Appendix

A Look at the Startup Report

The INIT-Scope Startup Report

The INIT-Scope Report begins with a preliminary introduction, which aside from the date and time is always the same. You might note that you can use this report to determine what time you last signed on to your computer.

INIT-Scope's Startup Report
INIT-Scope is Copyright 1990
by
David P. Sumner

Date: 8/11/92 Time: 3:43 PM

All addresses are hex. All sizes, quantities, and id's are decimal.

Next comes a description of the basic environment this consists of the name of the computer and its keyboard type.

The Environment Consists of:
Computer: Power Book 170
PowerBook Keyboard

After this follows a description of the computers hardware. In particular, this portion of the report shows the type of processor, floating point coprocessor, and Memory Management Unit in the computer, the version of QuickDraw installed, and whether the computer is in 32-bit mode and has virtual memory on.

If virtual memory is on, then much of the trap history report will be missing.

Hardware Configuration Is:

Processor: 68030
MC 68882 Floating Point Coprocessor
68030 MMU
32-bit (vers 1.3) Color QuickDraw

Has VIA1
Has VIA2
Has ASC
Has SCC
Has SCSI Original (based on the 53C80)

Has 32-bit capability, but is not in 32-bit mode.

Next comes the values of some important values stored in low memory globals. The last of these is the system version in use. This appears as a hexadecimal value, so the $701 in the report below indicates that the system in use is 7.1.

```
          Low Memory Values:
          --------------------------
BufPtr:  $726EB0
APPL Zone   $B1000
Heap End:    $FDFF4
AppLimit  $3F9196
System Zone    $2000
Free in System Heap (bytes): 124756
Largest Free Block in System: 120864
Top of Memory:   $801770  (8394608 Total Bytes of Ram).
Screen Base:   $E08000
Sound Buffer:   $F14000
System Version:     $701
```

After this is a listing of all the VBL (Clock interrupt) routines that are currently active at the time INIT-Scope is activated.
These routines are called every 1/60th of a second (i.e. every tick).

```
Initially Active VBL Procedures:
----------------------------------------
            $86D7FA
            $8E5E78
            $8E68BA
             $7B8A
             $3E25E
             $3E398
           $80074058
             $77198
           $8005ECCC
----------------------------------------
```

Next you will see the address of any global variable that is being monitored. You can set an address for monitoring by editing the 'GloB' resource in ResEdit (there is a template for this resource in INIT-Scope, so it is easy to edit). In this example no global value is being monitored. The global must represent a long integer (4 bytes). If the value stored in the 'GloB' resource is $FFFFFFFF, then no global variable will be monitored.

```
Global being monitored: <None>
```

If a global variable is being monitored, then its value will be checked after every trap call (from anywhere), and whenever its value changes, the change will be noted in the report.

For example, a portion of a report that is monitoring $304 might look like this:

```
 $A090  SysEnvirons
  $A009  Delete
  •••Global Variable at     $304 Changed to:    $1E7F7
  •••Global Variable at     $304 Changed to:   $0
  •••Global Variable at     $304 Changed to:    $1E7F8
  •••Global Variable at     $304 Changed to:   $0
```

$A008 Create
$A00C GetFileInfo

(Only calls from the INIT are shown in this report.)

In this case, the global variable at $304 changed at least four times between the INIT's call to Delete and Create. It may have changed other times as well; the value at $304 was only checked after very trap call. (So there were several calls made from outside the INIT).

It is important to understand the difference between the global variable referred to here, and the global variables reported as changed by each INIT. The latter variables are checked prior to the execution of an INIT and again at the end of the INIT. INIT-Scope reports those locations whose values changed in a special section prior to the trap history portion of the report. The global referred to here is monitored after *every* trap call (ROM Included) and any changes in its value are reported in the trap history portion of the report. Consequently, you will have a very good idea of when these values changed since you will know which trap calls took place before and after the change.

Next comes the Startup Log. This portion of the report is a detailed accounting of the activities of each INIT as it loads into RAM and executes.

Startup Log Follows:

The report on each item begins with essentially the same information. First is an indication of whether the file being accessed is an INIT (system extension), cdev (control panel), or RDEV (Chooser item).

After this comes the actual size of the code that is loading (some files may contain several INITs).

The next thing mentioned is the use of High Ram. Every INIT will load its code and resources into one or both of two places—the system heap or a protected portion of memory termed High Ram; this is the portion of the Mac's memory that lies above the address in the low memory variable *BufPtr*. So INIT-Scope's Report first gives the value of BufPtr at the end of the INIT's execution. If this value is different from what it was at the beginning, then the INIT used some of High Ram for storage. The report indicates how many bytes of High Ram were used.

Next the report shows how many bytes of the system heap are used by the INIT. In fact, this portion of the report shows how much system memory the INIT indicated that it might need, how much the system heap actually expanded, and how much of the system heap the INIT used. The INIT file contains a 'sysz' resource whose first long word is the amount of memory that the file requests from the system for all of its INITs.

The report continues with the value of the largest free block of memory contained in the system heap. The difference between this value and the total free space in the system heap is a rudimentary measure of the heap's fragmentation.

This portion of the report also shows the Application Heap at the time this INIT loaded. This is the portion of memory into which the INIT code is initially loaded.

Finally, the report gives the Handle to the INIT. This is likely of interest only if you also requested a trap history report. In that case you will be able to tell when the INIT's code accesses the handle to the INIT.

Our example output for the INIT Disk Doubler™ looks like this:

```
        >DiskDoubler™ INIT  ( INIT) <
        ---------------------------------
Size of this INIT in bytes: 11574
BufPtr:  $726EB0
High Ram Used (bytes): 0
Requested System Heap space (bytes): 196608
System Heap Expansion (bytes): 65536
System Heap Used (bytes): 184262
Application Heap:    $E1944 -   $E3144
Free in System Heap (bytes): 32044
Largest Free Block in System: 27400  Diff= 4644
INIT Handle:    $5D3DC
```

From this we can see that Disk Doubler stored all its code into the system heap
and did not use any of the Mac's High Memory. It uses over 184,000 bytes of memory.

Now it is the intent of most INITs to alter the operation of some part of the Macintosh in some particular (hopefully desirable) way. They do this by installing hooks into the calls to the Mac's internal ROM routines, or by installing other accessible code resources. The next part of the INIT-Scope Startup Report lists all such installations in chronological order.

```
          Patches / Installations
          ---------------------------
  $A937 Patch:    $96870   DrawMenuBar
  $A9EA           Patch:   $96880   Pack3 (StdFile)
  $A9FE           Patch:   $968A0  PutScrap
  $A000 Patch:    $96850   Open
```

So in the case of Disk Doubler, we see that it initially patches four traps. Clearly this INIT intends to alter the way the Standard File mechanism works (that's what the Pack3 trap is all about), it needs to keep its menus on the menubar whenever the menubar is redrawn (so it patches DrawMenuBar), and since it also needs to know when a file is opened up, it patches the Open trap.

If the Show Resource Info button is highlighted in the INIT-Scope cdev, then you will also see a complete listing of all resources used during the INIT’s execution.

Some of these resources may be really used by ROM calls made by the INIT and so are only used by the INIT indirectly. Probably, you will choose to keep this option turned off most of the time.

After the Patches/Installations portion of each INIT's report, there is a listing of all the low-memory global variables altered by the INIT. This portion of the report uses the addresses stored in INIT-Scope's 'HOOK' resource. INIT-Scope checks the value of each of these global variables prior to the execution of the INIT and again after it terminates. INIT-Scope reports any changes to these variables. The addresses in the 'HOOK' resource have been carefully selected to represent those variables that would reasonably be changed during the execution of an INIT. In particular variables that would most likely be changed by the ROM are not reported by INIT-Scope. For example, the location of the mouse may change during the execution of an INIT, but that change is not likely related to any action of the INIT, and so it is not reported.

Similarly, the value stored at RMgrHiVars is the *type* of the last resource accessed by the Resource Manager, and consequently it generally changes numerous times during the execution of each INIT. To report this as having changed by the INIT would be silly, and so INIT-Scope does not report such a change. Only values that are meaningful are watched by INIT-Scope.

However, if you should want to monitor the changes to other variables, you need only add their names and addresses to the 'HOOK' resource.

Here is a sampling of the initial portions of the report for several INITs.

```
          ----------------------------------
          >AutoMenus II  ( cdev) <
          ----------------------------------
Size of this INIT in bytes: 3692
BufPtr:  $6D32C6
High Ram Used (bytes): 0
Requested System Heap space (bytes): 32768
System Heap Expansion (bytes): 25816
System Heap Used (bytes): 4100
Application Heap:    $524E0 -   $540E0
Free in System Heap (bytes): 31692
Largest Free Block in System: 30820  Diff= 872
INIT Handle:    $1D8BC

            Patches / Installations
          ---------------------------
Notification request Installed. Record is at:     $DB8C
--- String: You are using a demonstration copy of AutoMenus II.  See the about box in the Control Panel to learn how you can become a registered user.
  $A973 Patch:    $1E2AA StillDown
  $A93D           Patch:    $1E39C  MenuSelect
VBLProc installed at:    $1E6C0  Count:       $1  Phase:   $0
-------------------------------------------------------------------------------------------------------------
```

>SndControl (cdev) <

Size of this INIT in bytes: 14758
BufPtr: $6D3070
High Ram Used (bytes): 0
Requested System Heap space (bytes): 40960
System Heap Expansion (bytes): 0
System Heap Used (bytes): 29586
Application Heap: $B043C - $C603C
Free in System Heap (bytes): 95896
Largest Free Block in System: 93196 Diff= 2700
INIT Handle: $1D7CC

Patches / Installations

$A00F Patch: $4E3FE MountVol
$A9C8 Patch: $4E31E SysBeep
$A017 Patch: $4E372 Eject
$A00E Patch: $4E3B8 UnMountVol
$A9F4 Patch: $4E4DC ExitToShell
$A9F2 Patch: $4E4EC Launch
$A9A0 Patch: $4E55C GetResource
$A92D Patch: $4E51E CloseWindow
$A9C9 Patch: $4E4FC SysError
$A97B Patch: $4D14A InitDialogs
Shut Down Routine at: $4E6D4 Called before: Power Off
Shut Down Routine at: $4E6EC Called before: Restart
Shut Down Routine at: $4E704 Called before: Closing Drivers

Low Memory Globals Altered:

($29A) JGNEFilter Changed from: $36D0E To: $4CFC2

>PopChar (cdev) <

Size of this INIT in bytes: 632
BufPtr: $6D3070
High Ram Used (bytes): 0
Requested System Heap space (bytes): 13518
System Heap Expansion (bytes): 0
System Heap Used (bytes): 13892
Application Heap: $8759C - $8999C
Free in System Heap (bytes): 34908
Largest Free Block in System: 32524 Diff= 2384
INIT Handle: $876B0

Patches / Installations

$A937 Patch: $36E3C DrawMenuBar
$A912 Patch: $36E76 InitWindows

Low Memory Globals Altered:

($29A) JGNEFilter Changed from: $2007F096 To: $36D0E
($9CE) ToolScratch Changed from: $FFFFFFFF To: $80033A40

```
---------------------------------------------------------------------------------------------------------------
          ---------------------------------
          >QuicKeys 2™  ( cdev) <
          ---------------------------------
Size of this INIT in bytes: 214
BufPtr:  $6D3070
High Ram Used (bytes): 0
Requested System Heap space (bytes): 61440
System Heap Expansion (bytes): 29028
System Heap Used (bytes): 57772
Application Heap:    $8E700 -   $99F00
Free in System Heap (bytes): 6164
Largest Free Block in System: 6152  Diff= 12
INIT Handle:    $8E81C

            Patches / Installations
          --------------------------
Driver Installed. RefNum is -63
  $A97B          Patch:    $37AC0 InitDialogs
  $A001 Patch:   $37DC8 Close
  $A031 Patch:   $3BF70          GetOSEvent
  $A030 Patch:   $3C10A OSEventAvail
  $A976 Patch:   $3BF0E GetKeys
  $A974 Patch:   $3B8F6          Button
  $A972 Patch:   $3B6F0          GetMouse
  $A93D          Patch:    $3B0DA MenuSelect
  $A937 Patch:   $3B466          DrawMenuBar
  $A850 Patch:   $3B958          InitCursor
  $A851 Patch:   $3B9BE SetCursor
  $A91EPatch:    $3AE60 TrackGoAway
  $A83B          Patch:    $3ADCA          TrackBox
  $A9B5          Patch:    $3AEF0 SystemMenu
  $A92C          Patch:    $3B70E FindWindow
  $A968 Patch:   $3B7E2 TrackControl
  $A971 Patch:   $3B03A EventAvail
  $A970 Patch:   $3B08C GetNextEvent
  $A9B7          Patch:    $37DF8 CloseDeskAcc
  $A02F Patch:   $37EF4 PostEvent
Shut Down Routine at:    $37CE8   Called before: Closing Drivers
---------------------------------------------------------------------------------------------------------------
```

Trap History

Finally, if the ‘Trap History’ option was selected in the cdev, then he report shows all the traps executed by each INIT. In fact, you may choose to see all the traps executed during the time INIT executes; this includes trap calls from the ROM. It is recommended that you generally not choose to do this since it can produce exceptionally long reports.

Not only is each trap call shown in the report, but also the parameters passed to the call. Moreover, in the case of resource traps, the handle returned by the call is also shown (unless the value is the same as the value returned by the last resource call). One word about how the value of the handle is obtained is in order. INIT-Scope makes use of the (apparent) fact

that each call to GetResource

and other resource manager calls store the value of the returned handle at the address of RMgrHiVars+4. If the value stored at this address changes, INIT-Scope assumes that the value stored at this address is the value of the handle returned by a previous call to the resource manager.

The parameters passed to many other traps are shown as well. For example you will see such information as:

$A260 HFSDispatch GetWDInfo

This makes it much easier to determine what this particular call to HFSDispatch is all about.

Note that INIT-Scope gives the exact sizes of parameters in bytes. This makes it possible to determine the exact physical extent of various code segments in memory. for example, when you see,

BlockMove #Bytes: 80 From: $4BA3E To: $3D803A

you can surmise that the data that begins at $3D803A extends for 80 bytes. If you also see that is the address of a patch to GetNextEvent, then you know exactly what portion of RAM memory holds this patch code.

An examples of a typical trap history reports follow.

For Sound Control

```
        Trap History
        ----------------
$A128 RecoverHandle $A00480C8
$A9A0 GetResource noLD #1
$A746 GetTrapAddress StdLine
$A746 GetTrapAddress Unimplemented
$A346 GetTrapAddress SysEnvirons
$A746 GetTrapAddress Unimplemented
$A090 SysEnvirons
$A746 GetTrapAddress Unimplemented
$A346 GetTrapAddress Gestalt
$A1AD Gestalt $666F6C64 fold
$A02E BlockMove #Bytes: 80 From: $4BA3E To: $3D803A
$A746 GetTrapAddress Unimplemented
$A346 GetTrapAddress Gestalt
$A1AD Gestalt $666F6C64 fold
$A346 GetTrapAddress SysEnvirons
$A746 GetTrapAddress Unimplemented
$A090 SysEnvirons
$A260 HFSDispatch GetWDInfo
$A146 GetTrapAddress Unimplemented
$A746 GetTrapAddress HOpenResFile
$A20A OpenRF
$A001 Close
$A214 GetVol
$A260 HFSDispatch GetWDInfo
$A215 SetVol
$A9C4 OpenRFPerm
$A9AF ResError
```

```
$A015  SetVol
$A99A  CloseResFile
$A346  GetTrapAddress  MountVol
$A346  GetTrapAddress  Eject
$A346  GetTrapAddress  UnMountVol
$A746  GetTrapAddress  SysBeep
$A746  GetTrapAddress  ExitToShell
$A746  GetTrapAddress  Launch
$A746  GetTrapAddress  SysError
$A746  GetTrapAddress  InitDialogs
$A746  GetTrapAddress  CopyMask
$A746  GetTrapAddress  CopyBits
$A746  GetTrapAddress  GetResource
$A746  GetTrapAddress  CloseWindow
$A055  StripAddress  $A004AA4E
$A055  StripAddress  $A004A96E
$A055  StripAddress  $A004A9C2
$A055  StripAddress  $A004AA08
$A055  StripAddress  $A004AB2C
$A055  StripAddress  $A004AB3C
$A055  StripAddress  $A004AC34
$A055  StripAddress  $A004AC34
$A055  StripAddress  $A004ABAC
$A055  StripAddress  $A004AB6E
$A055  StripAddress  $A004AB4C
$A055  StripAddress  $A004AD24
$A055  StripAddress  $A004AD3C
$A055  StripAddress  $A004AD54
$A055  StripAddress  $A004979A
$A055  StripAddress  $A0049612
$A055  StripAddress  $A004B774
$A025  GetHandleSize     $1D7CC
$A11A  GetZone
$A01B  SetZone      $2000
$A11E  NewPtr   Size (bytes): 14758
$A02E  BlockMove  #Bytes: 14758  From:    $480C8   To:    $4BA78
$A055  StripAddress  $6004CFC2
$A346  GetTrapAddress  PMgrOp
$A746  GetTrapAddress  Unimplemented
$A247  SetTrapAddress  MountVol
$A647  SetTrapAddress  SysBeep
$A247  SetTrapAddress  Eject
$A247  SetTrapAddress  UnMountVol
$A647  SetTrapAddress  ExitToShell
$A647  SetTrapAddress  Launch
$A647  SetTrapAddress  GetResource
$A647  SetTrapAddress  CloseWindow
$A647  SetTrapAddress  SysError
$A647  SetTrapAddress  InitDialogs
$A055  StripAddress  $6004E6D4
$A895  ShutDown
$A055  StripAddress  $6004E6EC
$A895  ShutDown
$A055  StripAddress  $6004E704
$A895  ShutDown
$A01B  SetZone     $B043C
$A9A0  GetResource   InIc  #1   ResHandle:    $B04B8
```

```
$AA1E  GetCIcon   ResHandle:    $B0498
```

$A86E InitGraf
$A86F OpenPort
$AA1F PlotCIcon
$A87D ClosePort
$AA25 DisposCIcon
$A014 GetVol
$A015 SetVol
$A9BA GetString ResHandle: $B049C
$A746 GetTrapAddress Unimplemented
$A346 GetTrapAddress Gestalt
$A1AD Gestalt $666F6C64 fold
$A02E BlockMove #Bytes: 80 From: $4BA3E To: $3D7EE2
$A746 GetTrapAddress Unimplemented
$A346 GetTrapAddress Gestalt
$A1AD Gestalt $666F6C64 fold
$A346 GetTrapAddress SysEnvirons
$A746 GetTrapAddress Unimplemented
$A090 SysEnvirons
$A260 HFSDispatch GetWDInfo
$A146 GetTrapAddress Unimplemented
$A746 GetTrapAddress HOpenResFile
$A20A OpenRF
$A001 Close
$A214 GetVol
$A260 HFSDispatch GetWDInfo
$A215 SetVol
$A9C4 OpenRFPerm
$A9AF ResError
$A015 SetVol
$A81F Get1Resource sdId #0 ResHandle: $B04B0
$A81F Get1Resource STR #128 ResHandle: $B053C
$A9AD RmveResource $B053C
$A992 DetachResource $B049C
$A9AB AddResource ÅÚÄ #61
$A99A CloseResFile
•••Global Variable at $304 Changed to: $1E961
$A81F Get1Resource kWhE #1
•••Global Variable at $304 Changed to: $0
ResHandle: $B0494
$A015 SetVol
$A994 CurResFile
$A746 GetTrapAddress Unimplemented
$A346 GetTrapAddress Gestalt
$A1AD Gestalt $666F6C64 fold
$A02E BlockMove #Bytes: 80 From: $4BA3E To: $3D8026
$A746 GetTrapAddress Unimplemented
$A346 GetTrapAddress Gestalt
$A1AD Gestalt $666F6C64 fold
$A346 GetTrapAddress SysEnvirons
$A746 GetTrapAddress Unimplemented
$A090 SysEnvirons
$A260 HFSDispatch GetWDInfo
$A146 GetTrapAddress Unimplemented
$A746 GetTrapAddress HOpenResFile
$A20A OpenRF
$A001 Close
$A214 GetVol

$A260 HFSDispatch GetWDInfo

```
$A215  SetVol
$A9C4  OpenRFPerm
$A9AF  ResError
$A015  SetVol
$A81F  Get1Resource   volu  #7
```