Amiga®

ENHANCER SOFTWARE

FEATURING *AMIGADOS*

Version 1.2

INCLUDING

KICKSTART™ 1.2

WORKBENCH™ 1.2

EXTRAS 1.2

Includes Revised
Amiga Basic™ &
PC Utilities

CONTENTS

AMIGA ENHANCER

The AMIGA ENHANCER package includes:

1. New versions (Version 1.2) of the Kickstart, Workbench, and Extras software disks. The Extras disk includes the new PC Utilities software, which allows you to format and copy 5¼ inch disks in MS-DOS format.
2. The Enhancer documentation, which includes:
   —Introduction to Amiga Update
   —The AmigaDOS Manual Update
   —Amiga ROM Kernel Reference Manual Update
   —Intuition Amiga System Software Update
   —Introduction to PC Utilities

Changes made to the Version 1.2 release to improve system reliability may result in incompatibility with early releases of some application programs. Contact the developer's customer support for information on update releases for V1.2. You may encounter problems with the following programs on V1.2: Music Studio, Deluxe Video (using sampled sound), Seven Cities of Gold, and Transformer.

Following are summaries of the Enhancer documentation.

Introduction to Amiga Update

This consists of individual sentence or paragraph changes which are keyed to page numbers in the original document. In some cases the change may be more than a page long.

Changes occur on the Copyright page at the front of the book; in Chapters 4, 5 and 7; and in Appendix A.

Major new features described in the update pages include:

Page 4–32:	A RAM disk icon that appears when a RAM disk is created by a program or by using CLI.
Page 5–2:	A new Expansion drawer on the Workbench.
Page 5–3, 7–6:	Additional printers supported.
Page 7–5:	A new gadget, Change Serial, is used in place of the Baud Rate Gadget in Preferences.
Page A–4, 5, 6, 7, 8:	New notepad features.
Page A–9:	New menu shortcuts.
Page A–22:	Many new keyboard configurations are available. These can be defined by using a new Workbench tool called SetMap. Details on using SetMap (including illustrations of each new keyboard) are given in this *Introduction to Amiga Update*.
Page A–22:	A new tool, KeyToy, is added to the Extras disk.
Page A–22:	New GraphicDump and Say icons have been added to the Workbench.
Page A–22:	New information on printer escape sequences, including a list of ANSI X3.64 style commands.
Page A–22:	Support for both NTSC and PAL video standards.

The Amiga DOS Manual Update

This consists of individual sentences or paragraphs changes that are keyed to page numbers in the original document.

Amiga ROM Kernel Reference Manual Update

This consists of individual sentences or paragraphs changes that are keyed to page numbers in the original document.

Intuition Amiga System Software Update

This describes changes to and makes general observations about the revision to Intuition that is part of the Version 1.2 Kickstart or ROM software. The discussion is keyed to topics (screens, windows, gadgets, etc.), rather than to the specific page numbers of the original documentation.

Introduction to PC Utilities

This describes the tools provided by the PC Utilities software. (Additional details are available on screen in the form of prompts displayed as the PC Utilities tools are being used.)

INTRODUCTION TO AMIGA

UPDATE FOR VERSION 1.2

Introduction to Amiga: Update to Revision D for Amiga® System Software Release 1.2

In this section, you'll find changes to revision D of *Introduction to Amiga* for the 1.2 release of the Amiga system software.

Copyright page:

Amiga is a registered trademark of Commodore-Amiga, Inc.

Amiga Transformer is a trademark of Commodore-Amiga, Inc.

Apple and ImageWriter are trademarks of Apple Computer, Inc.

Epson is a registered trademark of Epson America, Inc.

IBM is a registered trademark of International Business Machines Corporation.

Okidata is a registered trademark and Microline is a trademark of Oki America, Inc.

Page 4–8:

When you drag an icon on the Workbench, you see a copy of the icon move. The Pointer does not change shape when you drag an icon.

Page 4–8:

If you use Extended selection to select more than one icon, you can drag all the icons you selected at the same time. Hold down the SHIFT key and select the icons; when you select the final icon, **do not release the Selection button; instead, release the SHIFT key.** You can now move the icons you've selected by moving the mouse. When you're finished moving the icons, release the Selection button.

Page 4–15: If you select an icon and then choose Rename from the Workbench menu, the gadget that appears is automatically selected. You can change the gadget's name as soon as the gadget appears. If you click the Menu button when this gadget is displayed, the gadget is no longer selected. If this happens, select the gadget by pointing anywhere within the gadget and clicking the Selection button.

Page 4–21: The text before the first two illustrations reads as follows:

Moving Windows in Front of Other Windows

When windows overlap, one window appears in front of the others. To move a window in front of other windows, select the *Front Gadget:*

You can also move a window to the front by pointing to the icon you selected to open the window, then double-clicking the Selection button.

The text before second two illustrations on page 4-21 reads as follows:

Pushing Windows Behind Other Windows

To move a window behind other windows with which it overlaps, select the *Back Gadget:*

Page 4–27: If you drag a screen so that part of it is below the bottom of the display, the Amiga takes advantage of any extra display area at the bottom of a monitor and displays as much of the screen as it can. One result of this is that icons and gadgets in a screen may be visible at the extreme bottom of the display, but may be lower than the area you can point to with the Pointer.

Page 4–30: *Autorequest requesters* are requesters that contain only two gadgets: a Retry or Continue Gadget, and a Cancel Gadget. There are Amiga-key shortcuts for selecting the gadgets in these requesters. To select Retry or Continue, hold down the left Amiga key, then with the key still down, press the V key. To select Cancel, hold down the left Amiga key, then with the key still down, press the B key.

Page 4–31: When you initialize a disk, the disk drive light stays on until the initialization is complete.

Whenever you initialize a disk from the Workbench, a Trashcan icon appears in the disk drawer for the disk. If the Workbench disk you used to set up the Workbench (the **SYS:** disk) is in a disk drive, the Trashcan icon from that Workbench disk appears on the disk you're initializing. (If you used the Icon Editor to change the Trashcan icon on a Workbench disk and you'd like that icon to appear on newly initialized disks, be sure to use that disk to set up the Workbench.) If the **SYS:** disk is not currently in a disk drive, the same Trashcan that appears on new Workbench disks will appear in the disk drawer of a newly initialized disk.

Page 4–30: Requesters are automatically selected when they appear.

Page 4–30: A *system requester* is a requester that appears when AmigaDOS or another program needs a response from you. An example is a requester that asks you to insert a disk.

A system requester can appear in any screen. If the screen containing a system requester is obscured by another screen, the screen containing the requester is brought to the front; this ensures that the requester is visible. When you've replied to the requester, you can move the screen to the back by selecting the screen's Back Gadget (one of the two gadgets at the upper right of the screen). If the requester appears in the Workbench screen, you can also move the screen back by holding down the left Amiga key while you press the M key.

Page 4–32: A *RAM disk* is an area of the Amiga's random access memory used for operations that normally involve disks. When you use the CLI to create a RAM disk, or when a program you're using creates a RAM disk for you, an icon for the RAM disk now appears on the Workbench. (You cannot create a RAM disk directly from the Workbench. To learn about RAM disks, see the *AmigaDOS Manual* published by Bantam Books.)

Once an icon for a RAM disk appears on the Workbench, it remains there until you reset or turn off your Amiga.

Page 4–33: *String Gadgets* are gadgets that accept text. An example is the gadget that appears when you choose Rename from the Workbench menu. When you select a String Gadget, a Text Cursor appears in the gadget. If you select a String Gadget that already contains text, you can point to the place in the text where you want the Text Cursor to appear.

Page 4–33: Programs can select String Gadgets for you. When a String Gadget appears with a Text Cursor in it, it's selected and you can begin typing. The gadget is selected until you select another gadget, requester, or window.

Page 5–2: The Expansion drawer on the Workbench is new. To use a new device, such as a hard disk drive, with your Amiga, you drag the icon that represents the driver file for the device into the Expansion drawer. The driver file and its icon are supplied by the manufacturer of the device.

Page 5–2: *Interlace screens* (described in the notes about Preferences that follow) appear to "flicker" on many monitors. This flickering can cause eyestrain and other problems. "Long-persistence" monitors are available that greatly reduce the perceived flicker. These monitors can be valuable in applications that require interlace screens.

Page 5–3:

5¼" disk drives, such as the Amiga Model 1020 drives used with the Amiga Transformer™, can be used as external storage devices for the Amiga. Installation instructions are provided with Model 1020 drives; for other 5¼" drives, see your Amiga dealer for the proper cables and instructions.

When using 5¼" disks on the Amiga, a system requester may appear that asks you to insert a disk. The Amiga can detect when you insert a 3½" disk, but it cannot detect when you insert a 5¼" disk. After you insert a 5¼" disk, you must enter the AmigaDOS **diskchange** command followed by the name of the disk drive in order for the Amiga to recognize the disk.

Page 5–3:

The Amiga supports four additional dot-matrix printers:

- The Apple™ ImageWriter™ II
- The Okidata® Microline™ 92
- The Okidata® Microline™ 192
- The Okidata® Microline™ 292

The Microline 92 and 192 printers are each available in two versions: a standard version and a version compatible with IBM® dot-matrix printers. To use the standard version, select Okidata_92 from the list of printer choices in Preferences. To use a version compatible with IBM printers, select CBM_MPS1000 from the list. The Microline 292 printer can be used with two different "personality cards": one that makes it compatible with IBM printers and another that, when used with a different ribbon, lets you print in color. To use the Microline 292 as an IBM compatible printer, select CBM_MPS1000 from the list of printers in Preferences. To use the Microline 292 as a color printer, select Epson_JX-80 from the list.

Page 7–2: There are Front and Back Gadgets and Drag Bars in all the Preferences windows. The Preferences Edit Pointer display is a screen rather than a window; it lacks the Front and Back Gadgets, and although it has a Drag Bar, you cannot drag it to a new location.

Page 7–2: The date in Preferences is displayed Day/Month/Year.

Page 7–2: The clock in Preferences advances when Preferences is open.

Page 7–3: There is a Preferences setting called Workbench Interlace. By changing this setting to On, you double the number of horizontal lines that make up the Workbench screen. The On and Off Gadgets that control this setting are at the upper right of the Preferences window.

For a new Workbench Interlace setting to take effect, first save the setting by selecting the Save Gadget, then reset the Amiga.

On many monitors, the colors in an interlaced screen may appear to "flicker." You can often reduce this effect by changing the screen's colors. For the Workbench, you can use Preferences to change the colors. Experiment to find the colors that work best on your monitor.

Page 7–4: For the CLI icon to appear, you must select the On Gadget to the right of "CLI" in Preferences, then select the Save Gadget to save the new setting. If the System drawer is closed, open it to see the icon. If the System drawer is already open when you leave Preferences, you must close it and reopen it to make the icon appear.

Page 7–5:

In place of the Baud Rate Gadget in Preferences, there is a gadget labeled Change Serial. When you select this gadget, a new window appears; in this window are gadgets you use to change settings for serial communication. You may need to change these settings to meet the requirements of a particular application or add-on device. The serial settings are as follows:

- **Baud Rate.** This is the number of bits transferred through the serial connector each second. The current rate is shown immediately below the words Baud Rate; to change it, select either the Up or Down arrow to the right of the rate.

- **Buffer Size.** The Serial Buffer is an area of memory set aside for serial communication. Its size is shown immediately below the words Buffer Size; to change it, select either the Up or Down arrow to the right of the size.

- **Read Bits.** This is the number of bits expected for each character received through the serial connector. You can select either 7 or 8 bits.

- **Write Bits.** This is the number of bits that are sent through the serial connector for each character. As with the Read Bits setting, you can select either 7 or 8 bits.

- **Stop Bits.** Stop bits are bits added to the end of a character that mark where it ends. You can select either 1 or 2; this is the number of bits added to each character sent through the serial connector, and the number expected at the end of each character that is read.

- **Parity.** Parity checking is a method for reducing transmission errors. Select Even to specify even parity, Odd for odd parity, or None for no parity checking.

- **Handshaking.** This setting lets you specify one of two methods to control the flow of information through the serial connector. You can select xON/xOFF, RTS/CTS, or None if you do not need either method.

When you're through changing these settings, select OK to confirm your selections or Cancel to cancel them. Selecting either OK or Cancel returns you to the Preferences window.

Page 7–6: There are three new choices in the Preferences printer list:

- ImagewriterII (for The Apple ImageWriter II)
- Okidata_292 (for the Okidata Microline 292)
- Okidata_92 (for the Okidata Microline 92)

For more information about these and other supported printers, see the notes for Chapter 5 in this package.

Page 7–6: If you add a new printer driver file to the Workbench disk, a new entry will appear in the Preferences printer list.

Page 7–7: The "generic" selection in the Preferences printer list is the default selection.

Page 7–8: When printing graphics on Epson® and other dot-matrix printers, narrow blank lines may appear on printouts. If this occurs, selecting the Custom setting for Paper Size in Preferences may correct the problem.

Page A–4: When the Clock's alarm goes off, the window for the Clock appears in front of all other windows on the Workbench.

There are many new Notepad features and options, which among other things allow you to do the things described below (see the descriptions that follow for manual pages A–4 through A–9):

- Set tabs using the keyboard TAB key
- Open the Notepad from disk without loading fonts
- Change the default location on the screen where the Notepad opens
- Change the Notepad's default font on both an overall and a note by note basis
- Move the cursor from line to line and page to page using keyboard commands
- Scroll up and down using a Scroll Gadget
- Delete, copy and move text using an Edit Menu and the Clipboard
- Change the default Print As option from Graphic to Draft
- Change the default printer option to Formfeed
- Select an Automatic Wordwrap option

All of these features are described below.

Page A–4: When entering text into a Notepad note, pressing the TAB key adds enough spaces to the left of the Text Cursor to move it to the next predefined tab stop. The first of the Notepad's predefined tab stops is at the eighth character position to the right of the left margin; additional tabs stops are eight columns apart.

Page A–4: You can open the Notepad without loading fonts from disk as follows:

1. Select the Notepad icon, choose Info from the Workbench menu, select the ADD Gadget, and select the "tool types" string gadget—or see Page A-22 and follow that format if you want to be consistent.

2. Type in "FLAGS=nofonts", press the RETURN key, and select the Save Gadget.

If you open the Notepad without loading fonts from disk, you can choose the new Read Fonts item in the Project menu to read the fonts into memory.

Page A-4:

To specify the size and position of the Notepad window whenever you open the Notepad follow the steps below:

1. Select the icon for the Notepad or the icon for a note you've saved, choose Info from the Workbench menu, select the ADD Gadget, and select the "tool types" string gadget—or see Page A-22 and follow that format if you want to be consistent.

2. Type in "WINDOW=" followed by (1) a three-digit number specifying the number of pixels from the left edge of the Workbench screen on the left edge of the window, (2) a comma, (3) a three-digit number specifying the number of pixels from the top edge of the Workbench screen to the top edge of the window, (4) a comma, (5) a three-digit number specifying the width of the window, in pixels, (6) a comma, and (7) a three-digit number specifying the height of the window, in pixels.

   An example: WINDOW=100,070,050,090

3. Press the RETURN KEY and select the SAVE Gadget.

If the Notepad cannot create the window you've specified, it will ignore your request.

You can also specify the position and size of the window for a note you've saved by selecting the icon for the note, then performing steps 2 and 3 listed above.

Page A-4:

To specify the default font for Notepad whenever you open the Notepad follow the steps below:

1. Select the Notepad icon, choose Info from the Workbench menu, select the ADD Gadget, and select the "tool types" string gad-

get—or see Page A-22 and follow that format if you want to be consistent.

2. Type "FONT=" followed by the name of the font (as shown in the Notepad's Font menu), a period, and a number for the font size (for example, FONT=emerald.20) then press the RETURN key.

3. Select the SAVE Gadget.

Page A-4: There is a limit to the number of lines you can include in a Notepad note. This limit varies; it depends on the size of the fonts you use for your note. (With the plain, 9-point Topaz font, the original default font for the Notepad, you can enter 73 lines.) When you reach the limit, characters you type no longer appear in the Notepad window.

Page A-5: Holding down the SHIFT key while you press the Up cursor key moves the Text Cursor to the top of the current Notepad page. Holding down the SHIFT key while you press the Down cursor key moves the Text Cursor to the bottom of the current page. Holding down the SHIFT key while you press the left cursor key moves the Text Cursor to the beginning of the line. Holding down the SHIFT key while you press the right cursor key moves the Text Cursor to the end of the current line.

Page A-5: The page number of whatever page is being displayed appears in the Notepad's Previous Page Gadget.

Page A-5: There are Up and Down Scroll Gadgets at the right of the Notepad window. Selecting the Up Scroll Gadget (the arrow-shaped gadget immediately below the Previous Page Gadget) moves the text in the window up one line. Selecting the Down Scroll Gadget (the gadget immediately above the Sizing Gadget) moves the text in the window down one line.

Page A–5:

There is an Edit menu for the Notepad that lets you perform several text editing functions in the Notepad. To cut text and put it on the Clipboard, first point to the start of the text you want to cut and double-click the Selection button, then point to the end of the text and double-click the Selection button again. The selected text is highlighted. To cut the highlighted text, choose Cut from the Edit menu. (If you decide not to cut the text you've selected, choose Cancel from the Edit menu.) To leave the highlighted text where it is and put a copy of it on the Clipboard, choose Copy from the Edit menu.

You can also select text to cut or copy by positioning the Text Cursor at the beginning of the text, choosing Mark place from the Edit menu, moving to the end of the text, then choosing Mark place again.

Each time you cut or copy text, the text replaces the previous contents of the Clipboard.

To paste text from the Clipboard into a note, point to the place in your note where you want the text to appear, click the Selection button to reposition the Text Cursor, then choose Paste from the Edit menu.

Page A–5:

You can search for and replace text in a note. To search, choose Find from the Edit menu. In the requester that appears, select the gadget labeled "Find:", enter the text you want to search for, then press the RETURN key. If you want to replace this text, select the gadget labeled "Repl:", enter the text you want to use as a replacement, then press the RETURN key. When you're done entering text, select the Next Gadget to find the next occurrence of the text, or Last to find the previous occurrence. Select CANCEL if you change your mind about finding the text.

Once you've entered text in these gadgets, you can choose Find Next to find the next occurrence. Find Last to find the previous occurrence, and Replace to replace text.

Page A–6: If you save a project that you've changed with the Notepad without specifying an AmigaDOS directory name, the project is saved in the same drawer as the Notepad.

Page A–6: In the Notepad's submenu for Print As, the Graphic option is the original default. To have the Draft option chosen automatically whenever you open the Notepad, do the following:

1. Select the Notepad icon, choose Info from the Workbench menu, select the ADD Gadget, and select the "tool types" string gadget—or see Page A-22 and follow that format if you want to be consistent.
2. Type in "FLAGS = draft" and press the RETURN key.
3. Select the SAVE Gadget.

Page A–6: When you choose Save As, or when your note doesn't yet have a name and you choose Save, the String Gadget in the requester that appears is automatically selected. This means that you can type the name for the note as soon as the requester appears; you do not need to select the gadget before typing.

Page A–6: When you save a note, the last font you chose when the Global font option was on becomes the default font for the note. (The default font for a note is the one that is chosen automatically when you reopen the note.)

Page A–6: After you save a note, you can specify a different default font for the note as follows:

1. Select the icon for the note, choose Info from the Workbench, select the ADD Gadget, and select the "tool types" string gadget—or see Page A-22 and follow that format if you want to be consistent.
2. Type "FONT = " followed by the name of the font (as shown in the Notepad's Font menu), a period, and a number for the font size (for example, FONT = emerald.20) and press the RETURN key.
3. Select the Save Gadget.

Page A–7: When you print a note, a blank line is no longer added automatically to the top of each page. To add blank lines, point to Print As from the Project menu, then choose Form-feeds from the submenu that appears.

Page A–7: To have the Form-feeds option chosen automatically whenever you open the Notepad follow the steps below:

1. Select the Notepad icon, choose Info from the Workbench menu, select the ADD Gadget, and select the "tool types" string gadget—or see Page A-22 and follow that format if you want to be consistent.
2. Type "FLAGS = formfeed" and press the RETURN key.
3. Select the SAVE Gadget.

Page A–8: You can change fonts within a note. First check to see if the new Global font option in the Format menu is chosen. (The Global font option is chosen by default when you open the Notepad unless you make one of the changes described below. You can tell that it's chosen if a check mark appears to the left of the Global font item in the Format menu.) If the option is chosen, choose the Global font item to turn it off. You can change the font and size of newly entered text by choosing a new item from the Font menu.

Page A–8: You can specify whether or not the Global font option is chosen when you open the Notepad or a note as follows:

1. Select the icon for the Notepad, choose Info from the Workbench menu, select the ADD Gadget, and select the "tool types" string gadget—or see Page A-22 and follow that format if you want to be consistent.
2. To have the Global font option chosen automatically when you open the Notepad or a note, type "FLAGS = global" and press the RETURN key. To specify that the Global font option NOT be chosen when you open the Notepad or a note, type in "FLAGS = local" and press the RETURN key.
3. Select the SAVE Gadget.

If you enter FLAGS = global, font changes in a note are not preserved when you reopen the note; the characters all become the current default font.

Page A–9:

Word wrap is a new option in the Format menu. Word wrap is chosen when you open the Notepad; when it's chosen, the Notepad ends each line of text at the end of a word. A check mark to the left of the Word wrap menu item indicates that the option is chosen. To turn off the word wrap feature, choose the Word wrap item. To turn the feature back on, choose the Word wrap item again.

To turn off the Word wrap feature automatically whenever you open the Notepad follow these steps:

1. Select the Notepad icon, choose Info from the Workbench menu, select the ADD Gadget, and select the "tool types" string gadget—or see Page A-22 and follow that format if you want to be consistent.

2. Type "FLAGS = nowrap" and press the RETURN key.

3. Select the SAVE Gadget.

Page A–9:

You can make an entire note appear in the same font by choosing Remove fonts from the Format menu.

Page A–9:

You can remove all style changes (changes from and to italic, bold, underline, or plain text) in a note by choosing Remove styles from the Format menu.

Page A–9:

There are a number of new menu shortcuts. These shortcuts are shown to the right of the menu items. To use these shortcuts, hold down the right Amiga key, then with the key still held down, press the other key shown:

O	Open
S	Save
Q	Cancel
X	Cut
&	Paste
C	Copy
M	Mark place
F	Find
+	Find Next
–	Find Previous
R	Replace
P	Plain
I	Italic
B	Bold
U	Underline

In addition, entering CTRL–L redraws what appears in the Notepad window. (To enter a CTRL–L character, hold down the CTRL key on the keyboard while you press the L key, then release both keys.)

Page A–22:

There are now several different keyboards available for the Amiga. SetMap is a new Workbench tool that lets you use other keyboards. You use SetMap to select the correct "key map": a list that tells your Amiga the right character to print for each key on a keyboard.

The SetMap tool is in the System drawer on the Workbench. To use SetMap, open the System drawer, then follow the steps below:

1. Select the SetMap icon.

2. Choose Info from the Workbench menu.

3. In the requester that appears, select the ADD Gadget to the right of the words TOOL TYPES.

4. Select the String Gadget between the ADD Gadget and the words TOOL TYPES.

5. Type "KEYMAP=" followed by the name of a new key map. The key maps available are:

d	for German keyboards
e	for Spanish keyboards
f	for French keyboards
gb	for British keyboards
i	for Italian keyboards
is	for Icelandic keyboards
s	for Swedish keyboards
dk	for Danish keyboards
n	for Norwegian keyboards
cdn	for French Canadian keyboards
usa	for standard United States keyboards
usa0	This is the key map provided with the 1.1 release of the Amiga system software. Some programs may require this key map to function properly.
usa2	for Dvorak keyboards

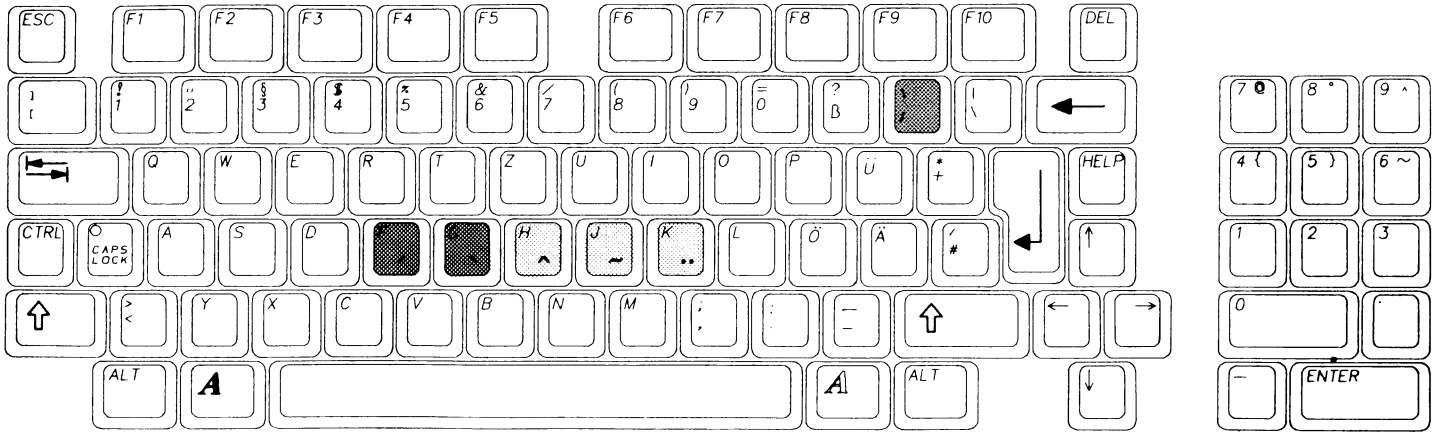
   When you've finished entering the name of the key map, press the RETURN key.

6. Select the SAVE Gadget at the lower left of the requester.

7. When the Workbench reappears, point to the SetMap icon, then double-click the Selection button.

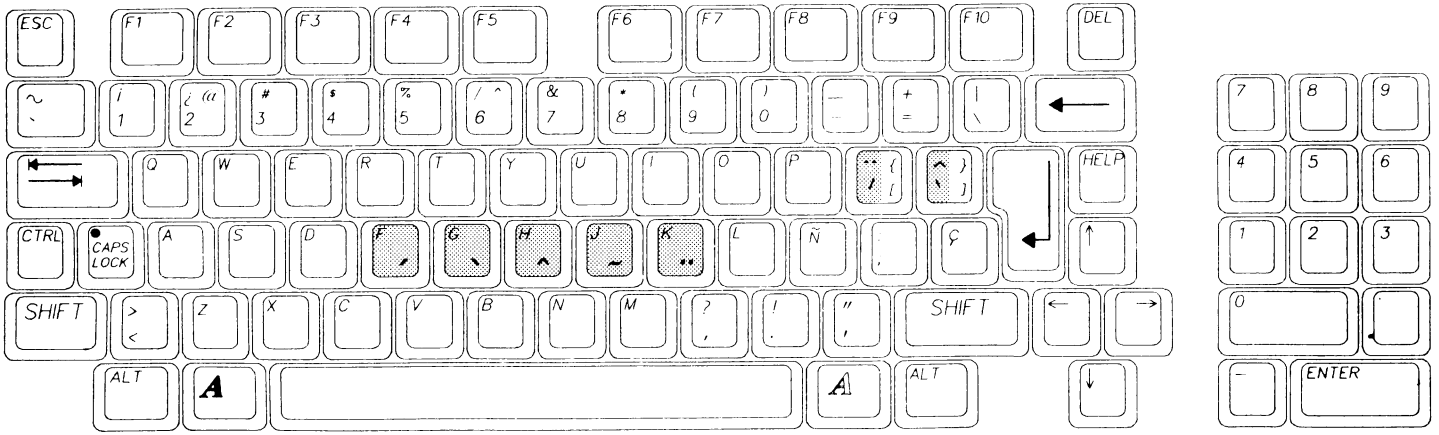
Not all of the keyboards listed above, and shown on the following pages, are currently manufactured.

GERMAN KEYBOARD (d)



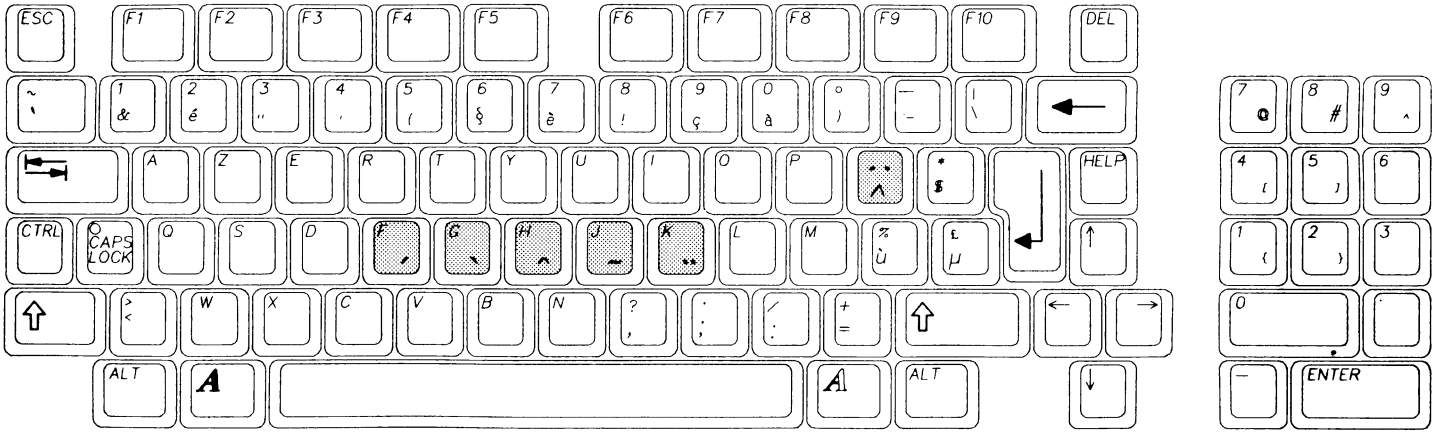
NOTE: Light tinted keys denote "dead" keys. Dark tinted keys are "double-dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

SPANISH KEYBOARD (e)



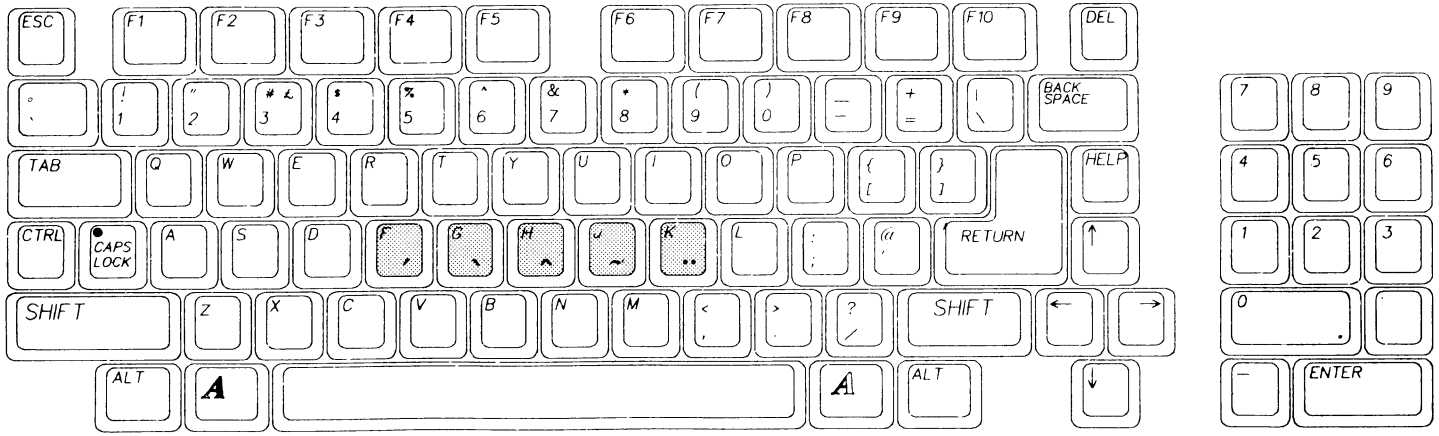
NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

FRENCH KEYBOARD (f)



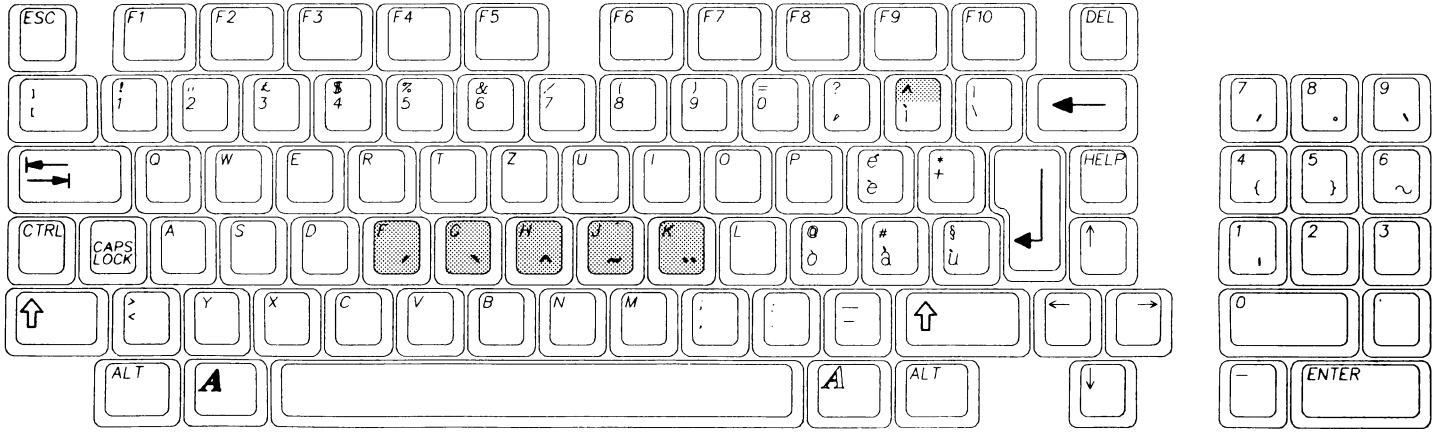
NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

BRITISH KEYBOARD (gb)



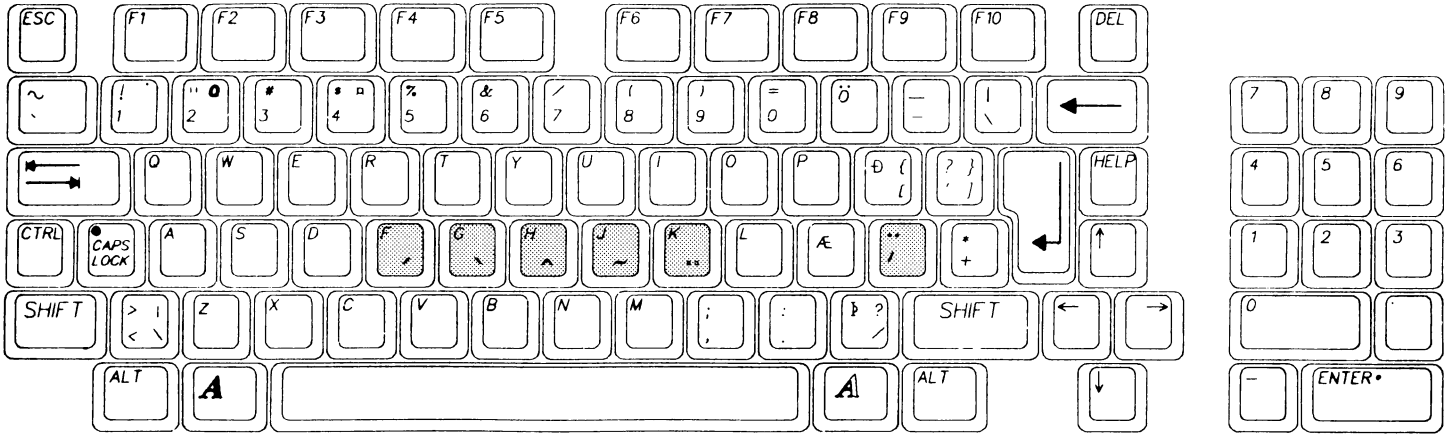
NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

ITALIAN KEYBOARD (i)



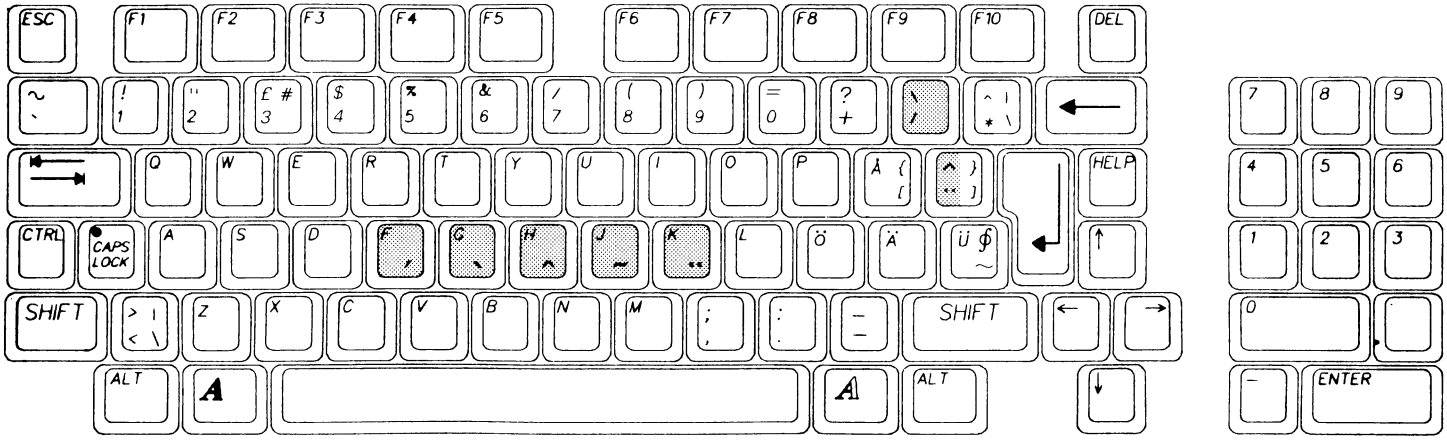
NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

ICELANDIC KEYBOARD (is)



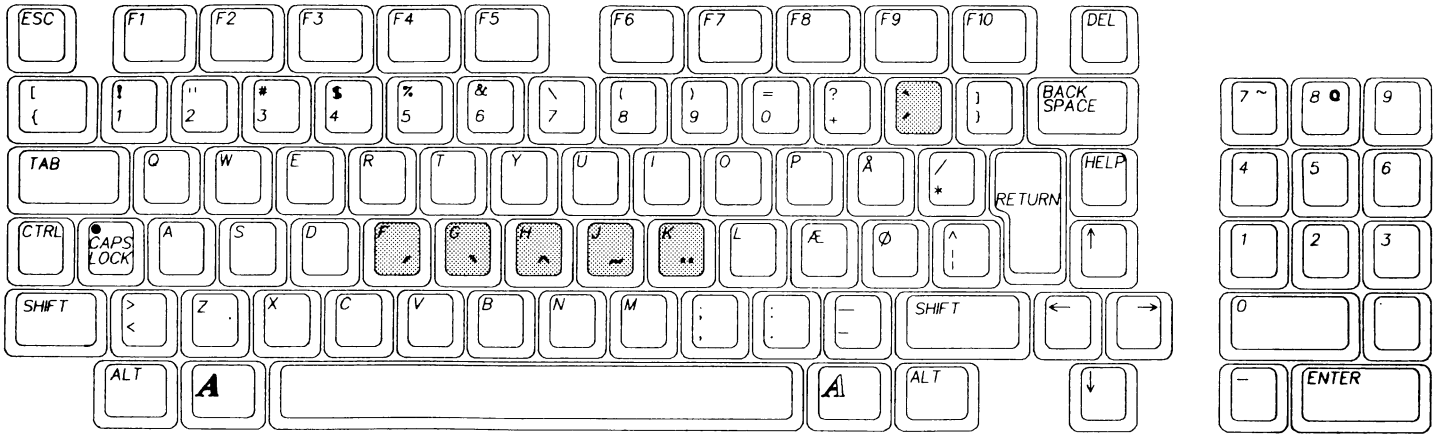
NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

SWEDISH/FINNISH KEYBOARD (s)



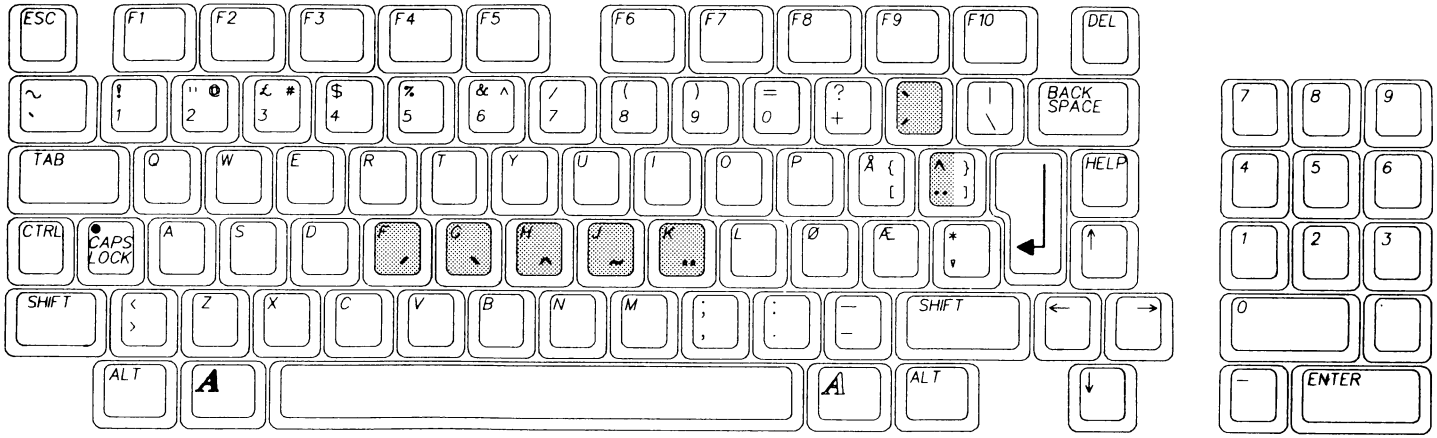
NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

DANISH KEYBOARD (dk)



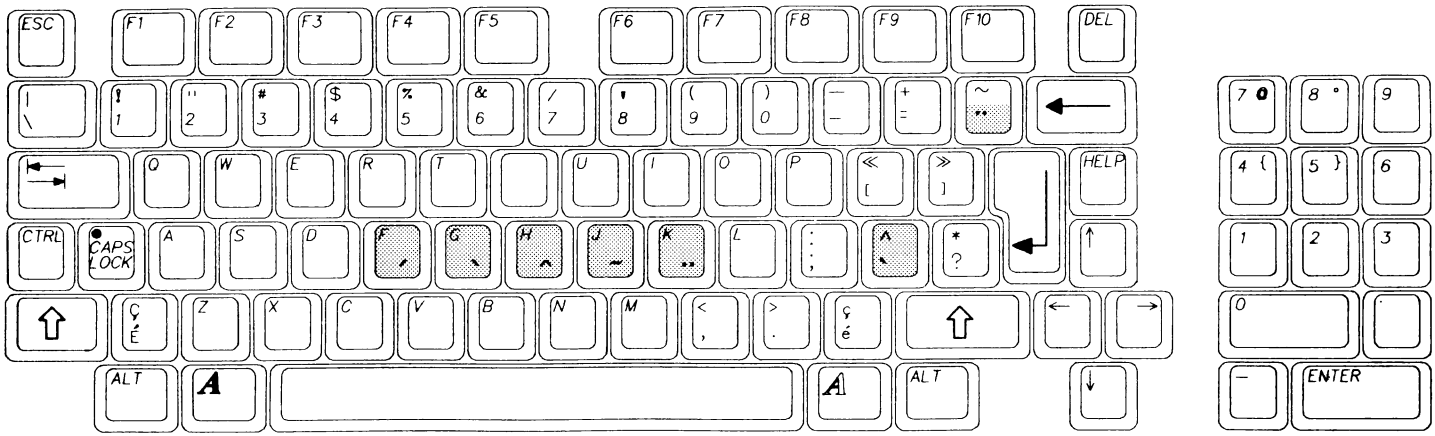
NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

NORWEGIAN KEYBOARD (n)



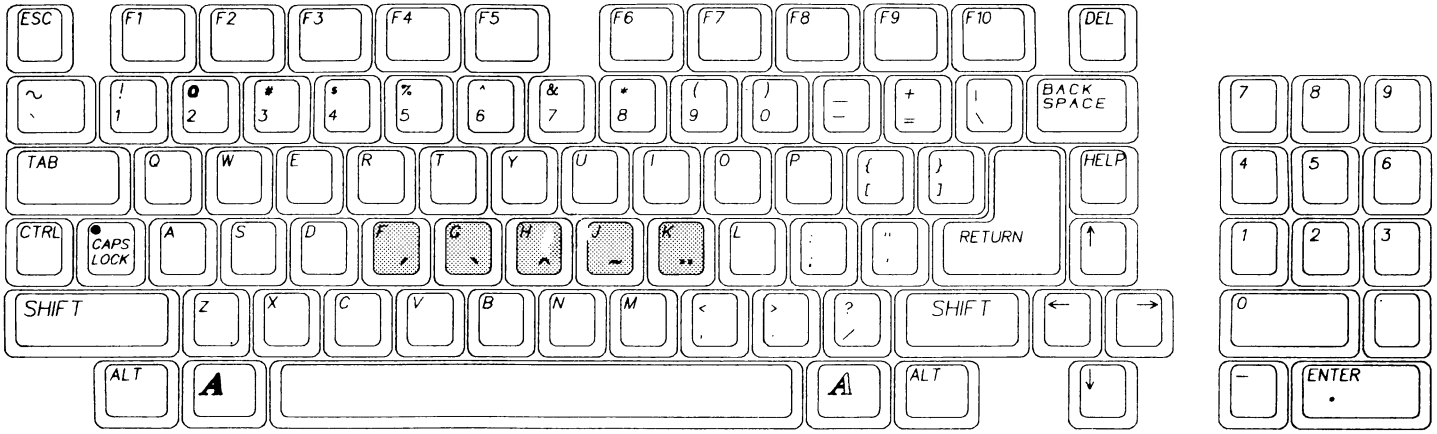
NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

FRENCH CANADIAN KEYBOARD (cdn)



NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

STANDARD U.S. KEYBOARD (usa)



NOTE: Darker keys denote "dead" keys. Characters shown on the bottoms of these keys are not actually imprinted on the key caps.

Each key map has a set of alternate characters. The alternate character set includes special symbols and diacritical marks used in each language. To enter an alternate character, hold down the ALT key while pressing another key on the keyboard.

Included among the alternate characters in the new key maps are "dead" keys. These keys allow you to add accents and other diacritical marks to selected characters. When you press a "dead" key, the cursor does not advance; thus the diacritical mark and the next character you type appear in the same position. Not all characters can be combined with "dead" keys.

KEYTOY, a new tool on the Extras disk, allows you to separately display each key map.

GraphicDump and Say

Two new features have been added to the Workbench. These are GraphicDump and Say. GraphicDump lets you print entire screen images, including menus and icons, just as they appear on your monitor. Say lets you type words and sentences, which the Amiga then "says" using its audio capabilities.

Both GraphicDump and Say have icons in the System drawer in the Workbench window.

GraphicDump

To use GraphicDump, open the System drawer of the Workbench and double-click on the GraphicDump icon. After a delay of approximately 10 seconds, the entire screen image that is foremost on the monitor is "dumped" to the printer. (You must have a graphics printer connected to your Amiga and properly selected in Preferences.)

The 10-second delay allows you time to move images on the screen, open menus, or move the current Workbench screen to the back and let another screen image be printed.

Say

Use the Say feature by opening the System drawer and double-clicking on the Say icon. Two new windows will open. One is labeled "Phoneme window" and the other is labeled "Input window."

Activate the input window by moving the pointer into it and clicking with the mouse. Now type the words or sentences you want the Amiga to "speak" and press Return. You'll hear the Amiga repeating your words.

If you type long lines or sentences, don't press Return when you reach the right edge of the input window. The text will automatically wrap around the next line. Don't worry if the line is broken in the middle of a word.

You can use the standard Amiga window manipulation techniques to move or change the size of both the phoneme and input windows.

After you typed in your words and pressed Return, you may have noticed that text appears in the phoneme window just before the Amiga speaks. This display is a phonetic interpretation of your words (that is, it is an interpretation of the sounds necessary to speak the words. Say translates what you type into sounds using a phonetic alphabet and phonetic rules. This translation is printed in the phoneme window.

To use and enjoy Say you don't need to learn all the rules of phonetics. However, since phonetic spelling is based on how words sound rather than how they are conventionally spelled, you should think in terms of sounds and at the same time watch how Say spells different words (or sounds) in the input window. A good dictionary will be helpful for looking up the pronunciation of words you are uncertain about.

You can change the voice, pitch, and speed of your Amiga's speech by choosing different options in the phoneme window.

You have a choice of a male, female, robot, or natural voice. Select which voice you want by typing the appropriate code below in the phoneme window:

–m	for male voice
–f	for female voice
–r	for robot voice
–n	for natural voice

For example, to change to the female voice type –f in the phoneme window and press return.

To change the pitch of the voice, in the phoneme window type –p followed by a number of 65 through 320, for example:

–p89 <return>

The higher the number you type, the higher the voice's pitch will be.

To change the speed of the voice, in the phoneme window type –s followed by a number of 40 through 400, for example:

–s321 <return>

The higher the number you type, the faster the voice will speak.

To close Say just move the pointer into the input window, activate the window by clicking with the mouse, and then press return without typing any words first.

Printer Escape Sequences

The Amiga printer device accepts standard escape sequences that are translated into escape sequences specific to the printer selected in Preferences. When using the printer device or printer handler (PRT:), use these standard sequences to implement special printer features. If the feature is not supported by your printer or driver, the sequence will be ignored.

To send a command to the printer from CLI follow the steps below.

1. Redirect the keyboard to the printer by typing the following:

   copy * to prt:<return>

2. Wait until disk access stops.

3. Type an escape sequence, for example the NLQ On command:

   <ESC>[2"z

   What you type will not be echoed on the screen.

4. Hold down the Control key and press the backslash key (\) to terminate the keyboard file.

If your printer supports NLQ mode, NLQ will now be on. Any file you COPY or TYPE to prt: will be printed in near letter quality. You can also use the four steps above to create small printer command files. For example, you could replace the prt: in step 1 with ram:NLQon. Then the four steps will create a file called NLQon in ram:. To turn on NLQ, copy ram:NLQon to prt:

ANSI X3.64 Style Commands

Note: ESC is the Escape key if you are typing a sequence from the keyboard. In BASIC, it is CHR$(27). In C /033 can be used.

Label	Escape Sequence	Feature
aRIS	ESCc	reset
aRIN	ESC#1	initialize
aIND	ESCD	line feed
aNEL	ESCE	return line feed
aRI	ESCM	reverse line feed
aSGR0	ESC[0m	normal character set
aSGR3	ESC[3m	italics on
aSGR23	ESC[23m	italics off
aSGR4	ESC[4m	underline on
aSGR24	ESC[24m	underline off
aSGR1	ESC[1m	boldface on
aSGR22	ESC[22m	boldface off
aSFC	ESC[30m to ESC[39m	set foreground color
aSBC	ESC[40m to ESC[49m	set background color

Label	Escape Sequence	Feature
aSHORP0	ESC[0w	normal pitch
aSHORP	ESC[2w	elite on
aSHORP1	ESC[1w	elite off
aSHORP4	ESC[4w	condensed fine on
aSHORP3	ESC3w	condensed off
aSHORP6	ESC6w	enlarged on
aSHORP5	ESC[5w	enlarged off
aDEN6	ESC[6"z	shadow print on
aDEN5	ESC[5"z	shadow print off
aDEN4	ESC[4"z	doublestrike on
aDEN3	ESC[3"z	doublestrike off
aDEN2	ESC[2"z	NLQ on
aDEN1	ESC[1"z	NLQ off

aSUS2	ESC[2v	superscript on
aSUS1	ESC[1v	superscript off
aSUS4	ESC[4v	subscript on
aSUS3	ESC[3v	subscript off
aSUS0	ESC[0v	normalize the line
aPLU	ESC1	partial line up
aPLD	ESCK	partial line down
aFNT0	ESC(B	US character set
aFNT1	ESC(R	French character set
aFNT2	ESC(K	German character set
aFNT3	ESC(A	UK character set
aFNT4	ESC(E	Danish I character set
aFNT5	ESC(H	Swedish character set
aFNT6	ESC(Y	Italian character set
aFNT7	ESC(Z	Spanish character set
aFNT8	ESC(J	Japanese character set
aFNT9	ESC(6	Norwegian character set
aFNT10	ESC(C	Danish II character set
aPROP2	ESC[2p	proportional spacing on
aPROP1	ESC[1p	proportional spacing off
aPROP0	ESC[0p	proportional spacing clear
aTSS	ESC[n E	set proportional offset
aJFY5	ESC[5 F	auto left justify
Label	**Escape Sequence**	**Feature**
aJFY7	ESC[7 F	auto right justify
aJFY6	ESC[6 F	auto full justify
aJFY0	ESC[0 F	auto justify off
aJFY3	ESC[3 F	letter space (justify)
aJFY1	ESC[1 F	word fill (auto center)
aVERP0	ESC[0z	1/8″ line spacing
aVERP1	ESC[1z	1/6″ line spacing
aSLPP	ESC[nt	set form length n
aPERF	ESC[nq	perf skip n (n > 0)
aPERF0	ESC[0q	perf skip off

aLMS	ESC#9	left margin set
aRMS	ESC#0	right margin set
aTMS	ESC#8	top margin set
aBMS	ESC#2	bottom margin set
aSTBM	ESC[Pn1;pn2r	top and bottom margins
aSLRM	ESC[Pn;Pn2s	left and right margins
aCAM	ESC#3	clear margins
aHTS	ESCH	set horizontal tab
aVTS	ESCJ	set vertical tabs
aTBC0	ESC[0g	clear horizontal tab
aTBC3	ESC[3g	clear all horizontal tabs
aTBC1	ESC[1g	clear vertical tabs
aTBC4	ESC[4g	clear all vertical tabs
aTBCALL	ESC#4	clear all horizontal and vertical tabs
aTBSALL	ESC#5	set default tabs
aESTEND	ESC[Pn"x	extended commands

ADDITIONAL NOTES

The system software now supports both the NTSC and PAL video standards. If a special custom chip for PAL has been installed in your Amiga, 256 horizontal lines appear on normal video displays; 512 horizontal lines appear on interlaced displays.

The new software also detects the frequency of the AC line current (50 or 60 Hz) and automatically uses the right frequency for internal timekeeping.

THE AMIGADOS MANUAL

UPDATE FOR VERSION 1.2

Page 11:

The RAM: device now writes to files in blocks of 512 bytes.

To create a RAM: device each time you boot, edit your startup file to include the line:

dir ram:

Page 43:

When you enter the ASSIGN LIST command, full directory names now appear.

Page 43:

There is a new command, ADDBUFFERS:

Format: ADDBUFFERS df<x>: <nn>
Purpose: to reduce disk access time by adding sector cache buffers

Specification:

Adds <nn> buffers to the list of sector caches for drive <x>. Adding additional buffers can significantly reduce disk access time. The tradeoff is that the command uses up memory; each buffer added reduces memory by approximately 500 bytes.

The improvement in disk access time is less pronounced when you add more than 25-30 extra buffers.

Example:

ADDBUFFERS dfl: 25

Adds 25 buffers to the sector caches for disk drive dfl:. If you are unsure about the number of buffers to add for your application, this is a good number to start with.

Page 43:

There is a new command, BINDDRIVERS:

Format: BINDDRIVERS
Purpose: To bind device drivers for add-on hardware

Specification:

The BINDDRIVERS command is normally part of a startup script. It is used to bind device drivers found in the directory SYS:Expansion to add-on hardware that has been automatically configured by the expansion library. (For end users, this means that if icons for expansion hardware are in the Expansion drawer on the Workbench, the hardware will be configured automatically when they boot up.)

Example:

BINDDRIVERS

Page 45:

There is a new command, CHANGETASKPRI:

Format: CHANGETASKPRI <priority>
Purpose: To change the priority of CLI tasks

Specification:

The Amiga uses the priority numbers to determine which of the tasks currently running it must attend to. Normally most user tasks have a priority of 0, and the time and instruction cycles of the CPU are time-sliced among them.

The CHANGETASKPRI command changes the priority of the CLI task from which it's run. Tasks started from this CLI inherit its priority.

Although priority numbers can range from −128 to +127, you should only enter values from −5 to +5 to avoid disrupting important system tasks. The command does not check for the value of <priority>.

Example:

CHANGETASKPRI 5

This CLI task, and any other tasks started from it, will have priority over any user tasks created without the CHANGETASKPRI command.

Page 45: Because of improved disk caching and a new disk layout, you can improve your Amiga's performance by copying an existing disk onto a different disk with the COPY ALL command.

Page 48: You can now halt output from the DIR command by entering CTRL-C.

Page 49: There is a new command: DISKCHANGE:

Format: DISKCHANGE <dr>
Purpose: To tell AmigaDOS that you've changed disks in a 5¼" disk drive

Specification:

Enter the DISKCHANGE command to inform AmigaDOS that you've changed disks in drive <dr>:. This is necessary for 5¼" disk drives that do not detect when a disk has been changed.

Example:

A requester appears that asks you to insert BigDisk into drive df2:. If df2: is a 5¼" drive, you must insert the disk and enter the command:

DISKCHANGE df2:

before AmigaDOS will continue.

Page 49:

DISKCOPY now works with any disks (including hard disks, disk partitions, and 5¼" disks) that have been mounted with the MOUNT command providing that those disks and partitions are the same size.

Page 50:

There is a new requester for Diskcopy:

Diskcopy:
Failed to open icon.library
Retry Cancel

The requester appears when Diskcopy cannot gain access to the icon library kept on the Workbench disk. This can happen if you do the following:

Copy c:diskcopy TO ram:
[Remove the Workbench disk.]
DISKCOPY
[A requester appears that asks you for the disk that has the LIBS: directory on it; this is normally the Workbench disk.]
[Select the Cancel gadget in the requester.]

If the requester appears, insert the Workbench disk.

(Technical note: Diskcopy retries the OpenLibrary call when this occurs. The retry flag is DISKINSERTED.)

Page 52:

There is a new command, DISKDOCTOR:

Format: DISKDOCTOR <dr>:
Purpose: To fix a corrupted disk in drive <dr>:

Specification:

If AmigaDOS detects a corrupted disk, it will display a message saying that the disk could not be validated. (A disk may be corrupted if, for example, you remove it from a drive while the disk drive light is still lit.) DISKDOCTOR restores as much of the file structure as it can. After you've run DISKDOCTOR, you should copy all the files that you want to keep onto another disk, and then reformat the corrupted disk.

Example:

AmigaDOS has detected a corrupted disk if you see requesters like:

Volume
Textfiles
is not validated

or

Error validating disks
Disk is unreadable

If the disk is in drive dfl:, enter:

DISKDOCTOR dfl:

After DISKDOCTOR has finished, it will display the message:

Now copy files required to a new disk and reformat this disk.

Page 54:

When you use EXECUTE, it now creates a T: directory if one doesn't already exist.

Page 65:

FORMAT now works with any disks (including hard disks, disk partitions, and 5¼" disks) that have been mounted with the MOUNT command. The syntax of the FORMAT command is:

FORMAT DRIVE <drivename>: NAME <diskname> [NOICONS]

Unless you include the NOICONS option, a Trashcan icon will be added to the disk you format. You can stop the command before it is finished by entering CTRL–C.

When used from the CLI, FORMAT now re-searches the command directory path if the first search fails. This is done under the assumption that the user may have inserted a disk as a result of a requester making one or more of the path directories valid. Thus, the path is searched twice before the command fails.

Page 65:

The INITIALIZE command is a new Workbench menu option. Like the FORMAT command, it works with any disks (including hard disks, disk partitions, and 5¼" disks) that have been mounted with the MOUNT command.

When you use INITIALIZE, a Trashcan icon is added to the disk. If you have your SYS: disk in any drive it will use the Trashcan image on that disk for the new trashcan. Otherwise the new Trashcan image that appears on new Workbench disks is added. To provide compatibility with the old FORMAT command, INITIALIZE has a NOICON option to make completely blank disks. To stop the initialization of a disk before INITIALIZE is finished, enter CTRL-C at the keyboard.

Page 72:

There is a new command, MOUNT:

Format: MOUNT <dr>:
Purpose: To make a new device (such as a disk drive) available.

Specification:

To use a new device with the Amiga, you must first create an entry for the device in the file devs:Mountlist. Then you enter the MOUNT command to make the device available. MOUNT commands are normally included in the file s/Startup–Sequence.

Normally hardware manufacturers will supply the proper procedures for installing their devices. If not, the file devs:Mountlist contains an entry for using an A1020 5¼" disk drive, which you can modify for other devices.

The format for an entry in the dev:Mountlist file is:

```
DF2:    Device = trackdisk.device
        Unit = 2
        Flags = 1
        Surface = 2
        BlocksPerTrack = 11
        Reserved = 2
        PreAlloc = 11
        Interleave = 0
        LowCyl = 0; HighCyl = 39
        Buffers = 5
        BufMemType = 3
```

You can include more than one description on the same line if you separate the descriptions with semicolons (;). Numbers are assumed to be decimal unless they are preceded by 0x in which case they are treated as hexadecimal. Comments may be placed in the file as in C programs.

If the device is not a storage device, you can use this format for the entry.

AUX Handler = 1:aux–handler
Stacksize = 700
Priority = 5

Page 72: There is a new FROM option for the NEWCLI. The new format for the NEWCLI command is:

NEWCLI [FROM <filename>]

Including FROM <filename> causes the new CLI you create with the NEWCLI command to execute the commands in <filename> when it's started.

Page 73: The Notepad can now be run from the CLI. The syntax is:

[run] notepad [[[−q] <filename>] | ?]

where the −q option opens the Notepad without reading in fonts from disk, <filename> is the name of a file to be opened, and the ? option shows the syntax of the command without opening the Notepad.

Page 73: There is a new command: PATH:

Format: PATH [SHOW] | [ADD <directoryname> [, <directoryname>] . . .] | [RESET <directoryname> [, <directoryname>] . . .]

Purpose: To add, see, or change the directories in which AmigaDOS searches for commands to execute.

Specification:

The PATH command lets you add, see, or change "search paths": directories that AmigaDOS searches when looking for a program to execute. By default, AmigaDOS searches the current working directory, then the SYS:c directory for a program. Enter the PATH command with the SHOW parameter or without parameters to show the current search paths. Enter the PATH command with one or more directory names to specify search paths; these paths replace any paths you defined before. Enter PATH ADD followed by one or more directory names to add paths to those you've already defined.

You can add up to ten directories with one PATH ADD command; names of the directories must be separated by at least one space. The ADD keyword is optional.

Replace the existing search path with one of your own by typing PATH RESET followed by the names of the directories. The existing search path (except for the current directory and SYS:C) is erased and the new one substituted.

Restore the search path to its default state (searching only the current directory and SYS:c), by typing PATH RESET. This command undoes the effect of the PATH ADD command in a startup file.

To see a listing of the current search path, type PATH SHOW or just PATH.

Keywords can go at the beginning or the end of a command line. For example, you can type PATH <file1> <file2> RESET or PATH RESET <file1> <file2>

Examples:

PATH SHOW (or PATH)
displays the directories in which AmigaDOS will search for commands. A typical search path would be:

Current directory
Workbench:System
C:

PATH ADD SYS:newcommands (or PATH SYS:newcommands)

AmigaDOS will then search for commands located in SYS:newcommands directory. (By default, the s/Startup-Sequence file includes a PATH ADD SYS:System command.)

Current directory
Workbench:System
Workbench:newcommands
C:

You may want to add this command to your s/Startup-Sequence file so that AmigaDOS automatically searches the SYS:newcommands any time you use that disk. If you look at the s/Startup-Sequence file, you will see that this is how SYS:System has been added to the search path.

Page 74: The PROTECT command has been modified to alter only the lower four bits of the protection field and preserve the other bits. It previously set all bits to 1. Bit 4 of the protection field (the archive bit) is now cleared whenever a file that has been written to is closed or a directory is updated. This allows an archiving program to scan a disk and detect those files that have been altered since it last ran. It causes the protect function to set the archive bit to mark the file as archived.

Page 77: Entering the RUN command no longer changes the priority of your CLI.

Page 78:

There is a new command, SETDATE:

Format: SETDATE <file> <date> [<time>]
Purpose: To change the timestamp associated with a file or directory.

Specification:

This command lets you change the timestamp associated with a file or directory.

Page 78:

There is a new command: SETMAP:

Format: SETMAP <mapfilename>
Purpose: To change the key map for the keyboard to one for another country.

There are now alternate key maps for keyboards used in countries other than the U.S.A. See the 1.2 update to Introduction to Amiga for more information about keymaps and illustrations of available keyboards.

The :devs/keymaps directory contains the names of the keymap files available on the Workbench disk.

To get back the default key map stored in read-only memory, enter the CLI command:

SETMAP usa

You can edit your startup script to include a SETMAP command.

Page 78:

There is a new command, SETTASKPRI:

Format: SETTASKPRI <priority>
Purpose: To change the priority of a CLI task.

Specification:

You use the SETTASKPRI command to change the priority of the CLI task from which you enter the command. In turn, tasks started from this CLI inherit its priority.

SETTASKPRI does not check for appropriate values for <priority>. To avoid disrupting important system tasks, enter values from −5 to +5 for <priority>.

Page 107: When you run EDIT, it now creates a T: directory if one doesn't already exist.

Page 174: Step 3 in the example for ExNext( ) should read: "Keep calling ExNext( ) until it fails. Then, if the error code held in IoErr( ) is equal to ERROR_NO_MORE_ENTRIES, you're finished."

Page 176: There is a new accessMode for opening files: MODE_READWRITE (= 1004). This mode opens an old file with an exclusive lock. In addition, you can now use the identifier MODE_READONLY as a synonym for MODE_OLDFILE.

Page 180 WaitForChar ( ) cannot be used on noninteractive streams.

Page 183 If a command you call with the Execute( ) function calls Exit( ), Execute( ) returns the value passed to Exit( ).

Page 241: DISKED now works with any disks (including hard disks, disk partitions, and 5¼" disks) that have been mounted with the MOUNT command.

Page 254: Printa & stripa now handle debug hunks.

Page 272:

There is a new packet action type: SetFileDate (= 34). You use this new packet type to set the date of a file or directory to a specified value. The first argument is a lock and the second argument is an APTR to an AmigaDOS date stamp returned by the DateStamp( ) routine.

There is another new packet action type: SetRawMode (= 994). This new mode is used to change the CON: device so that it behaves like the RAW: device, and to change it back to its normal state. Include the TRUE argument to select raw mode (so that it behaves like RAW:). Include the FALSE argument to return the console device to its normal state.

There is, however, a difference between the RAW: device and the CON: device in raw mode. CON: in either normal or raw mode accepts the escape sequences to enable and disable the conversion of a linefeed character to carriage return + linefeed. RAW: does not translate linefeeds. (The sequences are CSI 20h to enable translation and CSI 201 to disable it.)

AMIGA ROM KERNEL REFERENCE MANUAL

UPDATE FOR VERSION 1.2

Amiga ROM Kernel Reference Manual:
Update to the Addison-Wesley Edition for
Amiga System Software Release 1.2

Changes to Volume I: Exec

Page ix: Exec now supports the 68881 math coprocessor in a multitasking environment.

Page 25: Exec has new Signal Semaphore functions: InitSemaphore( ) and ObtainSemaphore( ). See exec.doc for more information.

Page 65: Support for the Amiga Expansion Architecture is now included in Exec. As a result, many system data structures, such as SysBase and GfxBase, should appear in fast memory if it is available.

Page 72: There are two new Exec functions that deal with memory. CopyMem( ) allows you to move memory quickly and safely. AddMemList( ) adds memory to the system. See exec.doc for more information.

Page 82: Exec can now find code modules in ram at system bring-up time. This new "romtags in ram" facility will allow developers to override individual Kickstart modules and add new modules.

Page A–2: Exec now displays different text in an alert if the alert is one that is recoverable.

Page B–1: Exec will ignore the old ExecBase if the version or revision of exec has changed.

Changes to Volume II: Libraries and Devices

Page iii: The library version number for this release is 33.

Page 3: The 1.2 system software now supports both the NTSC and PAL video standards. If a special custom chip for PAL has been installed in an Amiga, there are 256 horizontal lines on the display (512 if you've chosen Workbench Interlace in Preferences). You also get the 50 Hz refresh rate required by the PAL standard.

THIS RELEASE ALSO DETECTS THE FREQUENCY OF THE AC LINE CURRENT (EITHER 50 or 60 Hz) AND AUTOMATICALLY USES THE RIGHT FREQUENCY WHEN UPDATING THE AMIGA'S INTERNAL CLOCK.

Page 13: There is a new graphics function, SetRGB4CM( ).
It sets one color register for a ColorMap structure. This function can be used to set up a ColorMap before linking it into a viewport.
See graphics.doc for more details.

Page 14: There is a new graphics routine, SetRGB4CM( ).
It is similar to SetRGB4( ) except that it takes a ColorMap as a parameter instead of a ViewPort.

A note to programmers: don't appropriate the ColorTable pointer in ColorMap. Amiga reserves the right to use additional bits for the color map in future releases.

Page 26:

To maintain compatibility with different standard screen sizes, programmers should get the normal display dimensions of a screen from GfxBase–>NormalDisplayRows and GfxBase–>NormalDisplayColumns. For NTSC the height is currently set to 200 at power up and to 256 for PAL at power up.

There are also new values in GfxBase for the maximum value for row and column in a ViewPort structure. Programmers should get these maximum values from GfxBase–>MaxDisplayRows and GfxBase–>MaxDisplayColumns.

Programmers should use the NTSC and PAL bits in GfxBase to maintain compatibility with future changes. For 1.2, using these bits will yield either 262 or 312 lines per frame.

Page 42:

There is a new graphics macro, DrawCircle.
It calls DrawEllipse( ) with a centerpoint and radii to create a circular outline in a rastport using the current pen color, at the specified position. It clips the result in a layered rastport.
See graphics/gfxmacros.h for macro definition.

Page 42:

There is a new graphics function, DrawEllipse( ).
It draws an elliptical outline into a rastport using the current pen color, at the specified position. It clips in layered rastports.
See graphics.doc for more details.

Page 42:

There is a new graphics macro, AreaCircle.
It calls AreaEllipse( ) with a centerpoint and radii to create a filled circle in a rastport using current pattern, drawmode, and pen colors. User must have initialized valid TmpRas and AreaInfo structures. This routine may be integrated with calls to other AreaFill routines (e.g. AreaMove, AreaDraw, AreaEnd).
See graphics/gfxmacros.h for macro definition.

Page 42: There is a new graphics function, AreaEllipse( ).
It is an AreaFill routine that fills an ellipse in the specified rastport using current pattern, drawmode, and pen colors. User must have initialized valid TmpRas and AreaInfo structures. This routine may be integrated with calls to other AreaFill routines; (e.g. AreaMove, AreaDraw, AreaEnd).
See graphics.doc for more details.

Page 42: There is a new graphics function, BltBitMapRastPort( ).
It blits from source bitmap to destination rastport using specified minterm, clipping the result in a layered rastport.
See graphics.doc for more details.

Page 42: There is a new graphics function, BltMaskBitMapRastPort( ).
It blits from source bitmap to destination rastport with masking of source image, clipping the result in a layered rastport.
See graphics.doc for more details.

Page 43: The WritePixel( ) function now waits for the blitter to complete between each of the blits. It is a little slower as a result.

Page 45: The AreaEnd( ) function now returns 0 if no error occurred.

Page 48: RectFill( ) still has a bug in COMPLEMENT mode: It complements all bit-planes rather than only those planes in which the foreground pen is non-zero.

To maintain compatibility with software written under 1.1, THIS BUG WILL NOT BE FIXED.

Page 51: Using ScrollRaster( ) for simple_refresh layers scrolls the appropriate existing damage region.

Page 79:

There is a new graphics function: AttemptLockLayerRom( ).
It attempts to obtain a lock on a layer. It does not block if a lock cannot be obtained. It returns the result of the attempt to caller.
See graphics.doc for more details.

Page 81:

There is a new graphics function: AttemptLockLayerRom( ).
It tries to obtain a layer lock, but does not block if attempt was unsuccessful. It returns a boolean value to indicate success or failure in obtaining the lock.

Page 95:

There is a new graphics function, InstallClipRegion( ).
It installs a transparent Clip region in a layer. Subsequent graphics calls will be clipped to this region. It returns a pointer to the previous Clip region.
See layers.doc for more details.

Page 95:

BeginUpdate( ) now assumes it has enough memory until it discovers otherwise.

Page 96:

There is a new graphics function, ClearRectRegion( ).
It clips away any portion of the region that exists inside of the rectangle, and leaves the result in the region. It returns a code indicating whether it succeeded or failed to the to calling program.
See graphics.doc for more details.

Page 96:

There is a new graphics function, OrRegionRegion( ).
It performs the logical inclusive OR of two regions. It returns a code indicating whether it succeeded or failed to the to calling program, and leaves the result in the second region.
See graphics.doc for more details.

Page 96: There is a new graphics function, XorRegionRegion( ).
It performs the logical exclusive OR of two regions. It returns a code indicating whether it succeeded or failed to the to calling program, and leaves the result in the second region.
See graphics.doc for more details. graphics/AndRegionRegion

Page 96: There is a new graphics function, AndRegionRegion( ).
It performs the logical AND of two regions. It returns a code indicating whether it succeeded or failed to the to calling program, and leaves the result in the second region.
See graphics.doc for more details.

Page 106: Programmers who can expect a performance improvement of about 15% in code that uses Gels.

Page 108: Sprite movement is now restricted to the limits imposed by the Amiga hardware. As a result, the absolute starting position of a Sprite must not occur before line 20.

Page 195: The proper order of parameters for the SetFont( ) function should be SetFont(rp, font).

Page 201: There is a new font, Topazll, for use in interlaced screens.

Page 225: The audio device now resets audio channels whenever they are allocated instead of only when they are freed or stolen.

Page 263: The actual step rate for the trackdisk device is now down to 3.6msec. To learn how to fine-tune the step rate, see the descriptions of tdu_StepDelay,tdu_SettleDelay in trackdisk.h. These parameters are extremely hardware dependent! This means that values that may work for a 68000 may not work for a 68010 as well as a 68020 or a faster clock, etc.

Page 297: The new console device re-establishes a keymap conversion buffer size of 127. This means that keys can be mapped to longer strings.

Page 390: MARK/SPACE is now supported by serial device.

The serial device now waits 12 seconds, rather than 90 micro-seconds, before giving up on CTS/RTS handshaking.

Page 413: All printers now have a defined extended character set for characters with values greater than 159.

Page 418: The printer device now only reloads the printer dependent portion when necessary.

Page 418: The printer device now reinitializes the printer only when printer preferences have been changed.

Page 418: Tasks calling the printer.device work always put up error requesters in the workbench screen.

Page 420: The printer device only generates an aRIN (which produces a line-feed) if printer preference variables are changed.

Page 422: In the printer device, DISABLE/ENABLE now surround the printer status checking code.

Page 427: The printer device now supports PAL screen sizes.

Page 455: The Mathffp library now supports Ceil, Floor.

Page 474: IEEEDPFix( ) in mathieeedoubbas.library now returns 0 instead of the error value 107 when there is an overflow.

Page 483: Workbench arranges to have its autorequest messages be marked with its name ("Workbench") in the title bar.

Page 486: Workbench now looks for initialize as either "sys:system/initialize" or "sys:system/format"

Page D-48: The name DPI in GfxBase has been changed to DPM; it now represents DotsPerMeter for both X and Y values in displays. Use these values to calculate display aspect ratios. These values will be different for NTSC and PAL.

Page D-122: The Okimate-20 now supports colored text.

Page D-122: The driver for the Apple ImageWriter II printer now has improved characters in the extended character set.

Page D-122: There is a new driver for the IBM compatible Okidata 292.

NAME

DumpRPort — dump the specified RastPort to a graphics printer.

FUNCTION

Print a rendition of the supplied RastPort, using the supplied ColorMap, position and scaling information, as specified in the printer preferences.

IO REQUEST

io_Message	mn_ReplyPort set if quick I/O is not possible
io_Command	PRD_DUMPRPORT
io_Flags	IOB_QUICK set if quick I/O is possible
io_RastPort	ptr to a RastPort.
io_ColorMap	ptr to a ColorMap.
io_Modes	the 'modes' flag as from a ViewPort structure the upper word is reserved and should be zero
io_SrcX	the x offset into the RastPort
io_SrcY	the y offset into the RastPort
io_SrcWidth	the x size in the RastPort to be printed
io_SrcHeight	the y size in the RastPort to be printed
io_DestCols	. . .
io_DestRows	these two parameters describe the size of the area to print to on the printer, as described below.
io_Special	interpretation of Dest parameters: • if SPECIAL_MIL is set, then the associated parameter is specified in thousandths of an inch on the printer. • if SPECIAL_FULL is set, then the dimension is set to the maximum possible (as determined by the printer limits or the configuration limits, whichever is less). • if SPECIAL_FRAC is set, the parameter is taken to be a longword binary fraction of the maximum for that dimension. • if ASPECT is set, one of the dimensions may be reduced to preserve the aspect ratio of the print. • if all bits for a dimension are clear, the parameter is specified in printer pixels.

- if SPECIAL_DENSITY(1–4) is set, then the printer specific driver has the option of selecting a different dots per inch density (dpi) than the default one. As of this writing, the printer specific modules supporting this feature are the HP_LASERJET & the HP_LASERJET_PLUS. For these two printers, the densities are 75, 100, 150 & 300 dpi respectively. The HP_LASERJET always defaults to 75 dpi. The HP_LASERJET_PLUS defaults to 100 dpi if the preferences is set to DRAFT quality and 150 dpi with LETTER quality.
- if SPECIAL_CENTER is set, then the picture will be horizontally centered on the paper.

There exist rules for the interpretation of io_DestRows and io_Destcols that may produce unexpected results when they are not greater than zero and io_Special is zero. They have been retained for compatibility. The user will not trigger these other rules with well formed usage of io_Special.

HINTS

The printer selected in preferences must have graphics capability to use this command.

Color printers may not be able to print black and white or greyscale pictures — specifically, the Okimate 20 cannot print these with a color ribbon: use a black one.

If the printer has an input buffer option, use it.

If the printer can be uni- or bi-directional, select uni-directional; this produces a much cleaner picture and in some cases a faster printout.

Please note that the width and height of the printable area on the printer is in terms of pixels and bounded by the following:

a) WIDTH = (RIGHT_MARGIN — LEFT_MARGIN + 1) / CHARACTERS_PER_INCH
b) HEIGHT = LENGTH / LINES_PER INCH

For BGR printer support, the YMC values in the printer specific render.c functions equate to BGR respectively, i.e. yellow is blue, magenta is green, and cyan is red.

The special rules for io_DestRows and io_DestCols (WHICH TAKE EFFECT ONLY IF IO_SPECIAL IS 0) are:

a) DestCols>0 & DestRows>0 — use as absolute values. ie. DestCols = 320 & DestRows = 200 means that the picture will appear on the printer as 320 × 200 dots.

b) DestCols = 0 & DestRows>0 — use the printers maximum number of columns and print DestRows lines. ie. if DestCols = 0 and DestRows = 200 than the picture will appear on the printer as wide as it can be and 200 dots high.

c) Destcols = 0 and DestRows = 0 — same as above except the driver determines the proper number of lines to print based on the aspect ratio of the printer. ie. This results in the largest picture possible that is not distorted or inverted. Note: As of this writing, this is all the call made by such program as DeluxePaint, GraphicCraft, and AegisImages.

d) DestCols>0 & DestRows = 0 — use the specified width and the driver determines the proper number of lines to print based on the aspect ratio of the printer. ie. If you desire a picture that is 500 pixels wide and aspect ratio correct, use DestCols = 500 and DestRows = 0.

e) DestCols<0 or DestRows>0 — the final picture is either a reduction or expansion based on the fraction |DestCols| / DestRows in the proper aspect ratio.
Some examples:

1) if DestCols = −2 & DestRows = 1 then the printed picture will be 2× the AMIGA picture and in the proper aspect ratio. (2× is derived from |−2| / 1 which gives 2.0)

2) if DestCols = −1 & DestRows = 2 then the printed picture will be ½× the AMIGA picture in the proper aspect ratio. (½× is derived from |−1| / 2 which gives 0.5)

INTUITION
AMIGA SYSTEM SOFTWARE

UPDATE
FOR
VERSION 1.2

Intuition Version 33 (v1.2 Amiga System Software Release)

1. INTRODUCTION

This document describes changes and key observations for the revision to Intuition which is part of the v1.2 Kickstart or ROM software.

This exposition is directed at the software developer who is already familiar with the Intuition library, who has read the Intuition Reference Manual carefully in the areas discussed. Particular familiarity is assumed of the usage and methods of the Intuition Direct Communications Message Port (IDCMP).

The "autodocs" are a reference resource for Intuition and the entire system. Autodocs are function description sections, which are automatically extracted from the source itself. An entry in the Intuition autodocs (intuition.doc) might be written as "the BeginRefresh( )autodocs." A reference to a function description in the Layers library might be layers.library/BeginUpdate( ) or layers/BeginUpdate( ).

The autodocs appear in volume 2 of the Rom Kernel manual, but the most up-to-date versions are released on a disk with the developers' distribution of the system software.

Many technical aspects of Intuition relate closely to the Layers library, which supports clipping and saving of offscreen areas of windows. It is very useful to understand the basics of this library, but it should be rarely, if ever, used by an application program directly.

A word on the multi-tasking nature of Intuition might be helpful. As far as your program is concerned, Intuition has "two parts" in that Intuition library code may be run on two schedules: your program (assuming your program has a single task accessing Intuition) and the Input Device. The latter is a task that runs on the user-interaction schedule as an input handler (the entry point is called Intuition( )). It appears to your code almost as if it were running as an interrupt, since it runs at high priority and thus can preempt your task.

The synchronization of the input schedule and your program, and the arbitration of data structures shared by your program and the Intuition input handler are the two most delicate topics which are discussed below.

2. SCREENS

Height and Position

In support of PAL displays (which support more lines than their US counterparts), a method is provided to open screens to their full height on any display. This is done by specifying the value STDSCREENHEIGHT in the NewScreen.Height field (this constant is defined in intuition.h).

If you intend to provide your own CUSTOMBITMAP for the screen, you need to know how big the screen will be before opening it. This is done by examining the public field GfxBase->NormalDisplayRows. Note that that field specifies the non-interlaced count (on US machines it is 200).

A new feature: screens of less than full height are not constrained to the bottom portion of the display. When opened, screens will appear (safely) at the display position specified in NewScreen.TopEdge. The user can drag short screens up to the top of the display. The display area at the bottom of the display below the bottom of a short screen is solid in the background color of the front screen.

At the bottom of the display, beyond NormalDisplayRows, the image of the front-most screen is continued and may be visible. The mouse is constrained, however, so that it remains in the "working region" of the display View structure, which extends only from the top of the view to NormalDisplayRows.

Workbench Screen Inquiry

Programs opening windows on the Workbench Screen may inquire to find out its size, the size of its Menu bar area, and so on, by using the function GetScreen Data( ). This function will open the Workbench screen if it happens to be closed. This is best suited to use by a program about to open a window on the Workbench screen.

Workbench Screen Inquiry also applies to custom screens, but the need for that will be rare.

New Flags for NewScreen

Some new features are provided and invoked by setting new flags in the NewScreen .Flags field. These values are defined in intuition.h.

SCREENBEHIND — indicates that when the screen is opened, it should be behind all other screens. Among other uses, this method allows a program to prepare imagery in the screen, change its colors, and so on, bringing it to the front when it becomes presentable.

SCREENQUIET — addresses the problem of doing fancy viewpoint operations in your custom screen. A screen opened with this flag will not have a title bar nor visible gadget rendering, but the dragging and depth arrangement facilities provided by Intuition still function. In order to completely prevent Intuition from rendering into your screen, you must also catch the menu button in each window in the screen, using MENUVERIFY or RMBTRAP.

A Warning on Dual Playfields

Setting the DUALPF flag in the NewScreen.Flags field is not the best method of obtaining a dual playfield viewport for your screen. It is better to open a standard screen, passing to Intuition (or letting Intuition create) only one of your playfield bitmaps (the front one). Then you allocate and set up a second BitMap, its bit-planes, and a RasInfo structure. Install these into new screen's viewport, change the viewport modes to include DUALPF, MakeScreen( ), and RethinkDisplay( ). An example appears at the end of this document. This method keeps Intuition rendering (gadgets, menus, windows) in a single playfield.

3. WINDOWS

ActivateWindow( )

This function allows a program to activate windows, hopefully in a way that doesn't confuse or frustrate the user. An example of a system program using this function on a (long, narrow) window along with the related function ActivateGadget( ) (see below) is the Workbench Rename operation.

See the autodocs for this function, as its use is not entirely straightforward.

Window Borders

Window borders are no longer excessively updated when a window is made active, the window's title is changed, or the window is brought to the front. Window borders are refreshed, when necessary, through the window's layer's DamageList, which further mimimizes the visual activity (essentially under the auspices of Begin/ End Refresh( )).

A lot of attention has gone into establishing the correct damage region for a smart refresh window. It is now possible for application programs to refresh their smart windows in between BeginRefresh( ) and EndRefresh( ) and get good results. The refreshing of gadgets can be more complicated, and is covered below, in Section 3, Gadgets.

The drag gadget in the top border is rendered in a more solid pattern which appears more stable in an interlaced display.

Dimensions

The position/dimensions of windows when opened now undergo error checking. The maximum dimensions (specified as NewWindow.MaxWidth/MaxHeight) are now unsigned quantities, and may be legally set to a maximum by using the value $0 \times FFFF$, better expressed as (0).

Refreshing

Be sure to use Begin/EndRefresh( ) as directed in the manual, or set the NOCAREFRESH flag in NewWindow.Flags. When using Begin/EndRefresh( ), restrict your operations to simple rendering. Avoid calls that may lock the LayerInfo, or get complicated in Intuition, since BeginRefresh( ) leaves the window's Layer(s) locked. All rendering functions of Intuition or Graphics are OK. RefreshGadgets( ) is OK, but probably unnecessary. AutoRequest( ) is to be avoided, and therefore all disk related DOS calls.

A new function, RefreshWindowFrame( ), is provided in the event that your program has been trashing window borders and wants to clean up.

RMBTRAP

The RMBTRAP bit in Window.Flags may be modified on the fly by your program. To be entirely proper, it should be set or cleared as an atomic operation, which can be done in assembler, using a hot compiler, or by surrounding it with Forbid/Permit( ).

4. GADGETS

4.1 General

Gadget List Functions

The functions to add, remove, and refresh gadgets to windows or requesters: (AddGadget( ), RemoveGadget( ), and RefreshGadgets( )) have analogies which apply to lists of gadgets linked by the Gadget.NextGadget field. Each function (AddG List( ), RemoveGList( ), and RefreshGList( )) takes a parameter determining how many gadgets in a list are to be operated on. In each case, operation terminates if a NextGadget field contains NULL; a parameter value of -1 means that the entire list (until NULL) is processed.

The function NewModifyProp( ) has the same parameter, with regard to the old ModifyProp( ). The new function takes a gadget count parameter to allow finer control of the refreshing of gadgets. See Section 3.2, Gadget Rendering for the behavior of each of these.

Also refer to the autodocs for more information on these gadget functions.

ActivateGadget( )

A String (or Integer) Gadget may be "activated" under program control. If successful, this function has the same effect as the user clicking the SELECT button when the mouse pointer is within the bounds of the gadget. Subsequent keystrokes accomplish entry and editing on the gadget's string. This function will fail if the window or requester containing the gadget is not active. It will also fail if the user is in the middle of some other interaction, such as menu or proportional gadget operation. See the ActivateGadget( ) autodoc section.

Custom Screen Gadgets

These gadgets have never been supported, and are not in 1.2. The commercial version of the manual reflects this observation.

Gadget Flags

The TOPBORDER flag bit in Gadget.Activation can be set to indicate to Intuition that a gadget must be refreshed after Intuition has rendered in the top border area of a window.

The REQGADGET flag (for Requester Gadgets) must reflect the truth. As an assistance, Intuition sets or clears this flag appropriately when gadgets are added or windows/requesters are opened.

The SELECTED flag is now set even for system gadgets when they are in use. We don't advise you rely heavily on such state flags of data structures that are not assigned to your program.

4.2 Gadget Refreshing by Intuition

Gadgets are refreshed by Intuition whenever a layer operation has damaged the layer of the window or requester to which they are attached. In the processing of the REFRESHWINDOW message, the typical program needn't call RefreshGadgets( ) at all.

Intuition's refresh of the gadgets of a damaged layer is done "Through the Layer's Damage List." This means that rendering is clipped to the layer's Damage Region — the part of the window's layer which needs refreshing because it has been exposed by a layer operation.

To be precise, Intuition calls layers/BeginUpdate( ) and EndUpdate( ) (Intuition user (i.e., your) equivalents are Begin/EndRefresh( )) so that rendering is restricted to the Region Layer.DamageList (see Windows, above, and the Begin/EndRefresh( ) function descriptions in the Intuition manual, and layers/BeginUpdate in the autodocs).

Gadgets which are positioned (GRELBOTTOM GRELRIGHT) or sized (GRELWIDTH, GRELHEIGHT) relative to the dimensions of their window pose a problem when the window is sized, since the visuals for these gadgets must change, and are not necessarily in the damage region.

Therefore, Intuition must add the original and new visual regions for such relative gadgets to the Damage Region before it goes about refreshing gadget rendering.

Relative gadgets that have Border, Image, of Intuitext imagery extending beyond their respective select boxes should be refreshed by the program itself (using RefreshGlist( )) when receiving a Newsize event for the window. This gadget refreshing should not be done between BeginRefreshing( ) and EndRefresh( ), in order to present the visuals beyond the extent of the damage region.

4.3 Gadget Refreshing by Your Program

If you add gadgets to your window or requester (using AddGlist( ), or AddGadgets( )) you must subsequently call RefreshGList( ) or RefreshGadgets( ) to get the image of your gadget drawn.

Several calls actually cause the refresh of more than one gadget. The behavior under v1.1 of these functions has been unchanged in v1.2 This was not true for several beta releases. In each case, an alternative function or method is available for new programs to better control the refreshing of gadgets so that only modified gadgets need be redrawn.

The functions which refresh gadgets in v1.1 are: RefreshGadgets( ), OnGadget( ), OffGadget( ), and ModifyProp( ). Their behavior and alternatives are described below:

Refresh Gadgets( )

In v1.1, this function is documented as refreshing all gadgets from the one whose address is passed as a parameter to the end of the list (linked by Gadget.NextGadget). If applied to a requester gadget, though, ALL gadgets in the requester were redrawn. This is the case in (the final releases of) v1.2, for compatibility.

The alternative is to use RefreshGList( ), which takes a parameter to determine exactly which gadgets are to be refreshed.

ModifyProp( )

This function simply modifies some values in the PropInfo structure of a proportional gadget and calls RefreshGadgets( ), the excessive behavior of which is described above.

The alternative is to call NewModifyProp( ). This new function takes a parameter in the manner of RefreshGList( ) to control the gadget refreshing activity.

OnGadget( ) and OffGadget( )

These functions simply clear or set (resp.) the GADGDISABLED flag of the gadget, and then call RefreshGadgets( ). There are no alternative functions for these, since they each can be implemented by manually modifying the GADGDISABLED flag and calling RefreshGList( ), as described below.

Some programs use RefreshGadgets( ) (better, RefreshGList( )) to update the display after they have made state changes to the gadgets. The types of changes include: the

SELECTED flag for Boolean Gadgets to implement home-rolled MUTUALEXCLUDE for Boolean Gadgets, the GadgetText of some gadget to change its label, the GADGDISABLED flag, and the contents of StringInfo.Buffer of a String Gadget. (See Section 3.6.1 Mutual Exclusion for an example.) On this subject, we offer the following:

> Be sure to RemoveGadget( ) any gadget before altering it. Understand that Boolean Gadgets rendered with Borders (instead of Images) or highlighted with surrounding boxes (GADGHBOX) are handled very simply by Intuition, and that complicated transitions done by your program (and in some cases the user's own actions) can get the rendering out of phase.

4.4. String Gadgets

The most important change in string gadgets is the ActivateGadget( ) function described under GADGETS.

When the user selects a string gadget with the mouse, the gadget's cursor moves to the position of the mouse.

Several bugs have been fixed. It is now possible to use RELWIDTH string gadgets. Integer (LONGINT) string gadgets now have the LongInt value updated whenever the textual contents of the gadget changes, and again, for good measure, when the gadget is de-activated.

4.5. Proportional Gadgets

There is a new function NewModifyProp( ) which is the same as ModifyProp( ) with additional control over gadget refreshing. See the sections above for details.

Highlighting by alternate knob image (GADGHIMAGE) is now supported, but you should make the alternate image the same size as the normal knob image.

Proportional gadgets are often used for scrolling graphics or textual information. To do this in your program (assuming vertical scrolling), it is necessary to convert the relationships between the visible fraction of the display and the VertBody value, and the top line of the display and the VertPot value.

The first relationship is simple: make the Body value represent "one-fourth." This can be expressed as

```
VertBody = (ULONG) (visible_lines * 0xFFFF)/total_lines;
```

If you are hard-up for cycles and can avoid zero visible_lines, this seems to work nicely:

```
VertBody = ((ULONG)(visible_lines<<16)-1)/total_lines;
```

Now, note that when you are displaying the last "page," line 75 is the top line in view (start line numbers from zero).

Therefore, when the VertPot value is a maximum (0xFFFF) you need to convert this to 75, and so on. This can be done like so:

```
top_line = ((ULONG)(total_lines-visible_lines)*VertPot+(1<<15))>>16;
```

The inverse of this operation is needed if you wish to initialize the pot for some particular value of top_lines.

```
VertPot = MIN(0xFFFF, ((top_line_input<<16))/(total_lines-visible_lines));
```

The MIN macro is used because "unity" is actually 0xFFFF.

4.6. Boolean Gadgets

Some minor bug fixing: Gadget Text may be used with gadgets highlighted by Alternate Image display (GADGHIMAGE). Selecting gadgets while other Intuition display activity is going on no longer confuses the system.

4.6.1. Mutual Exclusion

Intuition managed Mutual Exclusion of Boolean Gadgets is still not provided; rather, a flexible method of doing it yourself is recommended:

Remove a Boolean gadget from the window or requester it is attached to (RemoveGadget( )). Set or clear the SELECTED flag to reflect the state of the gadget you desire to display to the user. Replace the gadget (AddGadget( )) and refresh its imagery (RefreshGList( )).

More than one gadget can be processed using RemoveGList( ) and AddGList( ), with a single call to RefreshGList when done.

It is strongly recommended that you play these games only with Boolean gadgets rendered with images and highlighted by complement mode rendering. Further, the Activation Type should be GADGIMMEDIATE (and not RELVERIFY, nor TOGGLE-SELECT) with your state changes executed upon receiving the GADGETDOWN message.

4.6.2. Masked Gadgets

A new feature allows non-rectangular Boolean gadgets, with some restrictions. An auxilliary bit plane called a Mask is associated with a gadget. When the user selects within the select box of the gadget, a further test is made to see if the selection point is contained in the mask. Only if so does the interaction count as a gadget "hit".

If the gadget has highlight type GADGHCOMP the complement rendering is restricted to the mask, which allows, for example, an oval gadget which highlights nicely, only within the oval.

However, there are some shortcomings. The gadget image is not rendered through the mask. For example, in the case of an oval mask the image is still a rectangle, and when it is displayed; it will clobber the corner areas even though they are outside of the oval. This means they can't be crowded together without care.

Likewise, the ghosting of a disabled gadget does not respect the mask, so ghosting of the corners around an oval may be visible, depending on the colors involved.

An example appears at the end of this note.

5. MENUS

5.1. Flags

MENUTOGGLE

This flag can be set by your program, and now behaves as the Intuition manual describes. The author apologizes for not finding the bug in time to get this into 1.1. Set this flag for a CHECKIT menu (sub-)item and the item can be selected to turn the checkmark off, as well as on.

MENUSTATE

This flag is set in Window.Flags when the menus of that window are in use. Beware: in typical event-driven programming, such a state variable is not on the same time-table as your input message handling, and should not be used to draw profound conclusions in your program. It is better to synchronize yourself with the menu handling using MENUVERIFY.

5.2. Shortcuts and MENUVERIFY

The idea behind MENUVERIFY (and to some degree SIZEVERIFY and REQVERIFY) is to synchronize your program with Intuition's menu handling sessions. The motivating reason was to allow your program to arbitrate access to your screen's bitmap, so that Intuition doesn't put menus in the way of your drawing. This circumstance is obviated to some extent by the fact that Intuition will lock all layers on a screen before any menus are displayed, and if your rendering is solely through windows you will be automatically locked out until menus are done.

Some programs use MENUVERIFY to permit them to intercept the right mouse button, thereby to use it for their own purposes. Other programs use MENUVERIFY to briefly suspend menu operations while they restore Wild Phenomena before menu operations proceed. These phenomena may be illegible colors of the screen or double buffering and related ViewPort operations.

In any case, it is vital to know when menu operations terminate. This is typically detected by watching for the MENUPICK IDCMP message. If you intercepted (MENU-CANCEL) the menu operations, you will instead receive MOUSEBUTTONS message with code equal to MENUUP. Menu shortcut keystrokes, for compatibility, also respect MENUVERIFY. They are always paired with a MENUPICK message (unlike v1.1) so that your program knows the menu operation is over.

Use of MENUVERIFY was recommended only with a healthy set of caveats, and with a publicly available program you needed named KillVerify( ), simply because there was not deadlock-safe way to turn MENUVERIFY off. Now you may call ModifyIDCMP( ) with confidence to turn MENUVERIFY and the other VERIFY IDCMP options off. It is important that you do so if you ever do anything that directly or indirectly has you waiting for Intuition (since Intuition may be waiting for you).

Intuition now handles its internal functions automatically, so you can safely open and close windows, change your menus (Clear/SetMenuStrip), and play with your gadgets.

You cannot, however, wait for a gadget or mouse event without checking also for any MENUVERIFY event messages that may require your response.

The most common problem area is System Requesters (AutoRequest( )). Before AutoRequest( ) returns control to your program, Intuition must be free to run and accept a response from the user. If the user presses the menu button, Intuition will wait for you to MENUVERIFY and there is a deadlock.

THEREFORE: USE THE FUNCTION ModifyIDCMP( ) TO TURN OFF ALL VERIFY MESSAGES BEFORE YOU CALL AutoRequest( ) OR THE DOS(!!!) SINCE MANY ERROR CONDITIONS IN THE DOS REQUIRE USER INPUT IN THE FORM OF AUTOREQUESTS.

5.3. Miscellaneous

Highlighting

Item and SubItem highlighting by Alternative Text display now works. The item must be a text item (ITEMTEXT) and the highlighting flag is set to HIGHIMAGE. MenuItem.SelectRender must, of course, point to an IntuiText structure.

Menu shortcut keys 'M' and 'N' now work.

Try not to leave space between the select boxes of your menu items and (especially) your subitems. When the pointer moves off of one subitem into the gap between it and the next subitem, the entire submenu is erased and redrawn, which doesn't look real sharp.

6. REQUESTERS

Pointer Relative Requesters

By setting the POINTREL Flag in a Requester, and installing the requester as a Double Menu Requester (Set/ClearDMRequest( )), you get a requester that comes up under the mouse pointer when the user double clicks the right mouse button.

The values of the fields Requester.RelLeft/Top determine the point on the requester the mouse will be over (experiment). The requester position will be restricted in an attempt to make it entirely visible (i.e., contained in its window).

Noisy Requesters

You may set the NOISYREQ flag if you do not want the presence of a Requester to inhibit your normal input flow. Gadgets and menus will remain activated, but you will be able to hear the keyboard and the mouse. This technique is used in the new keyboard feature of AutoRequest( ) (see below).

Flags

The REQACTIVE flag bit (in Requester.Flags) is now always set and cleared as your requesters are posted and removed. The active requester has always been indicated by the value of Window.FirstRequest.

System Requests (AutoRequest( ))

By using the new NOISYREQ Requester Flag, AutoRequest( ) now allows the user to satisfy a system request from the keyboard. The key strokes left-Amiga-V and left-Amiga-B correspond to the left and right system request gadgets, respectively.

When a System Request is posted (using AutoRequest( ) or BuildSysRequest( )) it will move the screen it appears in to the front (if it is not already the foremost). This change was made after too many users were rebooting their machines when programs that used custom screens appeared to "lock-up" — due to the notice DOS put up on the workbench screen. Now these users need to learn to re-arrange the screens after they have satisfied a Request which has pre-empted their work, since the screens are not re-arranged to their original state.

User Rendering

A Requester appears in a Layer. You may render to this layer through its RastPort, which can be found as Requester.ReqLayer.rp. The Requester layer is smart, so that your rendering is preserved, but if the window is sized it may damage your work, so that you will need to refresh your rendering.

IDCMP Changes

The REQSET and REQCLEAR messages are now sent for each Requester that is put up or removed (resp.) instead of only for the first and last (resp.) This change provides more information for those programs trying to use more than one requester, especially if they are also trying to be tricky with ActiveGadget( ).

7. ALERTS

If a recoverable Alert cannot be displayed (because of low-memory), DisplayAlert( ) will return FALSE, as if the user had selected CANCEL. Formerly, the Alert would be converted into a dead-end alert (Guru Meditation) claiming that Intuition was in a panic, and that just didn't seem fair. Note that a System Request (AutoRequest( ) or BuildSystemRequest( )) will convert into a recoverable alert, so this change means that numerous errors in low memory conditions which appeared to crash the machine won't be doing so in 1.2.

The part of the recoverable alert message which says "RIGHT BUTTON CANCEL" is displayed in 1.2, fix of a bug in 1.1.

8. LOCKING AND RE-ENTRANCY

General Locking

The multi-tasking access of Intuition data structures is now arbitrated by the use of ExecSignalSemaphores (see the autodocs). This should eliminate incidences of clashing use of windows, Intuition linked lists, the dreaded busy WaitForbid( ), Screen RastPort confusion, and the like. In the alpha and beta stages, these problems were all replaced with system deadlocks, at one time or another, and finally were all eliminated in the final versions.

The application programmer's exposure to Intuition locking is minimized. A pair of entry points named LockIBase( ) and UnlockIBase( ) are provided to gain access to locks which assure the integrity of the state of IntuitionBase (the intuition.library static data) and of the linked lists of Screens and Windows. The autodoc sections for these functions explains in great detail the intended use and cautions pertaining to these functions. In particular, be VERY careful that you pass the parameters described, or you may be opening vulnerabilities in the system that will not appear as problems until your program is running in a heavily-laden multi-tasking environment.

The author-programmer makes this request: don't get tricky with these locks.

Intuition's Use of Your RastPort

Intuition has many rendering chores: screen and window titles and borders, gadgets, menus, and so on. In v1.2, Intuition will use a copy of the RastPort of the screen in which the rendering is to take place. This copy will determine the bitmaps the rendering will end up in, and often the font and similar modal information.

One thing Intuition sets each time is the mask value of the RastPort. It is set to all ones (0xFFFF). If you wish to restrict Intuition's rendering to all bitplanes of your screen, you may change the Depth and Planes values in Screen.RastPort.BitMap. This will only affect rendering into the screen itself, which consists of the Screen title and gadgets, and menus. Window gadgets are not fooled, since they use the mask in the window's layer's rastport, which you should not be changing.

9. IDCMP and INPUT EVENTS

Keyboard Messages

VANILLAKEY IDCMP Class (which does not appear in the original version of the manual) has not been enhanced from 1.1, for compatibility. It provides a translation from RAWKEY input event using the system default keymap, but only sends an IDCMP message if the translation results in a single byte (which is passed in the Code field of the VANILLA-KEY IDCMP message).

Most programs will prefer to use the RAWKEY IDCMP class, and perform their own RawKeyConvert( ). See also DeadKeyConvert( ), below.

Dead Keys

The IntuiMessages are converted from InputEvents. InputEvents now (in v1.2) contain information for the processing of so-called Dead Keys, which refers to a technique of generating a diacritical using two separate keystrokes. (See release notes on keymaps.) This information is passed in RAWKEY IDCMP messages in a location pointed to by the IntuiMessage.IAddress field. The following example shows how to use this information and the ConsoleDevice function RawKeyConvert( ) to convert RAWKEY messages into character values.

```
/* must have the ConsoleDevice opened to use RawKeyConvert( ) */
struct Device *ConsoleDevice; /* external declaration */
struct IOStdReq ioreq;
```

```
...
OpenDevice("console.device", -1, &ioreq, 0);
ConsoleDevice = ioreq.io_Device;

...

/* DeadKeyConvert( )
* returns:
* -2 if msg is not class RAWKEY
* same as RawKeyConvert otherwise:
* buffer length if <= kbsize
* -1 else
*/
DeadKeyConvert(msg, kbuffer, kbsize, kmap)
struct IntuiMessasge *msg;
UBYTE*kbuffer;
int kbsize;
struct KeyMap *kmap;
{
   static struct InputEvent ievent = {NULL, IECLASS_RAWKEY, 0, 0, 0};

   if (msg->Class != RAWKEY) return (-2);

   /* pack input event */
   ievent.ie_Code = msg->Code;
   ievent.ie_Qualifier = msg->Qualifier;

   /* get previous codes from location pointed to by IAddress
   * this pointer is valid until IntuiMessage is replied.
   */
   ievent.ie_position.ie_addr = *( (APTR *)msg->IAddress);
   return (RawKeyConvert(&ievent, kbuffer, kbsize, kmap) );
}
```

Verify Messages

See the discussion in Section 4, Menus on the MENUVERIFY IDCMP Flag. All of that information applies to the two other verify messages: REQVERIFY AND SIZEVERIFY. A special note: a bug somewhere in the input handling stream through Intuition may confuse window sizing by the user if your program takes too much time responding to the SIZEVERIFY message. Please be quick.

MOUSEMOVE

Your program will not be sent MOUSEMOVE messages while Intuition has the layers of your screen locked (during menu operations and window sizing/dragging). This avoids problems of messages accumulating while your program is blocked trying to render to a layer which Intuition has locked.

Bug Fixes

DELTAMOVE messages work. Be advised that if you have a DELTAMOVE IDCMP flag set, your MOUSEBUTTONS messages will also have relative values, instead of the absolute window position of the mouse.

NEWPREFS messages are sent if the NEWPREFS IDCMP flag is set. Sending it used to be determined by some other flag values, by coincidence.

Duplicate mouse messages for users of RMBTRAP are no longer sent.

The INTUITICKS messages are paced: until you reply to one, no subsequent one will be sent to you. This was not true in v1.1, but is fixed in 1.2. This has the effect of making the messages appear to arrive half as often, so programs using INTUITICKS to drive animations appear to run slower.

A NEWSIZE message will be sent when sizing are operations completed, even if the size of the window did not actually change. This is the way of 1.1, but was missing from some of the development releases of v1.2.

10. MISCELLANEOUS

Pointer Position Error

There was a bug in v1.1 graphics/MoveSprite( ) and resultingly in intuition/SetPointer ( ). For compatibility, we are forced to leave it in. The error is in Sprite positioning. To compensate for it, you must tell Intuition that the "hot spot" of a pointer sprite is one pixel to the left of the position actually intended. The Preferences pointer editor adds this fudge factor; only changes to the pointer done by your program must compensate for this error.

Input Event Food Chain

The Intuition input handler now no longer discards linked lists of input events longer than 10 in length, and can handle linked lists of input events in string gadgets. This is only important to programs which add input events to the Input Device stream either using their own handler or by using the IND_WRITEEVENT command of the Input Device (see the documentation on the Input Device).

Absolute Mouse Position

Intuition now handles pointer position input events. The input event class IECLASS _POINTERPOS can be used to specify a position for the mouse pointer RELATIVE TO THE INTUITION VIEW ORIGIN. The coordinates are provided in the pseudo-fields ie_X and ie_Y (these are really part of a union). Internally, Intuition will convert this event into the proper RAWMOUSE event, replacing ie_X/Y with a suitable relative mouse motion.

Note that the presence of the qualifier IEQUALIFIER_RELATIVEMOUSE indicates that an IECLASS_POINTERPOS input event specifies a relative mouse move, having the same effect as IECLASS_RAWMOUSE

PublicIntuitionBase

They include file intuition/intuition base.h contains a definition of the entire Intuition library data structure. It also contains very important warnings and specification of what part of this READ-ONLY data structure may be used in a supported way, and explains the role of the functions LockIBase( ) and UnlockIBase( ) in accessing the data it contains.

InstallClipRegion Support

This new function in layers.library allows specifying a region within a window to which rendering is clipped. Intuition needs to modify and restore this information to perform its gadget rendering and border refreshing, and does so in such a way as to be transparent to your program, in simple circumstances.

ReportMouse( )

This function had a problem in v1.1 that caused its parameters to be handled in the order opposite of that described in the Intuition manual. For a full explanation of the current state of this function, and an interesting lesson in software maintenance, see the autodoc section for this function.

PrintIText( )

This function now safely handles the case of the actual text pointer--IntuiText.IText--being NULL. This routine is used internally for all Intuition rendering of IntuiText structures in menus and gadgets.

Memory

Menu operations (to name one thing) handle low-memory conditions better than in v1.1. Specifically, menu and submenu pages will not be rendered when insufficient memory exists for the internal auxilliary bitmaps.

Display Beep( )

This function was not re-entrant in v1.1, and could leave the screen in a "highlighted" state. No more.

Mouse X and Y

These fields in Screens and Windows are now properly initialized when a screen or window is opened.

Fonts

Intuition now closes the fonts it opens. The font opened when a window is opened is stashed in the new Window field Window.IFont. Intuition will close this font when the window is closed, asnd won't try to close any font you have set in your window's rastport. You must close such fonts yourself.

11. DUAL-PLAYFIELD SCREEN EXAMPLE

```
*****************************************************************************************
* Example program for Dual-Playfield Screens
*
* copyright Commodore-Amiga, Inc., Sept. 1986. use at will
* author: jim mackraz (amiga!jimm)
*****************************************************************************************

* Turn the workbench into dual playfield.
*
```

```
* You can use the same trick for your own screens,
* which is the recommended method for creating dual-playfield
* screens. (Please don't really do this to the Workbench.)
*
* -Start with a new, single-playfield screen
* (don't set DUALPF in NewScreen.ViewModes)
* -Allocate a second playfield, set up a rastport for
* rendering into it, and install it into your open screen
* as shown here. Intuition will never know about or use your
* second playfield for its rendering (menus, gadgets, etc.).
* -Be sure to remove evidence of your deed before CloseScreen( ).
*/
#include <exec/types.h>
#include <exec/memory.h>
#include <intuition/intuition.h>

#define printf kprintf

struct Remember      *rememberkey = NULL;
struct Window      *getNewWind( );

struct IntuitionBase      *IntuitionBase;
struct      GfxBase      *GfxBase;
ULONG   flg = ACTIVATE | WINDOWCLOSE | NOCAREREFRESH | WINDOW-
            DRAG | WINDOWDEPTH | SIMPLE_REFRESH;

ULONG   iflg = CLOSEWINDOW;

main( )
{
   struct IntuiMessage   *msg;
   struct Window *window = NULL;
   WORD   exitval = 0;

/* hold data from *msg */
   ULONG class;
```

```
/*specific for this test */
struct Screen *wbscreen;
struct RasInfo *rinfo2 = NULL;  /* second playfield rasinfo ... */
struct BitMap *bmap2 = NULL;  /* ... and bitmap          */
struct RastPort *rport2 = NULL; /* for rendering into bmap2*/
BOOL it_is_done = FALSE;        /* success flag            */
int        counter = 0;         /* for timing the visuals      */

if (!(IntuitionBase = (struct IntuitionBase*)
   OpenLibrary("intuition.library",0) ) )
{
  printf("NO INTUITION LIBRARY 0);

   exitval = 1;
   goto EXITING;
}

if (!(GfxBase = (struct GfxBase *)
   OpenLibrary("graphics.library",0)))
{
   printf("NO GRAPHICS LIBRARY0);
   exitval = 2
   goto EXITING;
}

/* get a window on the workbench */
window = getNewWind(320, 20, 300, 50, flg, iflg);
if (window = = NULL)
{
   printf("test: can't get window.0);
   exitval = 1;
   goto EXITING;
}

/*------ Add a second playfield for Workbench ---------- */

wbscreen = window->WScreen;          /* find it */
```

```
/* allocate second playfield's rasinfo, bitmap, and bitplane */
if (!rinfo2 = (struct RasInfo *)
   AllocMem(sizeof(struct RasInfo), MEMF_PUBLIC | MEMF_CLEAR) ) )
{
   printf("alloc rasinfo failed0);
   goto EXITING;
}

if (!(bmap2 = struct BitMap *)
   AllocMem(sizeof(struct BitMap), MEMF_PUBLIC | MEMF_CLEAR) ) )
{
   printf("alloc bitmap failed0);
   goto EXITING;
}

InitBitMap(bmap2, 1, wbscreen->Width, wbscreen->Height);

/* we'll use 1 plane.*/
if (!(bmap2->Planes[0] =
   (UWORD *) AllocRaster (wbscreen->Width, wbscreen->Height) ) )
{
   printf("alloc raster failed0);
   goto EXITING;
}

/* get a rastport, and set it up for rendering into bmap2 */
if (!(rport2 = (struct RastPort *)
   AllocMem(sizeof (struct RastPort), MEMF_PUBLIC) ) )
{
   printf("alloc rastport failed0);
   goto EXITING;
}
InitRastPort(rport2);
rport2->BitMap = bmap2;

SetRast(rport2, 0);

/* manhandle viewport: install second playfield and change modes      */
Forbid( );
```

```
rinfo2->BitMap = bmap2;      /* install my bitmap in my rasinfo      */

wbscreen->ViewPort.RasInfo->Next = rinfo2;
          /* install rinfo for viewport's second playfield      */

wbscreen->ViewPort.Modes | = DUALPF;
                    /* convert viewport          */
it_is_done = TRUE;

Permit( );

/* set my foreground color */
SetRGB4(&wbscreen->ViewPort, 9, 0, 0×F, 0);
/* color 9 is color 1 for second playfield of hi-res viewport */

/* put viewport changed into effect */
MakeScreen(wbscreen);
RethinkDisplay( );

drawSomething(rport2);

printf("test program ok0);

FOREVER
{
  if ((msg = (struct IntuiMessage *)GetMsg(window->UserPort)) = = NULL)
  {
    Wait(1<<window->UserPort->mp_SigBit);
    continue,
}

  class = msg->Class;
  ReplyMsg(msg);
    switch (class)
    {
    case CLOSEWINDOW:
      printf("event CLOSEWINDOW0);
      goto EXITING;
    default:
      printf("unknown event: class%1×0, class);
```

```
    }
  }
EXITING:
  /* clean up dual-playfield trick*/
  if (it_is_done)
  {
    Forbid( );
    wbscreen->ViewPort.RasInfo->Next = NULL;
    wbscreen->ViewPort.Modes & = DUALPF;
    Permit( );
    MakeScreen(wbscreen);
    RethinkDisplay( );
  }

  if (rport2) FreeMem(rport2, sizeof (struct RastPort) );
  if (bmap2)
  {
    if (bmap2->Planes[0])
    {
      FreeRaster(bmap2->Planes[0], wbscreen->Width, wbscreen->Height);
    }
    FreeMem(bmap2, sizeof (struct BitMap) );
  }
  if (rinfo2) FreeMem(rinfo2, sizeof (struct RasInfo) );

  if (window) CloseWindow(window);
  if (GfxBase) CloseLibrary(GfxBase);
  if (IntuitionBase) CloseLibrary(IntuitionBase);

  exit (exitval);
}

drawSomething(rp)
struct RastPort *rp;
{
  int width, height;
  int r, c;

  width = rp->BitMap->BytesPerRow * 8;
  height = rp->BitMap->Rows;
```

```
    SetAPen(rp, 1);

    for (r=0; r<height; r+=40)
      for (c=0; c<width; c+=40)
      {
        Move(rp, 0, r);
        Draw(rp, c, 0);
      }
}

struct Window * getNewWind(left, top, width, height, flg, iflg)
SHORT left, top, width, height;
ULONG flg, iflg;
{
    struct Window *OpenWindow( );
    struct NewWindow nw;

    nw.LeftEdge = (SHORT) left;
    nw.TopEdge = (SHORT) top;
    nw.Width = (SHORT) width;
    nw.Height = (SHORT) height;
    nw.DetailPen = (UBYTE) -1;
    nw.BlockPen = (UBYTE) -1;
    nw.IDCMPFlags = (ULONG) iflg;

    nw.Flags = (ULONG) flg;

    nw.First Gadget = (struct Gadget *) NULL;
    nw.CheckMark = (struct Image *) NULL;
    nw.Title = (UBYTE *) " DUAL Playfield Mole ";
    nw.Screen = (struct Screen *) NULL;
    nw.BitMap = (struct BitMap *) NULL;
    nw.MinWidth = (SHORT) 50;
    nw.MinHeight = (SHORT) 30;
    /* work around bug */
    nw.MaxWidth = (SHORT) nw. Width;
    nw.MaxHeight = (SHORT) nw. Height;
    nw.Type = (USHORT) WBENCHSCREEN;
```

```
  return ((struct Window *) OpenWindow(&nw));
}
```

12. MASKED BOOLEAN GADGET EXAMPLE

```
/*****************************************************************************************
* Example code fragment for Masked Boolean Gadget
*
* copyright Commodore-Amiga, Inc., May 1986. use at will
* author jim mackraz (amiga!jimm)
*****************************************************************************************/
/* come up with a mask of your choice */
extern UWORD testgmask[];      /* single Image plane, used here as both as mask and
image */
/* you probably want a fancier (multi-color) image */
struct Image testgimage = {
   0, 0, 32, 32, 1,
   testgmask,   /* using same bit-plane for image as for mask */
   0x012, 0x00, NULL
   };

struct BoolInfo testboolinfo = {
   BOOLMASK,/* this extension is for masked boolean values */
   testgmask, /* a single bit plane */
   0      /* reserved */
};

struct Gadget testgadget ={
   NULL, 50, 50, 32, 32,
   GADGHCOMP | GADGIMAGE | SELECTED,
   GADGIMMEDIATE | RELVERIFY | TOGGLESELECT | BOOLEXTEND, /* NEW */
   BOOLGADGET,
   &testgimage,
   NULL,
   NULL, /* text */
   0,          /* mutual exclude */
   &testboolinfo,              /* NEW */
   123,
   NULL
};
```

INTRODUCTION TO PC UTILITIES

NEW FOR VERSION 1.2

Introduction to PC Utilities

The PC Utilities software is contained on the Version 1.2 Extras disk. This software requires an Amiga 1020 5¼ inch drive or an equivalent 5¼ drive.

PC Utilities

There are two tools associated with the PC Utilities release, PCFormat and PCCopy. Both make use of A-1020 5¼" disk drives that are attached to your system. If you Mount these disks, i.e. tell AmigaDOS about them, they cannot be used by these utilities. If you get a message telling you that 5¼" drives are not available, ensure that they are plugged in, turned on, and that your start-up sequence does not perform a Mount command for them. Naturally, only one 5¼" drive is required.

PCFormat

All disks used with the PCCopy utility need to be correctly formatted. This utility will format them for you. Usually the default settings are what you want; it's nice to include a Volume name. Do not hit the Format gadget without a disk in the drive. You may Abort as soon as the disk starts Verifying, if you need to save time.

PCCopy

PCCopy can be invoked with an argument that describes the Amiga file or directory to use. The project ToPCCopy invokes the PCCopy tool with that different initial direction of the copy. It works because of the line FROM = Amiga in the Tool Types.

When copying vanilla ASCII files between the Amiga and the PC, take advantage of the text filtering gadget. Text-7 and Text-8 differ in that Text-7 ensures that the data does not have the high bit in the byte set. Both convert carriage-return linefeed pairs on the PC to linefeeds on the Amiga, or vice versa when copying to the PC.

To create subdirectories on the PC, include them in the path of the file you wish created. Subdirectories on the Amiga cannot be created by this means: you must duplicate an Empty directory from Workbench, or use MakeDir from the CLI.

Whenever PCCopy informs you of an error, it waits for you to hit the Abort gadget before continuing.

To exit PCCopy, hit the Cancel gadget or the window's Close box.