Libraries

COLLABORATORS

	TITLE : Libraries	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 14, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Libraries

1.1 Amiga® RKM Libraries: E Release 2 Compatibility

If you are developing new software or updating older software, you need to avoid compatibility traps. This comprehensive list of Release 2 compatibility problem areas can help you avoid and diagnose compatibility problems In addition, refer to the "General Amiga Development Guidelines" listed in Chapter 1.

1.2 E Release 2 Compatibility / General Compatibility Problem Areas

The following improper Amiga programming practices are likely to fail on new ROMs or hardware.

* Requiring all free RAM.

* Overwriting memory allocations. With 32-bit addresses, a 1-byte overwrite of a string array can wipe out the high byte of a pointer or stack return address. This bug could go unnoticed on a 24-bit address machine (e.g., A500) but crash the system or cause other problems on an A3000.

* Improper flags or garbage in system structures. A bit that means nothing under one OS may drastically change the behavior of a function in a newer version of the OS. Clear structures before using, and use correct flags.

* Misuse of function return values. Use function prototypes and read the Autodocs for the functions you are using. Some system functions

return just success or failure, or nothing at all (void). In such cases, the value which the function happens to return must not be used except as it is documented.

* Depending on unsupported side effects or undocumented behavior. Be sure to read the Autodocs, include file comments and other documentation.

* Assuming current choices, configurations or initial values. If the current possibilities are A, B, or C, do not assume C if it isn't A or B. Check specifically for the choices currently implemented, and provide default behavior for unexpected values.

Amiga debugging tools such as Enforcer, Mungwall and Scratch can find many program bugs that may affect compatibility. A program that is Enforcer/Mungwall/Scratch clean stands a much better chance of working well under current and future versions of the OS.

1.3 E Compatibility / Release 2 Changes That Can Affect Compatibility

There are several areas where Relase 2 OS changes and enhancements can cause compatibility problems for some software.

Exec

Serial Device

Preferences

Expansion

Timer Device

Workbench

Strap

Trackdisk Device

Layers

DOS

CIA Timers

Graphics

Audio Device

Other Hardware Issues

Fonts

Gameport Device

Intuition

CLI/Shell

1.4 E / Release 2 Changes That Can Affect Compatibility / Exec

* Do not jump to location $FC0002 -- the start of the ROM under 1.3 -- as part of performing a system RESET. The 2.04 Kickstart ROM has a temporary compatibility hack called "Kickety-Split" which is a redirecting jump at $FC0002. This hack does not appear on the A3000 ROM and due to space considerations will not appear on future machines.

* Everything has moved.

* The Supervisor stack is not in the same place as it was under 1.3. This has caused problems for some games that completely take over the Amiga. If your program goes into Supervisor mode, you must either respect allocated memory or provide your own Supervisor stack when taking over the machine.

* ExecBase is moved to expansion memory if possible. Before, ExecBase would only end up in one of two fixed locations. Now, ColdCapture may be called after expansion memory has been configured.

* Exception/Interrupt vectors may move. This means the 68010 and above Vector Base Register (VBR) may contain a non-zero value. Poking assumed low memory vector addresses may have no effect. You must read the VBR on 68010 and above to find the base.

* No longer tolerant of wild Forbid() counts. Under 1.3, sometimes this bug could go unnoticed. Make sure that all Forbid()s are matched with one and only one Permit() (and vice versa).

* When an Exec device gets an IORequest, it must validate io_Command. If the io_Command is 0 or out of range, the device must return IOERR_NOCMD and take no other action. The filesystem now sends new commands and expects older devices to properly ignore them.

* A fix to task-switching in Release 2 allows a busy task to properly regain the processor after an interrupt until either its quantum (4 vertical blanks) is up or a higher priority task preempts it. This can dramatically change the behavior of multitask programs where one task busyloops while another same-priority task Wait()s. See "

Task Switching

" in the "Additional Information" section below.

1.5 E / Release 2 Changes That Can Affect Compatibility / Expansion

* ExpansionBase is private - use FindConfigDev().

* Memory from contiguous cards of the same memory type is automatically merged into one memory pool.

1.6 E / Release 2 Changes That Can Affect Compatibility / Strap

* Romboot.library is gone.

* Audio.device cannot be OpenDevice()ed by a boot block program. See "
Audio Device
" below.

* Boot from other floppies (+5,-10,-20,-30) is possible.

* Undocumented system stack and register usage at Diag and Boot time have changed.

1.7 E / Release 2 Changes That Can Affect Compatibility / DOS

* DOS is now written in C and assembler, not BCPL. The BCPL compiler artifact which caused D0 function results to also be in D1 is gone. System patches in Release 2 that return some DOS function results in both D0 and D1 are not guaranteed to remain in the next release. Fix your programs! Use Scratch to find these problems in your code.

* DOS now has a real library base with normal LVO vectors.

* Stack usage has all changed (variables, direction).

* New packet and lock types. Make sure you are not passing stack garbage for the second argument to Lock().

* Process structure is bigger. "Rolling your own" Process structure from a Task fails. Use dos.library System() or CreateNewProc().

* Unless documented otherwise, you must be a process to call DOS functions. DOS function dependence on special process structures can change with OS revisions.

1.8 E / Release 2 Changes That Can Affect Compatibility / Audio Device

* Now not initialized until used. This means low memory open failure is possible. Check your return values from OpenDevice(). This also means audio.device cannot be opened during 2.0 Strap unless

InitResident()ed first. If OpenDevice() of audio.device fails during strap, you must FindResident()/InitResident() audio.device, and then try OpenDevice() again. There will be a small memory loss (until reboot) generated by the first opener of audio.device or narrator.device (memory used in building of audio.device's base).

1.9 E / Release 2 Changes That Can Affect Compatibility / Gameport Device

* Initial state of hardware lines may differ.

1.10 E / Release 2 Changes That Can Affect Compatibility / Serial Device

* Clears io_Device on CloseDevice() (since 1.3.2)

1.11 E / Release 2 Changes That Can Affect Compatibility / Timer Device

* The most common mistake programmers make with timer.device is to send off a particular timerequest before the previous use of that timerequest has completed. Use IO_Torture to catch this.

* IO_QUICK requests may be deferred and be replied as documented.

* VBLANK timer requests, as documented, now wait at least as long as the full number of VBlanks you asked for. Previously, a partial vertical blank could count towards your requested number. The new behavior is more correct and matches the docs, but it can cause VBlank requests to now take up to 1 VBlank longer under 2.0 as compared to 1.3. For example, a 1/10 second request, may take 6-7 Vblanks instead of 5-6 VBlanks, or about 15% longer.

1.12 E / Release 2 Changes That Can Affect Compatibility / Trackdisk Device

* Private trackdisk structures have changed. See trackdisk.doc for a compatible REMCHANGEINT.

* Buffer is freeable, so low memory open failure is possible.

* Do not disable interrupts (any of them), then expect trackdisk to function while they are disabled.

1.13 E / Release 2 Changes That Can Affect Compatibility / CIA Timers

* System use of CIA timers has changed. Don't assume how they're used.

* Don't depend on initial values of CIA registers.

* Don't mess with CIABase. Use cia.resource.

* If your code requires hardware level CIA timers, allocate the timers
  using cia.resource AddICRVector()! This is very important. Operating
  system usage of the CIA timers has changed. The new 2.0 timer.device
  ("Jumpy the Magic Timer Device") will try to jump to different CIAs
  so programs that properly allocate timers will have a better chance
  of getting what they want. If possible, be flexible and design your
  code to work with whatever timer you can successfully allocate.

* OS usage of INT6 is increasing. Do not totally take over INT6, and
  do not terminate the server chain if an interrupt is not for you.

1.14 E /Release 2 Changes That Affect Compatibility / Other Hardware Issues

* Battery-backed clock is different on A3000. Use battclock.resource
  to access the real-time clock if battclock.resource can be opened.

* A 68030 hardware characteristic causes longword-aligned longword
  writes to allocate a valid entry in the data cache, even if the
  hardware area shouldn't be cached. This can cause problems for I/O
  registers and shared memory devices. To solve this: 1) don't do that
  2) flush the cache or 3) use Enforcer Quiet. See the Motorola 68030
  manual under the description of the Write Allocate bit (which must be
  set for the Amiga to run with the Data Cache).

1.15 E / Release 2 Changes That Can Affect Compatibility / Intuition

* Private IntuitionBase variables have moved/changed. Reading ↩
them is
illegal. Writing them is both illegal and dangerous.

* Poking IntuitionBase MaxMouse variables is now a no-op, but stop
  poking when Intuition version is >35.

* If you are opening on the Workbench screen, be prepared to handle
  larger screens, new modes, new fonts, and overscan. Also see
  "
Fonts
" compatibility information.

* Screen TopEdge and LeftEdge may be negative.

* Left-Amiga-Select is used for dragging large screens. Do not use
  left-Amiga-key combinations for application command keys. The
  left-Amiga key is reserved for system use.

* For compatibility reasons, GetScreenData() lies if the Workbench screen is a mode only available after 1.3. It will try to return the most sensible mode that old OpenScreen() can open. This was necessary to prevent problems in applications that cloned the Workbench screen. To properly handle new modes, see LockPubScreen(), GetVPModeID(), and the SA_DisplayID tag for OpenScreenTags().

* Using combined RAWKEY and VANILLAKEY now gives VANILLAKEY messages for regular keys, and RAWKEY messages for special keys (fkeys, help, etc.)

* Moving a SIMPLE_REFRESH window does not necessarily cause a REFRESHWINDOW event because layers now preserves all the bits it can.

* Sizing a SIMPLE_REFRESH window will not clear it.

* MENUVERIFY/REQVERIFY/SIZEVERIFY can time out if you take too long to ReplyMsg().

* Menu-key equivalents are ignored while string gadgets are active.

* You can't type control characters into string gadgets by default. Use Ctrl-Amiga-char to type them in or use IControl Prefs to change the default behavior.

* Width and Height parameters of AutoRequest() are ignored.

* New default colors, new gadget images.

* JAM1 rendering/text in border may be invisible gadgets over default colors.

* The cursor for string gadgets can no longer reside outside the cleared container area. If your gadget is 32 pixels wide, with MaxChars of 4, all 32 pixels will be cleared, instead of just 24, as was true in 1.3.

* Applications and requesters that fail to specify desired fonts will get the fonts the user sets up in Font Preferences in Release 2. These could be much larger, or proportional in some cases. Screen and window titlebars (and their gadgets) will be taller when accommodating a larger font. Applications which open on the Workbench screen must adapt to variable size titlebars. Any application which accepts system defaults for its screen, window, menu, Text or IntuiText fonts must adapt to different fonts and titlebar sizes. String gadgets whose height is too small for a font will revert to a smaller ROM font. There are now 2 different user-specifiable default system fonts which affect different Intuition features. This can lead to mismatches in mixed gadgets and text. See the "Intuition Screens" chapter.

* Don't modify gadgets directly without first removing them from the gadget list, unless you are using a system function designed for that purpose, such as NewModifyProp() or SetGadgetAttrs().

* Don't rely on NewModifyProp() to fully refresh your prop gadget after you've changed values in the structure. NewModifyProp() will only

correctly refresh changes which were passed to it as parameters. Use Remove/Add/RefreshGList() for other kinds of changes.

* Custom screens must be of type CUSTOMSCREEN or PUBLICSCREEN. Other types are illegal. One application opens its screen with NewScreen.Type = 0 (instead of CUSTOMSCREEN, 0x0F). Then, when it opens its windows, it specifies NewWindow.Type of 0 and NewWindow.Screen of NULL, instead of Type = CUSTOMSCREEN and Screen = (their screen). That happened to work before, but no longer.

* Referencing IntuiMessage->IAddress as a Gadget pointer on non-Gadget IDCMP messages, or as a Window pointer (rather than looking at the proper field IntuiMessage->IDCMPWindow) may now cause Enforcer hits or crashes. The IAddress field always used to contain a pointer of some type even for IDCMP events for which no IAddress value is documented. Now, for some IDCMP events, IAddress may contain a non-address, possibly an odd value that would crash a 68000 based system).

* Using Intuition flags in the wrong structure fields (for example, using ACTIVEWINDOW instead of ACTIVATE). To alleviate this problem, 2.0 has introduced new synonyms that are less confusing than the old ones. For example, IDCMP_ACTIVEWINDOW and WFLG_ACTIVATE. This particular example of confusion (there are several) was the nastiest, since IDCMP_ACTIVEWINDOW, when stuffed into NewWindow.Flags, corresponds numerically to WFLG_NW_EXTENDED, which informs Intuition that the NewWindow structure is immediately followed by a TagItem, list which isn't there! Intuition does some validation on the tag-list pointer, in order to partially compensate. To make your compiler use the new synonyms only, add this line to your code before Intuition include files: #define INTUI_V36_NAMES_ONLY.

* Do not place spaces into the StringInfo->Buffer of a GACT_LONGINT string gadget. Under 1.3, this worked, but the 2.0 validation routine that checks for illegal keystrokes looks at the contents for illegal (i.e., non-numeric) characters, and if any are found assumes that the user typed an illegal keystroke. The user's only options may be shift-delete or Amiga-X. Use the correct justification instead.

* If you specify NULL for a font in an IntuiText, don't assume you'll get Topaz 8. Either explicitly supply the font you you need or be prepared to size accordingly. Otherwise, your rendering will be wrong, and the user will have to reset his Preferences just to make your software work right.

* Window borders are now drawn in the screen's DetailPen and BlockPen rather than the Window's pens. For best appearance, you should pass an SA_Pens array to OpenScreen(). This can be done in a backwards compatible manner with the ExtNewScreen structure and the NS_EXTENDED flag.

* The system now renders into the full width of window borders, although the widths themselves are unchanged. Window borders are filled upon activation and inactivation.

* Window border rendering has changed significantly for 2.0. Note that the border dimensions are unchanged from 1.x (Look at

Window->BorderLeft/Top/Right/Bottom if you don't believe us!). If your gadget intersects the border area, although it may have looked OK under 1.3, a visual conflict may occur under 2.0. If Intuition notices a gadget which is substantially in the border but not declared as such, it treats it as though it were (this is called "bordersniffing"). Never rely on Intuition to sniff these out for you; always declare them explicitly (see the Gadget Activation flags GACT_RIGHTBORDER, etc.). See "

Intuition Gadgets and Window Borders

"

in the "Additional Information" section below.

1.16 E / Release 2 Changes That Can Affect Compatibility / Preferences

* Some old struct Preferences fields are now ignored by SetPrefs() (for example FontHeight). SetPrefs() also stops listening to the pointer fields as soon as a new-style pointer is passed to Intuition (new-style pointers can be taller or deeper).

* Preferences ViewX/YOffset only applies to the default mode. You cannot use these fields to move the position of all modes.

* The Preferences LaceWB bit is not necessarily correct when Workbench is in a new display mode (akin to GetScreenData()).

1.17 E / Release 2 Changes That Can Affect Compatibility / Workbench

* The Workbench GUI now has new screen sizes, screen top/left offsets, depths, modes, and fonts.

* Default Tool now searches paths.

* New Look (boxed) icons take more space.

* Do not use icons which have more 1bits set in PlanePick than planes in the ImageData (one IFF-to-Icon utility does this). Such icons will appear trashed on deeper Workbenches.

* New Look colors have black and white swapped (as compared to 1.3).

* The Workbench screen may not be open at startup-sequence time until some output occurs to the initial Shell window. This can break startup-sequence-started games that think they can steal WB's screen bitplanes. Do not steal the Workbench screen's bitplanes. (For compatibility, booting off pre-2.0 disks forces the initial screen open. This is not guaranteed to remain in the system.) Use startup code that can detach when RUN (such as cback.o) and use CloseWorkbench() to regain the screen's memory. In addition, see "

Workbench and Startup

" in the "Additional Information" section
below.

1.18 E / Release 2 Changes That Can Affect Compatibility / Layers

* Use NewLayerInfo() to create, not FattenLayerInfo(), ThinLayerInfo(), InitLayers().

* Simple-refresh preserves all of the pixels it can. Sizing a SIMPLE_REFRESH window no longer clears the whole window.

* Speed of layer operations is different. Don't depend on layer operations to finish before or after other asynchronous actions.

1.19 E / Release 2 Changes That Can Affect Compatibility / Graphics

* Do not rely on the order of Copper list instructions. The Release 2 MrgCop() function builds different Copper lists to that of 1.3, by including new registers in the list (e.g., MOVE xxxx,DIWHIGH). This changes the positions of the other instructions. We know of one game that 'assumes' the BPLxPTRs would be at a certain offset in the Copper list, and that is now broken on machines running 2.0 with the new Denise chip.

* Graphics and layers functions which use the blitter generally return after starting the final blit. If you are mixing graphics rendering calls and processor access of the same memory, you must WaitBlit() before touching (or deallocating) the source or destination memory with the processor. For example, the Text() function is faster in Release 2, causing some programs to trash partial lines of text.

* ColorMap structure is bigger. Programs must use GetColorMap() to create one.

* Blitter rtns decide ascend/descend on 1st plane only.

* Changing the display mode of an existing screen or viewport while open is still not a supported operation.

* GfxBase DisplayFlags and row/cols may not match Workbench screen.

* Do not hardcode modulo values - use BitMap->BytesPerRow.

* If the graphics Autodocs say that you need a TmpRas of a certain size for some functions, then you must make that the minimum size. In some cases, before 2.0, you may have gotten away with using a smaller TmpRas with some functions (for example Flood()). To be more robust, graphics now checks the TmpRas size and will fail the function call if the TmpRas is too small.

* ECS chips under 2.0 generate displays differently. The display window

registers now control DMA.

* LoadRGB4() used to poke colors into the active copperlist with no protection against deallocation of that copperlist while it was being poked. Under 2.0, semaphore protection of the copperlist was added to LoadRGB4() which makes it totally incorrect and extremely dangerous to call LoadRGB4() during an interrupt. The general symptom of this problem is that a system deadlock can be caused by dragging one screen up and down while another is cycling. Color cycling should be performed from within a task, not an interrupt. In general, the only functions which may be safely called from within an interrupt are the small list of Exec functions documented in the "Exec Interrupts" chapter.

1.20 E / Release 2 Changes That Can Affect Compatibility / Fonts

* Some font format changes (old format supported).

* Private format of .font files has changed (use FixFonts to create).

* Default fonts may be larger, proportional.

* Topaz is now sans-serif.

* Any size font will be created via scaling as long as TextAttr.Flags FPF_DESIGNED bit is not set. If you were asking for an extreme size, like size 1 to get smallest available, or 999 to get largest available, you will get a big (or very very small) surprise now.

* Do not use -1 for TextAttr.Flags or Styles, nor as the flags for AvailFonts (one high bit now causes AvailFonts to return different structures). Only set what you know you want. A kludge has been added to the OS to protect applications which currently pass -1 for AvailFonts flags.

1.21 E / Release 2 Changes That Can Affect Compatibility / CLI/Shell

* Many more commands are now built-in (no longer in C:). This can break installation scripts that copy C:commandname, and programs that try to Lock() or Open() C:commandname to check for the command's existence.

* The limit of 20 CLI processes is gone and the DOSBase CLI table has changed to accommodate this. Under V36 and higher, you should use new 2.0 functions rather than accessing the CLI table directly.

* Shell windows now have close gadgets. The EOF character is passed for the close gadget of a Shell. This is -1L with CON: getchar(), and the Close Gadget raw event ESC seq with RAW:.

* Shells now use the simple-refresh character-mapped console.

* By default, CON: now opens SIMPLE_REFRESH windows using the V36/V37 console character mapped mode. Because of some differences between character mapped consoles, and SMART_REFRESH non-mapped consoles, this may cause incompatibilities with some applications. For example, the Amiga private sequences to set left/top offset, and set line/page length behave differently in character mapped console windows. The only known work-around is to recompile asking for a CON: (or RAW:) window using the SMART flag.

* Simple refresh/character mapped console windows now support highlighting and copying text with the mouse. This feature, as well as pasting text should be transparent to programs which use CON: for console input, and output. Pasted text will appear in your input stream as if the user had typed it.

* While CONCLIP (see s:startup-sequence) is running, programs may receive "<CSI>0 v" in their input stream indicating the user wants to paste text from the clipboard. This shouldn't cause any problems for programs which parse correctly (however we know that it does; the most common problems are outputting the sequence, or confusing it with another sequence like that for FKEY 1 which is "<CSI>0~").

* The console.device now renders a ghosted cursor in inactive console windows (both SMART_REFRESH, and SIMPLE_REFRESH with character maps). Therefore, rendering over the console's cursor with graphics.library calls can trash the cursor; if you must do this, first turn off the cursor.

* Some degree of unofficial support has been put in for programs which use SMART_REFRESH console windows, and use graphics.library calls mixed with console.device sequences to scroll, draw text, clear, etc. This is not supported in SIMPLE_REFRESH windows with character maps, and is strongly discouraged in all cases.

* Closing an Intuition window before closing the attached console.device will now crash or hang the machine.

* Under 1.2 and 1.3, vacated portions of a console window (e.g., areas vacated because of a clear, or a scroll) were filled in with the character cell color. As of V36 this is no longer true, vacated areas are filled in with the global background color which can be set using the SGR sequence "<ESC>[>##m" where ## is a value between 0-7. In order to set the background color under Release 2, send the SGR to set background color, and a form feed to clear the screen.

* Note that SIMPLE_REFRESH character mapped consoles are immediately redrawn with the global background color when changed--this is not possible with SMART_REFRESH windows.

1.22 E Release 2 Compatibility / Additional Information

Workbench and Startup

1.23 E / Additional Information / Task Switching

The 1.3 Kickstart contained two task switching bugs. After an interrupt, a task could lose the CPU to another equal priority task, even if the first task's time was not up. The second bug allowed a task whose time was up to hold on to the CPU either forever, or until a higher priority task was scheduled. Two busy-waiting tasks at high priority would never share the CPU. Because the input.device runs at priority 20, usually the effect of these bugs was masked out for low priority tasks. The ExecBase->Quantum field had little effect because of the bugs.

For 2.0, a task runs until either its Quantum is up, or a higher priority task preempts it. When the Quantum time is up, the task will now lose the CPU. The Quantum was set to 16/60 second for 1.3, and 4/60 second for 2.0.

In general, the 2.0 change makes the system more efficient by eliminating unnecessary task switches on interrupt-busy systems (for example, during serial input). However, the change has caused problems for some programs that use two tasks of equal priority, one busy-waiting and one Wait()ing on events such as serial input. Previously, each incoming serial character interrupt would cause task context switch allowing the event-handling task to run immediately. Under 2.0 the two tasks share the processor fairly.

1.24 E / Additional Information / Intuition Gadgets and Window Borders

If 2.0 Intuition finds a gadget whose hit area (Gadget LeftEdge/TopEdge/ Width/Height) is substantially inside the border, it will be treated as though it was declared in the border. This is called "bordersniffing". Gadgets declared as being in the border or detected by Intuition as being in the border are refreshed each time after the border is refreshed, and thus aren't clobbered.

Noteworthy special cases:

1) A gadget that has several pixels not in the border is not bordersniffed. An example would be an 18-pixel high gadget in the bottom border of a SIZEBBOTTOM window. About half the gadget will be clobbered by the border rendering.

2) A gadget that is not substantially in the border but has imagery that extends into the border cannot be sniffed out by Intuition.

3) A gadget that is substantially in the border but has imagery that extends into the main part of the window will be sniffed out as a border gadget, and this could change the refreshing results. A common trick to put imagery in a window is to put a 1x1 or 0x0 dummy gadget at window location (0,0) and attach the window imagery to it. To support this, Intuition will never bordersniff gadgets of size 1x1

```
    or smaller.

    All these cases can be fixed by setting the appropriate
    GACT_xxxBORDER gadget Activation flag.

 4) In rare cases, buttons rendered with Border structures and JAM1 text
    may appear invisible under Release 2.
```

The height of the window's title bar is affected by the current font settings. See the discussion of "Screen Attributes" in the "Intuition Screens" chapter. To predict your window's titlebar height before you call OpenWindow():

```
    topborder = screen->WBorTop + screen->Font->ta_YSize + 1
```

The screen's font may not legally be changed after a screen is opened.

Be sure the screen cannot go away on you. This is true if:

1) You opened the screen yourself.

2) You currently have a window open on the screen.

3) You currently hold a lock on this screen (see LockPubScreen()).

IntuiText rendered into a window (either through PrintIText() or as a gadget's GadgetText) defaults to the Window RastPort font, but can be overridden using its ITextFont field. Text rendered with the Text() function appears in the Window RastPort font.

The Window's RPort's font shown above is the initial font that Intuition sets for you in your window's RastPort. It is legal to change that subsequently with SetFont().

1.25 E / Additional Information / Workbench And Startup

Under 1.3, the Workbench Screen and initial Shell (CLI) opened before the first line in s:startup-sequence. Some naughty programmers, in an attempt to recover memory, would search for the bitplane pointers and appropriate the memory for their own use. This behavior is unsafe.

By default 2.0 opens the initial CLI on the first _output_ from the s:startup-sequence. This allows screen modes and other parameters to be set before the user sees the screen. However, this broke so many programs that we put in the "silent-startup" hack. A disk installed with 1.3 install opens the screen as before. A disk installed under 2.0 opens silently. Never steal the Workbench bitplanes. You don't know where they are, how big they are, what format they may be in, or even if they are allocated. Recovering the memory is a bit tricky.

Under 2.0, simply avoid any output from your s:startup-sequence. If your program opens a screen it will be the first screen the user ever sees. Note that if ENDCLI is ever hit, the screen will pop open.

Under 1.3, after ENDCLI, use CloseWorkbench() to close the screen. This

also works under 2.0. Loop on CloseWorkbench() with a delay between loops. Continue looping until CloseWorkbench() succeeds or too much time has passed. Note that a new program called EndRun is available for starting non-returning programs from the startup-sequence. EndRun will reduce memory fragmentation and will close Workbench if it is open. EndRun.lzh will be available in Commodore's Amiga listings area on BIX.