client machine).
- If using Acorn Access returns to the host machine and returns the access permission for ShareBoot to read-only export, except for Scrap (read/write).

5.1.5 End-User Option Save

This is done by the end user working from a client machine, to save preferences such as which applications get auto-run on boot. He/she:

- Sets the machine up as they want it to boot up.
- Opens a directory viewer for their URD (Level4) or a disc area they can write to on Acorn Access.
- Saves the Desktop save file from the TaskManager 'nut' to this viewer. If using Level4 they use the filename '!ArmBoot'. If using Acorn Access, they use any name they wish, eg. a name derived from their personal name.

5.2 Boot (standard version)

The user interface of the Boot application will not be materially altered by RISC OS Black. The functionality of Boot will be altered in detail by removing some components as their features are added elsewhere. The utilities in !Boot.Library implementing command line keywords will be superseded by the BootCommands module (see 5.6) under Black (but the library versions must still exist in ArmBoot and ShareBoot for older operating systems). If a part of ARMovie is placed in ROM, it will of course not be duplicated in Boot. See the associated 'Black ROM and Disc Structures Functional Specification', 1303,006/FS, for more details of what is to be moved from Boot to ROM.

5.3 ArmBoot and ShareBoot

The ArmBoot and ShareBoot applications are almost identical. Their

differences are: name (as required for Level4 and Acorn Access respectively); ArmBoot logs the user ('Boot') off at the end of its boot sequence; and ShareBoot contains Scrap and ArmBoot doesn't. ArmBoot can't contain Scrap, since the Boot URD it resides in is world read-only. Acorn Access allows more flexibility on this, so Scrap can be included in ShareBoot.

The sister applications are both closely related to Boot. Their user interface, structure and contents are basically the same, their functionality differing from Boot in the following respects:

a) ArmBoot and ShareBoot will both 'work' - go through a boot sequence without failing with compatibility errors - on clients running all versions of RISC OS from version 2.00 onwards. Operating system version dependent features will be automatically disabled rather than attempted producing errors on early RISC OS versions. The RO200Hook and RO310Hook obey files will be replaced with directories of the same names containing !Run files and RISC OS 2 and 3.10 resources so that instead of producing errors, they obey the !Run files and load resources as required by the earlier RISC OS versions. Note that System contains directories for modules specific to versions of RISC OS, so System and its contents do not need to be placed in these 'Hook' directories. Example resources that do are RISC OS 2 Fonts within RO200Hook, and NewLook within RO310Hook.

b) ArmBoot and ShareBoot both contain machine configuration subdirectories into which the MchConfig application can be used to save CMOS files.

c) When a client machine boots with ArmBoot or ShareBoot, if a CMOS file is stored within a subdirectory with a name mapping to the client's machine address then that CMOS image will be automatically loaded into the client's CMOS - overriding any temporary configuration set up by users. If CMOS is locked on the client machine, either by software or by the hardware link, CMOS is not loaded.

d) At the end of its boot sequence, ArmBoot logs the user off the fileserver as pseudo-user 'Boot' so they can log on next to their own account.

e) ArmBoot and ShareBoot both contain a directory Library with utility programs. Boot applications under Black do not need this because these commands are present in ROM.

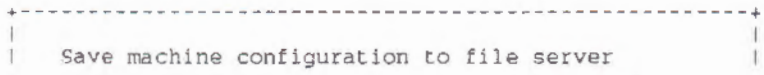
5.4 Boot (cut-down application)

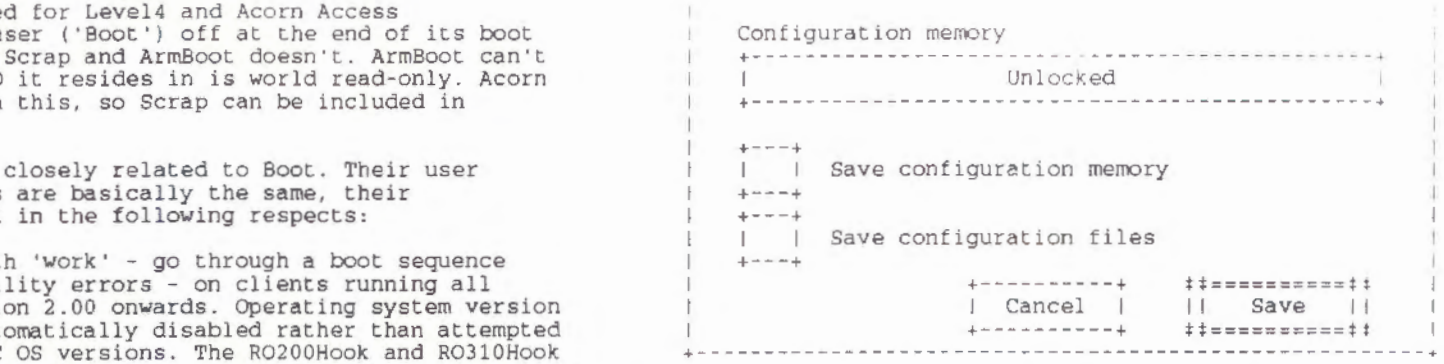
This is the standard Boot application cut down for floppy disc use by the removal of ARMovie (application and data), Configure (the application) and Fonts. Double clicking on this version of Boot displays the error 'You cannot reconfigure this machine'.

5.5 MchConfig

A new application MchConfig will be written to allow a system manager to configure individual stations separately. This will be based on SaveCMOS. It differs from SaveCMOS in that it does not have a Restore button/action.

When you double click on MchConfig, it will offer a dialogue box





The option buttons are initialised to 'Save configuration memory' only, as indicated above. The read-only text field below 'Configuration memory' is used to indicate the current settings of the CMOS lock and the keyboard link. It contains the text 'Unlocked' (as in the example above), 'Hardware locked', 'Software locked' or 'Hardware and software locked'. If in any state other than 'Unlocked', the only option available is 'Save configuration files': the others are shaded.

Choosing 'Save' will display a SaveAs dialogue initialised to the machine-specific directory associated with the machine being used (unlike SaveCMOS, which silently saves the CMOS file within itself on disc); and choosing 'OK' on will save the chosen aspects of machine configuration in it (creating it if it doesn't already exist). To do this, it:

a) Determines which Filesystem has been configured.

b) Uses the unique machine address to calculate a directory name which is guaranteed to be unique for this computer, using the algorithm that Scrap uses. This name will be denoted <machine>.

c) Initialises the SaveAs filename field to '!<boot-app>.MchConfig.<machine>' so most IT coordinators using MchConfig are aware where the files are being saved to, but merely need to click on OK to perform the save. The SaveAs box icon is a directory icon, to indicate that a directory will be created. If dragged, it must be dragged to a Filer viewer.

d) If saving configuration memory, saves the CMOS of the machine into !RO<version>CMOS, and if saving configuration files, copies the entire contents of <Choices$Dir> into the new directory.

s) When new copies are made of the configuration files, the Choices$* variables will be updated to refer to the new locations.

To reverse the effect of MchConfig, the system administrator must delete the unwanted files from inside the machine-specific directory.

5.6 BootCommands relocatable module

This implements the following *commands:

*AddApp <Application name>
*AppSize <Memory size in Kb>K
*Do <Command>

```
*IfThere <Filename> Then <Command>
*Repeat <Command> <Directory> [-Directories] [-Applications]
        [-Files | -Type <file type>] [-CommandTail <cmdtail>] [-Tasks]
*SafeLogon [[:]<station number>]:<File server name>]
        <user name> [[:<CR>]<Password>]
*LoadCMOS <file name>
```

These *commands are the ones implemented by library utilities in !Boot.Library on RISC OS 3.5, together with 2 new ones - SafeLogon and LoadCMOS.

SafeLogon is a command that executes Logon in an idempotent fashion: i e, it checks first to see if the user is already logged on to the given file server, and if so, does nothing. (If NFS is the current filing system, it does not interfere in this way: this is NetFS-specific.) This change allows Desktop files saved from the Switcher icon to work when saved as a user's personal startup file (either as &.!ArmBoot in their own URD or using some local ShareFS convention).

Note that the boot applications will alias "logon" to "safelogon" and leave this alias set, so that it is still set when a user comes to log on and uses a desktop save file. This will be noticeable to users who use a repeated *logon command to effectively do a *bye then *logon - they will have to do the *bye as well as the second *logon.

LoadCMOS is a command that takes a CMOS-image file, and loads it into the CMOS of the computer. All of the CMOS is overwritten, apart from:

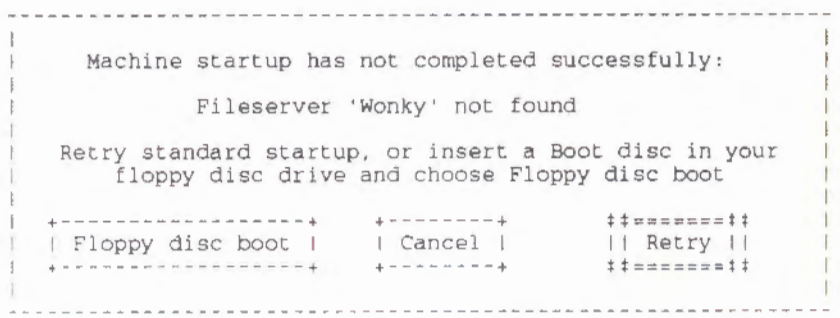
- the station number
- the current year
- the DST flag

since these are either not expected to be configured incorrectly, or else are expected to change in normal use. After the new CMOS has been loaded, the CMOS checksum will be recalculated.

None of these commands will have any dependency on disc resources such as Scrap.

5.7 Desktop

When the Desktop starts up it will check the existence of the system variable Boot$Error. If found, it will display a dialogue box. The overall layout of the dialogue box is:



Retry is the default action button. The error message reported is of course just an example, assuming that Boot$Error has been found to be "Fileserver 'Wonky' not found".

'Retry' has the effect of performing a shutdown and hard reset.

Since this dialogue behaviour will only be implemented by the RISC OS Black version of the Kernel and Desktop modules, it will not appear on machines with earlier RISC OS versions booting from ArmBoot or ShareBoot. Such earlier machines will (as they do at the moment) display error messages at a command line, then lose them and enter the desktop if the configured language is the desktop (default).

5.8 Kernel

The kernel will be modified so that if it fails to locate the <boot-app> determined by the CMOS settings (the configured file system, file server, etc), it will save the error message responsible for the failure in a system variable Boot$Error.

5.9 Configure

Configure will be supplied within Boot, ArmBoot and ShareBoot in the same way that it is within RISC OS 3.5 Boot (though with the expectation that IT coordinators may well choose to remove it elsewhere).

It will be modified in the following respects:

a) Instead of always editing files in Boot, as it does under RISC OS 3.5, it will understand that ArmBoot or ShareBoot may instead be the boot application. It determines which of these to edit by looking at the setting of the configured filesystem: ArmBoot for Net, ShareBoot for Share, Boot for anything else. (If the user wants to edit more than one of these, he/she must use the *command to change the configuration. Configure remembers which is the case at startup: it will not be confused if it changes while it is running.)

b) In the case of a network boot-app, it will always use the files found in <Choices$Dir>. In the sequel, these files are referred to as the "configuration files." In other words, the configuration files are one of:

!<boot-app>.MchConfig.<machine>.Boot.PreDesk.Configure and
!<boot-app>.MchConfig.<machine>.Boot.Tasks.Configure

where <boot-app> is one of ArmBoot, ShareBoot; or

!<boot-app>.MchConfig.RO<>Hook.Boot.PreDesk.Configure and
!<boot-app>.MchConfig.RO<>Hook.Boot.Tasks.Configure

where <boot-app> is one of ArmBoot, ShareBoot; or

!<boot-app>.Choices.Boot.PreDesk.Configure and
!<boot-app>.Choices.Boot.Tasks.Configure

where <boot-app> is one of Boot, ArmBoot, ShareBoot,

and in any case may be referred to as <Choices$Dir>.PreDesk.Configure and <Choices$Dir>.Tasks.Configure.

c) Its System merging feature is updated to allow merging of the more complex System applications as defined in §6.10.

d) If the configured file system is Net or Share, Configure users (ie. the IT coordinators) are warned every time they change global state, ie. change options which will result in changes to files within ArmBoot/ShareBoot which are common to all users - change either of the last two pairs of files above.

f) In order to edit the global configuration files, it is necessary to use a machine with no local ones. If there is no such machine, editing the global ones is pointless anyway, so this is not a problem.

5.10 System

There is no user-interface change to this application. A new version of SysMerge will be produced to allow harmonious updating of existing and new Systems.

6. Programmer Interface

6.1 Overview

There is no requirement for programming skills on the part of the IT coordinator. The following information should not be needed by any user of the system, but may be useful to those writing applications with special requirements.

6.2 Boot

The program interface to a Black Boot application remains as specified in the RISC OS 3.5 PRM supplement.

6.3 ArmBoot and ShareBoot

The program interface to a Black ArmBoot or ShareBoot application is broadly as specified in the RISC OS 3.5 PRM supplement. However, a program is not expected to modify these files, since they are typically shared over a network and will not be writable.

ArmBoot and ShareBoot will be based on the RISC OS 3.5 Boot application with changes as developed in the prototype versions of ArmBoot and ShareBoot produced by Carl Sellers and David Walker for the Training Centre network.

Where RISC OS versions are used for directory names, etc., they are derived from the version of the UtilityModule on the machine in question.

This implies the following structure:

```
!<boot-app>.!Boot
            !Help
            !Run
            !Sprites
            !Sprites22
Choices.Boot.Desktop
            PreDesk.BandLimit
                    Configure
                    (etc)
            PreDesktop
            Tasks.Configure
                  (etc)
       <other-choices>
Library.AddApp
       AppSize
       Do
       FontMerge
       IfThere
       Repeat
       SafeLogon
       LoadCMOS
MchConfig.<machine>.<cmos-image>
                    Boot.Desktop
                         PreDesk.BandLimit
                                 Configure
                                 (etc)
                         PreDesktop
                         Tasks.Configure
                               (etc)
                    <other-choices>
                    (repeated as required)
Resources.!ARMovie
         !Configure
         !Fonts
         !Scrap (ShareBoot only)
         !System
         Configure.2DTools
                   ClrMonitor
                   FontChange
                   Monitors.Acorn.AKF50 (etc)
                   Textures.T1 (etc)
Utils.BootRun
     DeskRun
     HOff
     HOn
     RO200Hook.Boot.Desktop
                    PreDesk.BandLimit
                            Configure
                            (etc)
                    PreDesktop
                    Tasks.Configure
                          (etc)
               <other-choices>
     RO310Hook.Boot.Desktop
                    PreDesk.BandLimit
                            Configure
                            (etc)
                    PreDesktop
                    Tasks.Configure
                          (etc)
               <other-choices>
     RO350Hook.Boot.Desktop
                    PreDesk.ARPlayer
                            BandLimit
                            Configure
                            DPMSUtils
                            WimpUtils
```

```
                                    Configure
                                    (etc)
                               PreDesktop
                               Tasks.!Boot
                                     !ROMPatch
                                     Configure
                                     (etc)
                           <other-choices>
                    VProtect
```

The !Run file is executed by the kernel as the machine starts up, and it performs the following sequence:

```
IF    there is a saved CMOS file
         !<boot-app>.<machine>.!RO<version>CMOS
THEN  load it
FI

IF    there is a machine-specific directory !<boot-app>.<machine>.Boot
         for this machine
THEN  Set Choices$Dir to !<boot-app>.<machine>
ELSE  IF    there is an operating system-specific directory
               !<boot-app>.RO<version>Hook.Boot for this machine
      THEN  Set Choices$Dir to !<boot-app>.RO<version>Hook
      ELSE  Set Choices$Dir to !<boot-app>.Choices
      FI
FI

Set an alias so Net:Logon translates to SafeLogon
IF    <version> >= 3.50
THEN  Set an alias so Desktop_SetPalette translates to '|' - so that
         such lines in desktop save files silently have no effect.
FI

FOR   each file in Resources
DO    Filer_Boot it
OD

FOR   each application in <boot-app>$Dir.^.Apps
DO    AddApp it
OD

/<Choices$Dir>.PreDesktop
FOR   each file in <Choices$Dir>.PreDesk
DO    take the action according to its type
OD

/<Choices$Dir>.Desktop
FOR   each file in <Choices$Dir>.Tasks
DO    take the action according to its type
OD
```

Some points to note:

- The above code is operating system version-independent. If we create a version 4.00 of RISC OS, an RO400Hook boot, if present, would be performed by such a machine; if one were not present, the standard boot would happen anyway. This maintains forward and backward compatibility.

- The variables Choices$* are always set to the values documented in the RISC OS 3.5 PRM supplement. This will always be useful for applications which wish to read options settings, and may be useful to allow a system manager to set up commonly used preferences in a network environment.

- The contents of the directories PreDesk and Tasks are at the discretion of the system manager. The diagram above shows the contents of a RISC OS 3.5 Boot application in the RO350Hook directory, as would normally be the case. Where possible, fixed versions of these will be used. In particular, RomPatch is 3.5-specific: if there is a requirement for patching of the Black ROM, this would be a done by a RomPatch in the main Choices directory, also as shown above.

- The contents of RO{200,310,350}Hook are designed to allow working under older versions of the operating system. Under RISC OS 3.5, they are obey files, run from <boot-app>.!Run rather than <boot-app>.Utils.BootRun and <boot-app>.Utils.DeskRun. In the prototype referred to above they are replaced by directories which themselves conform to the specification of a 3.5-style Choices directory. This allows consistent configuration across all platform types.

- Although Black will be providing IfThere, Repeat, AddApp etc as ROM-based *commands, they must also be present in the library in order to support RISC O S 2, 3.1 and 3.5 machines.

- The PreDesk sequence will define Alias$Logon to be SafeLogon. SafeLogon will be provided in !<boot-app>.Library, and also in the BootCommands module, q v.

- The CMOS images are held inside the <machine> directory for a given machine. The name given to the file is derived from the version of the operating system under which it was created, as follows:

RO200	!RO200CMOS
RO201	!RO201CMOS
RO300	!RO300CMOS
RO310	!RO310CMOS
RO311	!RO311CMOS
RO350	!RO350CMOS
Black	!RO360CMOS

When a machine starts up, it will only look for a file that matches its own version number. In this way, if a machine is upgraded but retains its station number, it will not load incompatible CMOS.

The file names all start with '!' in order to avoid conflict with the recommendation that applications save their own configuration data in directories that do not start with '!'.

- The BandLimit commands are platform-specific, rather than OS-specific. If the default value from the OS-specific directory is not suitable for a certain machine, that machine will have to be equipped with its own configuration files using MchConfig, and its BandLimt file (which is an obey file containing a *VIDCBandwidthLimit command) edited "by hand."

6.4 Boot (cut-down application)

The program interface to a Black Boot application remains as specified in the RISC OS 3.5 PRM supplement. The contents are reduced in a way consistent with the original design, leaving the following.

```
!Boot.!Boot
     !Help
     !Run
```

```
!Sprites
!Sprites22
Choices.Boot.Desktop
            PreDesk.Configure
            PreDesktop
            Tasks.!ROMPatch (if necessary)
                  Configure
Library.FontMerge
Resources.!Scrap
         !System
         Configure.2DTools
                   ClrMonitor
                   FontChange
                   Monitors.Acorn.AKF50
                   Textures.T1
Utils.BandLimit
      BootRun
      DeskRun
      HOff
      HOn
      VProtect
```

6.5 MchConfig

MchConfig will be coded as specified above. The address used will be derived using the same algorithm as Scrap, which guarantees consistency on all Acorn networks.

6.6 BootCommands relocatable module

The BootCommands module will be based on the code existing in the RISC OS 3.5 library applications, although one of them will have to be converted from BASIC. ROM space considerations will probably imply an assembler implementation.

6.7 Desktop

If there is an error in the boot sequence, <Boot$Error> will have been set to the text of the message. The Desktop module will be altered to check this variable, report the error if there is one, and offer the dialogue box pictured above. (This the user to see the when no boot-app is found.)

6.8 Kernel

There is no new programmer interface to the kernel.

6.9 Configure

This will merge !System applications to form output !System applications as described below in section 6.10. Modules from a RISC OS 2 !System.Modules will be placed in !System.Modules, from a RISC OS 3.1 !System.Modules in !System.310.Modules, etc. A check will then be performed on the version numbers of uncompressed relocatable modules placed in these directories to ensure that more recent versions of identical modules are not placed in earlier OS version directories - which would result in the more up to date module only being loaded on older machines - an error hard for the user to disentangle.

6.10 System

A new System, which allows multiple operating system version to share resources, will be adopted. This is also based on the Carl Sellers/David Walker prototype, with an extension to provide consistency with ArmBoot applications. It will have a directory structure

```
!System.!Boot
        !Help
        !Run
        !Sprites
        !Sprites22
        !Sprites23
        Modules.(modules as required)
                ...
        310.Modules.(modules as required)
                    ...
        350.Modules.(modules as required)
                    ...
        360.Modules.(modules as required)
                    ...
        SetOSVer
        SetPath
```

The top level (!System.Modules) is for modules that will run under 2.00 (and therefore normally all subsequent versions), !System.310.Modules is for those that require RISC OS 310, etc. (The reason for this is to ensure backwards compatibility with Systems written for 2.00 machines.)

After running !System, the following variables will be set up (where each component on the right hand side is prefixed by the value of Obey$Dir):

```
System$Dir        !System

System$Path       !System.
                     (under RISC OS 2.00, 2.01, 3.00)
                  !System.310.,!System.
                     (under RISC OS 3.10, 3.11)
                  !System.350.,!System.310.,!System.
                     (under RISC OS 3.50)
                  !System.360.,!System.350.,!System.310.,!System.
                     (under Black)
```

In addition, when System is seen by the Filer, these variables will be set up if they are not already set. Measures will be taken to ensure that System$Path does not exceed the 256 character limit - eg. defining an additional $Path variable, and defining System$Path in terms of that.

System$Dir is the same under all versions of RISC OS, since some older applications refer to <System$Dir>.Modules rather then System:Modules.

This allows complete flexibility for providing extension modules that will work under any previous operating systems.

7. Standards

All enhancements to system software will be made by extended existing components in a way consistent with their current state, and in accordance with the Style Guide.

8. Data Interchange

No data interchange protocols are specified here.

9. Data Formats

No data formats are specified here.

10. External Dependencies

All of the application work and most of the module work can proceed using existing hardware platforms.

11. Acceptance Tests

11.2 Boot (standard version)

This should implement all of the user interface documented in the RISC OS 3.5 PRM supplement, and be comparable in size.

11.3 ArmBoot and ShareBoot

These should implement all of the user interface documented in the RISC OS 3.5 PRM supplement for Boot, and in a default configuration be of comparable size to a standard Boot application.

The time taken to boot a network-only machine on a network with 3 others also booting should not be in excess of 2 minutes.

It must be possible to put a standard Territory application in a root directory to load territory-specific files for localisation.

11.4 Boot (cut-down application)

This should implement all of the user interface documented in the RISC OS 3.5 PRM supplement, and should use 600K or less of disc space.

11.5 MchConfig

This should implement all of the features described above.

11.6 BootCommands relocatable module

This should implement all of the features described above. The code sizes of each of the new commands referred to above must not exceed more than twofold the size of the file that currently implements them:

AddApp	908 bytes
AppSize	232 bytes
Do	98 bytes
IfThere	140 bytes
Repeat	4Kbytes

The new commands (SafeLogon, LoadCMOS) must not exceed 2K in size each.

11.7 Desktop

This should implement all of the features described above and not grow in size by more than 5K.

11.8 Kernel

This should implement all of the features described above and not grow in size by more than 1K.

11.9 Configure

This should implement all of the features described above and not grow in size by more than 5K.

12. Development Test Strategy

12.1 Boot

The provision of a directory for configuration settings in a networked environment will be tested by doing 'Save choices' from Printers, in each of the following environments:

- a vanilla network-only machine
- a network-only machine with its own MchConfig directory
- any other version of RISC OS (e g, RISC OS 3.1)

Also, a test will be made that the Scrap directory is available in each of these cases, by dragging a text file from Edit onto the Printers icon.

12.2 MchConfig

MchConfig will be coverage-tested, in order to ensure that each line of code is exercised.

This will involve running on machines with file system configured to Net, Share, and another filing system; and with/without the CMOS protection link and FSLock protection enabled.

It will also be the case that a 'Save choices' operation from the Printer Manager continues to work (saving the configuration to a different place)

after a MchConfig save of configuration files.

12.3 BootCommands

A machine will be started with

- a saved Desktop Boot file containing a Logon command to its own Boot file server, and
- saved CMOS.

This will exercise all the facilities of the BootCommands module.

12.4 Desktop

This will be tested by configuration the file server to an unknown name and rebooting: the error 'File server 'Unknown' not known' must be presented to the user.

12.5 Kernel

The test in 12.4 will also be sufficient to test the new kernel feature.

12.6 Configure

Starting with a local disc, the Wimp Font will be changed to Trinity.Medium, then the configured file system changed to Net, then the font changed back. At each stage, the contents of !Boot.Choices.PreDesk.Configure will be checked. Then Configure will be started again, and the same changes made (this time changing the configured file system back to ADFS): this time, it must be Net::FS.&.!ArmBoot.Choices.PreDesk.Configure that is edited.

A System containing all the optional components described above will be merged into the Boot System, and the results checked. Standard RISC O S 2 and 3 Systems will also be merged.

The standard Fonts directory will be merged with the empty Fonts in a cut-down Boot, and the result checked.