PROGRAMMER'S MANUAL

P·I·X·A·R

REFERENCE GUIDE

MEMO: Pixar Manual Pages
TO: Pixar Customers
FROM: Pixar Documentation Group
DATE: December 2, 1986

Welcome to the Pixar *man* pages. We have tried to match this book to the Pixar Software Release 1.2 and make it easy to use. It is modeled on the UNIX documentation for *man* pages. Each *man* page describes one or more routines related to each other. Use the permuted index to find the page that holds a specific routine.

TABS: You will find two kinds of tabs to help divide the tradition UNIX sections into subsections. The major tabs have the familiar meaning, while the minor tabs correspond to subsections (e.g., libraries, etc.).

PAPER: Grey paper denotes material (table of contents, permuted index, etc.) that will direct you to the pages in each section. The pink pages at the end of the book are for your comments.

DATES: Each manual page bears its own date of last revision on the bottom.

PATHS: Pixar manual pages are located in */usr/pixar/man*. To get on-line manual help (*man*) for *xxxx*, type:

man -P /usr/pixar/man xxxx

Or you can save typing by using:

alias pixman "man -P /usr/pixar/man"

Then simply type:

pixman xxxx

(The above *alias* can also be put in the *.cshrc* file in your home directory to have the *pixman* command available permanently.)

BUGS: Mail in the pink comment forms, or use electronic mail to submit on-line comments and suggestions (e.g., *mail pixar!bugs*).

NAME

README – introduction to Pixar Manual Pages

DESCRIPTION

Welcome to the Pixar *man* pages. We have tried to match this book to the Pixar Software Release 1.2 and make it easy to use. It is modeled on the UNIX documentation for *man* pages. Each *man* page describes one or more routines related to each other. Use the permuted index to find the page that holds a specific routine. Here are some notes on general use of the Pixar man pages.

Tabs You will find two kinds of tabs to help divide the tradition UNIX sections into subsections. The major tabs have the familiar meaning, while the minor tabs correspond to subsections (e.g., libraries, etc.).

Paper Grey paper denotes material (table of contents, permuted index, etc.) that will direct you to the pages in each section. The pink pages at the end of the book are for your comments.

Dates Each manual page bears its own date of last revision on the bottom.

Paths Pixar manual pages are located in */usr/pixar/man*. To get on-line manual help (*man*) for *xxxx*, type:

man -P /usr/pixar/man xxxx

Or you can save typing by using:

alias pixman "man -P /usr/pixar/man"

Then simply type:

pixman xxxx

(The above *alias* can also be put in the *.cshrc* file in your home directory to have the *pixman* command available permanently.)

Bugs Mail in the pink comment forms, or use electronic mail to submit on-line comments and suggestions (e.g., *mail pixar!bugs*).

Note

TABLE OF CONTENTS

1. Commands and Application Programs

3. User Libraries

3H. Host Routines

3C. Chap Routines

4. Special Files

5. File Formats

7. Miscellaneous

8. Maintenance

PERMUTED INDEX

NAME

intro – introduction to shell-level Pixar utilities

DESCRIPTION

This section describes the shell-level programs for interacting with the Pixar Image Computer.

Many frame buffer programs accept the following options.

–fb *fbname*
: Most of these programs will read the *FBDEFS* environment variable for finding the default frame buffer window. Specify a frame buffer window on a command line using the **–fb** *fbname* option. It is either a string delimited by "" (quotation marks), or a frame buffer as defined in the *LFBDEFS* environment variable. This option is described in more detail in *fbdefs*(7) and *FbGetDefs*(3H).

–srcfb *fbname*
–tmpfb *fbname*
–dstfb *fbname*
: Many programs use source, temporary and destination frame buffers rather than a single **–fb** option. Specify these frame buffer windows in the same manner as for the **–fb** option. If any of **srcfb, dstfb**, or **tmpfb** are not specified, but their corresponding windows are specified, they will each default to the *FBDEFS* environment variable. If any of **srcfb**, **dstfb**, or **tmpfb** are not specified, nor are their corresponding windows, then **srcfb** will default to **dstfb**, **dstfb** will default to **srcfb**, and **tmpfb** will default to **dstfb**. If none of these are specified, the program will read the *FBDEFS* environment variable for finding the default frame buffer window. This set up may sound complicated, but in practice yields the most intuitively expected results.

–w *xmin xmax ymin ymax*
: Denotes a subwindow of the frame buffer, expressed relative to the top left corner of the frame buffer.

–src *xmin xmax ymin ymax*
–tmp *xmin xmax ymin ymax*
–dst *xmin xmax ymin ymax*
: Denotes subwindows of frame buffers **srcfb, tmpfb** and **dstfb** respectively, expressed relative to the top left corner of their respective frame buffers.

–ch *selectchan*
: The channel select option; sets a write mask for the window. *selectchan* is a sequence of 1 to 4 characters from the set {r, g, b, a, R, G, B, A}, without repetition. For example: **–ch** rb will select the red and blue channels, **–ch** a selects only the alpha channel, **–ch** aGrB will select all four channels.

–shuffle *shufflechan*
: The channel shuffle option; sets a permutation of the channels for the window. *shufflechan* is a sequence of 4 characters from the above set, for example: RRRR gbaA rgrg are all legal.

Where used, the optional argument *color* is short for the usual color specification [red [green blue [alpha]]], where the color (0, 0, 0, 0) is assumed if the argument is not given, the color (red, red, red, red) is assumed if only red is specified, and the color (red, green, blue, 0) is assumed if alpha is not specified.

Many of these programs do not deal with the frame buffer at all; among these are the Chap assembler, linker and loader.

SEE ALSO

intro(3H), intro(3C)

DEVELOPMENT TOOLS

Name	*Page*	*Description*
charm	charm.1	– Chap runtime monitor
chas	chas.1	– Chap assembler
chc	chc.1	– Chap compiler
chcmp	chcmp.1	– compare a Chap object file to the downloaded version
chd	chd.1	– Chap disassembler
chld	chld.1	– Chap link editor
chload	chload.1	– download a Chap object file and start it running
chmap	chmap.1	– display Chap symbol table
chnm	chnm.1	– print name list of a Chap object file
chranlib	chranlib.1	– convert archives to Chap random libraries
chsize	chsize.1	– size of a Chap object file
dumi	dumi.1	– examine and modify Dumi registers

GRAPHICS TOOLS

Name	*Page*	*Description*
blur	blur.1	– applies a box filter to the framebuffer
cbars	cbars.1	– video colorbar generator
cha	cha.1	– perform linear arithmetic on framebuffer channels
clamp	clamp.1	– clamp the contents of a framebuffer to [0..2048]
clr	clr.1	– framebuffer clear
conv	conv.1	– convolve a framebuffer image with a 3x3 filter
copy	copy.1	– copy utility for portions of the framebuffer
crc	crc.1	– compute a Cyclic Redundency Check (CRC) on a framebuffer
gamma	gamma.1	– set gamma-corrected colormap
gt	gt.1	– get a frame buffer image from a picture file
gtinfo	gtinfo.1	– type out picture file information
guide	guide.1	– display fieldguide in framebuffer
hg	hg.1	– Take the histogram of a picture
loop	loop.1	– framebuffer animation tool
merge	merge.1	– merge two frame buffer windows onto a third
perm	perm.1	– permutations of the frame buffer
pixinit	pixinit.1	– initialize the pixar and the configuration tables
ramp	ramp.1	– ramp framebuffer window horizontally or vertically
resize	resize.1	– resize utility for portions of the framebuffer
rotate	rotate.1	– rotate utility for portions of the framebuffer
scale	scale.1	– scale framebuffer RGBA intensitiesn
see	see.1	– display a frame buffer image from a variety of types of picture file
sv	sv.1	– save frame buffer into picture file
tool	tool.1	– framebuffer tool
video	video.1	– video board utility

NAME

charm – Chap runtime monitor

SYNOPSIS

charm [**–x**] [**–I***dir*] [*chap-device*]

DESCRIPTION

charm is the Chap runtime monitor. With *charm* a user may interactively interrogate the state of a Chap, load and link-edit Chap code, and control the execution of programs running in a Chap. *charm* uses the Chap diagnostic interface and the facilities described in *chap*(4).

Options:

–x instructs *charm* to open the specified (or default) Chap device with exclusive access; this overrides the normal shared access.

–I may be used to specify directories in which files to be read with **$<** or **$<<** (see below) may be found. Normally, *charm* searches only in the directory "/usr/pixar/lib/charm". Multiple directories may be specified in this way. A specific Chap may be designated using Chap device; *charm* uses "/dev/chap0" by default.

charm ignores QUIT signals; INTERRUPT signals cause return to the next *charm* command.

When *charm* is ready to accept commands from the keyboard, it prompts with "> " and waits for input. In general, requests to *charm* are of the form

[*address*] [,*count*] [*command*] [;]

If *address* is present, the current address, referred to as "*dot*", is set to *address*. Initially, *dot* is set to 0. For most commands *count* specifies how many times the command should be executed. The default *count* is 1. *address* and *count* may be expressions.

EXPRESSIONS

charm processes two types of expressions: those involving scalar quantities, and those involving vectors (of length 4). Where two scalar expressions are combined, the obvious arithmetic is performed. Combining two vector expressions results in a component-by-component application of the appropriate operator. When a vector and a scalar are combined, the scalar is combined with each element of the vector to generate a vector result. Constants are considered scalars. 4-way registers (e.g., the ALU accumulator) are treated as vector expressions.

. The value of *dot*.

\+ The value of *dot* incremented by the current increment.

^ The value of *dot* decremented by the current increment.

" The last *address* typed.

integer A number. The prefixes "0x" and "0X" force interpretation in hexadecimal radix; the prefix "0" forces interpretation in octal radix; "0t" and "0T" force interpretation in decimal radix. If no prefix appears, then the *default* radix is used; see the **$d** command. The default radix is initially decimal. The hexadecimal digits are 0123456789abcdefABCDEF with the obvious values.

integer.fraction
A 16-bit Pixar fixed-point number. If the fraction is followed by an **e** or **E**, the number is treated as a component value (eleven bits of fraction). If the fraction is followed by an **f** or **F**, the number is treated as a coefficient value (fourteen bits of fraction). By default, fixed-point numbers are treated as component values. The *integer* portion of a fixed-point number must be specified in base ten, either explicitly with a "0t" prefix, or implicitly by setting the input radix to 10; see the **$d** command.

<name The value of *name*, which is either a variable name or a register name. *charm* maintains 36 variables: **a-z** and **0-9**. The register names are the same as those used by the Chap assembler; §4.4 of the *Charm Reference Manual* provides a complete list.

symbol A *symbol* is a sequence of upper or lower case letters, underscores or digits, not starting with a digit. The backslash character \ may be used to escape other characters. The value of the *symbol* is found by first checking the list of known registers then, failing there, looking in the symbol table.

(*exp*) The value of the expression *exp*.

Monadic Operators

**exp* The contents of the tessellated scratchpad location addressed by *exp*.

@*exp* The contents of the untessellated scratchpad location addressed by *exp*.

−*exp* Integer negation.

˜*exp* Bitwise complement.

!*exp* Logical negation.

Dyadic Operators

Dyadic operators are left associative and less binding than monadic operators.

e1+e2 Integer addition.

e1−e2 Integer subtraction.

*e1*e2* Integer multiplication.

e1%e2 Integer division.

e1&e2 Bitwise conjunction.

e1|e2 Bitwise disjunction.

e1#e2 Round *e1* up to the next multiple of *e2*.

COMMANDS

Most commands consist of a verb followed by a modifier or list of modifiers. The following verbs are available.

?*f* Locations starting at *address* in instruction RAM are printed according to the format *f*. *dot* is incremented by the sum of the increments for each format letter.

/*f* Locations starting at *address* in scratchpad RAM are printed according to the format *f* and *dot* is incremented as for "?".

=*f* The value of *address* itself is printed in the styles indicated by the format *f*. (This may not be used with the **i** format.).

A format *f*, consists of one or more characters that specify a style of printing. Each format may be preceded by a decimal integer that is a repeat count for the format letter. While stepping through a format, *dot* is incremented by the amount given for each format letter. If no format is given, the last format is used.

Lower-case letter formats used with the / operator force *charm* to interpret the address as a tessellated address; upper-case letters cause the address to be interpreted as un-tessellated.

The format letters available are as follows:

o Print the value in octal (**O** for untessellated).
d Print the value in decimal (**D** for untessellated).
x Print the value in hexadecimal (**X** for untessellated).
u Print the value as an unsigned decimal number (**U** for untessellated).
e Print the value as an 11-bit fixed point number (**E** for untessellated).
f Print the value as a 14-bit fixed point number (**F** for untessellated).
i Print the value as a machine instruction.
a Print the value of *dot* in symbolic form. Symbols are checked to ensure that they have an appropriate type as indicated below:

/local or global data symbol
?local or global text symbol
=local or global absolute symbol
(A for untesselated).

p Print the addressed value in symbolic form using the same rules for symbol lookup as **a** (**P** for untessellated).

b Print the value of *dot* in the form *pixel.component*, where the specified *component* is one of "RGBA" (**B** for untessellated).

z Print the addressed value in the form *pixel.component*, as for the **b** format (**Z** for untessellated).

t When preceded by an integer, tabs to the next appropriate tab stop. For example, **8t** moves to the next 8-space tab stop.

r Print a space.

n Print a newline.

"..." Print the enclosed string.

^ *dot* is decremented by the current increment; nothing is printed.

+ *dot* is incremented by 1; nothing is printed.

– *dot* is decremented by 1; nothing is printed.

c Print the value as an ASCII character. Control characters are printed as ^X and the delete character is printed as ^?.

s *n* Print a string of characters (terminated by a null byte).

newlineR

Repeat the previous command with a *count* of 1.

[?/]w *value ...*

Write a 1-word *value* into the addressed locations. If the command is **W**, the address is treated as untessellated. If the address expression is 4-way, the value is written to each of the four components. Multiple values are written into consecutive locations. If a *count* is specified, the write command is repeated *count* times with *dot* incremented each time (useful for clearing a block of scratchpad).

>name *dot* is assigned to the variable or register named. If a 4-way register is specified and *dot* is a scalar expression, its value is assigned to each component of the register. Assigning a vector expression to a variable causes it to be treated later as a vector expression.

$*modifier*

Miscellaneous printing commands. The available *modifiers* are:

<f Read commands from the file *f*. If this command is executed in a file, further commands in the file are not seen. If *f* is omitted, the current input stream is terminated. If a *count* is given, and is zero, the command will be ignored. The value of count will be placed in variable *9* before the first command in *f* is executed.

<<f Similar to < except it can be used in a file of commands without causing the file to be closed. Variable *9* is saved during the execution of this command and restored when it completes. There is a (small) finite limit to the number of << files that can be open at once.

>f Append output to the file *f*, which is created if it does not exist. If *f* is omitted, output is returned to the terminal.

a Print the scratchpad address registers. If *count* is specified, only the first *count* registers are displayed.

b Print all breakpoints and their associated counts and commands.

c Print a stack backtrace. The backtrace shows the value of the **pc**, **lc**, and **runflag** at each level in the stack. If *count* is given, then only the first *count* frames are printed. If *address* is specified, the backtrace commences at that stack level.

d Set the default radix to *address* and report the new value. Note that *address* is interpreted in the (old) current radix. If no radix is specified, *charm* reports the the current radix.

e Print the names and values of external symbols. If an *address* is specified, it is interpreted as a symbol table type; the possible values are: 2 (*absolute* symbols), 4 (*text* symbols), 6 (*data* symbols), and 8 (*bss* symbols).

l Print the names and values of local symbols. Any *address* specified is interpreted as for **e**.

m Print the segment (load) map.

p Print the contents of the Pbus registers and the Pbus data buffer. If *count* is given, only the first *count* entries in the Pbus data buffer are displayed.

q Exit from *charm* ($Q and ^D work as well).

r Print the registers of each ALU, the loop counter, the stack pointer, and the instruction addressed by the **pc**. *dot* is set to **pc**. If *address* is specified, it is interpreted as a bitmask of processors for which ALU registers should be displayed. If *count* is given, only the first *count* ALU registers are displayed.

u Print the name of each unresolved symbol and the modules in which the symbols are referenced.

v Print all non-zero variables in hexadecimal.

x Print the contents of the crossbar.

y Print the contents of the Yapbus registers.

S Set the limit for symbol matches to *address* (default 255).

W Set the page width for output to *address* (default 80).

:*modifier*

Manage the execution of the Chap. The available *modifiers* are:

b*c* Set a breakpoint at *address*. The breakpoint is executed *count*−1 times before causing a stop. Each time the breakpoint is encountered the command *c* is executed. If this command is omitted or sets *dot* to zero, then the breakpoint causes a stop.

d Delete the breakpoint at *address*.

c The Chap is continued. If *address* is given, then the processor is continued at this address. Breakpoint skipping is the same as for *r*.

f*c* Specify a set of commands *c* to be executed each time the Chap is stopped by a : command. More explicitly, the "format" commands are executed after each single-step, next, run, continue, or halt command is completed. If a command string is not specified, the current one is displayed.

h Halt the Chap.

l*f* Load and bind a *chas* output file *f*. *charm* will try to resolve any undefined external references in *f* from code currently resident in the Chap. Failure to resolve references is reported on the standard output. The file is searched for in the list of directories shown with the **:p** command. If a file is not specified, the last file specified in a **:u** or **:l** command is used. If no "last file" is available, *charm* tries to load the file "chap.out", or failing, "chas.out".

n As for **:s** except that if the current instruction contains a jump to subroutine sequencer instruction, the subroutine is run at full speed with the Chap halted at the instruction immediately following the return. If the Chap had not previously been started with an r command, the **n** command will do this.

p*p* Set the "load searchpath" to *p*. The path is a list of directories in which to search for loadable files. Searchpaths must be separated by colons. If no path is specified, the current load path is displayed. The default load path is ".:/usr/lib:/usr/pixar/lib".

r Begin execution of the Chap. If *address* is given explicitly, then the program is entered at this point; otherwise the program is entered at its standard entry point. *Count* specifies how many breakpoints are to be ignored before stopping.

s As for **:c** except that the Chap is single stepped *count* instructions. If the Chap had not previously been started with an **r** command, the **s** command will do this.

u*f* Unload the file *f*. That is, reclaim the instruction and scratchpad memory associated with file and remove the related allocation information from the symbol table. When no file is specified, *charm* searches for a file as described under the **:l** command.

VARIABLES

charm provides a number of variables. Certain named variables are set initially by *charm* and used in the print commands (see below). Numbered variables are used to communicate various dynamically changing values.

- 0 The last value printed.
- 1 The last immediate field of an instruction.
- 2 The previous value of variable 1.
- 9 The count on the last **$<** or **$<<** command.
- a Number of registers to print with the **$a** command.
- f "Runflag" to use in limiting printing with the **$r** command.
- p Number of data buffer entries to print with the **$p** command.
- r Number of registers to print with the **$r** command.
- s Number of registers to print with the **$s** command.

REGISTERS

charm allows Chap data registers to be referred to symbolically. Register names are identical to those used by the Chap assembler *chas* wherever possible. A component of a vector register may be specified with [*exp*], where *exp* is an expression as described above. A **sysbus** register is specified, as in *chas*, **sysbus**<*exp*> where, once again, *exp* is an expression. The following list shows the names of registers as understood by *charm*.

a0, a1	Pbus address registers	
acc	ALU accumulator	4-way
admux	address portion of the crossbar	4-way
b0, ..., b15	Scratchpad base address registers	
i0,..., i15	Scratchpad index registers	
lc	Loop counter	
lsp	Least significant part of multiplier output	4-way
msp	Most significant part of multiplier output	4-way
multx	Multiplier X-input	4-way
multy	Multiplier Y-input	4-way
pc	Program counter	
pcsr	Pbus control status register	
r0, ..., r31	ALU internal registers	4-way
rdmux	read portion of the crossbar	4-way
rf	Runflag	
sp	Stack pointer	
sysbus	Sysbus shared data register	
status	Chap status register	
wrmux	write portion of the crossbar	4-way
ycsr	Yapbus control status register	

FILES

/dev/chap0 default Chap device to use
/usr/pixar/host/bin/charm

SEE ALSO

chc(1), chas(1), intro(3H), ChapLoad(3H), chap(4G), chap.out(5)

DIAGNOSTICS

Types 'Charm' when there is no current command or format. Comments about inaccessible files, syntax errors, etc. Exit status is 0, unless the last command failed or returned nonzero status.

BUGS

The $c command sometimes doesn't work. You can't write instruction memory.

NAME

chas – Chap assembler

SYNOPSIS

chas [**−wsS**] [**−o** *output*] [*file*]

DESCRIPTION

chas is an assembler for the Chap. *chas* takes one or more input files (or standard input if no files are specified) and generates a relocatable object file suitable for use with the Chap link editor, *chld*(1), or dynamic loader, *ChapLoad*(3H). *chas* is most normally accessed through the *chc*(1) program, which first passes the input file through the C preprocessor. The options to *chas* are:

−w Suppress warning messages.

−s Enable messages indicating new instructions generated as the result of the special bit.

−S Print the symbol table after all input has been processed.

−o Place the relocatable object file in *output* instead of the default file *chas.out*.

FILES

chas.out default name for output file
/usr/pixar/host/bin/chas

SEE ALSO

Chap Assembler Reference Manual
chc(1), chld(1), ChapLoad(3H)

BUGS

Incorrect relocation information is generated for "loop" constructs using expressions that require the special bit. Statements that cause *chas* to generate a special bit instruction, and that modify operands to be supplied to the ALU, are not handled correctly.

NAME

chc – Chap compiler

SYNOPSIS

chc [*options*] [*files*]

DESCRIPTION

chc is the Chap compiler (more of an assembler than anything else). *chc* accepts several types of arguments.

Arguments whose names end with '.s' are taken to be Chas source programs; they are assembled, and each object program is left in a file whose name is that of the source with '.o' substituted for '.s'. The '.o' file is normally deleted if a single Chas program is compiled and loaded all at once (see **–c** option below).

The following options are interpreted by *chc*. See *chld*(1) for load-time options.

–c Suppress the loading phase of the compilation, and force an object file to be produced even if only one program is compiled.

–w Suppress warning diagnostics.

–E Run only the macro preprocessor on the named Chas programs, and send the result to the standard output.

–C Prevent the macro preprocessor from removing comments.

–o *output*
Name the final output file *output*. If this option is used, the file 'chap.out' will be left undisturbed.

–D*name=def*
Define the *name* to the preprocessor, as if by '#define'. If no definition is given, the name is defined as "1".

–U**Sname*
Remove any initial definition of *name*.

–I**Sdir* '#include' files whose names do not begin with '/' are always sought first in the directory of the *file* argument, then in directories named in **–I** options, then in directories on a standard list.

Arguments are taken to be either loader option arguments, or Chas-compatible object programs, typically produced by an earlier *chc* run, or perhaps libraries of Chas-compatible routines. These programs, together with the results of any compilations specified, are loaded (in the order given) to produce an executable program with the name *chap.out*.

FILES

file.s	input file
file.o	object file
chap.out	loaded output
/tmp/chas?	temporary
/lib/cpp	preprocessor
/usr/pixar/host/bin/chas	assembler
/usr/pixar/host/bin/chc	compiler
/usr/pixar/host/bin/chld	loader
/usr/pixar/include	standard directory for '#include' files

SEE ALSO

charm(1), chld(1), chas(1)

DIAGNOSTICS

The diagnostics produced by Chas itself are intended to be self-explanatory. Occasional messages may be produced by the loader.

NAME

chcmp – compare a Chap object file to the downloaded version

SYNOPSIS

chcmp [**–s**] [**–l**] [**–f** *device*] [*file*]

DESCRIPTION

chcmp compares the contents of a Chap object file against the contents of instruction and scratchpad memories. If no object file is specified, *chcmp* tries to use *chap.out* or (failing) *chas.out*. The default device is /dev/chap0.

Options:

–f Used to specify an alternate device.

–l Normally, *chcmp* will report the first place where the two files differ, then exit. If the –l flag is specified, *chcmp* will report all differences.

–s If specified, *chcmp* will produce no output; instead its termination status will indicate whether the files compare.

FILES

/usr/pixar/host/bin/chcmp
chap.out
chas.out
/dev/chap0

SEE ALSO

chap.out(5), chload(1), chld(1), chd(1)

BUGS

chcmp is useful only with *chload*(1). Chap object files that have been relocated by *chld* obviously will not compare "correctly".

NAME

chd – Chap disassembler

SYNOPSIS

chd [**–f**] [**–n**] [*files*]

DESCRIPTION

chd prints a listing of the specified Chap object files, disassembling instructions in the text segment.

Options:

–f If specified, each field of a Chap instruction that differs from the "default" value assigned it by the assembler is displayed.

–n If specified, *chd* does not print the customary text or data segment address in the first column; this is useful mostly for comparing object files with the UNIX command *diff*.

When *chd* is invoked without specifying any object files, it tries to open the file "chap.out". Should that fail, *chd* will then try to open the file "chas.out".

FILES

chap.out primary default input file
chas.out secondary default input file
/usr/pixar/host/bin/chd

SEE ALSO

charm(1), chas(1)

DIAGNOSTICS

The diagnostics are intended to be self-explanatory.

BUGS

chd uses an ancient disassembly algorithm which differs significantly from that used by *charm*(1); it should be rewritten to use *charm*'s algorithm and the symbol table associated with each object file.

NAME

chld – Chap link editor

SYNOPSIS

chld [*options*] [*file*]

DESCRIPTION

chld combines several object programs into one, resolving external references, and searching libraries. In the simplest case, several object *files* are given, and *chld* combines them, producing an object module that can be either executed on a Chap or become the input for a further *chld* run. (In the latter case, the **–r** option must be given to preserve the relocation bits.) The output of *chld* is left on **chap.out.**

The argument routines are concatenated in the order specified. The entry point of the output is the beginning of the first routine (unless the **–e** option is specified).

If any argument is a library, it is searched exactly once at the point it is encountered in the argument list. Only those routines defining an unresolved external reference are loaded. If a routine from a library references another routine in the library, and the library has not been processed by *chranlib*(1), the referenced routine must appear after the referencing routine in the library. Thus, the order of programs within libraries can be important. The first member of a library should be a file named '__.SYMDEF', which is understood to be a dictionary for the library as produced by *chranlib*(1); the dictionary is searched iteratively to satisfy as many references as possible.

The symbols '_etext', '_edata' and '_end' are reserved, and, if referred to, are set to the first location above the program, the first location above initialized data, and the first location above all data respectively. It is erroneous to define these symbols.

chld understands several options. Except for **–l**, they should appear before the file names.

–A This option specifies incremental loading, i.e., linking is to be done in a manner so that the resulting object may be read into an already executing program. The next argument is the name of a file whose symbol table will be taken as a basis on which to define additional symbols. Only newly linked material will be entered into the text and data portions of **chap.out,** but the new symbol table will reflect every symbol defined before and after the incremental load. This argument must appear before any other object file in the argument list.

–T May be used as well as **–A**, and will be taken to mean that the newly linked segment will commence at the corresponding address (which must be a multiple of 1024). The default value is the old value of _end.

–D Take the next argument as a hexadecimal number and pad the data segment with zero-filled bytes to the indicated length.

–d Force definition of common storage even if the **–r** flag is present.

–e The following argument is taken to be the name of the entry point of the loaded program; location 0 is the default.

–l*x* This option is an abbreviation for the library name '/usr/pixar/chap/lib/lib*x*.a' where *x* is a string. A library is searched when its name is encountered, so the placement of a **–l** is significant.

–M Produce a primitive load map, listing the names of the files to be loaded.

–o The *name* argument after **–o** is used as the name of the *chld* output file, instead of *chap.out*.

–r Generate relocation bits in the output file, so it can be the subject of another *chld* run. This flag also prevents final definitions from being given to common symbols, and suppresses the 'undefined symbol' diagnostics.

–S 'Strip' the output by removing all symbols except locals and globals.

–T The next argument is a hexadecimal number which sets the text segment origin. The default origin is 0.

−t (“trace”) Print the name of each file as it is processed.

−u Take the following argument as a symbol and enter it as undefined in the symbol table. This is useful for loading wholly from a library, since initially the symbol table is empty and an unresolved reference is needed to force the loading of the first routine.

−X Save local symbols except for those whose names begin with ‘L’. This option is used by *chc*(1) to discard internally-generated labels while retaining symbols local to routines.

−x Do not preserve local (non-.globl) symbols in the output symbol table; only enter external symbols. This option saves some space in the output file.

−y*sym* Indicate each file in which *sym* appears, its type and whether the file defines or references it. Many such options may be given to trace many symbols.

FILES

/usr/pixar/chap/lib/*x*.a	libraries
chap.out	output file
/usr/pixar/host/bin/chld	

SEE ALSO

chas(1), ar(1*), chc(1), chranlib(1)

* See the appropriate UNIX programmer's manual page.

NAME

chload – download a Chap object file and start it running

SYNOPSIS

chload [**–f** *device*] [**–h**] [**–r**] [**–s** *startsym*] [**–v**] [*files*]

DESCRIPTION

chload link-edits and relocates one or more relocatable object files created by *chas*(1) or *chld*(1), downloading the resulting program into a Chap. If the link-edit process is successful, the Chap is set running at the start of the first file.

Options:

–f Unless this is specified, *chload* downloads programs into /dev/chap0.

–h Used to link-edit and load the files but not start the Chap running.

–r Resets the symbol table before starting the loading process.

–s Forces *chload* to start the Chap running at a specific location in the program when a symbol name is specified with the –s flag.

–v Normally, *chload* operates quietly. The –v flag causes it to print messages regarding each file loaded out of a library and the location in instruction memory at which the Chap is set running.

–lx This option is an abbreviation for the library name '/lib/lib*x*.a', where *x* is a string. If that does not exist, *ld* tries '/usr/lib/lib*x*.a'. A library is searched when its name is encountered, so the placement of a –l is significant (/usr/pixar/chap/lib*.a).

SEE ALSO

chc(1), chld(1), ChapLoadGo(3H), ChapLoad(3H), chap.out(5)

NAME

chmap – display Chap symbol table

SYNOPSIS

chmap [**–gbdtulrm**] [**–f** *device*] [**–i**] [*symbol-name*]

DESCRIPTION

chmap prints the symbol table associated with a Chap. This symbol table, used by the Chap link-editor loader *chload*(1), reflects the known contents of the Chap's scratchpad and instruction memories. The default symbol table displayed is that associated with "/dev/chap0"; the **–f** flag may be used to specify an alternate *device*. If no arguments are specified *chmap* displays the entire contents of the symbol table. Otherwise, only the values of the specified *symbols* are shown. If a file name symbol is specified, *chmap* displays all the symbols defined in that file.

Each symbol name is preceded by its value (blanks if undefined) and one of the letters **U** (undefined), **A** (absolute), **T** (text segment symbol), **D** (data segment symbol), **B** (bss segment symbol), or **f** file name. If the symbol is local (non-external) the type letter is in lower case.

The following options may be used to specify only a subset of the symbols in the symbol table:

–g Print only global (external) symbols.

–b Print only symbols defined in bss segments.

–d Print only symbols defined in data segments.

–t Print only symbols defined in text segments.

–u Print only undefined symbols.

–l Print only local (not external) symbols.

–r May be used to force *chmap* to list the address and segment of each reference to a symbol.

–m Displays a "load map" identical to the **$m** command of *charm*(1). This display shows each file that has been loaded into the Chap and the locations of the file's segments.

–f May be used to specify an alternate *device*.

–i May be used to initialize the symbol table and the operating system resource allocation maps. While it is often useful for resetting the Chap, it should be used with care, since this request deletes all information about resident code and data.

Text segment symbol values are divided by the size of a Chap micro-instruction. Data and bss segment symbol values are divided by two to give a word offset in scratchpad.

FILES

/dev/chap0	default Chap device
/usr/pixar/host/symtab/*	symbol table files
/usr/pixar/host/bin/chmap	

SEE ALSO

chnm(1), chload(1), chapsym(5)

NAME

chnm – print name list of a Chap object file

SYNOPSIS

chnm [**−gnopru**] [*file*]

DESCRIPTION

chnm prints the name list (symbol table) of each Chap object *file* in the argument list. If an argument is an archive, a listing for each object file in the archive will be produced. If no *file* is given, the symbols in "chap.out" are listed.

Each symbol name is preceded by its value (blanks if undefined) and one of the letters **U** (undefined), **A** (absolute), **T** (text segment symbol), **D** (data segment symbol), **B** (bss segment symbol), **Q** (qualifier symbol), or **C** (common symbol), or **f** file name. If the symbol is local (non-external) the type letter is in lower case. The output is sorted alphabetically.

Options are:

−g Print only global (external) symbols.

−n Sort numerically rather than alphabetically.

−o Prepend file or archive element name to each output line, rather than only once.

−p Don't sort; print in symbol-table order.

−r Sort in reverse order.

−u Print only undefined symbols.

Text segment symbol values are divided by the size of a Chap micro-instruction (96 bits). Data and bss segment symbol values are divided by two to give a word offset in scratchpad.

SEE ALSO

ar(1*), ar(5*), chap.out(5)

* See appropriate page in UNIX Programmer's Manual.

NAME

chranlib – convert archives to Chap random libraries

SYNOPSIS

chranlib *archive*

DESCRIPTION

chranlib converts each *archive* to a form the Chap loader can load efficiently. *chranlib* does this by adding a table of contents called __.SYMDEF to the beginning of the archive. *chranlib* uses *ar (1)* to reconstruct the archive, so that sufficient temporary file space must be available in the file system that contains the current directory.

FILES

/usr/pixar/host/bin/chranlib

SEE ALSO

chld(1), ar(1*), lorder(1*)

BUGS

Because generation of a library by *ar*(1*) and randomization of the library by *chranlib* are separate processes, phase errors are possible.

* See appropriate page in UNIX Programmer's Manual.

NAME

chsize – size of a Chap object file

SYNOPSIS

chsize [*object-file*]

DESCRIPTION

chsize prints the (decimal) sizes of the text, data, and bss portions, and their sum in hex and decimal, of each object-file argument. Text sizes are printed in terms of 96-bit instructions, while data and bss sizes are in terms of the 16-bit words. If no file is specified, *chap.out* is used.

FILES

/usr/pixar/host/bin/chsize
chap.out

SEE ALSO

chap.out(5)

NAME

dumi – examine and modify Dumi registers

SYNOPSIS

dumi [**iena**] [**–iena**] [**reset**] [**–reset**] [**init**] [**peek** *addr*] [**poke** *addr data*]

DESCRIPTION

dumi is a simple program used to peek and poke values into or through the Dumi interface. If no arguments are given, *dumi* prints the contents of the Dumi control status register. Arguments are interpreted as commands and processed one at a time as follows:

iena Set the interrupt enable bit in the csr.

–iena Clear the interrupt enable bit in the csr.

reset Set the reset bit in the csr.

–reset Clear the reset bit in the csr.

init Initialize the Dumi by setting the reset and interrupt enable bits in the csr.

peek Display the value in Sysbus address *addr*.

poke Try to poke the specified *data* value into the location at the given Sysbus *addr* .

dumi catches faults generated by peeks and pokes on the Sysbus and prints the message "Bus error".

This program is useful as a hardware diagnostic aid. Use of this program should normally be limited to that purpose.

FILES

/usr/pixar/host/bin/dumi

SEE ALSO

dumi(4), mctrl(8)

NAME
blur – applies a box filter to the framebuffer

SYNOPSIS
blur [[*height*] width] [*options*]

DESCRIPTION
blur computes a new value for each pixel by averaging it with its *height**width* neighbors. The box filter is a *height*-by-*width*-sized kernel containing all ones. It makes a horizontal pass of the framebuffer followed by a vertical pass. The box filter is truncated at the edges of the framebuffer. If the filter width is not given, it is assumed to be equal to the height. (The default is 11.)

Options:

–ch Allows channel selection. For example, **–ch** rg indicates that only the red and green channels are to be filtered. By default, all four channels are blurred.

–src *xmin xmax ymin ymax*
See *intro*(1). Only the pixels within the window are blurred; the box filter is truncated at the window edges.

–dst *xmin xmax ymin ymax*
See *intro*(1).

–y Don't do vertical pass.

–x Don't do horizontal pass.

–h Set high pass.

–b *weight*
Set *weight* of blurred image *fbname* for *PirlBoxFilter*(3H).

–c *weight*
Set *weight* of center pixel.

–n *count*
Set number of times.

–srcfb *fbname*
See *intro*(1).

–dstfb *fbname*
See *intro*(1).

DIAGNOSTICS
Height and *width* should both be odd numbers. If not, they are made odd by adding one and a message is printed. If *height* and *width* exceed the framebuffer (or window) dimensions, a modulus is applied and a message is printed.

FILES
/usr/pixar/host/bin/blur

SEE ALSO
PirlBoxFilter(3H), fbdefs(7)

NAME

cbars – video colorbar generator

SYNOPSIS

cbars [–c] [–f] [–r]

DESCRIPTION

cbars generates colorbars in a Pixar framebuffer for video test and calibration. It generates colored bars at 75% full saturation, a white bar at 75% of full white, and a black bar at 0% of full white. The default pattern has eight bars in the upper three-fourths of the window. The pluge pattern is drawn in the bottom one-fourth of the screen. This pattern contains a 0% black (NTSC encoder jacks this up to 7.5%), a full 100% white, and an alternating series of blacks: 0%, 2.5%, 0%. It is used to adjust the brightness of a monitor. The brightness control is turned until the 2.5% bar is just visible. (The 2.5% bar corresponds to the 10% bar of the standard pluge pattern.) The colormap is set by *cbars* to a gamma of 2.3.

The following options, which alter the pattern in the lower one-fourth of the screen, are available:

–c Draw the CBS standard colorbars in which the bottom one-fourth contains the eight bars in reverse order with every other one set to black. The resulting colorbars possess the following feature: When the red and green guns are turned off, the pattern becomes alternating blue and black bars with perhaps a slight discontinuity in the blue bars at the one-fourth screen position. By adjusting the "hue" and "chroma" (or "color") knobs on an NTSC monitor showing this pattern, these discontinuities can be made to disappear. In this state, the NTSC monitor is correctly color adjusted.

–f Draw full length bars. The bottom one-fourth is identical horizontally to the upper three-fourths.

–r Draw reverse bars at the bottom. This is different from the default pattern in that every other bar is not set to black.

SEE ALSO

PirlCbars(3H), fbdefs(7)

BUGS

Is missing the +Q and –I bars of some standard colorbar generators. Is missing the darkest pluge bars of some standard colorbar generators. Uses 64-pixel bars for even subdivision of 512 instead of the 512/7 and 512/12 width bars of some standard colorbar generators.

NAME

cha – perform linear arithmetic on framebuffer channels

SYNOPSIS

cha [*S*] [**–r** *rr* [*rg* [*rb* [*ra* [*rk*]]]]]
[**–g** *gr* [*gg* [*gb* [*ga* [*gk*]]]]]
[**–b** *br* [*bg* [*bb* [*ba* [*bk*]]]]]
[**–a** *ar* [*ag* [*ab* [*aa* [*ak*]]]]]

DESCRIPTION

cha performs a linear transformation on the pixels of a framebuffer by treating the R, G, B and A pixel values of each input pixel as the vector [R G B A 1] and post-multiplying this vector by a 4x5 matrix. This matrix is specified on the command line as follows: if the *S* argument is given, the matrix is

```
[S 0 0 0
 0 S 0 0
 0 0 S 0
 0 0 0 S
 0 0 0 0]
```

If the **–r**, **–g**, **–b** or **–a** flags are given, their arguments appear in the matrix as

```
[rr gr br ar
 rg gg bg ag
 rb gb bb ab
 ra ga ba aa
 rk gk bk ak]
```

For example, if R, G, B, and A represent the original red, green, blue, and alpha channel values, and if R' represents the new red channel value, then *cha* computes R' = *rr**R + *rg**G + *rb**B + *ra**A + *rk*, and similarly for the other channels, if desired, with coefficients specified as floating point numbers. Note that channel values are NOT clamped to the range [0, 1.0E].

If any of the arguments to these flags but the last are omitted, they default to the last value given. Thus,

–r *rr rg*

generates the column

```
[ rr
  rg
  rg
  rg
  0  ]
```

If both the *S* and flag arguments are given, the scale factor *S* is applied to the diagonal of the matrix, effectively scaling the input vector before the matrix is applied.

–srcfb *fbname*
See *intro*(1).

–dstfb *fbname*
See *intro*(1).

–src *xmin xmax ymin ymax*
See *intro*(1).

–dst *xmin xmax ymin ymax*
See *intro*(1).

SEE ALSO

PirlCha(3H)

NAME

clamp – clamp the contents of a framebuffer to [0..2048]

SYNOPSIS

clamp [*options*]

DESCRIPTION

clamp sets any pixels greater than 1.0E(2048) to 1.0E, and any pixels less than 0.0E to 0.

Options:

–srcfb *fbname*
See *intro*(1).

–dstfb *fbname*
See *intro*(1).

–src *xmin xmax ymin ymax*
See *intro*(1).

–dst *xmin xmax ymin ymax*
See *intro*(1).

–ch *selectchan*
See *intro*(1). Use specified channels only (e.g., –ch rgb, –ch a, –ch AR).

FILES

/usr/pixar/host/bin/clamp

SEE ALSO

PirlClamp(3H), PWClamp(3C), SSClamp(3C)

NAME

clr – framebuffer clear

SYNOPSIS

clr [*red* [*green blue* [*alpha*]]] [*options*]

DESCRIPTION

clr clears a Pixar framebuffer to a color, each of whose components are specified (*red, blue, green, alpha*) as an integer in the range [0, 2048]. No argument implies a clear to (0, 0, 0, 0). One argument implies a clear to (*red, red, red, red*). Three arguments implies a clear to (*red, green, blue*, 2048). Four arguments implies a clear to (*red, green, blue, alpha*).

Options:

–w *xmin xmax ymin ymax*
See *intro*(1).

–n Clear the complement of any specified window.

–row *y* [*xmin xmax*]
–col *x* [*ymin ymax*]
Special cases of the **–w** option for clearing a given row or column or, optionally, a subset of the given row or column.

–ch *selectchan*
See *intro*(1). Use specified channels only (e.g., –ch rgb, –ch a, –ch AR).

–fb *fbname*
See *intro*(1).

FILES

/usr/pixar/host/bin/clr

SEE ALSO

PirlClear(3H)

DIAGNOSTICS

clr will complain about invalid argument values and invalid window descriptions.

NAME

conv – convolve a framebuffer image with a 3x3 filter

SYNOPSIS

conv [*options*]

DESCRIPTION

conv convolves a frame buffer image with a 3x3 filter.

Options:

–k *k00 k01 k01 k10 k11 k12 k20 k21 k22*
: specifies the values of each entry in the kernel. The first value is the left-most entry in the uppermost row. Subsequent entries complete that row and then move to the next row.

–laplace
: Use a Laplacian kernel. This is equivalent to "–k 0 –1 0 –1 4 –1 0 –1 0".

–n normalize the kernel so the sum of all the weights add up to 1.

–s *scale* multiplies each element of the kernel by *scale*.

–srcfb *fbname*
: See *intro*(1).

–dstfb *fbname*
: See *intro*(1).

–src *xmin xmax ymin ymax*
: See *intro*(1).

–dst *xmin xmax ymin ymax*
: See *intro*(1).

–ch *selectchan*
: See *intro*(1).

FILES

/usr/pixar/host/bin/conv

SEE ALSO

PirlConvolve3x3(3H)

NAME

copy – copy utility for portions of the framebuffer

SYNOPSIS

copy [*options*]

DESCRIPTION

copy copies a source window on the framebuffer to a destination window on the framebuffer. The source window is copied to all the destination windows, and several destination windows may be specified on the command line.

copy also supports multiple destination logical framebuffers. If more than one destination framebuffer is specified (with the **–dstfb** command), then the source framebuffer is copied to each of the destination framebuffers. Naming the framebuffers via *LFBDEFS* makes it easier to maintain multiple images on the same framebuffer.

If several destination framebuffers and windows are given, each window is defined relative to the corresponding framebuffer definition. If only one destination framebuffer is given, and multiple destination windows, each window is defined relative to that framebuffer. Similarly, if only one destination window is given, and multiple destination framebuffers, this window is applied for each framebuffer definition during the copy.

If the destinations pixel window(s) are smaller than the source pixel window then the pixels are clipped to the destination pixel windows.

Options:

–swap Swap the specified windows instead of just copying.

–src *xmin xmax ymin ymax* See *intro*(1).

–dst *xmin xmax ymin ymax* See *intro*(1).

–srcfb *fbname* See *intro*(1).

–dstfb *fbname* See *intro*(1).

–ch *selectchan* See *intro*(1).

EXAMPLES

copy –src 0 255 0 255 –dst 256 511 256 511

This copies the 256x256 pixel window at the origin of the framebuffer into the 256x256 pixel rectangle at (256,256).

copy –srcfb q0 –dstfb q1

This copies the pixel window defined by the *fbdef* q0 into the pixel window defined by *fbdef* q1.

copy –srcfb q1 –src 0 49 0 19 –dst 100 149 0 19

This copies the 50x10 pixel window *relative to fbdef q1*, into a 50x10 pixel window, relative to the default *fbdef* read from the *FBDEFS* environment variable. (See *intro*(1) for a discussion of default settings of frame buffers.)

copy –srcfb q0 –dstfb q1 –swap

This swaps the pixel window defined by the *fbdef* q0 and the pixel window defined by *fbdef* q1.

FILES

/usr/pixar/host/bin/copy

SEE ALSO

PirlCopy(3H), fbdefs(7)

NAME

crc – compute a Cyclic Redundency Check (CRC) on a framebuffer

SYNOPSIS

crc

DESCRIPTION

crc computes a CCITT standard CRC value for a framebuffer window. The CRC values for each channel are printed on *stdout.*

Options:

–fb *fbname*
See *intro*(1).

–w *xmin xmax ymin ymax*
See *intro*(1).

FILES

/usr/pixar/host/src/bin/crc.c
/usr/pixar/host/src/lib/libpirl/crc.c

SEE ALSO

Chad(3H), PirlCrc(3H)
SSCrc(3C), PWCrc(3H)

DIAGNOSTICS

none

BUGS

NAME

gamma – set gamma-corrected colormap

SYNOPSIS

gamma [*exponent*]

DESCRIPTION

gamma assumes that a video monitor's nonlinearities may be approximated by an exponential curve with an exponent traditionally called ''gamma''. With no *exponent* argument, *gamma* sets the colormap to compensate for a gamma of 2.3.

The program *gamma* is actually a shell script containing the following command:

```
video –gamma $*
```

FILES

/usr/pixar/host/bin/gamma
/usr/piar/host/bin/video

SEE ALSO

video(1)

NAME

gt – get a frame buffer image from a picture file

SYNOPSIS

gt [*options*] *file...*

DESCRIPTION

gt brings picture(s) from "tile-based" picture file(s) into a frame buffer. The picture file must conform to Pixar's standards for tile-based picture files (see "The Format of Stored Pictures," in the *Pixar Programmers' Manual*). The PIXPATH of the environment is used to find the picture(s). Each picture header is read to determine the picture size, tile size, and component information.

Eventually, only those components in the stored picture will be written into the frame buffer. With the current frame buffer interface however, selective channel writing can be done only with a slow read-modify-write sequence. Currently, then, missing RGB channels are zero by default and a missing alpha channel is unity. A red-channel-only picture is now written as (red, zero, zero, unity). The **–ch** flag should be used to assure the selective channel writing.

When matting is requested, the target frame buffer is assumed to be matted to black. The user should override this default when the alpha channel of the target image is unassociated with the RGB channels.

Options:

–fb *fbname*
See *intro*(1).

–o *x y* causes the picture to be offset as it is decoded.

–w *xmin xmax ymin ymax*
See *intro*(1).

–t *n* specifies that only tile number *n* be read into the frame buffer.

–l print the label stored with the picture.

–clr clears the frame buffer between images.

–nc don't display cursor while getting the picture

–v (verbose) elicits typeout of the frame buffer and recovered file name.

–ch *selectchan*
See *intro*(1).

–host force host to decode picture.

FILES

/usr/pixar/host/bin/gt

SEE ALSO

sv(1), gtinfo(1), PicCreat(3H), PicRead(3H), PicClose(3H), fbdefs(7)

DIAGNOSTICS

gt will die if it cannot open the file or the frame buffer.

NAME

gtinfo – type out picture file information

SYNOPSIS

gtinfo *file*

DESCRIPTION

gtinfo gives details of a "tile-based" picture file. Running the command *gtinfo /usr/pixar/demo/pix/1984* should print something like the following:

```
/usr/pixar/demo/pix/1984:
[no label]
picture size              :  1024   768
tile size                 :  1024   768
picture offset            :  0      0
1 tile in 8192 byte blocks.
8 encoded bits per channel :  RGBA   matted to black
Tile status               :  tile 0 complete
```

The PIXPATH of the environment is used to find the picture. The picture header is read to determine the picture size, tile size, and component information.

FILES

/usr/pixar/host/bin/gtinfo

SEE ALSO

gt(1), sv(1), PicCreat(3H), PicRead(3H), PicClose(3H)

NAME

guide – display fieldguide in framebuffer

SYNOPSIS

guide [**–n**] [**–c** [*red* [*green* [*blue* [*alpha*]]]]]

DESCRIPTION

guide displays a conventional field-guide in a framebuffer by complementing the high bit of the green channel. Upon completion, the program waits for a carriage return for recomplementing the framebuffer, (thus resetting it to its original contents).

Options:

–n Overrides the wait for a carriage return, forcing an exit after the first complement is performed.

–c Writes the fieldguide permanently into a framebuffer using the specified color. It uses the conventional color specification similar to *clr*(1), except that component values are clamped as in *clamp*(1).

–fb *fbname*
See *intro*(1).

FILES

/usr/pixar/host/bin/guide

NAME

hg – Take the histogram of a picture

SYNOPSIS

hg [*options*]

DESCRIPTION

hg generates a histogram of the pixels in the frame buffer. It tallies the number of pixels at each intensity level and prints out the minimum and maximum values found for each color.

Options:

–fb *lfbdef*
use logical frame buffer.

–v verbose mode. Prints out the number of pixels found for each of 256 intensities.

–w *xmin xmax ymin ymax*
limit the histogram to the specified window.

–scale produces a form compatible with *scale*(1).

NAME

loop – framebuffer animation tool

SYNOPSIS

loop [*options*]

DESCRIPTION

loop is a framebuffer animation tool. It allows the user to view a series of stored rectangular framebuffer images in sequence, simulating animation. Images are stored in consecutive order within the framebuffer, row by row. The number of images in a single row should be the maximum number that will fit in the width of the framebuffer. Monochromatic images may be viewed using the **bw** option. When this option is used, four sequential images are stored in the red, green, blue, and alpha channels of each image rectangle. The video display freezes on each frame for the specified amount of time (see **kbd** option to modify the number of frames/sec displayed), and then moves to the next frame.

The program can be controlled either via the mouse buttons or the keyboard (see **kbd** option below). The mouse should only be used outside of the window system to avoid any side-effects. If the keyboard option is used, keys '1','2', and '3' correspond to the Left, Middle, and Right buttons on the mouse. The key 'q' corresponds to simultaneously pressing the Left and Right mouse buttons which exits the program.
The Middle button toggles between single step and continuous modes.

Single Step:

Right: forward one frame
Left: backward one frame

Continuous:

Right: changes direction to forward, successive hits cycle speed
Left: changes direction to backward, successive hits cycle speed

The following options are available, with numbers in brackets representing the default values (used if no argument is given).

–blank *n*	Blank frames at end of loop [0]
–bw	Run in black and white (single channel images)
–cont	Start off in continuous mode
–count *n*	Use count instead of mouse
–fsize *x y*	framebuffer memory dimensions *x y* [1024 4096]
–file *commands*	read commands from file
–frames *n*	Number of frames (max that will fit)
–help	Print mouse instructions
–kbd	Use keyboard instead of mouse (will give directions)
–o *x y*	starting frame offset *x y* [0 0] (pixel dimensions)
–rock	Rock loop back and forth
–s *x y*	frame size *x y*
–speeds *S1 S2 S3 S4*	Set loop speeds in frac. of secs [1/24 1/12 1/6 1/3]
–start *n*	Frame to start on
–video	use video speeds [1/30 1/15 1/7.5 1/3.25]
–vsize *x y*	video dimensions *x y*
–zoom *factor*	zoom factor (defaults to fill screen if not specified)

EXAMPLES

To rock back and forth across a 16 frame film loop of 256x256 monochrome images, the following command would be issued: **loop -bw -kbd -rock -frames 16 -s 256 256**

BUGS

Frame size **–fsize** should be settable with **–fb** *fbname*.

NAME

merge – merge two frame buffer windows onto a third

SYNOPSIS

merge [*fgfbname*] *operator* [*bgfbname*] [**to** [*dstfbname*]]
[**–lf** *coeffspec*]
[**–lb** *coeffspec*]
[**–fpt** *xmin ymin*]
[**–bpt** *xmin ymin*]
[**–dpt** *xmin ymin*]
[**–s** *width height*]

DESCRIPTION

Pixels from foreground frame buffer *fgfbname* are merged into the pixels of the (possibly different) background frame buffer *bgfbname*, with output to the (possibly different) frame buffer *dstfbname*.

fgfbname, *bgfbname* and *dstfbname* are frame buffers (discussed in *lfbdefs*(7)), specified either as a quoted delimited string or as the name of a frame buffer in the *LFBDEFS* environment variable. Windowing offsets in a frame buffer are given with the **–fpt, –bpt** and **–dpt** arguments. The size of the merge window is clipped to the intersection of all windows and one of size (*width, height*).

An *operator* is one of the operators listed below, as described in *Compositing Digital Images*, included in the *Pixar Programmer's Manual.* Note that the operators and *to* are keywords to *merge*, so that it is an error for any *lfbdef* to have the same name as an operator (or *to*).

clear	Clear the destination window
copy	Copy the foreground
noop	Copy the background
over	(''merge foreground over background'') Copy both foreground and background, copying foreground where they intersect.
under	(''merge foreground under background'') Copy both foreground and background, copying background where they intersect.
out	(''use foreground held out by background'') Copy those parts of the foreground lying outside the background
in	(''use background held out by foreground'') Copy those parts of the background which intersect the foreground
above	(''copy foreground above background'') Like **in**, but also copies background pixels lying outside the foreground
below	(''copy background above foreground'') Opposite of **above**
xor	(''foreground or background, but not both'') Copies foreground **and** background, except where they intersect.
plus	(''add pixels'') Sums the pixel values.
plusin	(''sum pixels in intersection'') Takes the sum of the two images, writing the result where the background appears.
plusbelow	(''sum pixels above background'') Mix pixels where foreground and background intersect; copy background elsewhere.
plusabove	(''sum pixels above foreground'') Mix pixels where foreground and background intersect; copy foreground elsewhere.

A *coeffspec* gives a weighting coefficient for the channels of the foreground or background. It is designated by either 1, 2, or 4 floating point numbers within slashes. The *coeffspec* / .7 / is equivalent to / .7 .7 .7 .7 /, which effects a dissolve to 70% of each channel. The *coeffspec* / .4 .5 .6 / is equivalent to / .4 .5 .6 1. /, which darkens the pixels to 40% of red, 50% of green, 60% of blue.

SEE ALSO

fbdefs(7), PirlMerge(3H)

NAME

perm – permutations of the frame buffer.

SYNOPSIS

perm [*options*]

DESCRIPTION

perm has several options for permuting the order or rows and columns of a Pixar frame buffer, and some simple image processing (clamping, inversion, ax+b) options. All *perm* routines operate on a single pixel window.

Options:

–clamp Clamp a pixel's components within unit range (0 to 2048).

–not Subtract pixel components from unit range (2048 - value).

–axb *A B*
Compute *Ax+B* for each component *x* (2048 equals a coefficient of 1.0)

–u[p] *n* Circular shift up *n* lines.

–d[own] *n*
Circular shift down *n* lines.

–l[eft] *n*
Circular shift left *n* columns.

–r[ight] *n*
Circular shift right *n* columns.

–nofill Use regular shift instead of circular shift. The shifted window is clipped to the original window and the exposed area remains the same.

–rc Reverse the columns. Exchange the left and right.

–rr Reverse the rows. Exchange the top and bottom.

–shuffle *shufflechan*
Shuffle rgba (e.g., **–shuffle** rgab, **–shuffle** ggrr). Each component from the new pixel is copied from the specified source component. (See *intro*(1).)

–trans Transpose the framebuffer. That is, exchange the lower left with the upper right. This option works on the largest square in the given window, the one with the same upper left corner as the given window.

–ch *selectchan*
See *intro*(1).

–src *xmin xmax ymin ymax* See *intro*(1).

–dst *xmin xmax ymin ymax* See *intro*(1).

–srcfb *fbname* See *intro*(1).

–dstfb *fbname* See *intro*(1).

FILES

/usr/pixar/host/bin/perm

SEE ALSO

fbdefs(7)

BUGS

Only one permutation allowed per invocation. Additional specifications overwrite the previous options.

NAME

pixinit – initialize the pixar and the configuration tables

SYNOPSIS

pixinit

DESCRIPTION

pixinit runs a shell script to initialize the pixar and the configuration tables. *pixinit* has the same effect as issuing the following commands to each appropriate piece of installed hardware.

```
mctrl init mips 5555
video –init –gamma
dumi iena
chconfig –a –k 32
```

All installed hardware is initialized. Any loaded Chap programs and data are lost. The contents of the framebuffer are left intact.

FILES

/usr/pixar/host/bin/pixinit /usr/pixar/host/bin/mctrl /usr/pixar/host/bin/video /usr/pixar/host/bin/dumi

SEE ALSO

mctrl(4), video(1), dumi(1), chconfig(1)

NAME

ramp – ramp framebuffer window horizontally or vertically

SYNOPSIS

ramp [**–ct** [*c*]] [**–cb** [*c*]] [**–cl** [*c*]] [**–cr** [*c*]] [**–ul** [*c*]] [**–ur** [*c*]] [**–dl** [*c*]] [**–dr** [*c*]]
[fB–w *xmin xmax ymin ymax*] [**–ch** *selectchan*] [**–fb** *fbname*]

DESCRIPTION

ramp causes a ramp of colors to be placed in the specified framebuffer window (the entire visible framebuffer by default) and in the specified channels (all of them by default). The ramps are automatically dithered with a 3x3 ordered dither matrix. Several types of ramp are supported.

–ul, ur, dl, dr *color*
The most general case is the bilinear interpolation of the colors at the four corners of the given window, where the up left, up right, down left, and down right colors are specified by these four options. (''Up'' and ''down'' refer to the visual directions.)

–ct, cb, cl, cr *color*
set color of top, bottom, left, and right respectively.

–w *xmin xmax ymin ymax*
See *intro*(1).

–ch *selectchan*
See *intro*(1).

–fb *fbname*
See *intro*(1).

In all cases, the optional argument *c* is short for the usual color specification [red [green blue [alpha]]], where the color (0, 0, 0, 0) is assumed if the argument is not given, the color (red, red, red, red) is assumed if only red is specified, and the color (red, green, blue, 0) is assumed if alpha is not specified. Alternatively, a top-to-bottom ramp may be specified with the **ct** and **cb** options for the top and bottom colors, respectively. Similarly, a left-to-right ramp may be specified with the **cl** and **cr** options.

Three point ramps are possible if the specifications are consistent. For example, the **ul** and **ur** options may be used with the **cb** option rather than with the **dl** and **dr** options set to the same color.

FILES

/usr/pixar/host/bin/ramp

SEE ALSO

fbdefs(7)

NAME

resize – resize utility for portions of the framebuffer

SYNOPSIS

resize [*options*]

DESCRIPTION

resize resizes a source window on the framebuffer to a destination window on the framebuffer. The source window is resized into all the destination windows, and several destination windows may be specified on the command line.

resize also supports multiple destinaiton logical framebuffers. If more than one destination framebuffer is specified (with the **–dstfb** command), then the source framebuffer is copied to each of the destination framebuffers. Naming the logical framebuffers via *LFBDEFS* makes it easier to maintain multiple images on the same physical framebuffer.

If several destination framebuffers and windows are given, each window is relative to the corresponding framebuffer definition. If only one destination framebuffer is given, and multiple windows, each window is defined relative to that framebuffer. Similarly, if only one destination window is given, and multiple destination framebuffers, this window is applied for each framebuffer definition during the resize.

Options:

–ext *x y* Specify and horizontal and vertical filter extent size (default 4 4). Possible filter extents are two and four. The four pixel filter (cubic) gives the best possible resizing, as opposed to the two pixel filter (linear).

–src *xmin xmax ymin ymax*
See *intro*(1).

–dst *xmin xmax ymin ymax*
See *intro*(1).

–ch *selectchan*
See *intro*(1).

See *intro*(1).

EXAMPLES

resize –src 0 255 0 255 –dst 256 300 256 400
This resizes the 256x256 pixels square at the origin of the framebuffer into at 45x145 pixel rectangle at (256,256), using the default 4x4 cubic filter.

resize –srcfb q0 –dstfb q1 –ext 2 2
This resizes the pixel window defined by the *fbdef* q0 into the pixel window defined by *fbdef* q1, using the 2x2 linear filter.

resize –srcfb q1 –src 0 49 0 19 –dst 60 200 60 200
This resizes the 50x10 pixel window *relative to fbdef* q1, into a 201x201 pixel window, relative to the default *fbdef* read from the *FBDEFS* environment variable, using the default 4x4 cubic filter. (See *intro*(1) for a discussion of default settings of frame buffers.)

FILES

/usr/pixar/host/bin/resize

resize -src ∅ 1∅23 ∅ 767 -dst ∅ 595 ∅ 53∅
This resizes a 1024x768 HIDEF image to an NTSC size image.

NAME

rotate – rotate a framebuffer region

SYNOPSIS

rotate [*options*]

DESCRIPTION

rotate rotates, scales,and translates a source window about a center point on the framebuffer to a destination window on the framebuffer. The transformation is performed using a two-pass (horizontal and vertical) resampling algorithm.

Source and destination windows can be specifed for each pass of the resampling algorithm. The source and destination windows for each pass cannot partially overlap. Arguments **src** and **srcfb** define the transformation source window. Arguments **tmp** and **tmpfb** define the destination window for the first (intermediate) pass and the source for the second pass. **dst** and **dstfb** define the the destination for the second (final) pass. If only a **src** argument is given, the transformation is done in place. If a **tmp** argument is not specified, the **dst** argument is used as the destination for both passes; however, if the **dst** window is smaller than the **src** window, a **tmp** window of the size of the **src** window must be specified.

Options:

–src *xmin xmax ymin ymax*
Specify source transformation window

–tmp *xmin xmax ymin ymax*
Specify intermediate transformation window

–dst *xmin xmax ymin ymax*
Specify destination transformation window

–srcfb *lfbdef*
Specify source framebuffer (see FBDEFS(7))

–tmpfb *lfbdef*
Specify intermediate framebuffer

–dstfb *lfbdef*
Specify destination framebuffer

–ext *size*
Specify the filter extent size [default 4]. Possible filter extents are two and four. The four pixel filter (cubic) gives the best possible resizing, as opposed to the two pixel filter (linear).

–a *angle*
Specify the angle (in degrees) to rotate the picture by [default 0.0].

–s *sx sy* Specify the scale factors for the picture [default 1.0,1.0].

–c *cx cy*
Specify the center point of the image for rotation [default center of picture].

–noclr Don't clear the area underneath the destination window (this is faster)

–ch *selectchan*
Use specified channels only (e.g., **–ch rgb, –ch a, –ch AR**).

FILES

/usr/pixar/host/src/bin/rotate.c
/usr/pixar/host/src/lib/libpirl/affine.c

SEE ALSO

PirlRotate(3h), PirlAffine(3h), PirlShear(3h), PWShear(3c), FBDEFS(7)

NAME

scale – scale framebuffer RGBA intensities

SYNOPSIS

scale [**–hi** [*color*]] [**–lo** [*color*]] [**–HI** [*color*]] [**–LO** [*color*]] [**–zhi** [*color*]] [**–zlo** [*color*]] [**–src** *xmin xmax ymin ymax*] [**–dst** *xmin xmax ymin ymax*] [**–ch** *selectchan]* *[*–**srcfb** *fbname*] [**–dstfb** *fbname*]

DESCRIPTION

scale remaps the RGBA channels of a framebuffer by linearly mapping the domain [*lo, hi*] onto the range [*LO, HI*], channel by channel. All colors outside the range [*zlo, zhi*], again on a channel by channel basis, are ignored.

In all cases, the optional argument *color* is short for the usual color specification [red [green blue [alpha]]], where the color (0, 0, 0, 0) is assumed if the argument is not given, the color (red, red, red, red) is assumed if only red is specified, and the color (red, green, blue, 0) is assumed if alpha is not specified.

Options:

–hi [*red*[*green*[*blue*[*alpha*]]]]
Scale given color to **HI** [default = (0,0,0,0)].

–lo [*red*[*green*[*blue*[*alpha*]]]]
Scale given color to **LO** [default = (0,0,0,0)].

–HI [*red*[*green*[*blue*[*alpha*]]]]
Set **HI** color [default = white = (2048,2048,2048,2048)].

–LO [*red*[*green*[*blue*[*alpha*]]]]
Set **LO** color [default = clear = (0,0,0,0)].

–zhi [*red*[*green*[*blue*[*alpha*]]]]
Set **zhi** color [default = white = (2048,2048,2048,2048)].

–zlo [*red*[*green*[*blue*[*alpha*]]]]
Set **zlo** color [default = clear = (0,0,0,0)]. No intensities outside range [**zlo, zhi**] are changed.

–srcfb *fbname*
See *intro*(1).

–dstfb *fbname*
See *intro*(1).

–src *xmin xmax ymin ymax*
See *intro*(1).

–dst *xmin xmax ymin ymax*
See *intro*(1).

–ch *selectchan*
See *intro*(1).

FILES

/usr/pixar/host/bin/scale

SEE ALSO

fbdefs(7), clamp(1)

NAME

see – display an image from raster files of various formats

SYNOPSIS

see [*options*] *file...*

DESCRIPTION

see displays a raster image file that is stored in one of a variety of available formats. This program is ideal for displaying pictures that were not saved using the *sv* command.

Both 8 bit-per-channel and 16 bit-per-channel images can be displayed. With 8 bit-per-channel images, the option is available (with the **–sh** flag), to multiply each channel value. With 16 bit-per-channel images, it may be necessary to swap the bytes in each 16 bit channel word (using the **–swap** flag), depending on the machine used to create the image.

see will also display RGB channel images by filling in the value zero for the alpha channels.

Options:

–seek Skip the first *n* bytes of the file. (Note: *sv* pads each image file with an 8192 byte header.)

–8bw 8 bit black and white image file.

–16bw 16 bit black and white image file.

–8rgb 8 bit RGB image file.

–16rgb 16 bit RGB image file.

–8rgba 8 bit RGBA image file.

–16rgba
16 bit RGBA image file.

–sh Multiply each channel value by 8.

–swap Swap the bytes in each 16 bit channel word.

–fb *fbname*
See *intro*(1).

–w *xmin xmax ymin ymax*
See *intro*(1).

–ch *selectchan*
See *intro*(1).

FILES

/usr/pixar/host/bin/see

SEE ALSO

sv(1), gt(1), gtinfo(1), PirlGetRaster(3H), fbdefs(7)

NAME

sv – save frame buffer into picture file

SYNOPSIS

sv *file* [*options*]

DESCRIPTION

sv saves the picture contained in a frame buffer as a ''tile-based'' picture file on disk with mode 0444. If *file* exists, overwrite permission is requested unless the force option is selected. Files saved with alpha components are flagged as ''matted-to-black'', unless this is explicitly overridden. The created file conforms to Pixar's standards for tile-based picture files (see ''The Format of Stored Pictures'' in the *Pixar Programmer's Manual*) and can be retrieved with the program *gt*.

Options:

–fb *fbname*
See *intro*(1).

–ch *selectchan*
See *intro*(1).

–tu states that the picture is not ''matted-to-black'', that the alpha channel is unassociated with the RGB.

–mode *0ddd*
requests that the file be saved with *0ddd* protection.

–f forces the removal of any existing file.

–v (verbose) elicits typeout of the created file name.

–t *width height*
asks that a specific tile size be used. The default is the size of the picture.

–w *xmin xmax ymin ymax*
See *intro*(1). The default is the size of the frame buffer. The tile size is always limited to be no bigger than the picture size in either dimension.

–l *label* provides a label to be stored in the picture header.

–dump indicates that pixel information should be stored dumped rather than run-length encoded (the default).

–12bit indicates that pixel information should be stored with 12 bits per channel.

–cur *string*
set cursor string.

–nc don't display the cursor while saving picture (the default is to display the cursor).

–host force host to do the encoding. Normally, the Pixar does the encoding.

FILES

/usr/pixar/host/bin/sv

SEE ALSO

gt(1), gtinfo(1), PicCreat(3H), PicRead(3H), PicClose(3H)

DIAGNOSTICS

sv will die if it cannot create the file or open the frame buffer. A *PicCreat* error results if the tilesize is <= 0 in either dimension or the picture size is <= 0 in either dimension.

NAME

tool – framebuffer tool

SYNOPSIS

tool [*options*]

DESCRIPTION

tool is a framebuffer diagnostic tool. It provides a crosshair cursor that may be moved around the framebuffer display under keyboard control, where u=up, d=down, r=right, l=left. The contents of the pixel at the crosshair may be read or written. The display may be zoomed up (centered on the current crosshair location). Following is a complete set of one-key commands available:

u,d,r,l:	up, down, right, or left one pixel
U,D,R,L:	up, down, right, or left N pixels [default=32]
+,<return>,<down arrow>:	alternatives for d
-,^,<up arrow>:	alternatives for u
<space>,<right arrow>:	alternatives for r
<backspace>,<left arrow>:	alternatives for l
<tab>:	alternative R
0,1,2,3,4,5,6,7,8:	set hardware zoom to this value
9:	demagnify without centering
C:	exit program without removing crosshair
h,<home>:	move to screen center (home)
H:	hsv switch
c:	colormap switch
k,K:	crosshair display switch
m:	move to new location ('.' means current value)
M:	exit program without demagnifying
o:	move to screen upper left (origin)
p:	print pixel value and location
q:	exit program
s:	set pixel value
S:	set large xstep, ystep sizes [default=32 30]
v:	verbose switch
!:	escape to Shell for one command
?:	help

Options:

–p *x y* Prints the RGBA at the specified pixel location.

–s *x y* Sets the RGBA at the specified pixel location to the color specified with the **–c** command (see below).

–c [*r* [*g b* [*a*]]]

Specify a color for the –s command in the usual way. I.e., no args means color (0, 0, 0, 0); one arg means (*r, r, r, r*); three args (*r, g, b,* 2048); four args (*r, g, b, a*).

–r *range*

When (r, g, b, a) or (h, s, v, a) is printed to a terminal, this command causes each element to be remapped to [0, range], except for *h*, which is remapped to [0, 6*range]. Range is 2048 by default.

−fb *fbname*
See *intro*(1).

FILES

/usr/pixar/host/bin/tool

BUGS

Hardware zoom values 9-16 not available.

NAME

video – video board utility

SYNOPSIS

video [*options*]

DESCRIPTION

video is a general purpose shell-level interface to the Pixar video board controller. Roughly, this involves setting what area of the framebuffer memory is read out, and how it is interpreted. To find out specifically what the hardware can do, see the video** routines in *libvideo*(3H).

Options:

–file *dev-name*	video device [/dev/video0]
–init	initialize parameters
–base *n*	set base [10]
–red	display red
–green	display green
–blue	display blue
–rgb	display red-green-blue
–alpha	display alpha
–blank	blank video
–width *n*	set width
–height *n*	set height
–twidth *n*	set tile width [32]
–theight *n*	set tile height [24]
–zoom *n*	set magnification [1]
–x *n*	set x offset [0]
–y *n*	set y offset [0]
–start *n*	set starting tile block
–gamma *exponent*	set color map correction [2.3]
–on	turn cursor on
–off	turn cursor off
–ntsc	set ntsc format
–hidef	set high definition format
–freq *n*	set video controller frequency [VFREQ-HIDEF]
–format *n*	set video controller format [VFORM-HIDEF]
–verbose	display current settings

FILES

/usr/pixar/host/bin/video

SEE ALSO

VideoCmap(3H), VideoCursor(3H), VideoDisplay(3H), VideoFormat(3H), VideoOpen(3H)

BUGS

The *twidth* option doesn't stick: it has to be reset on each command if the argument is different than 32.

NAME

intro — introduction to Pixar library functions

DESCRIPTION

This section describes functions that may be found in various Pixar support libraries. There is a manual page for each library, named after the library. For example, *libpirl*(3H) gives a summary of each function in the Pixar Runtime Library.

The archive for each library resides in /usr/pixar/host/lib.

libpixar.a
: low-level routines comprising the basic host interface with the registers, buses, etc. of the Pixar. This library includes the dynamic loader of native Chap routines, routines to manipulate the video parameters and colormap, routines for accessing the diagnostic registers, and much more. However, there is little here that the end-use programmer should need to know about it; almost all the functionality of 'libpixar' is contained in the three libraries below.

libchad.a
: a high-level host-interface library for the Pixar in general and the Chap in particular. 'libchad' contains all 'Chad...' routines, and is required for using 'libpirl' and 'libpicio'.

libpirl.a library containing many C-callable ('Pirl...') routines for performing functions on the Chap. Typically, 'libpirl' functions only require descriptions of one or more sections of frame buffer memory, and the functions take care of all interface tasks, like manipulating Chad.

libpicio.a
: library of functions dealing with moving pictures between the frame buffer and external media, primarily disk files.

The libraries above are listed in order of dependence: to use Chad, 'libpixar' must be included in the list of libaries. Thus, if you write a program called 'myprog', which uses the **Pirl** package, it should have a command line that looks something like:

```
cc -o myprog myprog.o /usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a \
  /usr/pixar/host/lib/libpixar.a
```

libG.a library of general routines. For now, this is simply the resting place for FBGETDEF, the library module that standardizes frame buffer conventions.

SEE ALSO

intro(3), intro(3C), intro(3H), libpixar(3H), libpirl(3H), libpicio(3H), libchad(3H).

NAME

libchad – Pixar resource-management library

DESCRIPTION

Chad is a simplified library for allocating and maintaining dynamic scratchpad and instruction **RAM** on a Chap. *Chad* mimics conventional dynamic memory allocation with certain functional extensions. Routines are provided to allocate and deallocate space in a Chap's instruction space and scratchpad memory, adding facilities for allocating tile blocks and pixel windows, loading scratchpad, Sbus registers, the sysbus and image memory, and linking and executing microcode. This manual page summarizes the routines comprising *Chad* and discusses their calling conventions. Each routine is detailed in another manual page, and there is a tutorial introduction, *Programming the Pixar with Chad*, which is intended as an introduction to the system.

Chad maintains a separate block-storage environment for each Chap attached to a system. The environment is entered by *ChadBegin*(3H) and exited with *ChadEnd*(3H). These functions are detailed in *ChadBegin*(3H).

ROUTINES

Specific resources are allocated by calling *ChadAlloc*(3H) and released by *ChadFree*(3H). All the resources of a given class (**SPAD**, **RAM**, etc.) may be released by calling *ChadReset*(3H). This action is useful in recovering when resources are exhausted, since it releases all resources which were allocated by other processes, in effect performing a complete housecleaning.

The function *ChadBackup*(3H) is used to deallocate all resources younger than its argument. This routine has a role in error recovery, allowing a user routine to free up all resources used by it and any routines it calls.

A resource that has been deallocated for any reason is specially marked: its 'addr' field becomes negative. When this occurs, the resource may be reallocated by *ChadCheck*(3H). Each resource structure maintains information to reconstruct its space, but naturally, *ChadCheck*(3H) can only reallocate space, not initialize it.

Of the deallocation routines, all but *ChadFree*(3H) leave the *Chad* host structures intact, so that the 'addr' field may be checked. All of the above allocation and deallocation routines are documented in *ChadAlloc*(3H).

Once allocated, data may be written using *ChadWrite*(3H), and read using *ChadRead*(3H). In addition, individual pixels may be written to a *ChadFrame* resource using the macro **CHAD_SETPXL** and read using **CHAD_GETPXL**. All four of these procedures are listed in *ChadWrite*(3H).

Resources of type RAM (i.e., Chap functions, listed in section 3C of the Pixar manual pages) may be executed with *ChadGo*(3H). Execution proceeds asynchronously, with *ChadGo*(3H) returning before the Chap routine completes. The status of execution is checked with *ChadCPUBusy*(3H), which is non-zero as long as a routine is still running. The function *ChadWaitCPU*(3H) provides a busy wait, which does not return until the Chap routine completes. The manual page for these execution routines is *ChadGo*(3H).

All *Chad* routines return an error code, which is **NULL (CHAD_NOERROR)** for normal return. Once detected, a message explaining the error can be sent to a file with *ChadErrReport*(3H).

UNIVERSAL TRUTHS

Since several Chaps may be attached to a host, and since *Chad* maintains a separate environment for each, it is necessary to distinguish among them. This is done, where appropriate, with tokens of type *ChapID*. *ChadBegin*(3H), *ChadEnd*(3H), *ChadRead*(3H), *ChadWrite*(3H), *ChadReset*(3H), *ChadCPUBusy*(3H), *ChadWaitCPU*(3H) and *ChadReset*(3H) all require such a token as their first argument.

When pixel values are passed to *Chad* routines, it is in the form of the *RGBAPixelType* datatype defined in <pixeldef.h>

Most *Chad* routines take a variable number of arguments, allowing, for example, several allocations to be performed with a single function call. The last argument to each of these routines must be the special token, *NIX*. The arguments to *ChadAlloc*(3H), *ChadRead*(3H), *ChadWrite*(3H) are arranged in groups, called *resource specifications*. Each specification begins with a type token like **SPAD** or **RAM**. This is followed by a type-dependent number of arguments laid out in the manual pages. With the exception of the *FRAME* specification to *ChadRead*(3H) and *ChadWrite*(3H), the arguments of a specification are fixed in number and type.

All *Chad* resource types share certain characteristics. First, each has a field, named 'addr'. This field provides information to *Chad*, and it is set to an invalid (negative) value when the resource is deallocated. Second, each resource has a 'new' field which is set non-zero when a resource is first allocated and whenever it is reallocated. This allows the user to perform any initialization necessary. Finally, each resource retains the parameters to *ChadAlloc*(3H) used to allocate it; the function *ChadCheck*(3H) will recover resources up to but not including any initialization by the user.

The document *Programming the Pixar with Chad* discusses the principles behind *Chad* and gives operational examples. The manual pages listed below give more terse explanations.

ERRORS

All *Chad* routines return an error code, which is **NULL (CHAD_NOERROR)** for normal return. Once detected, a message explaining the error can be sent to a file with *ChadErrReport*(3H).

LIBRARIES

/usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

ChadBegin(3H), ChadAlloc(3H), ChadWrite(3H), ChadGo(3H), ChadErrReport(3H), ChadFrame(3H)

intro (1)	– list of shell-callable Pixar programs
intro (3C)	– list of libraries of device-resident routines
intro (3H)	– list of libraries of host-resident routines

LIST OF FUNCTIONS

Name	*Appears on Page*	*Description*
CHAD_GETPIXEL()	ChadFrame(3H)	– get a pixel from Pixar image memory
CHAD_SETPIXEL()	ChadFrame(3H)	– set a pixel in Pixar image memory
ChadAlloc()	ChadAlloc(3H)	– allocate Chad resources
ChadBackup()	ChadAlloc(3H)	– release recently-allocated resources
ChadBegin()	ChadBegin(3H)	– enter the Chad environment
ChadCPUBusy()	ChadGo(3H)	– is the previous Chap routine still executing?
ChadCPUWait()	ChadGo(3H)	– wait for the last Chap routine to complete
ChadCheck()	ChadAlloc(3H)	– confirm the continued existence of Chad resources
ChadEnd()	ChadBegin(3H)	– leave the Chad environment
ChadErrReport()	ChadErrReport(3H)	– explain an error by Chad
ChadFrame()	ChadFrame(3H)	– discussion of Chad frames
ChadFree()	ChadAlloc(3H)	– free Chad resources
ChadGo()	ChadGo(3H)	– execute a Chap routine
ChadLibs()	ChadAlloc(3H)	– include an archive in Chad's search list
ChadRead()	ChadWrite(3H)	– read Chap resources
ChadReset()	ChadAlloc(3H)	– reset a Chap
ChadWrite()	ChadWrite(3H)	– write to resources on the Chap

NAME

ChadAlloc(),
ChadFree(),
ChadLibs(),
ChadReset(),
ChadCheck(),
ChadBackup() – Chad resource allocation routines

SYNOPSIS

```
# include "/usr/pixar/include/chad.h"
ChadError ChadAlloc (chapid,
[SPAD, blockpp, nwords,]
[RAM, pcpp, sym,]
[PIXELS, blockpp, npixels,]
[TB, tbpp, firsttile, tileswide, tileshigh,]
[PW, pwpp, tbpp, xmin, xmax, ymin, ymax,]
[FRAME, framepp, tbpp, xmin, xmax, ymin, ymax,]
  NIX)
  ChadSpad *(*blockpp);
  ChadPC *(*pcpp);
  ChadTB *(*tbpp);
  ChadPW *(*pwpp);
  ChadFrame *(*framepp);
  int nwords, npixels, firsttile, tileswide, tileshigh,
     csr, xmin, xmax, ymin, ymax;
  char *sym;

ChadError ChadFree ([blockp,] [pcp,] [tbp,] [pwp,] [framep,] NIX)
ChadSpad *blockp;
ChadPC *pcp;
ChadTB *tbp;
ChadPW *pwp;
ChadFrame *framep;

ChadError ChadLibs (lib1,...,libN,NIX)
char *lib1,...,libN;

ChadError ChadReset (chapid, [RAM,] [SPAD,] [TB,] [PW,] NIX)
ChapID chapid;

ChadError ChadCheck (chapid, [blockp,] [pcp,] [tbp,] [pwp,] [framep,] NIX)
ChapID chapid;
ChadSpad *blockp;
ChadPC *pcp;
ChadTB *tbp;
ChadPW *pwp;
ChadFrame *framep;

ChadError ChadBackup(structp);
union {
  ChadSpad spad;
  ChadPC pc;
  ChadTB tb;
  ChadPW pw;
  ChadFrame frame;
} *structp;
```

DESCRIPTION

The routines *ChadAlloc* and *ChadFree* manage resources of several types on a Pixar's Chap processor. Several storage requests are combined in a single call using the allocation requests above. A *SPAD* request allocates dynamic space in the Chap's scratchpad memory. *PIXELS* requests are similar, but expressed in multiples of 4 words. *RAM* requests invoke the Chap dynamic loader (see *chap*(4) for more information) to load routines into the Chap's instruction memory, resolving undefined symbols from a list of libraries maintained by *Chad.* This list initially includes *libcolor.a, libpG.a, libpip.a, libpm.a, libpt.a* and *libpx.a* in '/usr/pixar/chap/lib'. Other libraries may be prepended to the list by calling *ChadLibs,* giving as arguments a set of full Unix file pathnames denoting the libraries to be used.

The *TB* specification requests that a Tile Block be allocated on the Chap. The tile block is a set of tiles of image memory, each tile being 32x32 pixels square. The specification gives the first tile, the number of tiles required in a row and the number of rows required.

PW gets a pixel window, a rectangular region of pixels within a tile block. This is specified by giving a rectangle expressed in pixels, with pixel (0,0) defined as the upper left corner of the tile block.

A *ChadFrame* (specified by *FRAME*) is similar to a pixel window, but maintains the notion of a current pixel which can be used for reading and writing into image memory. It is discussed in *ChadFrame*(3H).

Several routines for deallocating resources are provided, tailored to different situations. *ChadFree* is unique in that it 1) deallocates only those resources in its argument list, and 2) frees the *Chad* structure associated with the resource, so that the pointers passed to it are no longer valid.

Of the remaining deallocation routines, *ChadReset* is the most drastic. Rather than *Chad* pointers, *ChadReset* takes a (possibly empty) set of type tokens, deallocating all resources of those classes. This is useful primarily for correcting storage leaks. If no type tokens appear in the argument list, *ChadReset* performs a device reset of the Chap, deallocating all the resources of all types on that Chap, and also rendering invalid any resources being used by any other process running concurrently on that Chap. Under some circumstances, this can be considered unfriendly.

ChadBackup takes a single *Chad* pointer as it argument. The denoted resource is deallocated, together with all resources of any type allocated after it. This action is useful for error recovery, when a function must clean up all its resources and those allocated by those functions it calls.

All *Chad* resources contain an 'addr' field. When a resource is deallocated, this field is set to an invalid value, which is less than 0. Thus, the application can always check the continued survival of a resource without consulting *Chad* (It is an error to use a deallocated resource). However, *Chad*s have no access to resources in other processes, and so the 'addr' field can remain valid even when its resource has disappeared (this can only occur if more than one process is using a Chap at a time). A mechanism is thus required for confirming that the 'addr' field still has meaning. *ChadCheck* confirms the validity of the resources in its argument list; if any have been deallocated, it consults the *Chad*s structure associated with the resource and calls *ChadAlloc* to recover the resource if possible. Naturally, it is unable to reinitialize any data which may have been lost with the deallocation. Each structure contains a 'new' field, set to nonzero when the resource is reallocated.

The document *Programming the Pixar with Chad* discusses the principles behind *Chad* and gives operational examples. The manual pages listed below give more terse explanations, with *chad*(3H) being the most complete.

LIBRARIES

/usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

ERRORS

All *Chad* routines return an error code, which is **NULL (CHAD_NOERROR)** for normal return. Once detected, a message explaining the error can be sent to a file with *ChadErrReport*(3H).

SEE ALSO

ChadBegin(3H), libchad(3H), ChadWrite(3H), ChadGo(3H), ChadErrReport(3H), ChadFrame(3H)

The *Chad Tutorial*, in *The Pixar User's Manual*, serves as an introduction to Pixar programming using the Chad routines. Also recommended is the *Chap Programming Tutorial*, in the same source, which discusses, indirectly, many of the tasks **Chad** performs invisibly.

NAME

ChadBegin,
ChadEnd – Pixar resource-management library

SYNOPSIS

 # include "/usr/pixar/include/chad.h"
ChadError ChadBegin(chapid, exclusive)
ChapID chapid;
int exclusive; 0 => shared use of CHAP by processes owned by me; 1 => exclusive use of CHAP by this process

ChadError ChadEnd(chapid)
ChapID chapid;

DESCRIPTION

The *Chad* runtime library maintains an environment for protecting and managing resources used by a program on a Pixar's Chap processor. The two functions *ChadBegin* and *ChadEnd* initialize and terminate this environment for a particular Chap.

The document *Programming the Pixar with Chad* discusses the principles behind *Chad* and gives operational examples. The manual pages listed below give more terse explanations, with *chad*(3H) being the most complete.

LIBRARIES

/usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

ERRORS

All *Chad* routines return an error code, which is **NULL (CHAD_NOERROR)** for normal return. Once detected, a message explaining the error can be sent to a file with *ChadErrReport*(3H).

SEE ALSO

libchad(3H), ChadAlloc(3H), ChadWrite(3H), ChadGo(3H), ChadErrReport(3H), ChadFrame(3H)

The *Chad Tutorial*, in *The Pixar User's Manual*, serves as an introduction to Pixar programming using the Chad routines. Also recommended is the *Chap Programming Tutorial*, in the same source, which discusses, indirectly, many of the tasks **Chad** performs invisibly.

NAME

ChadErrReport　　– describe an error by Chad

SYNOPSIS

 # include "/usr/pixar/include/chad.h"
ChadError ChadErrReport(fp)
FILE *fp;

DESCRIPTION

When *Chad* encounters an error, *ChadErrReport* prints an explanation of the error to the given file. The error code is assumed to be the value of *ChadLastErr*, as set (and returned) by all other **Chad** routines.

The document *Programming the Pixar with Chad* discusses the principles behind *Chad* and gives operational examples. The manual pages listed below give more terse explanations, with *chad*(3H) being the most complete.

LIBRARIES

/usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

libchad(3H), ChadBegin(3H), ChadAlloc(3H), ChadWrite(3H), ChadGo(3H), ChadFrame(3H)

The *Chad Tutorial*, in *The Pixar User's Manual*, serves as an introduction to Pixar programming using the Chad routines. Also recommended is the *Chap Programming Tutorial*, in the same source, which discusses, indirectly, many of the tasks **Chad** performs invisibly.

NAME

ChadFrame,
CHAD_SETPIXEL,
CHAD_GETPIXEL – access to Pixar image memory

SYNOPSIS

```
# include "/usr/pixar/include/chad.h"
ChadError ChadAlloc(chapid, FRAME, framepp, tbpp, xmin, xmax, ymin, ymax, NIX)
ChapID chapid;
ChadFrame *(*framepp);
ChadTB *(*tbpp);
short xmin, xmax, ymin, ymax;

ChadError ChadWrite(chapid,
[ FRAME, framep, FRX, xval, ]
[ FRAME, framep, FRY, yval, ]
[ FRAME, framep, FRCSR, csr, ]
[ FRAME, framep, FRBFR, buffer, xmin, xmax, ymin, ymax, ]
  NIX)
ChapID chapid;
ChadFrame *framep;
RGBAPixelType buffer[][];
short xval, yval, csr, xmin, xmax, ymin, ymax;

ChadError ChadRead(chapid,
[ FRAME, framep, FRX, xvalp, ]
[ FRAME, framep, FRY, yvalp, ]
[ FRAME, framep, FRCSR, csrp, ]
[ FRAME, framep, FRBFR, buffer, xmin, xmax, ymin, ymax, ]
  NIX)
ChapID chapid;
ChadFrame *framep;
RGBAPixelType buffer[][];
short *xvalp, *yvalp, *csrp, xmin, xmax, ymin, ymax;

CHAD_SETPIXEL(framep, red, green, blue, alpha)
ChadFrame *framep;
unsigned short red, green, blue, alpha;

CHAD_GETPIXEL(framep, red, green, blue, alpha)
ChadFrame *framep;
unsigned short red, green, blue, alpha;
```

DESCRIPTION

The *ChadFrame* resource manages image memory, allowing programs to directly access pixels of Pixar image memory. Associated with the frame is a *current pixel*, given as an x/y offset within the frame. The frame is allocated with the call to *ChadAlloc*(3H) above, giving a pointer to a frame pointer, a pointer to a tile block pointer, and the bounding box of the frame relative to the coordinates of the tile block, where (0,0) is the upper left corner. The current pixel is set using *ChadWrite*(3H), which can also be used to write a rectangular region of the frame by passing a pointer to sufficient host storage, and the bounding rectangle to be written. For example,

ChadWrite(*CHAP0*, *FRAME*, fp, *FRBFR*, bufr, 20, 25, 30, 40, *NIX*);

would write 11 lines of 6 pixels each from the buffer *bufr*. All *ChadWrite*(3H) operations on frames have the obvious converse in *ChadRead*(3H).

Individual pixels of the frame can be written using *CHAD_GETPIXEL* and *CHAD_SETPIXEL*. These macros access the current pixel. The address of the current pixel can be set explicitly with *ChadWrite*(3H), as above. However, the address can also be set implicitly. The addressing register *CSR* controls automatic incrementation of the current pixel: as desired, the current pixel can be incremented or decremented in X or Y, after reads or writes. These actions are controlled by a bit mask obtained by the bitwise *or* of a set of predefined constants. Automatic address modification is enabled by **RP_ADDR_MOD**. Modification upon writing is set with **RP_WRITE_ADDR_MOD**, otherwise modification occurs upon pixel reads. The address is incremented if **RP_INC_ADDR_MOD** is set, and decremented otherwise. The first coordinate incremented or decremented is X if the **RP_X_ADDR_MOD** is set, otherwise modification occurs first in Y. Finally, when the **RP_AUTO_CR** bit is set, the second coordinate is modified whenever the first coordinate reaches its boundary. For example, the call

ChadWrite(*ChadOwner*(fp), *FRAME*, fp, *FRCSR*, (**RP_INC_ADDR_MOD | RP_X_ADDR_MOD | RP_AUTO_CR**), *NIX*);

sets the addressing mechanism to automatically pass through an image in scan line order, from top left to bottom right: pixel coordinates are incremented first in X then Y, and Y is incremented automatically at the end of each scan line (and, of course, X is restarted at the beginnning of the next scan line).

The document *Programming the Pixar with Chad* discusses the principles behind *Chad* and gives operational examples. The manual pages listed below give more terse explanations, with *chad*(3H) being the most complete.

LIBRARIES

/usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

ERRORS

All *Chad* routines return an error code, which is **NULL** (**CHAD_NOERROR**) for normal return. Once detected, a message explaining the error can be sent to a file with *ChadErrReport*(3H).

SEE ALSO

libchad (3H), ChadBegin(3H), ChadAlloc(3H), ChadGo(3H), ChadErrReport(3H)

The *Chad Tutorial*, in *The Pixar User's Manual*, serves as an introduction to Pixar programming using the Chad routines. Also recommended is the *Chap Programming Tutorial*, in the same source, which discusses, indirectly, many of the tasks **Chad** performs invisibly.

NAME

ChadGo,
ChadCPUBusy,
ChadWaitCPU – execute routines on a Chap

SYNOPSIS

 # include "/usr/pixar/include/chad.h"
ChadError ChadGo(pcp)
ChadPC *pcp;

ChadCPUBusy(chapid)
ChapID chapid;

ChadWaitCPU(chapid)
ChapID chapid;

DESCRIPTION

Previously-allocated *RAM* resources (Chap functions) allocated using *ChadAlloc*(3H) may be executed by calling *ChadGo*.

This execution routine is asynchronous, returning immediately when the Chap is started so that the host may continue its work concurrently. Synchronization may be accomplished by using *ChadWaitCPU*(3H) to busily await the completion of the Chap on its appointed routines, or *ChadCPUBusy*(3H) to test whether the Chap is active.

ChadCPUBusy and ChadWaitCPU are macros **#defined** in the include file chad.h.

The document *Programming the Pixar with Chad* discusses the principles behind *Chad* and gives operational examples. The manual pages listed below give more terse explanations, with *chad*(3H) being the most complete.

LIBRARIES

/usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

ERRORS

All *Chad* routines return an error code, which is **NULL (CHAD_NOERROR)** for normal return. Once detected, a message explaining the error can be sent to a file with *ChadErrReport*(3H).

SEE ALSO

libchad(3H), ChadBegin(3H), ChadAlloc(3H), ChadWrite(3H), ChadErrReport(3H), ChadFrame(3H)

The *Chad Tutorial*, in *The Pixar User's Manual*, serves as an introduction to Pixar programming using the Chad routines. Also recommended is the *Chap Programming Tutorial*, in the same source, which discusses, indirectly, many of the tasks **Chad** performs invisibly.

NAME

ChadWrite,
ChadRead – write/read Chap resources

SYNOPSIS

```
# include "/usr/pixar/include/chad.h"
ChadError ChadWrite (chapid,
[ SPAD, blockp, val, offset, ]
[ SPADARRAY, blockp, vals, nwords, offset, ]
[ PIXELS, blockp, pxvals, npixels, offset, ]
[ FRAME, framep, FRX, x, ]
[ FRAME, framep, FRY, y, ]
[ FRAME, framep, FRCSR, csr, ]
[ FRAME, framep, FRBFR, pxvals, xmin, xmax, ymin, ymax, ]
[ SYSBUS<0..15>, val, ]
[ R<0..31>, proc, val, ]
[ ACC, proc, val, ]
[ B<0..15>, val, ]
[ I<0..15>, val, ]
  NIX)

ChapID chapid;
ChadSpad *blockp;
int proc, nwords, npixels, offset, x, y, csr,
   xmin, xmax, ymin, ymax;
ChadFrame *framep;
CHAPVAL val, vals[ ];
RGBAPixelType pxvals[ ];

ChadError ChadRead(chapid,
[ SPAD, blockp, valp, offset, ]
[ SPADARRAY, blockp, vals, nwords, offset, ]
[ SPADTAB, blockp, vals, nwords, offset, ]
[ PIXELS, blockpp, pxvals, npixels, offset, ]
[ FRAME, framep, FRX, xp, ]
[ FRAME, framep, FRY, yp, ]
[ FRAME, framep, FRCSR, csrp, ]
[ FRAME, framep, FRBFR, pxvals, xmin, xmax, ymin, ymax, ]
[ SYSBUS<0..13>, valp, ]
[ R<0..31>, proc, valp, ]
[ ACC, proc, valp, ]
[ B<0..15>, valp, ]
[ I<0..15>, valp, ]
  NIX)

ChapID chapid;
ChadSpad *blockp;
int offset, nwords, npixels, proc, *xp, *yp, *csrp, xmin, xmax, ymin, ymax;
ChadFrame *framep;
RGBAPixelType pxvals[ ];
CHAPVAL *valp, vals[ ];
```

DESCRIPTION

Once *Chad* resources have been allocated, *ChadWrite* (3H) will download data to them, and *ChadRead* (3H) will read data from them. In addition to resources explicitly allocated with *ChadAlloc* (3H), these functions will address various system registers: **r0** through **r31** and **acc** refer to four-way ALU registers and accumulator, respectively; **b0** through **b15** and **i0** through **i15** refer to the base and index registers associated with access to scratchpad memory; *SYSBUS0* through *SYSBUS13* denote the memory-mapped sysbus registers of the Chap. *SYSBUS14* and *SYSBUS15* are reserved and unavailable via *Chad.* The *proc* parameter of the ALU registers is a bit mask indicating which processor gets or provides the value: *CHAD_PROCR*, *CHAD_PROCG*, *CHAD_PROCB* and *CHAD_PROCA* write to the red, green, blue and alpha processors, respectively. These may be combined bitwise to select any set of processors; *CHAD_ALLPROCS* is such a bit mask for them all.

SPAD writes individual words to Scratchpad memory, while *SPADARRAY* writes to contiguous blocks of words. *SPADTAB* does the same, but writes them in a manner consistent with the 'index' access mode of Chap routines. *SPADTAB* accesses are rare. *PIXELS* also reads and writes scratchpad, but in units of four words (pixels).

The argument specifications denoted by *FRAME* refer to frames of image memory. The *FRX* and *FRY* specifications set the x and y coordinate of the current pixel, where this is defined relative to the boundaries of the frame. The *FRCSR* argument sets the access register of the frame, giving access permission and address-modification information for pixel access. Further details are available in *ChadFrame*(3H).

The document *Programming the Pixar with Chad* discusses the principles behind *Chad* and gives operational examples. The manual pages listed below give more terse explanations, with *chad*(3H) being the most complete.

LIBRARIES

/usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

ERRORS

All *Chad* routines return an error code, which is **NULL (CHAD_NOERROR)** for normal return. Once detected, a message explaining the error can be sent to a file with *ChadErrReport*(3H).

SEE ALSO

libchad (3H), ChadBegin(3H), ChadAlloc(3H), ChadGo(3H), ChadErrReport(3H), ChadFrame(3H)

The *Chad Tutorial*, in *The Pixar User's Manual*, serves as an introduction to Pixar programming using the Chad routines. Also recommended is the *Chap Programming Tutorial*, in the same source, which discusses, indirectly, many of the tasks **Chad** performs invisibly.

NAME

getchaps(), getvideo(), getdumi(), getmctrl(), getdiskw() – PIXAR special filename determination routines

SYNOPSIS

getchaps(chaparray)
  char *(* chaparray[]);

getvideo(vidarray)
  char *(* vidarray[]);

getdumi(dumiarray)
  char *(* dumiarray[]);

getmctrl(mctrlarray)
  char *(* mctrlarray[]);

getdiskw(diskwarray)
  char *(* diskwarray[]);

DESCRIPTION

The routines *getchaps(), getvideo(), getdumi(), getmctrl(),* and *getdiskw()* are used to determine the special filenames for PIXAR interfaces. *getchaps()* finds all attached Chap names. *getvideo()* finds all attached video board names. *getdumi()* finds all attached DUMI interface board names. *getmctrl()* finds all attached memory controller board names. *getdiskw()* finds all attached disk window (disk buffer) names. These routines are in the libchad.a library.

The routines are called with the address of a variable declared to be a pointer to an array of character pointers. This variable is filled in with the address of a dynamically allocated array of pointers to the appropriate device names. A count of found device names is returned. If no device names are found, a count of 0 is returned.

The list of found devices is sorted by minor device number, from lowest to highest. Devices are found by searching the /dev directory for all special files which have a specific major device number, and, where appropriate, testing for the presence of hardware.

If the operating system is configured so that the PIXAR major device number is some value other than 34, a special file named */dev/dumi0* must exist with the correct PIXAR major device number. This file, if it exists, provides the PIXAR major device number for the search. If the file does not exist, a default major device number of 34 is assumed.

EXAMPLE

```
/* Print out the names of all Chap devices */
#include <stdio.h>

main()
{
   char ** array;
   int cnt;

   cnt = getchaps( &array );
   while( cnt-- )
      puts( *array++ );
}
```

ERRORS

A count of zero is returned if no device names are found. This indicates an abnormal condition for a host equipped with a PIXAR Image Computer. Check to see if the special files for the PIXAR interface exist in

/dev. Make sure that the PIXAR computer cage is connected and has power applied.

SEE ALSO

CHAP(4G), DUMI(4), MCTRL(4G), VIDEO(4G)

LIBRARY

libchad.a

NAME

FbGetDef,
FbSetFbDefs,
FbSetLfbDefs,
FbSetFbPath,
FbSetLfbPath – routines for defining and getting framebuffer definitions.

SYNOPSIS

```
#include <fbdefs.h>

LfbDefType * FbGetDef(fbname)
char *fbname;

void FbSetFbDefs(fbdefs)
char *fbdefs;

void FbSetLfbDefs(lfbdefs)
char *lfbdefs;

void FbSetFbPath(fbpath)
char *fbpath;

void FbSetLfbPath(lfbpath)
char *lfbpath;
```

DESCRIPTION

FbGetDef searches for *fbname* in the LFBPATH described in *fbdefs*(7), and returns a pointer to the *lfbdef* structure which describes the framebuffer information associated with *fbname*. The *pfb* element in the structure is a pointer to its *pfbdef*. The *pwptr* of the *LfbDefType* and the *tbptr* of the *FbDefType* are not set by *FbGetDef*, but may be used by the application program to store PW and TB pointers. If the *fbname* is a complete *fbdef* which is not in the *LFBPATH*, the *fbdef* is added to the ILFBDEFS, and a pointer to the structure is returned. If the *fbname* is only a name and is not in the *LFBPATH*, a null pointer is returned.

FbSetFbDefs takes a colon-separated list of *fbdefs* and prepends it to the environment *FBDEFS*.

FbSetLfbDefs takes a colon-separated list of *lfbdefs* and prepends it to the environment *LFBDEFS*.

FbSetFbPath takes a colon-separated list of *fbnames* and prepends it to the environment *FBPATH*.

FbSetLfbPath takes a colon-separated list of *lfbnames* and prepends it to the environment *LFBPATH*.

LIBRARY

libg.a

SEE ALSO

fbdefs(7), TB(3C), PW(3C)

NAME

libpicio – picture encoding library

DESCRIPTION

libpicio is a library of C-callable functions for encoding, decoding, loading and unloading picture files. The following documentation exists for the picture encoding library routines:

PicCreat (3H) discusses routines *PicCreat*, *PicOpen*, and *PicFind* for creating and opening picture files.

PicRead (3H) documents the *PicReadBuffer* and *PicWriteBuffer* routines for communications between picture tiles on disk and RGBA arrays in a program.

PicClose (3H) documents the *PicClose* routine.

Note the parallelism between the above routines and the creat, open, read, write, and close routines for normal files.

picDecode (3H) discusses the routines *picPreDecodeScanline*, *picDecodeScanline*, and *picPostDecodeScanline* used for decoding individual scanlines of picture tiles on disk. These routines are provided as a more convenient alternative to the *PicRead* call.

picEncode (3H) discusses the routines *picPreEncodeScanline*, *picEncodeScanline*, and *picPostEncodeScanline* used for encoding individual scanlines of picture tiles on disk. These routines are provided as a more convenient alternative to the Write call.

PicLabel (3H) discusses the routines *PicLseekLabel*, *PicReadLabel*, and *PicWriteLabel*, used for general reading and writing of arbitrary text information in the picture file.

The header file *picio.h* includes all the picture file definitions needed for programs using *picio* routines. The file header addresses, specifying the position of the label, picture descriptor, and tile map, for example, are listed in this file. User access to header information should be done through library routines; the addresses are convenient for visual decoding of an octal dump.

The major structure upon which pictures are based is the *PictureDescriptor*, defined in *picio.h*. Each picture file on disk contains a *PictureDescriptor* in its header; each picture file referred to in a program is accessed via a pointer to a *PictureDescriptor* held in memory. User access to this structure is normally handled via library routines; occasional reading of individual fields is best handled directly.

The currently stored fields of a *PictureDescriptor* are the height and width of the picture, the height and width of the tiles, the picture's format, storage, blocksize, and matting indicator, and the x-y offsets of the picture. *PictureDescriptors* for open picture files include an open file number. The picture height (*Pheight*) and width (*Pwidth*) are the height and width of the picture in pixels. The tile height (*Theight*) and width (*Twidth*) are the height and width of the tiles in pixels. Aside from being positive, these tile sizes have no restrictions. If not evenly divisible into the corresponding picture sizes, the rightmost and bottommost tiles will contain encoded pixel information not in the picture. We suggest that tiles not exceed 512 pixels in either dimension, so that 512-resolution frame buffers can be used for processing.

The picture format (*Pformat*) is a short which contains some union of the single bit entities **PF_R (8), PF_G (4), PF_B (2), PF_A (1)**, all of which are defined in *picio.h*. The picture storage (*Pstorage*) indicates whether the stored file is 8 or 12 bits, encoded or dumped. Appropriate macros (**PF_8BIT**, etc.) are included. Pictures are blocked to speed their recovery; the blocksize (*Pblocksize*) is tuned to the machine upon which the picture file was created (1024 on a VAX). The matting indicator (*Pmatting*) is important for files including both color and alpha information. The matte can be unassociated (**PM_NONE**) from the picture, associated (**PM_MTB**) in the standard *matted-to-black* fashion, or indirect (**PM_IND**) through the color map.

The x-y offsets (*Xoffset, Yoffset*) indicate how many pixels to translate the upper left corner of the picture from the standard upper left corner origin of the frame buffer.

Further macros listed in *picio.h* aid in computing the number of tiles (**PM_NTILES(pdptr)**) and number of components (**PM_NOFC(pdptr)**) of a picture.

LIBRARY

/usr/pixar/host/lib/libpicio.a

SEE ALSO

intro (1)	– list of shell-callable Pixar programs
intro (3C)	– list of libraries of device-resident routines
intro (3H)	– list of libraries of host-resident routines

PicClose (3H), PicCreat (3H), PicLabel (3H), PicRead (3H), picDecode (3H), picEncode (3H).

LIST OF FUNCTIONS

Name	*Appears on Page*	*Description*
PicClose	PicClose(3H)	– close a picture file
PicCreat	PicCreat(3H)	– create/open a picture file
PicFind	PicCreat(3H)	– open a picture file, searching for it using PIXPATH
PicGetFrame	PicFrame(3H)	– get pictures from frame buffer to picture file
PicLseekLabel	PicLabel(3H)	– determine the length of a picture label
PicOpen	PicCreat(3H)	– open a picture file
PicPutFrame	PicFrame(3H)	– put pictures from picture file into frame buffer
PicReadBuffer	PicRead(3H)	– read a picture tile
PicReadLabel	PicLabel(3H)	– read a picture label
PicSetForce	PicCreat(3H)	– force a picture-file overwrite
PicSetLabel	PicCreat(3H)	– label a picture
PicSetOffset	PicCreat(3H)	– set a picture's offset
PicSetPformat	PicCreat(3H)	– set format of a picture
PicSetPmatting	PicCreat(3H)	– set a picture matting indicator
PicSetPsize	PicCreat(3H)	– set size of a picture
PicSetPstorage	PicCreat(3H)	– set a picture storage flag
PicSetTsize	PicCreat(3H)	– set tile size of a picture
PicWriteBuffer	PicRead(3H)	– write a picture tile
PicWriteLabel	PicLabel(3H)	– write a picture label
picDecodeScanline	picDecode(3H)	– decode a picture scan line
picEncodeScanline	picEncode(3H)	– encode a picture scan line
picPostDecodeScanline	picDecode(3H)	– finish decoding a picture tile
picPostEncodeScanline	picEncode(3C)	– finish encoding a picture tile
picPreDecodeScanline	picDecode(3C)	– start decoding a picture tile
picPreEncodeScanline	picEncode(3C)	– start encoding a picture tile

NAME

PicClose　　– close a picture file

SYNOPSIS

#include <picio.h>

PicClose(channel)
PFILE *channel;

DESCRIPTION

PicClose closes the picture file on this *channel.*

LIBRARY

/usr/pixar/host/lib/libpicio.a

SEE ALSO

libpicio (3H)　　– overview of the picio library
PicCreat (3H), PicLabel (3H), PicRead (3H), picDecode (3H), picEncode (3H).

DIAGNOSTICS

This routine returns 0 if *channel* is not associated with an output file.

NAME

PicCreat,
PicSetPsize,
PicSetTsize,
PicSetPformat,
PicSetPstorage,
PicSetPmatting,
PicSetOffset,
PicSetForce,
PicSetLabel,
PicOpen,
PicFind – create/open a picture file

SYNOPSIS

```
#include <picio.h>

PFILE *PicCreat(filename, mode)
char *filename;
int mode;

PicSetPsize(Pwidth,Pheight)
int Pwidth,Pheight;

PicSetTsize(Twidth,Theight)
int Twidth,Theight;

PicSetPformat(PictureFormat)
int PictureFormat;

PicSetPstorage(PictureStorage)
int PictureStorage;

PicSetPmatting(PictureMatting)
int PictureMatting;

PicSetOffset(Xoffset,Yoffset)
int Xoffset,Yoffset;

PicSetForce(ForcedRemovalFlag)
int ForcedRemovalFlag;

PicSetLabel(labelptr)
char *labelptr;

PFILE *PicOpen(filename, type)
char *filename, *type;

PFILE *PicFind(filename, type)
char *filename, *type;

char *PicFindName;
```

DESCRIPTION

PicCreat creates a new picture file or prepares to rewrite an existing file called *filename*, and associates a picture descriptor (PFILE) with it. *PicCreat* returns a pointer to identify the picture descriptor in subsequent operations. If the file did not exist, it is given the mode *mode*, as modified by the process's mode mask. If the file exists and is protected against overwriting, the routine will ask permission to overwrite unless a call *PicSetForce(TRUE)* has been made.

The picture width (512), picture height (488), tile width (512), tile height (488), picture format (**PF_RGBA**), picture storage flag (**PS_8BIT**), picture matting indicator (**PM_MTB**), and picture offsets (0,0) (see <picio.h>) are all stored in the picture descriptor associated with this channel. The default values are shown in parentheses; these can be changed with calls to the various *PicSet...* routines.

An ASCII picture label is written if *PicSetLabel* has been called, passing a label pointer.

The picture format is one of (**PF_RGBA, PF_RGB, PF_R, PF_G, PF_B, PF_A**) corresponding to 4, 3, and 1 channel images. The picture storage should normally be **PS_8BIT** indicating that 8 bits of each channel is saved in a run-length-encoded manner. Dump mode is now supported, so natural images might well be **PS_8DUMP** to indicate an 8-bit per channel dumped format. Virtual frame buffers are saved as **PS_12DUMP** to indicate a 12-bit per channel dumped format. A 12-bit encoded storage scheme is also supported, flagged by **PS_12BIT.** The picture matting indicator is relevant only for pictures including an alpha channel, and should be **PM_MTB** for matted-to-black pictures and **PM_NONE** for unassociated pictures. Tile sizes are suggested to be no bigger than 512 by 512 so that current frame buffers can be used for manipulation of pictures on a tile by tile basis.

PicOpen opens the picture named by *filename* and associates a channel with it. *PicOpen* returns a pointer to be used to identify the picture descriptor in subsequent operations.

Type is a character string having one of the following values:

"r" open for reading

"w" open for writing

"r+" open for both

"w+" create and open for both.

PicFind is just like *PicOpen*, except that it searches PIXPATH to find *filename* before opening it. The global *PicFindName* is a char pointer pointing to the full specification of the opened file.

LIBRARY

/usr/pixar/host/lib/libpicio.a

SEE ALSO

libpicio (3H), open(2), creat(2), fopen(3)
PicClose (3H), PicLabel (3H), PicRead (3H), picDecode (3H), picEncode (3H).

DIAGNOSTICS

PicCreat, PicOpen and *PicFind* return the pointer NULL if *filename* cannot be accessed. *PicCreat* will return the pointer NULL if the header information is illegal.

NAME

picPreDecodeScanline,
picDecodeScanline,
picPostDecodeScanline – sequential decoding of tile from disk

SYNOPSIS

#include <picio.h>

picPreDecodeScanline(channel, tilenumber)
PFILE *channel; long tilenumber;

char * picDecodeScanline(channel, ptr)
PFILE *channel; RGBAPixelType *ptr;

picPostDecodeScanline(channel)
PFILE *channel;

DESCRIPTION

These routines are offered as an alternative to the *PicReadBuffer* routine, described elsewhere, which converts one entire tile from disk to memory buffer. These routines allow the sequential decoding of individual scanlines of a tile from disk to a scanline memory buffer. These routines are fragile in the sense that the described order must be followed exactly to produce a correctly decoded picture. If a tile has width *w* and height *h*, there should be an RGBAPixelType buffer of *w* pixels. There should be one call to *picPreDecodeScanline*, *h* calls to *picDecodeScanline*, and one call to *picPostDecodeScanline*.

picPreDecodeScanline initiates the decoding of a tile with tile number *tilenumber* in the picture header. Note that tile numbers begin at zero. Zero is returned if the tile does not exist. It is very possible to create pictures with some tiles missing. Whether this means "all pixels black" or "no picture at all" is your choice. *picDecodeScanline* decodes the next scanline of the named picture output *channel* into a scanline RGBA buffer beginning at *ptr*. A global variable *picDecodeEmpty* is set non-zero if the decoded scanline has an alpha channel that is zero everywhere. A global variable *picDecodeFull* is set non-zero if the decoded scanline has an alpha channel that is unity everywhere. Because the alpha channel defaults to unity, the decoding of any picture not including alpha will force picDecodeFull on at every scanline. *picDecodeScanline* returns a char pointer, which should point beyond the last pixel of the scanline buffer.

picPostDecodeScanline ends the decoding of this tile.

LIBRARY

/usr/pixar/host/lib/libpicio.a

SEE ALSO

libpicio (3H) – overview of the picio library
PicClose (3H), PicCreat (3H), PicLabel (3H), PicRead (3H), picEncode (3H).

BUGS

Abuse of the lseek pointer into the open picture file may wreak havoc.

DIAGNOSTICS

picPreDecodeScanline will return 0 if the tilenumber is bad or internal buffer space cannot be allocated. *picDecodeScanline* will return 0 if *picPreDecodeScanline* has not been called or if we reach the end of file in the midst of decoding. *picPostDecodeScanline* will return 0 if the number of calls to picDecodeScanline is not equal to the height of the tile.

NAME

picPreEncodeScanline,
picEncodeScanline,
picPostEncodeScanline – sequential encoding of a tile to disk

SYNOPSIS

#include <picio.h>

picPreEncodeScanline(channel,tilenumber)
PFILE *channel; long tilenumber;

char * picEncodeScanline(channel, ptr)
PFILE *channel; RGBAPixelType *ptr;

picPostEncodeScanline(channel)
PFILE *channel;

DESCRIPTION

These routines are offered as an alternative to the *PicWriteBuffer* routine, described elsewhere, which converts one entire tile from memory buffer to disk. These routines allow the sequential encoding of individual scanlines of a tile from a scanline memory buffer to disk. These routines are fragile in the sense that the described order must be followed exactly to produce a correctly encoded picture on disk. If a tile has width *w* and height *h*, there should be a RGBAPixelType buffer of *w* pixels. There should be one call to *picPreEncodeScanline*, *h* calls to *picEncodeScanline*, and one call to *picPostEncodeScanline*.

picPreEncodeScanline initiates the encoding of a tile and associates the encoded information with tile number *tilenumber* in the picture header. Note that tile numbers begin at zero. Zero is returned upon failure to write this pointer. *picEncodeScanline* uses a scanline RGBA buffer beginning at *ptr* to encode the next scanline of the named picture output *channel*.

picPostEncodeScanline ends the encoding of this tile.

LIBRARY

/usr/pixar/host/lib/libpicio.a

SEE ALSO

libpicio (3H) – overview of the picio library
PicClose (3H), PicCreat (3H), PicLabel (3H), PicRead (3H), picDecode (3H)

BUGS

Abuse of the lseek pointer into the picture being encoded will trash the encoding. Although concurrent encoding is supported, concurrent encoding of tiles in the same picture is not.

DIAGNOSTICS

picPreEncodeScanline returns 0 if the tilenumber is bad or internal buffer space cannot be allocated. *picEncodeScanline* returns 0 if *picPreEncodeScanline* has not been called or upon any disk write error (i.e., when there is no more space). *picPostEncodeScanline* returns 0 if the number of calls to picEncodeScanline is not equal to the height of the tile. This indicates a malformed tile.

NAME

PicGetFrame,
PicPutFrame - get/put pictures from frame buffer to picture file

SYNOPSIS

```
#include <picio.h>
#include <chad.h>

PicPutFrame( pdptr, fbptr, xoffset, yoffset, xmin, xmax, ymin, ymax, tile, channels, host )
PFILE *pdptr;
ChadFrame *fbptr;
int xoffset, yoffset, xmin, xmax, ymin, ymax;
int tile, channels, host;

PicGetFrame( pdptr, fbptr, xoffset, yoffset, host )
PFILE *pdptr;
ChadFrame *fbptr;
int xoffset, yoffset;
int host;
```

DESCRIPTION

These procedures copy images from picture files to and from frame buffer memory. The function *PicGetFrame* corresponds roughly to the command line program *sv*(1), and *PicPutFrame* corresponds to the program *gt*(1). These names are **not** backwards\(**-a host programmer made them up, that's all.

PicGetFrame copies an image from Pixar frame buffer memory into a picture file. The picture file is described by a picture descriptor pointer, *pdptr*, which is usually obtained by calling *PicCreat*(3H). The attributes of the picture file (for example, its dimensions and storage format), are set before the call to *PicCreat.* Functions to set these parameters are described in *PicCreat(3H).* The rectangular window within the Pixar frame buffer memory from which the picture is copied is described by a ChadFrame pointer, *fbptr*, which is obtained by a call to *ChadAlloc*(3H). The origin of the picture is given by the point (*xoffset*, *yoffset*). The coordinates of this point are relative to the frame buffer window, that is, (0,0) is the upper-left hand corner. As mentioned previously, the size of the picture is determined by the picture dimensions in the picture file descriptor. If the flag *host* is true, the picture is encoded on the host. Normally, this should be false allowing the Chap to do the encoding since this is much faster.

PicPutFrame copies an image stored in a picture file into Pixar frame buffer memory. A picture file descriptor, *pdptr*, corresponding to the already created picture file to be transferred, should be obtained using *PicFind*(3H). If the variable *tile* is less than 0, all the tiles in the picture file are transferred. If it is greater than or equal to 0, only the corresponding numbered tile is copied. A rectangular window in the frame buffer is described by a ChadFrame pointer, *fbptr*, which is obtained by calling *ChadAlloc.* A clipping rectangle can be specified by giving its x-range (*xmin*, *ymin*) and y-range (*ymin*, *ymax*). The coordinates of the clipping rectangle are specified relative the the pixel window (so that (0,0) is the upper-left corner). If the point (*xoffset*, *yoffset*) equals (0,0) the picture is copied so that its upper-left hand corner is aligned with the frame buffer window's upper-left hand corner. Positive x offsets move the picture to the right and positive y offsets move the picture down. The variable *channels* is a bit-mask that specifies the channels in the frame buffer, which are written (r=1, g=2, b=4, a=8). Finally, the boolean variable *host* controls whether the decoding takes place in the host or in the chap. Normally, this is set to 0 indicating that the chap does the decoding since this is much faster. Unfortunately, not all picture file formats can be encoded in the chap, and sometimes this flag is overridden.

EXAMPLES

The simplest code to copy a picture into the Pixar frame buffer would look something like this:

```
ChadAlloc(0, TB, &tbp, 0, 32, 24, FRAME, &fbp, &tbp, 0, 1023, 0, 767, NIX);

pdptr = PicFind( "reyes","r");
PicPutFrame( pdptr, fbp, 0, 0, 0, 1023, 0 767, -1, channels=0xf, host=0 );
```

```
PicClose(pdptr);
```

The simplest code to store a picture from the Pixar frame buffer would look something like this (assuming the same ChadFrame as above):

```
PicSetPsize(picturewidth, pictureheight);
PicSetTsize(picturewidth, pictureheight);
PicSetPformat(pictureformat);
PicSetOffset(xmin, ymin);

pdptr = PicCreat(filename,mode);
PicGetFrame( pdptr, fbp, xmin, ymin, host=0 );
PicClose(pdptr);
```

BUGS

Currently, only the following formats are encoded and decoded by the Chap: PF_RGBA, PF_RGB, PF_R.

If for some reason the chap cannot allocate all the resources it needs, the host performs the encoding and decoding.

LIBRARY

/usr/pixar/host/lib/libpicio.a

SEE ALSO

gt(1), sv(1), libpicio(3H), ChadAlloc(3H), PicCreat(3H).

NAME

PicLseekLabel,
PicReadLabel,
PicWriteLabel – handle picture labels

SYNOPSIS

#include <picio.h>

```
long PicLseekLabel(pdptr, offset, whence)
PFILE *pdptr;
long offset;
int whence;

int PicReadLabel(pdptr, buffer, nbytes)
PFILE *pdptr;
char *buffer;
int nbytes;

int PicWriteLabel(pdptr, buffer, nbytes)
PFILE *pdptr;
char *buffer;
int nbytes;
```

DESCRIPTION

The picture descriptor *pdptr* refers to a picture file open for reading or writing. The pointer for the file is set as follows:

> If *whence* is 0, the pointer is set to *offset* bytes.
>
> If *whence* is 1, the pointer is set to its current location plus *offset*.
>
> If *whence* is 2, the pointer is set to the end-of-label plus *offset*. The end-of-label is taken as the first null character after the start of the label.

The returned value is the resulting offset into the label; -1 is returned in case of invalid picture descriptor, seek to a position before the start of the label, or improper *whence*. The read/write pointer is untouched by the first and last errors; the pointer is set to zero after an attempt to seek a position before the start of the label.

PicReadLabel reads the next *nbytes* bytes of the picture label into the array *buffer*. It is not guaranteed that all *nbytes* bytes will be read. Furthermore, it may read past the end-of-label, leaving the read/write pointer beyond the first null character of the label. In any event, the number of characters read is returned as the value of the procedure.

If the returned value is less than *nbytes*, end-of-label has been reached and likely overrun. If you need accuracy in the length of label, use the returned value from *PicLseekLabel(pdptr,0,2)*.

PicWriteLabel writes the next *nbytes* bytes into the picture label from the array *buffer*. The number of characters written is returned. Writing a null character in the midst of the buffer will induce an end-of-label and set the read/write pointer beyond this end-of-label.

The *PicSetLabel(labelptr)* routine described along with *PicCreat* is equivalent to a *PicLseekLabel(pdtpr,0,0)* followed by a *PicWriteLabel(pdptr,labelptr,strlen(labelptr))* in writing the label and setting the label lseek pointer.

PicCreat() and *PicOpen()* set the read/write pointer of the label to zero.

LIBRARY

/usr/pixar/host/lib/libpicio.a

SEE ALSO

libpicio (3H) – overview of the picio library
PicClose (3H), PicCreat (3H), PicRead (3H), picDecode (3H), picEncode (3H).

NAME

PicReadBuffer,
PicWriteBuffer – tile to buffer I/O

SYNOPSIS

#include <picio.h>

PicReadBuffer(channel, ptr, tilenumber)
PFILE *channel; RGBAPixelType *ptr; long tilenumber;

PicWriteBuffer(channel, ptr, tilenumber)
PFILE *channel; RGBAPixelType *ptr; long tilenumber;

DESCRIPTION

PicReadBuffer reads, into a block beginning at *ptr*, pixel by pixel contents of tile *tilenumber* from the named picture input *channel*. It returns the number of scanlines actually read.

PicWriteBuffer writes the pixel by pixel contents of the buffer starting at *ptr* into tile number *tilenumber* to the named picture output *channel*. It returns the number of scanlines actually written.

Note that tile size information is accessible through the channel pointer.

LIBRARY

/usr/pixar/host/lib/libpicio.a

SEE ALSO

libpicio(3H)
read(2), write(2), PicOpen(3H), PicClose (3H), PicCreat (3H), PicLabel (3H), picDecode (3H), picEncode (3H).

DIAGNOSTICS

Pread and *Pwrite* return 0 upon end of file or error.

NAME

libpirl – Introduction to Pixar Resource Library

DESCRIPTION

Libpirl is a set of C-callable functions for working with rectangular pixel windows on the Pixar. Some routines, e.g., *PirlClamp*, *PirlReflectX*, accept only a pixel window in order to perform their operation. Most of the routines need additional parameters, such as a color, or numeric values (e.g., *PirlClear*, *PirlAddI*, respectively) in order to operate on the pixel window. A few routines, such as *PirlCopy* and *PirlSwap*, accept two pixel windows.

This library also includes several image processing routines such as boxfilter, convolve and blur.

It is the user's responsibility to begin and end the *Chad* environment, and allocate and deallocate valid pixel windows via *ChadAlloc*. *Libpirl* routines will allocate and deallocate temporary storage, such as scanline buffers, as needed.

Each of these routines returns the token *PIRL_NO_ERROR* if it completed successfully. Error checking and recovery is explained in more detail in the *The Pirl Tutorial.*

SEE ALSO

intro(1), intro(3C), intro(3H), libchad(3H)

LIST OF FUNCTIONS

Name	*Appears on Page*	*Description*
Mapfcn	PirlMakeMap(3H)	– Mapping function for PirlMakeMap
PirlAdd	PirlArithmetic(3H)	– add two pixel windows
PirlAffine	PirlAffine(3H)	– Perform an affine transformation on a pixel window
PirlAxb	PirlAxb(3H)	– compute new pixel = *a**pixel+*b* for a pixel window.
PirlBBox	PirlBBox(3H)	– determine the smallest rectangle that surrounds an image
PirlBegin	PirlBegin(3H)	– Pixar runtime environment entry/exit
PirlBeginLines	PirlLine(3H)	– Draw lines in a pixel window
PirlMulI	PirlAxb(3H)	– compute new pixel = *a**pixel for a pixel window.
PirlDivI	PirlAxb(3H)	– compute new pixel = pixel/*a* for a pixel window.
PirlAddI	PirlAxb(3H)	– compute new pixel = pixel+*b* for a pixel window.
PirlSubI	PirlAxb(3H)	– compute new pixel = pixel-*b* for a pixel window.
PirlBoxFilterX	PirlBoxFilter(3H)	– convolve pixel window buffer with 1-d pulse (box)
PirlBoxFilterY	PirlBoxFilter(3H)	– convolve pixel window buffer with 1-d pulse (box)
PirlCbars	PirlCbars(3H)	– display color bars
PirlCha	PirlCha(3H)	– perform linear arithmetic on framebuffer channels
PirlCircularShift	PirlShift(3H)	– circular shift pixel window contents in x and/or y
PirlClamp	PirlClamp(3H)	– clamp pixel between 0.0 (0) and 1.0 (0x800) for a pixel window
PirlClear	PirlClear(3H)	– clear pixel window to *color*
PirlConvolveX	PirlConvolve(3H)	– convolve a pixel window with a 1-d kernel
PirlConvolveY	PirlConvolve(3H)	– convolve a pixel window with a 1-d kernel
PirlConvolve3x3	PirlConvolve3x3(3H)	– convolve pixel window with a separable 3x3 matrix
PirlConvolve3x3s	PirlConvolve3x3(3H)	– convolve pixel window with a 3x3 kernel
PirlCopy	PirlCopy(3H)	– copy the source pixel window to the destination pixel window
PirlCopyGeneric	PirlCopy(3H)	– PirlCopy with user-specified axes, start and direction parameters
PirlCrc	PirlCrc(3H)	– performs a Cyclic Redundency Check (CRC) on a pixel window
PirlDiv	PirlArithmetic(3H)	– divide two pixel windows
PirlDisplay	PirlDisplay(3H)	– Display a pixel window on the monitor
PirlEnd	PirlBegin(3H)	– Pixar runtime environment entry/exit
PirlEndLines	PirlLine(3H)	– Draw lines in a pixel window
PirlErrReport	PirlErrReport(3H)	– print a descriptive error message explaining the last error
PirlGetBuf	PirlGetBuf(3H)	– Get/put a block of pixels from/to a pixel window
PirlGetFrame	PirlFrame(3H)	– get a buffer into a pixel window
PirlGetPic	PirlGetPic(3H)	– Get/save a pixel window from/to a disk file

PirlGetRaster	PirlGetRaster(3H)– get a raster image file into a pixel window	
PirlHistogram	PirlHistogram(3H)	– accumulate frequency histogram of a pixel window
PirlLineEdgeDesc	PirlLine(3H)	– Set line characteristics
PirlMakeMap	PirlMakeMap(3H)	– create a map for changing the colors of a pixel window
PirlMap	PirlMap(3H)	– remap the colors of a pixel window
PirlMerge	PirlMerge(3H)	– composite a foreground with a background
PirlMul	PirlArithmetic(3H)	– multiply two pixel windows
PirlNot	PirlNot(3H)	– subtract pixel value from 1.0 (0x800) for a pixel window
PirlPolyLine	PirlLine(3H)	– Set line characteristics
PirlPutBuf	PirlGetBuf(3H)	– Get/put a block of pixels from/to a pixel window
PirlPutFrame	PirlFrame(3H)	– put a a buffer into a pixel window
PirlRampX	PirlRamp(3H)	– draw a ramp into a pixel window
PirlRampY	PirlRamp(3H)	– draw a ramp into a pixel window
PirlRange	PirlRange(3H)	– find the minimum and maximum pixel values in a pixel window
PirlReflectX	PirlReflect(3H)	– reflect the pixel window around its center horizontal axis
PirlReflectY	PirlReflect(3H)	– reflect the pixel window around its center vertical axis
PirlResize	PirlResize(3H)	– copy the source pixel window to the destination pixel window
PirlSavePic	PirlGetPic(3H)	– Get/save a pixel window from/to a disk file
PirlSetChannelMask	PirlSetChannelMask(3H)	– set a pixel window's channel mask
PirlSetLineColor	PirlLine(3H)	– Set line characteristics
PirlSetLineEdge	PirlLine(3H)	– Set line characteristics
PirlSetLineMode	PirlLine(3H)	– Set line characteristics
PirlSetLineWidth	PirlLine(3H)	– Set line characteristics
PirlShift	PirlShift(3H)	– shift pixel window contents in x and/or y
PirlShuffle	PirlShuffle(3H)	– shuffle components of each pixel for a pixel window
PirlSub	PirlArithmetic(3H)	– subtract two pixel windows
PirlSwap	PirlSwap(3H)	– swap the source pixel window and the destination pixel window
PirlSweepX	PirlSweep(3H)	– copy a column of pixels repeatedly into a pixel window
PirlSweepY	PirlSweep(3H)	– copy a row of pixels repeatedly into a pixel window
PirlTranspose	PirlTranspose(3H)	– transpose a pixel window around the diagonal axis
PirlZoom	PirlZoom.3h	– Zoom in on a pixel window on the monitor

NAME

PirlAffine – Perform an affine transformation on a pixel window

SYNOPSIS

#include <chad.h>
#include <pirl.h>
PirlError
PirlAffine(srcpw,tmppw,dstpw,transf,clr,extent)
ChadPW *srcpw, *tmppw, *dstpw;
double transf[3][2];
int clr,extent;

DESCRIPTION

PirlAffine performs an affine transformation on a source pixel window and places the result in a destination pixel window. The transformation is performed using a two-pass (horizontal and vertical) resampling algorithm.

Source and destination pixel windows are specifed for each pass of the resampling algorithm. The source and destination pixel windows for each pass cannot partially overlap. **srcpw** defines the transformation source pixel window. **tmppw** defines the destination window for the first (intermediate) pass and the source for the second pass. **dstpw** defines the the destination for the second (final) pass. If the **dstpw** is smaller than the **srcpw**, the **tmppw** must be greater than or equal to the size of the **srcpw**. The transformation can be done in place, so **srcpw**, **tmppw**, and **dstpw** can all point to the same pixel window.

transf specifies the affine transformation matrix. The first column of the matrix defines the coefficients for x', such that: x' = Ax + By + C. The second column defines the coefficients for y', such that: y' = Ax + By + C.
clr is a flag wich specifies whether PirlAffine should clear out its borders.
extent specifies the filter width (in pixels). An extent of four will use a cubic filter. An extent of two will use a linear filter.

FILES

/usr/pixar/host/src/lib/libpirl/affine.c

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

Rotate(1), PirlRotate(3h), PirlShear(3h), PWShear(3c)

NAME

PirlAdd	– add two pixel windows
PirlSub	– subtract two pixel windows
PirlMul	– multiply two pixel windows
PirlDiv	– divide two pixel windows

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlAdd ( dstpw, srcpw )
ChadPW *dstpw, *srcpw;

PirlError
PirlSub ( dstpw, srcpw )
ChadPW *dstpw, *srcpw;

PirlError
PirlMul ( dstpw, srcpw )
ChadPW *dstpw, *srcpw;

PirlError
PirlDiv ( dstpw, srcpw )
ChadPW *dstpw, *srcpw;
```

DESCRIPTION

These procedures perform image arithmetic on two pixel windows. Pixel values within a window are treated as 11-bit fixed point quantities. Therefore, 2048*2048=2048 and 2048/2048=2048. *PirlAdd* computes *dstpw += srcpw*, *PirlSub* computes *dstpw –= srcpw*, *PirlMul* computes *dstpw *= srcpw*, and *PirlDiv* computes *dstpw /= srcpw*.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlCha(3H), PirlAxb(3H)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlAxb – compute new pixel = *a*∗pixel+*b* for a pixel window.
PirlMulI – compute new pixel = *a*∗pixel for a pixel window.
PirlDivI – compute new pixel = pixel/*a* for a pixel window.
PirlAddI – compute new pixel = pixel+*b* for a pixel window.
PirlSubI – compute new pixel = pixel-*b* for a pixel window.

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlAxb (pw, a, b)
ChadPW *pw;
RGBAPixelType *a,*b;

PirlError
PirlMulI (pw, a)
ChadPW *pw;
RGBAPixelType *a;

PirlError
PirlDivI (pw, a)
ChadPW *pw;
RGBAPixelType *a;

PirlError
PirlAddI (pw, b)
ChadPW *pw;
RGBAPixelType *b;

PirlError
PirlSubI (pw, b)
ChadPW *pw;
RGBAPixelType *b;
```

DESCRIPTION

PirlAxb computes a new pixel value by multiplying each pixel component by the appropriate components of *a* and adding *b*. The factors are four-way 11-bit values, where 1.0E (2048) equals 1.0. This function is useful for performing simple channel arithmetic.

PirlMulI and *PirlDivI* are equivalent to *PirlAxb* with the *b* pixel set to zero. The division function computes the reciprocal of *a*, then calls *PirlAxb* with this new value.

PirlAddI and *PirlSubI* set the *a* components to 1.0E (2048) before calling *PirlAxb*. The subtract routine subtracts *b* from the pixel value instead of adding it.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWAxb(3C), SSAxb(3C), PirlCha(3H)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

PirlDivI returns PIRL_AXB_DIV_BY_ZERO if asked to divide by a zero component.

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlBBox – determine the smallest rectangle that surrounds an image

SYNOPSIS

#include "/usr/pixar/include/pirl.h"

```
PirlError
PirlBBox (pw,background,xmin,xmax,ymin,ymax)
ChadPW *pw;
RGBAPixelType *background;
int *xmin, *xmax, *ymin, *ymax;
```

DESCRIPTION

PirlBBox finds the smallest rectangle (bounding box) that surrounds an image in the given pixel window. This can be used to make a smaller pixel window so that subsequent processing is performed on smaller images and takes less time.

The color *background* is used to determine whether the image data is present. If the color equals *background* image data is assumed not to be present; if, on the otherhand, the color is not equal to *background*, image data is assumed to be present. Normally, *background* is set to (0,0,0,0).

The edges of the bounding rectangle are returned in (*xmin, xmax, ymin, ymax*). These coordinates are relative to the pixel window. If the entire pixel window contains valid image data, (*xmin,ymin*) will equal (0,0), and (*xmax,ymax*) will equal the width and height of the pixel window, respectively.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWBBox (3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlBegin, PirlEnd – Pixar runtime environment entry/exit

SYNOPSIS

 # include "/usr/pixar/include/pirl.h"
 # define STD_TB FB_DESCRIP(0, 32, 24)
 # define BIG_TB FB_DESCRIP(0, 32, 128)
 # define HUGE_TB FB_DESCRIP(0, 32, 256)

should be TB_DESCRIP

extern PirlTB ThePirlTB;
extern PirlPW ThePirlPW;
extern Video ThePirlVideo;

TB

PirlError PirlBegin(chapid, FBDESCRIP(firsttile, nxtiles, nytiles))
ChapID chapid;
int firsttile, nxtiles, nytiles;

PirlError PirlEnd ( )

DESCRIPTION

Pirl (Pixar Runtime Library) provides simplified access to routines which access rectilinear regions of image memory on the Pixar Image Computer. It also maintains a simple runtime environment, which must be entered with *PirlBegin* () and exited with *PirlEnd* (). The former takes two arguments: a **ChapID** (almost invariably the constant **CHAP0** as defined for **Chad**), indicating which Chap **Pirl**'s microcode will run on; and a frame buffer descriptor, usually one of **STD_TB** (for 1024x768-pixel displays), **BIG_TB** (ditto, but including substantial offscreen frame buffer) or **HUGE_TB** (likewise, and including all offscreen frame buffer in the largest available Pixar Image Computer). **PirlBegin()** allocates **ThePirlTB** and **ThePirlPW**, a tile block and pixel window encompassing the tiles given in its argument list. These are a convenience to be used in other **Pirl** calls.

Pirl defines two data types, **PirlTB** and **PirlPW**. These are the same as **ChadTB** and **ChadPW**, respectively, except that **Pirl** retains a record of all pixel windows that were allocated via **PirlNewPW** (3H), so they can be automatically deallocated via **PirlEnd()**.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SOURCE

/usr/pixar/host/src/lib/libpirl/begin.c -- source for **PirlBegin()** and **PirlEnd ()**

SEE ALSO

libpirl(3H), PirlNewPW(3H)

The *Pirl Tutorial*, in *The Pixar Programmer's Manual*, serves as an introduction to Pixar programming using **Pirl**.

NAME

PirlBoxFilterX,
PirlBoxFilterY – convolve pixel window buffer with 1-d pulse (box)

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlBoxFilterX(pw, width, highpass, a, b)
ChadPW *pw;
int width, highpass;
double a, b

PirlError
PirlBoxFilterY(pw, width, highpass, a, b)
ChadPW *pw;
int width, highpass;
double a, b
```

DESCRIPTION

PirlBoxFilterX and *PirlBoxFilterY* performs a one-dimensional convolution of the image stored in the pixel window. An image can be convolved with a 2-d pulse function by first convolving in x, and then convolving in y.

The convolution is done by summing the center pixel plus the *width* pixels preceding and following it. Therefore, the total width of the pulse is *2*width+1* pixels. *PirlBoxFilterX* sums the pixels horizontally, while *PirlBoxFilterY* sums them vertically.

If the flag *highpass* is non-zero, the result of the convolution is subtracted from the value of the center pixel. This creates a highpass filter.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlConvolve3x3(3H), PirlConvolve(3H), PWBoxFilter(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlCbars – display color bars

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
#include "/usr/pixar/include/cbars.h"

PirlError PirlCbars(pw, type)
ChadPW *pw;
int type;
```

DESCRIPTION

PirlCbars displays color bars for 3-quarters the length of a pixel window, *pw*. These options are controlled by the *type*, as follows:

NORMAL	– draw conventional color bars
CBS	– draw CBS colorbars
FULL	– draw full-length bars
REVERSE	– draw reverse bars at the bottom

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

cbars(1)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

NAME

PirlCha – perform linear arithmetic on framebuffer channels

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"

PirlError PirlCha(pw, coeffs)
ChadPW *pw;
double coeffs[4][5];
```

DESCRIPTION

PirlCha generates a color at each pixel of a pixel window, *pw*. The channels of each output pixel are a linear combination of the original channel values, as given by the matrix *coeffs*. The first row of the matrix generates the new Red channel value by multiplying the input channel values by the first four elements, then adding the fifth.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWCha(3C), SSCha(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

NAME

PirlClamp – clamp pixel between 0.0 (0) and 1.0 (0x800) for a pixel window

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlClamp (pw)
ChadPW *pw;
```

DESCRIPTION

PirlClamp clamps each pixel in a pixel window to a minimum of 0.0 (0) and a maximum of 1.0E (2048). This is useful for removing the values outside this range occasionally produced by filtering.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

SSClamp(3C), PWClamp(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlClear – clear pixel window to *color*

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlClear (pw,color)
ChadPW *pw;
RGBAPixelType *color;
```

DESCRIPTION

PirlClear clears the pixel window to *color*.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWClear(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlConvolveX,
PirlConvolveY – convolve a pixel window with a 1-d kernel

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlConvolveX(pw, kernel, kernelsize)
ChadPW *pw;
double kernel[];
int kernelsize;

PirlError
PirlConvolveY(pw, kernel, kernelsize)
ChadPW *pw;
double kernel[]
int kernelsize;
```

DESCRIPTION

PirlConvolveX and *PirlConvolveY* convolve the image in the pixel window with a 1-dimensional kernel. In the *X* version, the kernel extends *kernelsize* pixels along the x axis. In the *Y* version, the kernel is aligned with the y axis. The result is placed in the same pixel window.

The kernel is an array of *kernelsize* doubles. The variable *offset* specifies which entry of the kernel matrix *kernel* corresponds to the center pixel. If offset is 0, the last entry of the kernel is aligned with the pixel being output. If *offset* is *kernelsize*/2, the kernel is centered. If *offset* is *kernelsize*, the first entry of the kernel is aligned with the pixel being output.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlConvolve3x3(3H), PWConv(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlConvolve3x3,
PirlConvolve3x3s – convolve pixel window with a 3x3 kernel

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlConvolve3x3(pw, kernel)
ChadPC *pw;
double kernel[3][3];

PirlError
PirlConvolve3x3s( pw, xkernel, ykernel )
ChadPW *pw;
double xkernel[3], ykernel[3];
```

DESCRIPTION

PirlConvolve3x3 convolves an image stored in the pixel window with the 3x3 matrix pointed to by *kernel*. The result is stored in the same pixel window.

PirlConvolve3x3s convolves an image with a separable 3x3 matrix. The image is first convolved horizontally with the 3-vector *xkernel* and then vertically with the 3-vector *ykernel*. This is done in one pass.

Where the kernel extends beyond the edge of the image, the pixel values are set to 0.

The inner loop of *PirlConvolve3x3* takes 14 ticks per pixel; the inner loop of *PirlConvolve3x3s* takes 19 ticks per pixel.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlConvolve(3H), PWc33(3C), PWc33s(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlCopy – copy the source pixel window to the destination pixel window
PirlCopyGeneric – PirlCopy with user-specified axes, start and direction parameters

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"

PirlError
PirlCopy (srcpw,dstpw)
ChadPW *srcpw;
ChadPW *dstpw;

PirlError
PirlCopyGeneric (srcpw,dstpw,srcRtn,srcStart,srcInc,srcAxis,
  dstRtn,dstStart,dstInc,dstAxis,spad)
ChadPW *srcpw;
ChadPW *dstpw;
int srcRtn,srcStart,srcInc,srcAxis,dstRtn,dstStart,dstInc,dstAxis;
ChadSpad *spad;
```

DESCRIPTION

PirlCopy copies the source pixel window to the destination pixel window. If the pixel windows overlap, the source window will be overwritten.

PirlCopyGeneric copies the source pixel window to the destination pixel window. This routine allows the user to specify several options for greater control of the copying operation. The *srcRtn* (a Chap routine) is used to extract a scanline from the framebuffer into *spad*. The *srcStart* and *srcInc* parameters specify the starting line, relative to the start of the source pixel window, and the increment (usually 1 or −1) in the direction of *srcAxis* (0 for x direction, 1 for y direction). Similarly, the *dstRtn* is used to copy the *spad* buffer into the framebuffer. *dstStart*, *dstInc*, *dstAxis* have the same semantics as their source counterparts.

The *spad* buffer must be large enough to hold the maximum number of pixels copied from the framebuffer by *srcRtn*.

For programming examples, see the source for *PirlCopy* and *PirlReflect*.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWCopy(3C), FSCopy(3C), SFCopy(3C), PirlSwap(3H)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Both pixel windows must be the same size and belong to the same Chap.

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

Both axis specifiers in *PirlCopyGeneric* must be the same (either 0 or 1).

NAME

PirlCrc – performs a Cyclic Redundency Check (CRC) on a pixel window

SYNOPSIS

```
#include <chad.h>
#include <pirl.h>
PirlError
PirlCrc (pw,CrcVals)
ChadPW *pw;
RGBAPixelType *CrcVals;
```

DESCRIPTION

PirlCrc() computes a CCITT standard CRC value for *pw*. The Crc values for each channel are returned in *CrcVals*.

LIBRARY

libpirl.a

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

SSCrc(3C), PWCrc(3H), crc(1H)

The **Chad** tutorial *Introduction to Chad*, and also the *Programming Tutorial* in *The Pixar Programmer's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all **Chad** errors apply.

NAME

PirlDisplay – Display a pixel window on the monitor

SYNOPSIS

 # include "/usr/pixar/include/pirl.h"

extern PirlTB ThePirlTB;

PirlError PirlDisplay(pxlwdw, xoffset, yoffset)
PirlPW *pxlwdw;
int xoffset, yoffset;

DESCRIPTION

PirlDisplay () sets the video board of a Pixar Image Computer to display the given pixel window on the monitor. Specifically, the upper left corner of the display displays the upper left corner of the pixel window. If *xoffset* and *yoffset* differ from 0, then they give the location of a pixel in the pixel window which is displayed at the upper left. Positive offsets are rightward and down, respectively.

The zoom rate (replication factor for the video scanout) of the display is unaffected by *PirlDisplay* (), but can be changed by *PirlZoom* (3H).

The offsets needn't lie within the pixel window, and they can be negative. However, the pixel specified **must** lie within the tile block in which the pixel window is defined.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SOURCE

/usr/pixar/host/src/lib/libpirl/begin.c -- source for **PirlDisplay()**

SEE ALSO

libpirl(3H), PirlZoom(3H)

The *Pirl Tutorial*, in *The Pixar Programmer's Manual*, serves as an introduction to Pixar programming using **Pirl**.

NAME

PirlErrReport – print a descriptive error message explaining the last error

SYNOPSIS

PirlErrReport(fp)
FILE *fp;

DESCRIPTION

PirlErrReport prints a descriptive error message to the file specified by *fp* (usually **stderr**). It uses the variable *PirlLastErr*, set by the *CHECK* macro. Since *Pirl* error messages are a superset of *Chad* error messages, this routine will call *ChadErrReport* for the descriptive messages if the last error was within *Chad.*

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

ChadErrReport, pirl.h (source for *CHECK* macro)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*,

NAME

PirlGetFrame,
PirlPutFrame   – get/put a buffer into a pixel window

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlGetFrame ( frame, buffer, xmin, xmax, ymin, ymax )
ChadFrame *frame;
RGBAPixelType *buffer;
int xmin, xmax, ymin, ymax;

PirlError
PirlPutFrame ( frame, buffer, xmin, xmax, ymin, ymax )
ChadFrame *frame;
RGBAPixelType *buffer;
int xmin, xmax, ymin, ymax;
```

DESCRIPTION

PirlGetFrame copies an array of pixels from the pixel window associated with *frame* to the *buffer*. *PirlPutFrame* copies an array of pixels from a *buffer* to a pixel window associated with *frame*. In both cases the buffer is assumed to contain *(xmax-xmin*+1) times *(ymax-ymin*+1) RGBA pixel values.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

ChadFrame(3H)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlGetBuf,
PirlPutBuf – Get/put a block of pixels from/to a pixel window

SYNOPSIS

 # include "/usr/pixar/include/pirl.h"
 # include "/usr/pixar/include/pixeldef.h"

PirlError PirlGetBuf (pw, pxlbuf, xmin, xmax, ymin, ymax)
PirlError PirlPutBuf (pw, pxlbuf, xmin, xmax, ymin, ymax)
PirlPW pw;
RGBAPixelType *pxlbuf;
int xmin, xmax, ymin, ymax;

DESCRIPTION

PirlGetBuf () reads a rectilinear subwindow of pixels from a pixel window into a hostside buffer. **PirlSaveBuf** () does the opposite. *pw* is the **Pirl** default pixel window **ThePirlPW**, or any other pixel window as allocated by **PirlNewPW** (). *xmin, xmax, ymin* and *ymax* describe the set of pixels involved, so that (*xmax-xmin*+1) * (*ymax-ymin*+1) pixels are read/written. The buffer *pxlbuf* must have at least this many pixels in it, in row-major order (the first row of pixels, followed by the second, etc.).

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SOURCE

/usr/pixar/host/src/lib/libpirl/buf.c -- source for **PirlGetBuf()** and
PirlPutBuf().

SEE ALSO

libpirl(3H), PirlBegin(3H), PirlNewPW(3H)
PirlGetPic (3H) discusses how to save (and restore) whole pixel windows to (and from) disk files.

The *Pirl Tutorial*, in *The Pixar Programmer's Manual*, serves as an introduction to Pixar programming using **Pirl**.

NAME

PirlGetPic,
PirlSavePic – Get/save a pixel from/to a disk file

SYNOPSIS

```
# include "/usr/pixar/include/pirl.h"
# include "/usr/pixar/include/picio.h"

PirlError PirlGetPic ( pic, pw, xoff, yoff)
PirlError PirlSavePic ( pic, pw, xoff, yoff)
PFILE *pic;
PirlPW pw;
int xoff, yoff;
```

DESCRIPTION

PirlSavePic () saves a pixel window in a picture file. **PirlGetPic** () gets pixels from a picture file into a pixel window. *pic* is a picture file descriptor such as is opened by **PicCreat** (3H) or **PicOpen** (). *pw* is the **Pirl** default pixel window **ThePirlPW**, or any other pixel window as allocated by **PirlNewPW** (). *xoff* and *yoff* are offsets: in the context of **PirlSavePic**(), the offset means that the pixel window origin (upper left pixel) is offset in the picture file. To **PirlGetPic** (), it means that the origin of the picture as stored in the file is offset with respect to the pixel window origin when copying it to the frame buffer. Thus, if the two *xoffs* and *yoffs* sum to 0 for a Get/Save pair on the same file, the pixels appear in the same place in the pixel window after restoration.

LIBRARIES

/usr/pixar/host/lib/libpirl.a   /usr/pixar/host/lib/libpicio.a   /usr/pixar/host/lib/libchad.a
/usr/pixar/host/lib/libpixar.a

SOURCE

/usr/pixar/host/src/lib/libpirl/getsv.c -- source for **PirlGetPic**() and **PirlSavePic**().

SEE ALSO

libpicio(3H), libpirl(3H), PirlNewPW(3H), PicCreat(3H)

The *Pirl Tutorial*, in *The Pixar Programmer's Manual*, serves as an introduction to Pixar programming using **Pirl**.

NAME

PirlGetRaster – get a raster image file into a pixel window

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlGetRaster ( pw, fd, mode, shift, swap )
ChadPW *pw;
int fd, mode, shift, swap;
```

DESCRIPTION

PirlGetRaster copies an array of pixels from the file whose file descriptor is *fd* into pixel window *pw*. If the flag *shift* is non-zero, each channel value will be multiplied by eight (this is useful for 8-bit per channel pictures.) If the flag *swap* is non-zero, the bytes in each channel word will be swapped (for 16-bit per channel pictures only.) The type of picture to be copied is controlled by *mode*, as follows:

BW8	– black and white image 8 bits-per-channel
BW16	– black and white image 16 bits-per-channel
RGB8	– RGB image 8 bits-per-channel
RGB16	– RGB image 16 bits-per-channel
RGBA8	– RGBA image 8 bits-per-channel
RGBA16	– RGBA image 16 bits-per-channel

For black and white images, *IFxCopy* is called, and for color images, *SFxCopy* is called.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

see(1), SFCopy(3C), IFCopy(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlHistogram – accumulate frequency histogram of a pixel window

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlHistogram (pw, histogram, size, component)
ChadPW *pw;
int histogram[];
register size, component;
```

DESCRIPTION

PirlHistogram accumulates a frequency histogram of one component of the pixels within a pixel window. The histogram table is an array of *size* long ints (32-bits). The routine assumes that 11-bit values are being examined; they are rescaled to fit into the *size* accumulators of the histogram array. The scaling is one-for-one, if *size*=4096. Since pixel values are signed fixed point numbers in the range [–.5E, 1.5E), the histogram entry for pixel value 0 is at location *size/4*.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlRange(3H)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlBeginLines,
PirlEndLines,
PirlPolyLine,
PirlSetLineAttributes,
PirlSetLineColor,
PirlSetLineEdge,
PirlSetLineMode,
PirlSetLineWidth – Draw lines in a pixel window and set line characteristics

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"

PirlError
PirlBeginLines()

PirlError
PirlEndLines()

PirlError
PirlPolyLine(pw,n,xy)
ChadPW *pw;
int n;
float xy[n][2]];

PirlError
PirlSetLineAttributes(name,linewidth,linemode,color,
                      edgetype,edgewidth,edgefunctionptr,
                      filtertype,filterwidth,filterfunctionptr)
char *name;
float linewidth;
PirlLineMode linemode;
RGBAPixelType color;
PirlEdgeType edgetype;
float edgewidth;
float *edgefunctionptr;
PirlFilterType filtertype;
float filterwidth;
float *filterfunctionptr;

PirlError
PirlSetLineColor(color)
RGBAPixelType color;

PirlError
PirlSetLineEdge(name)
char *name;

PirlError
PirlSetLineMode(mode)
PirlLineMode mode;

PirlError
PirlSetLineWidth(width);
float width;
```

DESCRIPTION

PirlBeginLines initializes all settable line attributes (e.g., line width, line mode, color, line edge characteristics, and antialiasing filter), to their default values and begins a state where line edge description tables are permanently loaded into the scratchpad of the Chap currently being used by Pirl. If PirlBeginLines is not called, then the current edge description table will be temporarily loaded into the scratchpad each time PirlPolyLine is called, and the scratchpad space freed after the PirlPolyLine call. The default values for the line attributes are color = RGBA = 2048, line mode = PM_MERGE, line width = 1 pixel, line edge type = PE_HARD with width 1 pixel, line filter type = PF_BOX with filter width = 1 pixel.

PirlEndLines causes all permanently allocated scratchpad resources to be freed. All calls to PirlPolyLine after a call to PirlEndLines will cause the current edge description table to be temporarily loaded into the scratchpad only for the duration of the PirlPolyLine call.

PirlPolyLine is called to draw *n* connected lines within the specified pixel window *pw*. The lines are drawn with the current line attributes (i.e., color, line mode, and line width), set by previous calls to PirlSetLineColor, PirlSetLineMode, and PirlSetLineWidth. The endpoints of the lines are expressed as floating point number pairs within the specified pixel window (e.g., x=0.0 to 1023.75, y=0.0 to 783.75). Lines will be drawn to quarter pixel resolution by PirlPolyLine.

PirlSetLineAttributes specifies all line attributes that may subequently be referred to by *name*. PirlSetLineAttributes merely establishes an attribute set that may be subsequently be passed to PirlSetLineEdge to cause the line attributes to have effect for subsequently drawn lines. Parameters passed to PirlSetLineAttributes include all the setable line attributes (color, line mode, and line width) plus parameters that are used to create an edge description table in the scratchpad of the current Chap with the edge characteristics specified by *edge, edgewidth,* and *edgefunctionptr* and with antialiasing performed using a filter that is specified by *filter, filterwidth,* and *filterfunctionptr*.

edge may be one of the following:
PE_GIVEN
PE_HARD
PE_RANDOM
PE_FELTTIP

The edge function is the profile of the edge used to determine the line's contribution to neighborhood pixels. The edge is represented by a table of alpha values as a function of distance to the line. The line is drawn by finding the distance from each pixel to the line segment, using the edge table to find the alpha, using the alpha in the classic expression (B + a*(F-B)), where F is the color of the line being drawn and B is the color of the background at the pixel. The edge function is convolved with a filter function to create an edge profile table. Lines are drawn subject to this edge profile table that determines the rolloff at the edges. Those edges occur *width* pixels away from the line center (and along semicircles at the endpoints). A line with a hard edge will appear as twice this width.

Predefined edge functions include PE_HARD, PE_RANDOM, and PE_FELTTIP. A PE_HARD edge falls shrply from 1.0 to 0.0. A PE_RANDOM edge jumps randomly between 1.0 and 0.0 throughout its width. A PE_FELTTIP edge is a gaussian function throughout its width. A PE_GIVEN edge is provided as a table pointed to by *edgefunctionptr. The table records the rolloff* starting near 1.0 and falling gradually to 0.0 with *edgewidth* entries.

filter may be one of the following:
PF_GIVEN
PF_SINC
PF_NONE
PF_BOX

Each of these is a convolution filter used to smooth out the edge function. Predefined filter functions include PF_SINC, PF_NONE, and PF_BOX. The PF_NONE alternative is provided to demonstrate the effects of improper anti-aliasing; it is not recommended. The PF_SINC function is the windowed sinc function (sinc(x)*sinc(x/2)). The extent of the filter *filterwidth* is measured in pixels and the recommended filter width is 1. For PF_SINC, this is the distance between the center of the function and its first zero. A PF_GIVEN filter is provided in a table pointed to by filterfunctionptr. All 2* *filterwidth* +1 values of the function should be provided, scaled between 0 and 1.

The edge description table will be created in the scratchpad of the current Chap by PirlPolyLine temporarily if PirlBeginLines has not been called, otherwise it will create the edge decription table in the scratchpad allowing the table to remain allocated for subsequent use. All allocated scratchpad resources will be freed by a call to PirlEndLines.

Calls to PirlSetLineAttributes with PE_GIVEN or PF_GIVEN require that a function be provided by the calling procedure that specify user-supplied edge and filter tables. The edgefunctionptr is not referenced unless PE_GIVEN is specified. The filterfunctionptr is not referenced unless PF_GIVEN is specified.

PirlSetLineColor sets the current color for which all subsequent lines will be drawn by PirlPolyLine until another color is specified in a call to PirlSetLIneColor or PirlSetLineAttributes.

PirlSetLineEdge sets the line attributes and edge characteristics for all subsequent lines drawn by PirlPolyline. The edge specified in the PirlSetLineEdge call must previously have been specified by a call to PirlSetLineAttributes.

PirlSetLineMode set the current mode for which all subsequent lines will be drawn by PirlPolyLine. Valid modes that may be specified are:
PM_REPLACE
PM_MERGE
PM_MAX

A default mode of PF_REPLACE will be used by PirlPolyLine if one has not been specified by a call to PirlSetLineMode.

PirlSetLineWidth set the current line width for all subsequent lines will be drawn by PirlPolyLine. The line width is expressed as a floating point number that will be interpreted to quarter pixel resolution by PirlPolyLine (e.g., line width = .25 to 255.75).

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

BUGS

Other edge types and filter types may be specified, but may produce unpredictable and undesirable results.

NAME

PirlMakeMap – Create a map for changing the colors of a pixel window
Mapfcn – Mapping function for PirlMakeMap

SYNOPSIS

```
#include <pixeldefs.h>
#include "/usr/pixar/include/pirl.h"

PirlError
PirlMakeMap(map, non-neg, clamped, mapfcn)
RGBAPixelType map[];
int non-neg, clamped;
int (*mapfcn)();

int mapfcn(val, pxl)
double val, pxl[4];
```

DESCRIPTION

PirlMakeMap() expresses a functional mapping between 12-bit Pixar pixel values as a table (**map**) indexed by these bits of those pixels. It must be given the table, a pointer to a function **mapfcn** mapping from the range [-.5,1.5). The *map* parameter has either 2048, 3072 or 4096 four-value elements, depending on the parameters *non-neg* and *clamped*: *non-neg* indicates that the map is to exclude negative pixel values, thus is 1024 entries smaller. *clamped* indicates that the map excludes values greater than 1, so is 1023 entries smaller. These flags are meant to be passed also to the function **PirlMap** (3H), which applies the map to a pixel window in Pixar image memory.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlMap (3H)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

It is a serious, undiagnosable error for the table passed to be too small for the state of **clamped** and **non_neg.**

Since this function allocates resources on the Chap that owns the pixel window, all **Chad** errors apply.

NAME

PirlMap – remap 4 components of a pixel window

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
#include <pixeldeft.h>

PirlError
PirlMap(pw, map, non_neg, clamped)
ChadPW *pw;
RGBAPixelType map[];
int non_neg, clamped;
```

DESCRIPTION

PirlMap changes the color values of a pixel window according to a function expressed as a map between pixel values in and pixel values out. Since pixel values are twelve bits wide, the map will normally contain 4096 four-channel elements. If the pixel values are all non-negative, a non-zero *non_neg* value will indicate that the map covers only those values, and is 1024 elements smaller as a result. Likewise, for pixels all less than or equal to 1.0, the *clamped* flag changes the assumed table size by 1023 elements. The principle benefit of these assumptions is minimizing the size of the table which must be loaded into the Chap's scratchpad. The function *PirlMakeMap*(3H) will construct the map table.

The mapping defined by the table gives each output channel value strictly as a function of that channel. Output in which a channel depends on the other channels in the pixel may be produced using *PirlCha*(3H).

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlMapComp(3H), PirlMakeMap(3H), PirlCha(3H)

PW4Map(3C), PWMap(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

PirlMap is most likely to fail due to the scratchpad being full. It is also a serious, undiagnosable error for the table as passed to be too small for the state of *clamped* and *non_neg*.

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlMapComp – map a single component through a color table to form a color image

SYNOPSIS

PirlMapComp(pwsrc, channel, table, tablesize)
ChadPW *pwsrc;
RGBAPixelType map[];
int channel, tablesize;

DESCRIPTION

PirlMapComp creates a color image from a single channel image by using the value in the single channel image as in index into a color table. The given component, *channel*, which must be a number from 0 to 3 (representing red through alpha) of the image stored in the source pixel window, *pwsrc*, are mapped and written back to the same window.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlMap(3H), PirlMakeMap(3H), PirlCha(3H)

SS4Map(3C), PWMap(3C)

SSRtoRGBALUT(3C), SSGtoRGBALUT(3C), SSBtoRGBALUT(3C), SSAtoRGBALUT(3C)

NAME

PirlMerge – composite a foreground with a background

SYNOPSIS

#include "/usr/pixar/include/merge.h"
#include "/usr/pixar/include/pirl.h"
PirlError
PirlMerge (fgd, bkg, dst, op, Lf, Lb)
ChadPW *fgd, *bkg, *dst;
MergeOp op;
RGBAPixelType *Lf, *Lb;

DESCRIPTION

PirlMerge implements the Porter-Duff compositing algegra on a pair of pixel windows, given by *fgd* and *bkg*, writing the composited pixels into *dst* (the dimensions of the three pixel windows must be identical, but the pixel windows themselves need not be distinct).

The paper "Compositing Digital Images," in *SIGGRAPH '84*, discusses the semantics of the merging operators. Briefly, compositing is performed by combining images using the fourth (alpha) channel in the image as a *matte* giving the opacity of image at each pixel. The assumption is that the interesting information in an image is confined to pixels with non-zero opacity, so that the matte may be used to allow backgrounds to show through, in proportion to the value of alpha.

The operator, given by *op*, is one of the following tokens defined in <merge.h>:

MergeOpCLEAR -- Clear the destination window

MergeOpCOPY -- Copy the foreground

MergeOpNOOP -- Copy the background

MergeOpOVER ("merge foreground over background")
-- Copy both foreground and background, copying foreground where they intersect.

MergeOpUNDER ("merge foreground under background")
-- Copy both foreground and background, copying background where they intersect.

MergeOpOUT ("use foreground held out by background")
-- Copy those parts of the foreground which lie outside the background

MergeOpIN ("use background held out by foreground")
-- Copy those parts of the background which intersect the foreground

MergeOpABOVE ("copy foreground above background")
-- Like **in**, but also copies background pixels lying outside the foreground

MergeOpBELOW ("copy background above foreground") -- opposite of **above**

MergeOpXOR ("foreground or background, but not both")
-- Copies foreground **and** background, except where they intersect.

MergeOpPLUS ("add pixels") -- Sums the pixel values.

MergeOpPLUSIN ("sum pixels in intersection")
-- Takes the sum of the two images, writing the result where the background appears.

MergeOpPLUSBELOW ("sum pixels above background")
-- Mix pixels where foreground and background intersect;
copy background elsewhere

MergeOpPLUSABOVE ("sum pixels above foreground")
-- Mix pixels where foreground and background intersect; copy foreground elsewhere

The pixel structures *Lf* and *Lb* give attenuation factors for the channels of the foreground and background images, respectively. They should be pixel values in the interval [-.5,1.5). Pixel values in this range may be obtained from floating-point values in the same range with the macro DBL2PXL.

SEE ALSO

Compositing Digital Images (*Pixar Programmer's Manual*) – the above paper, excerpted.
merge(1)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

ERRORS

PirlMerge will fail if it cannot allocate sufficient scratchpad memory for storing two full scan lines, or if there is no room in instruction memory for the Chap routines it uses. It also fails if the pixel windows it is given do not match in x and y dimensions.

BUGS

As discussed in the paper, this style of compositing is susceptible to failure on correlated data, for example when two images depict objects with adjacent edges.

NAME

PirlNewPW – Make a new pixel window under Pirl

SYNOPSIS

 # include "/usr/pixar/include/pirl.h"

PirlError PirlNewPW(pwp, xmin, xmax, ymin, ymax)
PirlPW *pwp;
int xmin, xmax, ymin, ymax;

DESCRIPTION

PirlNewPW() allocates a new pixel window under **Pirl** of the given dimensions. It returns a *PirlError* value, which should be checked for a nonzero (i.e., error) value.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SOURCE

/usr/pixar/host/src/lib/libpirl/begin.c -- source for **PirlBegin()** and
PirlEnd ()

SEE ALSO

libpirl(3H), PirlBegin(3H)

The *Pirl Tutorial*, in *The Pixar Programmer's Manual*, serves as an introduction to Pixar programming using **Pirl**.

NAME

PirlNot – subtract pixel value from 1.0 (0x800) for a pixel window

SYNOPSIS

#include "/usr/pixar/include/pirl.h"
PirlError
PirlNot (pw)
ChadPW *pw;

DESCRIPTION

PirlNot subtracts the pixel value from 1.0 (0x800) for each pixel in the pixel window.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWNot(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlRampX,
PirlRampY – draw a ramp into a pixel window

SYNOPSIS

#include "/usr/pixar/include/pirl.h"
PirlError
PirlRampX (pw, left, right)
ChadPW *pw;
RGBAPixelType *left, *right;

PirlError
PirlRampY (pw, top, bottom)
ChadPW *pw;
RGBAPixelType *top, *bottom;

DESCRIPTION

PirlRampX draws a ramp in which the pixels in a column are identical, and the pixels in a row are linearly interpolated between pixel values *l* and *r*. *PirlRampY* performs the converse operation.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlRange – find the minimum and maximum pixel values in a pixel window

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlRange (pw, min, max)
ChadPW *pw;
RGBAPixelType *min, *max;
```

DESCRIPTION

PirlRange finds the minimum and maximum values of each component within a pixel window.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWRange(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlReflectX – reflect the pixel window around its center horizontal axis
PirlReflectY – reflect the pixel window around its center vertical axis

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlReflectX (pw)
ChadPW *pw;

PirlError
PirlReflectY (pw)
ChadPW *pw;
```

DESCRIPTION

PirlReflectX turns a pixel window upside-down by reflecting it around its horizontal axis.

PirlReflectY mirrors a pixel window left-right by reflecting it around its vertical axis.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlCopyGeneric(3H), perm(1)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlResize – copy the source pixel window to the destination pixel window

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlResize(srcpw,dstpw,hextent,vextent)
ChadPW *srcpw;
ChadPW *dstpw;
int         hextent,vextent;
```

DESCRIPTION

PirlResize resizes the source pixel window to the destination pixel window. The two pixel windows must not overlap.

The hextent and vextent are the filter widths for horizontal and vertical scaling. An extent of four, a cubic filter, produces produces the best resized images, at the cost of using more intermediate scratchpad memory and more time. An extent of two, a linear filter, produces an acceptable resizing, but some image quality is lost.

Different horizontal and vertical filter widths may be specified.

For programming examples, see the source for *resize* shell command.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWResize(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Both pixel windows must belong to the same chap. The two pixels window must not overlap.

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlRotate – rotate a pixel window

SYNOPSIS

#include <chad.h>
#include <pirl.h>
PirlError
PirlRotate(srcpw, tmppw, dstpw, angle, sx, sy, cx, cy,clr,extent)
ChadPW *srcpw,*tmppw,*dstpw;
double angle,sx,sy,cx,cy;
int clr,extent;

DESCRIPTION

PirlRotate rotates, scales,and translates a source pixel window to a destination pixel window. The transformation is performed using a two-pass (horizontal and vertical) resampling algorithm.

Source and destination pixel windows are specifed for each pass of the resampling algorithm. The source and destination pixel windows for each pass cannot partially overlap. **srcpw** defines the transformation source pixel window. **tmppw** defines the destination window for the first (intermediate) pass and the source for the second pass. **dstpw** defines the the destination for the second (final) pass. If the **dstpw** is smaller than the **srcpw**, the **tmppw** must be greater than or equal to the size of the **srcpw**. The transformation can be done in place, so **srcpw**, **tmppw**, and **dstpw** can all point to the same pixel window.

angle specifies the number of degrees to rotate.
sx, sy specifies the scale factors for the horizontal and vertical dimensions.
cx, cy specifies the center of rotation.
clr is a flag wich specifies whether rotate should clear out its borders.
extent specifies the filter width (in pixels). An extent of four will use a cubic filter. An extent of two will use a linear filter.

FILES

/usr/pixar/host/src/lib/libpirl/affine.c

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

Rotate(1), PirlAffine(3h), PirlShear(3h), PWShear(3c)

NAME

PirlSetChannelMask – set a pixel window's channel mask

SYNOPSIS

#include "/usr/pixar/include/pirl.h"
PirlError
PirlSetChannelMask (pw,mask)
ChadPW *pw;
int mask;

DESCRIPTION

PirlSetChannelMask() sets the channel mask for the pixel window for future writing. The *mask* is a number between 0 (all channels off) and 0xf (all channels on). The channel mask is formed by OR-ing the following mask bits:

red - 0x1
green - 0x2
blue - 0x4
alpha - 0x8

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

SetMaskPW

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlShear – Shear a pixel window

SYNOPSIS

```
#include <chad.h>
#include <pirl.h>
PirlError
PirlShear(srcpw,dstpw,scale,off,inc,access,clr,extent,iw,ih)
ChadPW *srcpw,*dstpw;
double scale,off,inc;
int access,clr,extent,iw,ih;
```

DESCRIPTION"

PirlShear shears a source pixel window and places the result in a destination pixel window.

srcpw defines the source pixel window to shear.
dstpw defines the the destination pixel window.
scale specifies a scale factor for resizing each scanline
off specifes the offset for the first destination scanline.
inc specifes the incremental offset for each additional destination scanline.
access specifies the scanline access directions for the src and dst.
access must be one of the following defined options:

```
#define XIN_XOUT              0
#define XBACKWARDSIN_XOUT     1
#define YIN_XOUT              2
#define YBACKWARDSIN_XOUT     3
#define YIN_YOUT              4
#define YBACKWARDSIN_YOUT     5
#define XIN_YOUT              6
#define XBACKWARDSIN_YOUT     7
```

clr is a flag wich specifies whether shear should clear out its borders.
extent specifies the filter width (in pixels). An extent of four will use a cubic filter. An extent of two will use a linear filter.
iw,ih specify the input width and height of the src window.

FILES

/usr/pixar/host/src/lib/libpirl/affine.c

SEE ALSO

Rotate(1), PirlRotate(3h), PirlAffine(3h), PWShear(3c)

NAME

PirlShift – shift pixel window contents in x and/or y
PirlCircularShift – circular shift pixel window contents in x and/or y

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlShift(pw,x,y)
ChadPW *pw;
int x,y;

PirlError
PirlCircularShift(pw,x,y)
ChadPW *pw;
int x,y;
```

DESCRIPTION

PirlShift shifts the contents of a pixel window in *x* and/or *y*. The pixels shifted outside the pixel window are clipped. The original pixels are retained in the exposed area. *X* and *Y* may be positive or negative.

PirlCircularShift shifts the contents of a pixel window in *x* and/or *y*. The pixels shifted outside the pixel window are circularly shifted around the edge of the pixel window into the exposed area.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWShift(3C), perm(1)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlShuffle – shuffle components of each pixel for a pixel window.

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlShuffle (pw,perm)
ChadPW *pw;
char perm[4];
```

DESCRIPTION

PirlShuffle shuffles the contents of the pixel window using the string *perm.* This string forms a general purpose crossbar for shuffling of pixel components. For instance, the string ''rrrr'' places the red component of each pixel into all four components. ''grba'' exchanges the red and green components.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWShuffle(3C), perm(1)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

The permutation string must be four characters of [rgbaRGBA].

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlSwap – swap the source pixel window and the destination pixel window

SYNOPSIS

```
#include "/usr/pixar/include/pirl.h"
PirlError
PirlSwap (srcpw,dstpw)
ChadPW *srcpw;
ChadPW *dstpw;
```

DESCRIPTION

PirlSwap swaps the source pixel window and the destination pixel window.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWSwap(3C), FSCopy(3C), SFCopy(3C)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

Both pixel windows must be the same size, not overlap, and belong to the same Chap.

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlSweepX,
PirlSweepY　　– copy one scanline repeatedly into a pixel window

SYNOPSIS

#include "/usr/pixar/include/pirl.h"
PirlError
PirlSweepX (pw, line)
ChadPW *pw;
RGBAPixelType line[ ];

PirlError
PirlSweepY (pw, line)
ChadPW *pw;
RGBAPixelType line[ ];

DESCRIPTION

PirlSweepX and *PirlSweepY* copy a single line of pixels into a pixel window. *PirlSweepX* copies the line to every row in the pixel window. *PirlSweepY* copies the line to every column in the pixel window.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PirlClear(3H)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

The *line* buffer of pixel values must be of appropriate size. These routines cannot confirm this. They may also fail if insufficient resources are available on the Chap.

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlTranspose – transpose a pixel window around the diagonal axis.

SYNOPSIS

#include "/usr/pixar/include/pirl.h"
PirlError
PirlTranspose (pw)
ChadPW *pw;

DESCRIPTION

PirlTranspose() transposes a pixel window around its diagonal axis (0,0) to (N,N). Any orientation of the pixel window may be achieved by a combination of transpositions and reflections.

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SEE ALSO

PWTranspose(3C), PirlReflect(3H), perm(1)

The *Chad Tutorial*, and also the *Chap Programming Tutorial* in *The Pixar User's Manual*, discuss allocation of pixel windows.

ERRORS

The pixel window must be square or the token PIRL_NOT_SQUARE is returned.

Since this function allocates resources on the Chap that owns the pixel window, all *Chad* errors apply.

NAME

PirlZoom – Zoom in on a pixel window on the monitor

SYNOPSIS

 # include "/usr/pixar/include/pirl.h"

PirlError PirlZoom(zoomrate)
int zoomrate;

DESCRIPTION

PirlZoom () sets the video board of a Pixar Image Computer to duplicate pixels between the frame buffer and the display, in effect making the pixels larger by a factor given by *zoomrate*.

The top left pixel of the display is the base for the zoom; after zooming up, pixels will appear shifted to the right and down in proportion to their distance from the upper left.

The pixel in the frame buffer which is seen at upper left in the display is set by *PirlDisplay* (3H).

zoomrate must lie in the range [0..15].

LIBRARIES

/usr/pixar/host/lib/libpirl.a /usr/pixar/host/lib/libchad.a /usr/pixar/host/lib/libpixar.a

SOURCE

/usr/pixar/host/src/lib/libpirl/begin.c -- source for **PirlZoom()**

SEE ALSO

libpirl(3H), PirlDisplay(3H)

The *Pirl Tutorial*, in *The Pixar Programmer's Manual*, serves as an introduction to Pixar programming using **Pirl**.

NAME

libpixar – introduction to Host-resident Pixar library functions

DESCRIPTION

libpixar is a library of C-callable functions which provide a low-level interface to the Pixar. Much of this functionality has been absorbed into *Chad*, a simplified library for executing programs on the Pixar's Channel Processor.

This section describes functions used by programs executing on the host. The functions described in this section are found in the library *libpixar*.

FILES

/usr/pixar/lib/libpixar.a	low-level Pixar functions
/usr/pixar/lib/libpixar_p.a	profiled versions of libpixar.a functions

SEE ALSO

Programming the Pixar with Chad, in the *Pixar Programmer's Manual*,
video(1), mctrl(8), Chap*(3H), dumiopen(3H), video*(3H), mctrlopen(3H), dbopen(3H), dumi(4), Chap(4), video(4), mctrl(4)

intro(1)	– list of shell-callable Pixar programs
intro(3C)	– list of libraries of device-resident routines
intro(3H)	– list of libraries of host-resident routines
libchad(3H)	– library of high-level Chap interface routines

LIST OF FUNCTIONS

Name	*Appears on Page*	*Description*
ChapAfillSpad	Chapspad(3H)	–fill each Chap scratchpad memory with its address
ChapAllocFB	Chapmman(3H)	–allocate/free space in the framebuffer
ChapAllocRam	Chapmman(3H)	–allocate/free space in instruction RAM
ChapAllocSpad	Chapmman(3H)	–allocate/free space in scratchpad
ChapAppendArchives	Chapload(3H)	–manipulate dynamic loading archive list
ChapBeginLoad	Chapload(3H)	–begin/end a dynamic load or unload
ChapClose	Chapopen(3H)	–open/close a Chap device
ChapClrBpt	Chapbpt(3H)	–set and clear Chap breakpoints
ChapCont	Chaprun(3H)	–continue the current Chap program
ChapDumpRam	Chapram(3H)	–read/write multiple locations in Chap instruction memory
ChapDumpSpad	Chapspad(3H)	–read/write multiple locations in Chap scratchpad
ChapDynamicLoad	Chapload(3H)	–dynamically load files into a Chap
ChapDynamicUnload	Chapload(3H)	–dynamically unload files from a Chap
ChapDynamicUnloadAll	Chapload(3H)	–dynamically unload files from a Chap
ChapEndLoad	Chapload(3H)	–begin/end a dynamic load or unload
ChapExecInst	Chapinst(3H)	–manipulate Chap diagnostic instruction register
ChapFillSpad	Chapspad(3H)	–fill Chap scratchpad memory with a single value
ChapFreeFB	Chapmman(3H)	–allocate/free space in the framebuffer
ChapFreeRam	Chapmman(3H)	–allocate/free space in instruction RAM
ChapFreeSpad	Chapmman(3H)	–allocate/free space in scratchpad
ChapGetArchives	Chapload(3H)	–manipulate dynamic loading archive list
ChapGetConfig	Chapconfig(3H)	–get/set Chap configuration information
ChapGetFB	Chapmman(3H)	–allocate/free space in the framebuffer
ChapGetMap	Chapmman(3H)	–get allocation map from kernel
ChapGetRam	Chapmman(3H)	–allocate/free space in instruction RAM
ChapGetSDR	Chapreg(3H)	–get the value of shared data register
ChapGetSpad	Chapmman(3H)	–allocate/free space in scratchpad
ChapHalt	ChapRun(3H)	–halt the Chap
ChapLoad	Chapload(3H)	–dynamically load files into a Chap
ChapLoadFile	Chapold(3H)	–load a non-relocatable Chap object file [DEFUNCT]

ChapLoadGo	ChapLoadGo(3H)	–dynamically load and start up a file in the Chap
ChapLoadInst	Chapinst(3H)	–manipulate Chap diagnostic instruction register
ChapLoadRam	Chapram(3H)	–read/write multiple locations in Chap instruction memory
ChapLoadSpad	Chapspad(3H)	–read/write multiple locations in Chap scratchpad
ChapLoadSymbol	Chapload(3H)	–dynamically load files into a Chap
ChapMMan	Chapmman(3H)	–enable/disable memory management
ChapOpen	Chapopen(3H)	–open/close a Chap device
ChapReadAbusDev	Chapabus(3H)	–read a Chap Abus device
ChapReadAcc	Chapalu(3H)	–read a Chap alu register
ChapReadInst	Chapinst(3H)	–manipulate Chap diagnostic instruction register
ChapReadLc	Chapstack(3H)	–read stacked Chap register
ChapReadMbusDev	Chapmbus(3H)	–read/write Chap Mbus devices
ChapReadPc	Chapstack(3H)	–read stacked Chap register
ChapReadRam	Chapram(3H)	–read/write one location in Chap instruction memory
ChapReadReg	Chapalu(3H)	–read a Chap alu register
ChapReadRunflag	Chapstack(3H)	–read stacked Chap register
ChapReadRunflags	Chapstack(3H)	–read/write Chap runflag state
ChapReadSbus	Chapsbus(3H)	–read Chap Sbus
ChapReadSbusDev	Chapsbus(3H)	–read/write Chap Sbus device
ChapReadSpad	Chapspad(3H)	–read/write one location in Chap scratchpad
ChapReadXbar	Chapxbar(3H)	–read/write Chap crossbar state
ChapReset	Chapreset(3H)	–reset Chap and framebuffer state
ChapResetInterrupt	Chapwait(3H)	–reset Chap interrupt handling
ChapResetMap	Chapmman(3H)	–reset allocation map to default state
ChapRun	Chaprun(3H)	–run a Chap program
ChapRunAsync	Chaprun(3H)	–run a Chap program and return
ChapSetArchives	Chapload(3H)	–manipulate dynamic loading archive list
ChapSetInterrupt	Chapwait(3H)	–set Chap interrupt handling
ChapSetBpt	Chapbpt(3H)	–set and clear Chap breakpoints
ChapSetConfig	Chapconfig(3H)	–get/set Chap configuration information
ChapSetSDR	Chapreg(3H)	–set the value of shared data register
ChapSetVDR	Chapreg(3H)	–set the value of virtual data register
ChapStep	Chaprun(3H)	–single step a Chap program
ChapSymEnter	Chapsym(3H)	–Chap symbol table routines
ChapSymLookup	Chapsym(3H)	–Chap symbol table routines
ChapSymX	Chapsym(3H)	–Chap symbol table routines
ChapVdregBase	Chapreg(3H)	–get the base address of virtual data registers
ChapWaitForInterrupt	Chapwait(3H)	–pause waiting for an interrupt from a Chap
ChapWriteAcc	Chapalu(3H)	–write a Chap alu register
ChapWriteLc	Chapstack(3H)	–write stacked Chap register
ChapWriteMbusDev	Chapmbus(3H)	–read/write Chap Mbus devices
ChapWritePc	Chapstack(3H)	–write stacked Chap register
ChapWriteRam	Chapram(3H)	–read/write one location in Chap instruction memory
ChapWriteReg	Chapalu(3H)	–write a Chap alu register
ChapWriteRunflag	Chapstack(3H)	–write stacked Chap register
ChapWriteRunflags	Chapstack(3H)	–read/write Chap runflag state
ChapWriteSbusDev	Chapsbus(3H)	–read/write Chap Sbus device
ChapWriteSpad	Chapspad(3H)	–read/write one location in Chap scratchpad
ChapWriteXbar	Chapxbar(3H)	–read/write Chap crossbar state
ChapXSym	Chapsym(3H)	–Chap symbol table routines
DbClose	dbopen(3H)	–setup a disk buffer device for use
DbOpen	dbopen(3H)	–setup a disk buffer device for use
DumiClose	dumiopen(3H)	–setup a Dumi device for diagnostic use

NAME

ChapReadAbusDev – read a Chap Abus device

SYNOPSIS

#include < pixar/pixar.h >

u_short ChapReadAbusDev(chap, dev, bus)
CHAP *chap; u_int dev, bus;

DESCRIPTION

This routine reads the contents of an Abus device. The specific Abus must be supplied in *bus*. Abus devices are defined in the file *<pixar/chap.h>*.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

ChapReadReg,
ChapReadAcc – read a Chap alu register
ChapWriteReg,
ChapWriteAcc – write a Chap alu register

SYNOPSIS

#include < pixar/pixar.h >

ChapReadReg(chap, reg, bus)
CHAP *chap; u_int reg, bus;

ChapReadAcc(chap, bus)
CHAP *chap; u_int bus;

ChapWriteReg(chap, reg, rf, v)
CHAP *chap; u_int reg, rf; v;

ChapWriteAcc(chap, rf, v)
CHAP *chap; u_int rf; v;

DESCRIPTION

These routines read/write the contents of one 29116A RAM location (register) or accumulator in the pr cessors specified. The *bus* parameter selects the processor from which the data is to be retrieved. The parameter is a runflag to be used in selecting the processors to which data should be written. The *r* identifies a particular RAM location (0-31).

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

ChapSetBpt,
ChapClrBpt – set and clear Chap breakpoints

SYNOPSIS

#include < pixar/pixar.h >

ChapSetBpt(chap, addr)
CHAP *chap; u_short addr;

ChapClrBpt(chap, addr)
CHAP *chap; u_short addr;

DESCRIPTION

ChapSetBpt and *ChapClrBpt* set and clear breakpoints in the specified instruction; they operate by manipulating the *i_bpt* field of the micro-instruction word.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

ChapGetConfig,
ChapSetConfig – get/set Chap configuration information

SYNOPSIS

#include < pixar/pixar.h >

ChapGetConfig(chap, conf)
CHAP *chap; struct chapconf *conf;

ChapSetConfig(chap, conf)
CHAP *chap; struct chapconf *conf;

DESCRIPTION

ChapGetConfig returns a structure *conf*, containing information about the hardware configuration of a Cha and/or associated framebuffer. The configuration structure is defined in *<pixardev/chapioctl.h>* an described in *chap*(4).

ChapSetConfig is used to set the values in the configuration structure.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H), chap(4), Chconfig (8)

NAME

ChapExecInst,
ChapLoadInst,
ChapReadInst – manipulate Chap diagnostic instruction register

SYNOPSIS

#include < pixar/pixar.h >

ChapExecInst(chap, ip)
CHAP *chap; inst_t *ip;

ChapLoadInst(chap, ip)
CHAP *chap; inst_t *ip;

ChapReadInst(chap, ip)
CHAP *chap; inst_t *ip;

DESCRIPTION

These routines load, execute, or read a single Chap instruction. *ChapLoadInst* loads one instruction into the instruction register, but does not execute it. *ChapExecInst* loads an instruction and executes it. *ChapReadInst* returns the instruction currently in the instruction register.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

ChapLoad,
ChapLoadSymbol,
ChapDynamicLoad – dynamically load files into a Chap
ChapDynamicUnload,
ChapDynamicUnloadAll – dynamically unload files from a Chap
ChapBeginLoad,
ChapEndLoad – begin/end a dynamic load or unload

SYNOPSIS

#include < pixar/pixar.h >

ChapLoad(chap, file, locs)
CHAP *chap; char *file; ChapLoadLocs *locs;

ChapLoadSymbol(chap, symname, locs)
CHAP *chap; char *symname; ChapLoadLocs *locs;

ChapDynamicLoad(chap, file, locs)
CHAP *chap; char *file; ChapLoadLocs *locs;

ChapDynamicUnload(chap, file)
CHAP *chap; char *file;

ChapDynamicUnloadAll(chap)
CHAP *chap;

ChapBeginLoad(chap)
CHAP *chap;

ChapEndLoad(chap)
CHAP *chap;

DESCRIPTION

The routines described here provide a facility for link-editing and loading Chap object files at runtim This "dynamic loader" package allows any **relocatable** object file generated by the Chap assembl *chas*(1), or Chap link-editor, *chld*(1), to be linked with code already resident in a Chap. The dynam loader maintains a master symbol table of allocated memory and the definition of symbols resident in t machine.

Several routines exist for loading. These routines differ in terms of their interaction with libraries. T routine *ChapLoad* automatically searches a global list of libraries (set with *ChapSetArchives*, *ChapPrepe dArchives*, and *ChapAppendArchives*) to resolve any undefined references in a file being loaded. On t other hand, the routine *ChapDynamicLoad* loads a single file without searching the archives. In both case the *file* specified is scanned to find symbol definitions, then memory is allocated in instruction RAM a scratchpad to accomodate the downloading of the text, data, and bss segments. Finally, once the necessa resources have been allocated, the module is relocated as it is downloaded into the Chap. Symbols defin in the file are merged into the master symbol table and any previously undefined references to exter symbols defined in this module are resolved.

The *locs* parameter allows the user to specify locations in the Chap at which to load the file. Both *Ch pLoad* and *ChapDynamicLoad* return the location and sizes of each segment assigned to the file as well the entry point in the text segment.

ChapLoad and *ChapDynamicLoad* will load a module with undefined symbols, reporting each undefin symbol on the standard error output, but will not load a module that attempts to redefine existing symbo This means, for example, that two instances of the same file will not be loaded twice (presuming the sar symbols are defined in each file). Both routines return the number of undefined references in the load file, or −1 if an error was encountered.

The routine *ChapLoadSymbol* works similarly to *ChapLoad*, but initiates loading based on a symbol's name. This is only possible when the library containing the symbol has been disclosed to a previous *ChapSetArchives, ChapPrependArchives,* or *ChapAppendArchives* call.

To unload a file previously loaded, the routine *ChapDynamicUnload* should be used. This routine removes the *file*'s symbols from the master symbol table, releases instruction and scratchpad resources, and ''revokes'' references to symbols defined in the module being unloaded.

The routine *ChapDynamicUnloadAll* purges the symbol table of all files previously loaded. Note that this does not necessarily remove all symbols, or free up all resources.

ENVIRONMENT VARIABLES

CHAPDEBUG controls printing of debug messages. It is interpreted as a numeric value with bit fields. If CHAPDEBUG is not set, a value of zero is assumed. The bottom two bits control the display of loading messages on stdout as follows:

0 - Display no messages
1 - Display "." for each chap module loaded.
2 - Display filenames
3 - Display filenames and archive names.

Bit 2 (numeric value 4) enables warning messages for unsatisfied externals.

FILES

/usr/pixar/host/symtab/chap*	master symbol tables
/dev/chap*	Chap special devices

SEE ALSO

ChapOpen(3H), chload(1), chmap(1)

NAME

ChapLoadGo　　– dynamically load and start up a file in the Chap

SYNOPSIS

#include < pixar/pixar.h >

ChapLoadGo(chap, file, entry)
CHAP *chap; char *file, *entry;

DESCRIPTION

ChapLoadGo is the simplest interface to the dynamic loading facilities, *chapload*(3H). *ChapLoadG* checks to see if the module associated with the symbol *entry* is present in the Chap. If the code is n present, *file* is loaded and *ChapLoadGo* attempts to resolve any undefined references by searching a stai dard set of libraries (see below). Finally, the Chap is set running at the location associated with the *ent* symbol.

The *file* specified must be a relocatable object file created by *chas*(1) or *chld*(1). If no *entry* point specified, *ChapLoadGo* starts the Chap running at the first instruction in *file*.

ChapLoadGo attempts to recover from running out of space in the instruction or scratchpad memories t purging all resident code and data and starting over. If the second try fails for any reason, it gives up.

ChapLoadGo returns −1 if an error occurred during loading, 0 if it was able to accomplish the wo without purging the symbol table, and a positive value if it had to flush the symbol table. This last indic tion may be of importance in case the caller is holding references to data structures previously allocated scratchpad.

FILES

/usr/pixar/chap/lib/libpG.a	default library searched
/usr/pixar/chap/lib/libpt.a	default library searched
/usr/pixar/chap/lib/libpx.a	default library searched
/usr/pixar/symtab/chap*	master symbol tables
/dev/chap*	Chap special devices

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

chapopen(3H), chapload(3H), chload(1), chmap(1)

NAME

ChapReadMbusDev,
ChapWriteMbusDev – read/write Chap Mbus devices

SYNOPSIS

#include < pixar/chap.h >

u_short ChapReadMbusDev(chap, dev, bus)
CHAP *chap; u_int dev, bus;

ChapWriteMbusDev(chap, dest, rf, v)
CHAP *chap; u_int dest, rf; u_short v;

DESCRIPTION

ChapReadMbusDev reads the contents of a single Mbus device. *ChapWriteMbusDev* writes the contents of one or more Mbus devices. The *bus* argument to *ChapReadMbusDev* specifies which Mbus to read from. The *rf* parameter should be a runflag that enables one or more Mbuses for writing. The *dest* argument to *ChapWriteMbusDev* is a mask of devices to enable for writing, as specified in <pixar/chap.h>.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

ChapAllocFB, ChapFreeFB, ChapGetFB	– allocate/free space in the framebuffer
ChapAllocSpad, ChapFreeSpad, ChapGetSpad	– allocate/free space in scratchpad
ChapAllocRam, ChapFreeRam, ChapGetRam	– allocate/free space in instruction RAM
ChapGetMap	– get allocation map from kernel
ChapResetMap	– reset allocation map to default state
ChapMMan	– enable/disable memory management

SYNOPSIS

```
#include < pixar/pixar.h >

ChapAllocFB(chap, size)
CHAP *chap; int size;

ChapFreeFB(chap, addr, size)
CHAP *chap; u_short addr, size;

ChapGetFB(chap, addr, size)
CHAP *chap; u_short addr, size;

ChapAllocSpad(chap, size)
CHAP *chap; int size;

ChapFreeSpad(chap, addr, size)
CHAP *chap; u_short addr, size;

ChapGetSpad(chap, addr, size)
CHAP *chap; u_short addr, size;

ChapAllocRam(chap, size)
CHAP *chap; int size;

ChapFreeRam(chap, addr, size)
CHAP *chap; u_short addr, size;

ChapGetRam(chap, addr, size)
CHAP *chap; u_short addr, size;

#include<sys/map.h>
#include<pixardev/chapioctl.h>
ChapGetMap(chap, map, count, pmap)
CHAP *chap; int map, *count; struct map *pmap;

ChapResetMap(chap, map)
CHAP *chap; int map;

ChapMMan(chap, onoff)
CHAP *chap; int onoff;
```

DESCRIPTION

The routines described here interface to the memory management facilities provided by the Chap interfac *chap*(4). The device driver maintains three resource maps for tracking the use of Chap instructi memory, Chap scratchpad memory, and framebuffer memory (associated with a particular Chap or Chap: Host-based programs may allocate space from these resource maps with the *ChapAllocFB*, *ChapA locSpad*, and *ChapAllocRam* calls. To free a previously allocated resource, the equivalent "free" routi should be used. Note that when freeing space the address and size must be supplied. Multiple free ca may be coalesced into a single call by appropriately adjusting the address and/or size parameters.

In addition to the normal allocation interface, a ''get'' interface is also supported. In this form, the allocation specifies not only a size but also a location at which the allocation should be performed.

The allocation granularity is in the *natural* unit of the associated resource; size parameters should be adjusted accordingly.

framebuffer	tile block	32x32 pixels
scratchpad	pixel	4 words
instruction memory	instruction	96-bits

To retrieve an allocation map, or portion of, *ChapGetMap* should be used. The *map* parameter should be one of CALLOC_RAM, CALLOC_SPAD, or CALLOC_FB, as defined in *<pixardev/chapioctl.h>*.

The routine *ChapResetMap* may be used to reset one or more allocation maps to their default state (everything free). The *map* parameter may be one of the map identifiers described above or CALLOC_ALL (also defined in *<pixardev/chapioctl.h>*) in which case all maps are reset. Note that this routine should normally not be used directly as it can leave the resource allocation maps inconsistent with the contents of the symbol table. To cleanly flush the state of a Chap, the routine *ChapReset* should be used; see *ChapReset*(3H).

The *ChapMMan* call controls whether the device driver intercepts interrupts from the Chap and interprets them, potentially, as memory management requests. If *onoff* is non-zero, this ''memory management facility'' is enabled; supplying a zero value for *onoff* disables the facility. The memory management protocol used is fully described in *chap*(4).

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

mman(3C), ChapOpen(3H), ChapReset(3H), chap(4)

NAME

ChapOpen,
ChapClose　　– open/close a Chap device

SYNOPSIS

#include<pixar/pixar.h>

CHAP *ChapOpen(device, shared)
char *device; int shared;

ChapClose(chap)
CHAP *chap;

DESCRIPTION

ChapOpen opens the specified Chap device and initializes the data structures used by the library. The *dev ice* specifies a Chap special file as described in *chap*(4). The locking protocol enforced by the devic driver permits access to only one user at a time. Multiple processes owned by the same user may, how ever, access the same Chap depending on the *shared* parameter. If *shared* is 1, other processes requestin shared access may open the same Chap, otherwise noone else will be allowed to open the Chap. *ChapO pen* returns a pointer to a **CHAP** structure, which must be supplied in all subsequent calls to routines in th library. A 0 value is returned if *ChapOpen* was unable to open the Chap.

The *ChapClose* routine closes all open files, unmaps memory previously mapped to the Chap diagnosti registers, and frees up all dynamically allocated memory.

FILES

/dev/chap*　　Chap special files

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

chap(4)

DIAGNOSTICS

%s: Device in use. The device is currently in use.

%s: Device busy. The device is currently open for exclusive use by another processor owned by th caller.

In addition, there are various messages about running out of memory or being unable to open files. Th symbol table routines may also generate messages regarding the symbol table file.

NAME

ChapReadRam,
ChapWriteRam – read/write one location in Chap instruction memory
ChapLoadRam,
ChapDumpRam – read/write multiple locations in Chap instruction memory

SYNOPSIS

```
#include < pixar/pixar.h >

ChapReadRam(chap, addr, ip)
CHAP *chap; u_short addr; inst_t *ip;

ChapWriteRam(chap, addr, ip)
CHAP *chap; u_short addr; inst_t *ip;

ChapLoadRam(chap, addr, count, ip)
CHAP *chap; u_short addr, count; inst_t *ip;

ChapDumpRam(chap, addr, count, ip)
CHAP *chap; u_short addr, count; inst_t *ip;
```

DESCRIPTION

These routines read and write Chap instruction memory. *ChapReadRam* and *ChapWriteRam* read and write a single Chap instruction at the address specified. *ChapLoadRam* and *ChapDumpRam* perform the equivalent operations, but for multiple instructions.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

ChapSetSDR
ChapGetSDR
ChapSetVDR,
ChapVdregBase – Chap sysbus register routines

SYNOPSIS

#include < pixar/chapdiag.h >

ChapSetSDR(chap, reg, value)
CHAP *chap; int reg; int value;

int ChapGetSDR(chap, reg)
CHAP *chap; int reg;

ChapSetVDR(chap, reg, value)
CHAP *chap; int reg; int value;

int *ChapVdregBase(chap)
CHAP *chap;

DESCRIPTION

These routines access the sysbus (sixteen-bit wide shared data registers) of the chap from the host. There are sixteen sysbus registers which may be used for communication with a running chap program. There are two modes of communication with the chap through these registers. Both modes require first opening the chap with *ChapOpen()* and *ChapMMan()* calls.

The first method, via *ChapSetSDR()* and *ChapGetSDR()* requires no host handshaking with the chap. The register number, *reg* must be one of the first fourteen registers (0 through 13). Any sixteen bit value may be placed into the register by *ChapSetSDR(chap,reg,value)*. Similarly, a value may be read by the host from the sysbus with a *ChapGetSDR(chap,reg)* call, which returns the value in the register. The chap reads and writes the sysbus with the *chas* symbol *sysbus<n>*, where *n* is the register number. This symbol may be used like any other scalar chap symbol. See *chas(1H)*.

The second method allows explicit synchronization between the host and chap. The chap sysbus has 256 virtual data registers (VDRs) which are accessed with the *ChapSetVDR()* command. The chap must respond to this write command by acknowledging with the *chas* statement "*sysrel=1*" before the bus times out and the host faults. There is just enough time for the chap to retrieve the value from the sysbus and acknowledge the VDR transfer. These registers are called VDRs because they are simulated on the sysbus by sysbus registers 14 and 15. Sysbus<14> will contain the *reg* and sysbus<15> will contain the *value*. Obviously, sysbus<14> and sysbus<15> are off-limits to all SDR transfers. A more detailed explanation of the mechanism for synchronization may be found in the *Chap Programming Tutorial*.

The call *ChapVdregBase()* will return the base of the VDR addresses, which can be used to read the virtual data registers. Indirecting through this pointer (reference only as an unsigned short or short) will return the value in sysbus<15>.

NOTES

The shared data register sysbus<13> is used by several Pixar application programs, including *Chad*(3H)* for synchronization during SDR transfers. This register should also be considered off-limits, unless its use in the application is carefully understood.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H), Chad*(3H), ChapMMan(3H), Chas(1H)

NAME

ChapReset – reset Chap and framebuffer state

SYNOPSIS

#include < pixar/pixar.h >

ChapReset(chap)
CHAP *chap;

DESCRIPTION

ChapReset flushes the symbol table and resets all the resource allocation maps manipulated by the kernel. This call should be used with care as it destroys all information about loaded Chap code.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H), ChapMMan(3H), chap(4)

NAME

ChapRun,
ChapRunAsync,
ChapCont,
ChapHalt,
ChapStep – Chap runtime control

SYNOPSIS

#include < pixar/pixar.h >

ChapRun(chap, pc)
CHAP *chap; u_short pc;

ChapRunAsync(chap, pc)
CHAP *chap; int pc;

u_short ChapCont(chap)
CHAP *chap;

u_short ChapHalt(chap)
CHAP *chap;

ChapStep(chap, byinst)
CHAP *chap; int byinst;

DESCRIPTION

These routines are available for controlling the execution of a Chap. *ChapRun* initializes the Chap runflags to 0xf, sets the stack-pointer to 0, and starts the Chap running at the specified address. *ChapRun* does not return until the Chap hits a breakpoint or the user types an interrupt on the keyboard. *ChapRun* returns the pc of the instruction where the Chap was stopped.

ChapRunAsync starts the Chap running as in *ChapRun*, but returns immediately.

ChapCont restarts the Chap at the place where it was last stopped. This routine does not return until a breakpoint is encountered, or the user types an interrupt on the keyboard. *ChapCont* returns the pc of the instruction at which the Chap was stopped.

ChapHalt halts the Chap. The current value of the program counter is returned. The machine is never halted in the middle of an instruction.

ChapStep single steps the Chap either one instruction, or one clock tick, depending on the value of *byinst*. *byinst* indicates the number of instructions to step at one time. When single-stepping one instruction, instructions which have the special bit on are executed, but not counted.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

BUGS

It is inadvisable to single step into the middle of a Chap instruction and then examine the internal state of a Chap; the library does not preserve enough internal state to do this correctly.

NAME

ChapReadSbus – read Chap Sbus
ChapReadSbusDev,
ChapWriteSbusDev – read/write Chap Sbus device

SYNOPSIS

```
#include < pixar/pixar.h >

ChapReadSbus(chap)
CHAP *chap;

ChapReadSbusDev(chap, dev, reg)
CHAP *chap; u_int dev, reg;

ChapWriteSbusDev(chap, dev, reg, v)
CHAP *chap; u_int dev, reg; u_short v;
```

DESCRIPTION

ChapReadSbus returns the contents of the Sbus by unloading the current instruction and Sbus contents from the shadow register.

ChapReadSbusDev and *ChapWriteSbusDev* read and write the contents of an Sbus device. If the device specified in *dev* is a base or index register, the *reg* parameter is used to identify the particular register. Scalar devices are defined in the file *<pixar/chap.h>*.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

ChapReadSpad, ChapWriteSpad	– read/write one location in Chap scratchpad
ChapLoadSpad, ChapDumpSpad	– read/write multiple locations in Chap scratchpad
ChapFillSpad	– fill Chap scratchpad memory with a single value
ChapAfillSpad	– fill each Chap scratchpad memory with its address

SYNOPSIS

#include < pixar/pixar.h >

u_short ChapReadSpad(chap, addr, mode, comp)
CHAP *chap; u_short addr; u_int mode, comp;

ChapWriteSpad(chap, addr, mode, comp, v)
CHAP *chap; u_short addr; u_int mode, comp; u_short v;

ChapLoadSpad(chap, addr, count, data)
CHAP *chap; u_short addr, count, data[];

ChapDumpSpad(chap, addr, count, data)
CHAP *chap; u_short addr, count, data[];

ChapFillSpad(chap, addr, count, fill)
CHAP *chap; u_short addr, count; u_short fill;

ChapAfillSpad(chap, addr, count)
CHAP *chap; u_short addr, count;

DESCRIPTION

ChapReadSpad and *ChapWriteSpad* read and write one 16-bit component in scratchpad memory using the addressing *mode* specified. The *comp* parameter identifies the component to be stored/retrieved. The *addr* parameter specifies the address in scratchpad. This address is shifted left two bits before passing to the Chap. For component, pixel, and broadcast read accesses, the address is placed in base register 15. For index mode read accesses, the address is supplied from the immediate field of the loaded instruction. All write accesses use base register 15 to supply the address. The contents of base register 15 is left unchanged after the operation.

ChapLoadSpad loads *count* words of *data* into scratchpad memory at the address specified. Data is loaded untessellated.

ChapDumpSpad retrieves *count* words of data from scratchpad, beginning at location *addr*.

ChapFillSpad fills count words of scratchpad memory with the value *fill*. This is useful, for example, in zeroing memory. *ChapAfillSpad* fills *count* memory locations starting at *addr* with each location's untessellated word address; this is used mostly for diagnostics.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

ChapReadPc,
ChapReadLc,
ChapReadRunflag – read stacked Chap register
ChapWritePc,
ChapWriteLc,
ChapWriteRunflag – write stacked Chap register
ChapReadRunflags,
ChapWriteRunflags – read/write Chap runflag state

SYNOPSIS

```
#include < pixar/pixar.h >

u_short ChapReadPc(chap)
CHAP *chap;

u_short ChapReadLc(chap)
CHAP *chap;

u_short ChapReadRunflag(chap)
CHAP *chap;

ChapWritePc(chap, pc)
CHAP *chap; u_short pc;

ChapWriteLc(chap, lc)
CHAP *chap; u_short lc;

ChapWriteRunflag(chap, rf)
CHAP *chap; u_short rf;

ChapReadRunflags(chap, rf)
CHAP *chap; Runflags *rf;

ChapWriteRunflags(chap, rf)
CHAP *chap; Runflags *rf;
```

DESCRIPTION

These routines read and write registers saved and restored on the Chap runtime stack. *ChapReadPc*, *ChapReadLc*, *ChapReadRunflag*, *ChapWritePc*, *ChapWriteLc*, and *ChapWriteRunflag* manipulate the current value of each register. *ChapReadRunflags* and *ChapWriteRunflags* affect the *previous*, *current*, and *next* runflags (runflag values are passed in the *Runflags* structure defined in *<pixar/chapdiag.h>*).

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

ChapSymLookup,
ChapSymEnter,
ChapSymX,
ChapXSym – Chap symbol table routines

SYNOPSIS

#include < pixar/pixar.h >

LoadSym **ChapSymLookup(chap, name, force)
CHAP *chap; char *name; int force;

ChapSymEnter(chap, hp, cp)
CHAP *chap; LoadSym **hp, *cp;

ChapSymX(chap, sp)
CHAP *chap; LoadSym *sp;

LoadSym *ChapXSym(chap, i)
CHAP *chap; int i;

DESCRIPTION

These routines deal with the symbol table maintained for each Chap. The symbol table is automatically opened at the time the Chap is opened with *ChapOpen*. The routine *ChapSymLookup* may be used to convert a symbol's name to a pointer to the appopriate symbol table entry (the *force* parameter should always be 0, it is needed internally for forcing the installation of local symbols), while the *ChapSymX* and *ChapXSym* routines are used to convert symbol table pointers to indices and back again (the latter actually being a macro defined in the include file).

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H), ChapSym(5)

NAME

ChapWaitForInterrupt – pause waiting for an interrupt from a Chap
ChapSetInterrupt,
ChapResetInterrupt – set/reset Chap interrupt handling

SYNOPSIS

#include < pixar/pixar.h >

ChapWaitForInterrupt(chap)
CHAP *chap;

ChapSetInterrupt(chap, sig)
CHAP *chap; int sig;

ChapResetInterrupt(chap)
CHAP *chap;

DESCRIPTION

ChapWaitForInterrupt causes the program to pause awaiting a *user interrupt* from the specified Chap or an interrupt from the keyboard. *ChapWaitForInterrupt* returns 1 if it was interrupted by a keyboard interrupt, 0 otherwise.

ChapSetInterrupt enables interrupt handling by performing a CHAPIOSSIG *ioctl*(2) call. The previous interrupt handling state is saved in a private variable and used by *ChapResetInterrupt* to reset the previous interrupt handling state. When signals are enabled with *ChapSetInterrupt,* the signal *sig* is sent to the calling process each time a Chap user interrupt is delivered to the host. It is the caller's responsibility to enable a signal handling routine with *signal*(3C) or *sigvec*(2).

Beware that interrupts from the Chap may, surreptitiously, be intercepted by the device driver if "memory management" is enabled, see *ChapMMan*(3H).

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

sigvec(2), signal(3C), ChapOpen(3H), ChapMMan(3H), chap(4)

BUGS

ChapWaitForInterrupt only handles one Chap.

NAME

ChapReadXbar,
ChapWriteXbar – read/write Chap crossbar state

SYNOPSIS

#include < pixar/chapdiag.h >

ChapReadXbar(chap, xbar)
CHAP *chap; XbarState *xbar;

ChapWriteXbar(chap, xbar)
CHAP *chap; XbarState *xbar;

DESCRIPTION

These routines read/write the state of the crossbar. The structure *XbarState* is defined in *<pixar/chapdiag.h>*.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

ChapOpen(3H)

NAME

DbOpen,
DbClose – setup a disk buffer device for use

SYNOPSIS

#include<pixar/pixar.h>

DB *DbOpen(device, size)
char *device; int size;

DbClose(dbp)
DB *dbp;

DESCRIPTION

DbOpen and *DbClose* support the disk buffer associated with each Dumi device. The disk buffer allows high speed transfer of data by having the Dumi generate the Sysbus addresses for each word of data transferrred. This can significantly increase the transfer rate between the host and the Chap (or a device, such as a disk resident on the host's bus.)

DbOpen opens the specified disk buffer and maps sufficient memory to contain a disk window of *size* bytes. This area in the process's address space is made available through the *db_bp* memory of the returned DB structure. In normal operation, this pointer is then used in a *read* operation or one of the special purpose *ioctl* calls described in *chap*(4).

DbClose closes the disk buffer and unmaps the associated memory.

FILES

/dev/db* disk buffer special files

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

dumi(4), chap(4)

DIAGNOSTICS

%s: Device in use. The device is currently in use by another person.

In addition, there are various messages about running out of memory and being unable to open files.

NAME

DumiOpen,
DumiClose – setup a Dumi device for diagnostic use

SYNOPSIS

#include<pixar/dumireg.h>

DUMI *DumiOpen(device)
char *device;

DumiClose(dumi)
DUMI *dumi;

DESCRIPTION

DumiOpen opens the specified Dumi *device*. The device is a special file as described in *dumi*(4). *DumiOpen* maps the Dumi diagnostic registers into the calling process's address space and returns a pointer to that area as *d_dumi* in the returned structure.

DumiClose closes the Dumi and unmaps the associated diagnostic registers.

In supporting the Dumi the caller is expected to use definitions found in the include file *<pixardev/dumireg.h>*. In particular, this file defines the structure of the bank of diagnostic registers provided by the Dumi and mapped into the process's address space by the library. *<dumireg.h>* is automatically included by *<pixar/pixar.h>*.

FILES

/dev/dumi* Dumi special files

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

dumi(4)

DIAGNOSTICS

%s: Device in use. The device is currently in use.

In addition, there are various messages about running out of memory or being unable to open files.

NAME

MctrlOpen,
MctrlClose – setup a memory controller device for diagnostic use

SYNOPSIS

#include<pixar/pixar.h> #include<pixardev/mctrlreg.h>

MCTRL *MctrlOpen(device)
char *device;

MctrlClose(mctrl)
MCTRL *mctrl;

DESCRIPTION

MctrlOpen opens the specified memory controller *device*. The device is a special file described in *mctrl*(4). *MctrlOpen* maps the memory controller diagnostic registers into the calling process's address space and returns a pointer to that area as *mc_mctrl* in the returned structure.

MctrlClose closes the memory controller and unmaps the associated diagnostic registers.

In supporting the Pixar memory controller, the caller is expected to use the definitions found in *<pixardev/mctrlreg.h>*. In particular, this file defines the structure of the diagnostic registers provided by the memory controller and mapped into the process's address space by the library. mctrlreg.h is automatically included by *<pixar/pixar.h>*.

FILES

/dev/mctrl* memory controller special files

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

mctrl(4)

DIAGNOSTICS

%s: Device in use. The device is currently in use.

In addition, there are various messages about running out of memory or being unable to open files.

NAME

VideoGetColormap,
VideoSetColormap – get/set the video controller color map

SYNOPSIS

 # include "/usr/pixar/include/pixar/video.h"

VideoGetColormap(video, r, g, b)
VIDEO
***video; u_short r[1024], g[1024], b[1024];**

VideoSetColormap(video, r, g, b)
VIDEO
***video; u_short r[1024], g[1024], b[1024];**

DESCRIPTION

VideoGetColormap fills supplied arrays with the contents of the colormap.

VideoSetColormap sets the colormap with the values specified in the given arrays.

The Pixar colormap provides the information which the video board uses to assign output intensities to pixel values. Each channel, red, green, and blue, has a separate map. Each map contains 1024 entries. This reflects the fact that only the top 10 bits of the pixel value are used as input to the colormaps. (2^10 = 1024). Colormap entries, on the other hand, are 12-bit fractions. This means that that full intensity for a channel is represented by the value 4096, which is the value 1 shifted left 12 bit positions. Hence, to convert a floating point value *c* (0≤c<1), representing an intensity, into a colormap entry *ic*,

ic = (u_short) (c * 4096);

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

videoopen(3H), video(4) /usr/pixar/tutorial/contour.c

NAME

VideoLoadCursor,
VideoDumpCursor – set/get video controller cursor
VideoCursorOn,
VideoCursorOff – turn video controller cursor on/off
VideoSetCursor – set the location of a video controller cursor

SYNOPSIS

VideoLoadCursor(video, n, cp)
VIDEO *video; int n; CURSOR *cp;

VideoDumpCursor(video, n, cp)
VIDEO *video; int n; CURSOR *cp;

VideoCursorOn(video, n)
VIDEO *video; int n;

VideoCursorOff(video)
VIDEO *video;

VideoSetCursor(video, x, y)
VIDEO *video; int x, y;

char *sp;

DESCRIPTION

Cursors are defined by the CURSOR structure defined in *<pixar/video.h>*. The Pixar video controller is capable of controlling four hardware cursors. These cursors may be up to 128 pixels on a side. The hardware cursor does not affect the frame buffer memory. The color of the active points in the hardware cursor is "super-white", so it can be distinguished from any color in the regular image.

VideoLoadCursor loads the video controller's *n*th cursor with the values specified in *cp*. If the cursor dimensions are less than 128 pixels on a side, the remaining space is zero filled (invisible). A cursor's location (X and Y coordinates) is translated from the hardware cursor location according to the X and Y "hot spot" (offsets) specified in the CURSOR structure. Each bit in the cursor's data representation corresponds to a pixel. The first bit of a cursor's data representation corresponds to the upper left hand corner of the cursor, as represented on the screen.

VideoDumpCursor retrieves the representation for the *n*th cursor and stores it in the data area associated with *cp*. The height and width of the cursor are taken from the cursor structure. Data is returned in a format suitable for use with *VideoLoadCursor*.

VideoCursorOn and *VideoCursorOff* turn the display of cursor *n* on or off, respectively. If the cursor number specified is negative, the current cursor is assumed.

VideoSetCursor sets the position of the current cursor. The *X* and *Y* positions are relative to the window associated with *v*. If a cursor hasn't yet been loaded, this call has no effect.

The current cursor location may be found in the *v_x*, and *v_y* members of the VIDEO structure.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

videoopen(3H), video(4)

NAME

VideoGetParam	– get video controller display state
VideoZoom	– set video controller zoom
VideoSetDisplay	– set video controller display state

SYNOPSIS

VideoGetParam(video)
VIDEO *video;

VideoZoom(video, zoom)
VIDEO *video;
int zoom;

VideoSetDisplay(video, base, width, height, x, y, mode)
VIDEO *video;
int base, width, height, x, y, mode;

DESCRIPTION

VideoGetParam updates the internal library state maintained in the VIDEO structure *video*. It need never be called unless the video controller registers are manipulated directly through the hardware registers.

VideoZoom sets the current magnification value for the video controller. The video controller implements magnification by pixel replication. Magnification values out of range, less than ZOOM_MIN (1) or greater than ZOOM_MAX (16), are clamped at the extremes. The window is automatically adjusted to maintain the cursor in the same relative position on the screen.

VideoSetDisplay sets the display window according to the parameters specified. The upper left hand corner of the window is set to be (*x*, *y*) pixels offset from the specified *base* (with scaling applied to take into account the current magnification). The width and height of the display, in tiles, are set according to *width* and *height*. The mode should be one of:

VMODE_RGB	display the red, green, and blue channels
VMODE_ALPHA	display the alpha channel
VMODE_BLANK	blank the screen
VMODE_RED	feed the red channel to all three color guns
VMODE_GREEN	feed the green channel to all three color guns
VMODE_BLUE	feed the blue channel to all three color guns

If the X or Y offsets are out of range, they are clipped according to the dimensions of *v*.

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

videoopen(3H), video(4)

NAME

VideoGetFormat,
VideoSetFormat – get/set video controller display format

SYNOPSIS

int VideoGetFormat(video)
VIDEO *video;

VideoSetFormat(video, format, freq)
VIDEO *video;
int format, freq;

DESCRIPTION

VideoGetFormat returns the current video controller *format*, a packed value containing information about the video configuration and clock oscillator frequency. Format information is unavailable on older video controllers (a –1 value is returned).

VideoSetFormat sets the current format select and clock oscillator frequency. Video formats are a property of a PROM on the video controller. Each PROM contains configuration information for up to four different video formats. The *format* parameter selects the set of parameters in the PROM to use. The *frequency* parameter selects which crystal, of four possible, to use. In normal operation, both parameters will correspond to the same video format. The following formats are currently defined:

VFORM_HIDEF	Hi-definition (1024x768)
VFORM_NTSC	NTSC (512x488)

while the following frequency definitions exist:

VFREQ_HIDEF	Hi-definition (1024x768)
VFREQ_NTSC	NTSC (512x488)

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

video(1), videoopen(3H), video(4)

NAME

VideoOpen,
VideoClose　　– open/close a video controller

SYNOPSIS

#include<pixar/pixar.h>

```
VIDEO *VideoOpen(device, width, height, shared)
char *device;
int width, height, shared;
```

```
VideoClose(video)
VIDEO *video;
```

DESCRIPTION

VideoOpen opens the video controller associated with the character special device *device*. If *shared* is non-zero, the device is opened with shared access, otherwise it is opened for exlusive use by the caller. A window onto the framestore is created with the specified dimensions (measured in pixels). The pointer returned by *VideoOpen* should be used in subsequent calls to routines described here.

VideoClose closes a previously opened video controller window.

The video controller support routines access the device through registers mapped into a process's address space.

In supporting the video controller definitions, the files *<pixar/video.h>* and *<pixardev/videoreg.h>* are used. These files are automatically included by *<pixar/pixar.h>*.

FILES

/dev/video*　　video controller special files

LIBRARY

/usr/pixar/host/lib/libpixar.a

SEE ALSO

video(4G)

DIAGNOSTICS

%s: Device in use. The device is currently in use.

%s: Device busy. The device is currently open for exclusive use by another processor owned by the caller.

In addition, there are various messages about running out of memory or being unable to open files.

NAME

intro – introduction to Chap library functions

DESCRIPTION

This section describes functions that may be found in various Chap libraries. The functions described in this section are grouped into the following subsections:

libpip This library contains common image processing functions. Included are routines to perform 1-dimensional and 2-dimensional convolution, box filtering, image arithmetic, histograms and find minimum and maximum image values.

libpx This library contains routines to geometrically transform images. There are procedures to change the size of an image using linear, quadratic or cubic interpolation. A procedure exists to decrease the size of an image. Other procedures can be used to rotate and warp images.

libpt This library contains procedures to transfer pixels to and from frame buffer memory and scrathpad memory. Included are procedures to clear memory, to copy from frame buffer to frame buffer, to transpose and reflect a frame buffer and to circularly shift pictures. Procedures also exist to perform component shuffling. Finally, this library also contains functions to perform single pixel operations. These include multiplying, channel arithmetic, clamping an image to the range 0 to 1. The procedures to do compositing also lie in the library.

libpm This library contains common arithmetic functions such as extended precision arithmetic, square roots, reciprocals, random number generation, and matrix multiplication.

libpG This library contains general purpose procedures. Currently, it contains routines to manipulate the register stack.

libcolor This library contains routines for performing color-space transformations.

All Chap library procedures are described in these manual pages using a C-like syntax. It's important, however, when calling these procedures to know how arguments are passed to them. This subject is discussed is more detail in *Introduction to Pixar Programming with Chad.*

Arguments are passed to functions by placing them in arithmetic, base and index registers. There are 16 base registers named b0-b15, 16 index registers named i0-i15, and 32 arithmetic registers named r0-r31. The base and index registers contains a single value whereas the arithmetic registers contains 4-values. If a single value is to be passed via an arithmetic register it is usually assigned to all 4 of its components.

Arguments are placed in registers in the order in which they appear in the command list. Integer values are passed using arithmetic registers and pointer values are passed using base registers. In a few instances pointers are passed in index registers. This is indicated by the keyword *index.*

As an example here is the declaration of SSClamp and how it would be called on the Chap.

```
SSClamp(src, dst, n)
pixel *src, *dst;
int n;
        b0 = src;
        b1 = dst;
        r0 = n;
        jsr SSClamp;
```

The same function would be called from *Chad* as follows:

```
ChadWrite( Chap,
        B0, srcaddr,
        B1, dstaddr,
        R0, n,
        NIX );
```

In the following man pages, these types are assumed:

pw	- pixel window
pixel	- RGBA tesselated pixel (address a multiple of 4)
component	- CCCC tesselated pixel
int	- any memory location

Furthermore, the name *src* is used for a source buffer and *dst* for a destination buffer. If there is more than one source or destination, they are numbered.

LIBRARY

libpm, libpm.3c, libpt, libpt.3c

SEE ALSO

intro(3H), intro(3), The Chad Tutorial.

NAME

libcolor – introduction to Pixar color-transformation library

DESCRIPTION

libcolor contains routines for performing color-space transformations on the channel values of pixels.

LIBRARY

/usr/pixar/chap/lib/libcolor.a

SEE ALSO

intro(3C), libpG(3C), libpip(3C), libpm(3C), libpt(3C), libpx(3C)

LIST OF FUNCTIONS

Name	*Page*	*Description*
rgb2XYZ	rgb2XYZ(3C)	– convert red-green-blue values to unnormalized CIE coordinates
rgb2xyY	rgb2xyY(3C)	– convert red-green-blue values to CIE coordinates
SSClamp	SSClamp(3C)	– clamp pixel values to the range [0, 1.0E].
XYZ2rgb	XYZ2rgb(3C)	– convert unnormalized CIE coordinates to red-green-blue values

NAME

rgb2xyY – convert red-green-blue values to CIE coordinates

SYNOPSIS

rgb2xyY(src, n, dst, MatrixPtr)
pixel src, dst; register n;
pixel MatrixPtr;

DESCRIPTION

rgb2xyY converts red-green-blue pixel values to CIE coordinates (x,y,Y). *src* holds the address of the input buffer, *dst* holds the address of the output buffer (which may be the same as the input buffer), *MatrixPtr* holds the address of the transformation matrix and *n* holds the number of pixels in the buffers. The transformation matrix is a 3x4 matrix (the fourth column isn't used) that transforms (R,G,B) to (X,Y,Z) space. This matrix is dependent on the phosphors that properly represent the (R,G,B) values. *rgb2xyY* normalizes the resultant (X,Y,Z) and writes *x*, *y* and *Y* in the red, green and blue channels, respectively, of the output buffer. *x* and *y* are unsigned 16-bit fractions. *Y* is an integer. The alpha channel of the output buffer is unchanged.

The following procedure computes the matrices needed to transform between (r, g, b)-space and CIE-space for a particular monitor. Let $[R, G, B]$ be the measured CIE coordinates of the red, green and blue phosphors of the monitor. These may be obtained from the manufacturer or, better yet, measured directly with a colormeter. Let W be the CIE-coordinates of the white to which the monitor is balanced. This typically is $6500^o K$, the coordinates of which are (.3127, .3290, .3583).

First compute the weighting vector:

$$(l_r, l_g, l_b) = W \begin{bmatrix} R \\ G \\ B \end{bmatrix}^{-1}$$

Given this, the transformation matrices are computed as follows:

$$(X, Y, Z) = (r, g, b) \begin{bmatrix} l_r R \\ l_g G \\ l_b B \end{bmatrix}$$

$$(r, g, b) = (X, Y, Z) \begin{bmatrix} l_r R \\ l_g G \\ l_b B \end{bmatrix}^{-1}$$

LIBRARY

libcolor.a

SEE ALSO

xyY2dens(3C), dens2rgb(3C)

NAME

rgb2XYZ – convert red-green-blue values to unnormalized CIE coordinates

SYNOPSIS

rgb2XYZ(src, n, dst, MatrixPtr)
pixel src, dst; register n;
pixel MatrixPtr;

DESCRIPTION

rgb2XYZ converts red-green-blue pixel values to unnormalized CIE coordinates (IX, Y, Z). *src* holds the address of the input buffer, *dst* holds the address of the output buffer (which may be the same as the input buffer), *MatrixPtr* holds the address of the transformation matrix and *n* holds the number of pixels in the buffers. The transformation matrix is a 3x4 matrix (the fourth column isn't used) that transforms (R, G, B) to (X, Y, Z) space. This matrix is dependent on the phosphors that properly represent the (R, G, B) values. The alpha channel of the output buffer is unchanged.

The following procedure computes the matrices needed to transform between (r, g, b)-space and CIE-space for a particular monitor. Let $[R, G, B]$ be the measured CIE coordinates of the red, green and blue phosphors of the monitor. These may be obtained from the manufacturer or, better yet, measured directly with a colormeter. Let W be the CIE-coordinates of the white to which the monitor is balanced. This typically is $6500^o K$, the coordinates of which are (.3127, .3290, .3583).

First compute the weighting vector:

$$(l_r, l_g, l_b) = W \begin{bmatrix} R \\ G \\ B \end{bmatrix}^{-1}$$

Given this, the transformation matrices are computed as follows:

$$(X, Y, Z) = (r, g, b) \begin{bmatrix} l_r R \\ l_g G \\ l_b B \end{bmatrix}$$

$$(r, g, b) = (X, Y, Z) \begin{bmatrix} l_r R \\ l_g G \\ l_b B \end{bmatrix}^{-1}$$

LIBRARY

libcolor.a

NAME

SSClamp – clamp pixel values to the range [0, 1.0E].

SYNOPSIS

SSClamp(src, dst, count)
pixel *src, *dst; register count;

DESCRIPTION

SSClamp clamps pixel values to the range [0, 1.0E]. *src* holds the address of the input buffer, *dst* holds the address of the output buffer (which may be the same as the input buffer), *n* holds the number of pixels in the buffers. Pixel values less than zero are set to zero and pixel values greater than 1.0E (2048) are set to one.

LIBRARY

libcolor.a

NAME

XYZ2rgb – convert unnormalized CIE coordinates to red-green-blue values

SYNOPSIS

XYZ2rgb(src, n, dst, MatrixPtr)
pixel src, dst, MatrixPtr; register n;

DESCRIPTION

XYZ2rgb converts unnormalized CIE coordinates (X, Y, Z) to red-green-blue pixel values. *src* holds the address of the input buffer, *dst* holds the address of the output buffer (which may be the same as the input buffer), *MatrixPtr* holds the address of the transformation matrix and *n* holds the number of pixels in the buffers. The transformation matrix is a 3x4 matrix (the fourth column isn't used) that transforms (X, Y, Z) to (R, G, B) space. This matrix is dependent on the phosphors that properly represent the (R, G, B) values. The alpha channel of the output buffer is unchanged.

The following procedure computes the matrices needed to transform between (r, g, b)-space and CIE-space for a particular monitor. Let $[R, G, B]$ be the measured CIE coordinates of the red, green and blue phosphors of the monitor. These may be obtained from the manufacturer or, better yet, measured directly with a colormeter. Let W be the CIE-coordinates of the white to which the monitor is balanced. This typically is $6500^o K$, the coordinates of which are (.3127, .3290, .3583).

First compute the weighting vector:

$$(l_r, l_g, l_b) = W \begin{bmatrix} R \\ G \\ B \end{bmatrix}^{-1}$$

Given this, the transformation matrices are computed as follows:

$$(X, Y, Z) = (r, g, b) \begin{bmatrix} l_r R \\ l_g G \\ l_b B \end{bmatrix}$$

$$(r, g, b) = (X, Y, Z) \begin{bmatrix} l_r R \\ l_g G \\ l_b B \end{bmatrix}^{-1}$$

LIBRARY

libcolor.a

NAME

libpG – introduction to Pixar library of general-purpose Chap routines.

DESCRIPTION

libpG contains general purpose procedures for the Channel Processor. Currently, it contains routines to manipulate the register stack.

LIBRARY

/usr/pixar/chap/lib/libpG.a

SEE ALSO

intro(3C), libcolor(3C), libpip(3C), libpm(3C), libpt(3C), libpx(3C)

LIST OF FUNCTIONS

Name	*Page*	*Description*
ALLOC	mman(3c)	– memory management support
MFREE	mman(3c)	– memory management support
MGET	mman(3c)	– memory management support
initstack	stack(3C)	– initialize the register stacking mechanism
pushb, popb	stack(3C)	– save and restore base registers from scratchpad stack
pushi, popi	stack(3C)	– save and restore index registers from scratchpad stack
pushr, popr	stack(3C)	– save and restore ALU registers from scratchpad stack
pushv, popv	stack(3C)	– save and restore all volatile registers from scratchpad stack

NAME

ALLOC,
MFREE,
MGET – memory management support

SYNOPSIS

#include<pixar/mman.h>

acc ALLOC(map, size, elabel)
acc MFREE(map, addr, size, elabel)
acc MGET(map, addr, size, elabel)

DESCRIPTION

These macros implement the Host-Chap memory management protocol described in *chap*(4). Each macro takes a *map* parameter indicating whether **SPAD** (Chap scratchpad memory) or **FB** (framebuffer memory) is to be allocated, and the appropriate parameters for the request. The *size* and *addr* parameters must be in the appropriate allocation units: *pixels* for **SPAD** and *tiles* for **FB**. The *elabel* parameter is a program label to which the code will jump in case of an error. *ALLOC* and *MGET* return an unscaled result in acc. *MFREE* overwrites **acc** with a success/failure indication.

Sysbus registers 9-13 are used when communicating with the host.

LIBRARY

libpg.a

SEE ALSO

chap(4)

DIAGNOSTICS

A −1 is returned in **acc** from the host when an error is encountered.

NAME

initstack	– initialize the register stacking mechanism
pushb, popb	– save and restore base registers from scratchpad stack
pushi, popi	– save and restore index registers from scratchpad stack
pushr, popr	– save and restore ALU registers from scratchpad stack
pushv, popv	– save and restore all volatile registers from scratchpad stack

SYNOPSIS

initstack()

pushb()

popb()

pushi()

popi()

pushr(n)
accumulator n;

popr()

pushv()

popv()

DESCRIPTION

Certain assumptions are made across procedure calls about the sanctity of registers. The first 25% of registers are *volatile*, the last two registers are *off-limits*, and the others are *sacred*. Thus, the volatile registers are acc, **r0-r7**, **b0-b3**, **i0-i3**. The sacred registers are **r8-r29**, **b4-b13**, **i4-i13**. Do not use **r30**, **r31**, **b14**, **b15**, **i14**, or **i15**. **b15** is used as a stack pointer. **i15** is used as a temporary index register by these stacking routines. **i14** is used as a structure pointer by the "variables" package. **b14** is used as a temporary field pointer by the variables package. **r31** is used as a temporary by these routines.

Routines may use any volatile register for local variables, or to receive arguments or return values. Routines **must** restore sacred registers upon exit. Thus, if a routine requires more registers than the volatile ones, some of the sacred registers should be stored upon entry and restored before exit. A scratchpad stack is maintained for storing registers.

initstack initializes the stack. This is a good routine to call on the first line of every program, because many library routines expect to use the stacking calls. *pushb* stores all the sacred base registers, and *popb* restores them. *pushi* stores all the sacred index registers, and *popi* restores them. *pushr* stores the *n* registers **r8**, **r9**, ..., r(n+7) and *popr* restores them. *pushv* stores all volatile registers (**r0**, ..., **r7**, **b0**, ..., **b3**, **i0**, ..., **i3**). *popv* restores them.

LIBRARY

libpG.a

DIAGNOSTICS

All routines commonly return 0. The push routines return a negative value in **acc** if there is no more space. The pop routines breakpoint if they are called in the wrong order. The push routines leave indicator flags on the stack, and the pop routines verify the flags before touching the stack. Continuing from the breakpoint in a pop routine will result in a negative exit from that routine.

NAME

libpip – introduction to the Pixar image processing library

DESCRIPTION

libpip is a library of Chap routines for performing common image processing tasks. Included are routines to perform 1-dimensional and 2-dimensional convolution, box filtering, image arithmetic, histograms and find minimum and maximum image values.

LIBRARY

/usr/pixar/chap/lib/libpip.a

SEE ALSO

intro(3C), libcolor(3C), libpG(3C), libpm(3C), libpt(3C), libpx(3C)

LIST OF FUNCTIONS

Name	*Page*	*Description*
PWAdd, PWSub, PWMul, PWDiv	PWArithmetic(3C)	– add, subtract, multiply and divide scratchpad arrays
PWBBox	PWBBox(3c)	– determine the smallest rectangle that surrounds an image
PWBoxFilterX, PWBoxFilterY	PWBoxFilter(3C)	– convolve pixel window buffer with 1-d pulse (box)
PWConvX, PWConvY	PWConv(3C)	– convolve pixel window with a 1-d kernel
PWCrc	PWCrc(3C)	– performs a Cyclic Redundency Check (CRC) on a pixel window
PWHistogram	PWHistogram(3C)	– compute the histogram of a pixel window
PWMap	PWMap(3C)	– map a single component through a color table
PWRange	PWRange(3C)	– find the minimum and maximum values in a pixel window
PWc33	PWc33(3C)	– convolve pixel window with 3x3 filter
PWc33s	PWc33s(3C)	– convolve pixel window with 3x3 separable filter
SSAdd, SSSub, SSMul, SSDiv	SSArithmetic(3C)	– add, subtract, multiply and divide scratchpad arrays
SSBoxFilter	SSBoxFilter(3C)	– convolve scratchpad buffer with 1-d pulse (box)
SSConv	SSConv(3C)	– convolve scratchpad buffer with 1-d kernel
SSCrc	SSCrc(3C)	– performs Cyclic Redundency Check on scanline in scratchpad
SSRange	SSRange(3C)	– find minimum and maximum values in a scratchpad array
c33	c33(3C)	– convolve scratchpad buffers with 3x3 kernel
c33s	c33s(3C)	– convolve scratchpad buffers with 3x3 separable kernel
c55s	c55s(3C)	– convolve scratchpad buffers with 5x5 separable kernel
cdhg	dhg(3C)	– accumulate histogram of input component array
idhg	dhg(3C)	– accumulate histogram of input integer array

NAME

c33 – convolve scratchpad buffers with 3x3 kernel

SYNOPSIS

c33(src, dst, kernel, n, spadbuffer)
pixel **src, **dst;
int kernel[3][3];
register n;
index **spadbuffer;

DESCRIPTION

c33 convolves 3 scanlines stored in scratchpad memory with a 3x3 kernel and stores the result in scrathpad memory.

This procedure is designed to be called once per scanline (see for example, PWc33) so it needs to maintain a ring of scratchpad buffers. The 3 input buffers each have *n*+2 pixels and the output buffer has *n* pixels. The two extra pixels in the input buffers are used as padding and are normally filled with 0s. The index register *spadbuffer* is a pointer to an two entry array of pixel pointers. The first entry is the oldest scratchpad array and the second is the second oldest. Each time it is called the ring of buffers is cycled so that the spadbuffer[0] = spadbuffer[1], spadbuffer[1] = *src and *src = spadbuffer[0]. These input pointers point to the first real pixel, not the padded pixel.

kernel points to 9 11-bit coefficients that comprise the 3x3 kernel matrix. The first three entries in the matrix correspond to the oldest scanline, the next three to the second oldest scanline and the final three entries to the *src* scanline.

TIMING

The inner loop takes 19 ticks per pixel.

LIBRARY

libpip.a

SEE ALSO

PWc33(3C), PWc33s(3C), c33s(3C), c55s(3C), PWConv(3C)

NAME

c33s – convolve scratchpad buffers with 3x3 separable kernel

SYNOPSIS

c33s(src, dst, kernel, n, spadbuffer)
pixel **src, **dst;
int kernel[6];
register n;
index **spadbuffer;

DESCRIPTION

c33s convolves 3 scanlines stored in scratchpad memory with a 3x3 separable kernel and stores the result in scrathpad memory.

This procedure is designed to be called once per scanline (see for example, PWc33) so it needs to maintain a ring of scratchpad buffers. The 3 input buffers each have *n+2* pixels and the output buffer has *n* pixels. The two extra pixels in the input buffers are used as padding and are normally filled with 0s. The index register *spadbuffer* is a pointer to a two entry array of pixel pointers. The first entry is the oldest scratchpad array and the second is the second oldest. Each time it is called the ring of buffers is cycled so that the spadbuffer[0] = spadbuffer[1], spadbuffer[1] = *src and *src = spadbuffer[0]. These input pointers point to the first real pixel, not the padded pixel.

kernel points to 6 11-bit coefficients that comprise the horizontal and vertical 3 entry kernel matrices. The first three entries in the matrix correspond to the horizontal filter, the next three to the vertical filter.

TIMING

The inner loop takes 14 ticks per pixel.

LIBRARY

libpip.a

SEE ALSO

c33s(3C), c55s(3C), PWConv(3C)

NAME

c55s – convolve scratchpad buffers with 5x5 separable kernel

SYNOPSIS

c55s(src, dst, kernel, n, spadbuffer)
pixel **src, **dst;
int kernel[10];
register n;
index **spadbuffer;

DESCRIPTION

c55s convolves 5 scanlines stored in scratchpad memory with a 5x5 separable kernel and stores the result in scrathpad memory.

This procedure is designed to be called once per scanline (see for example, PWc55), so it needs to maintain a ring of scratchpad buffers. The 5 input buffers each have *n+2* pixels and the output buffer has *n* pixels. The two extra pixels in the input buffers are used as padding and are normally filled with 0s. The index register *spadbuffer* is a pointer to a four entry array of pixel pointers. The first entry is the oldest scratchpad array, the second is the second oldest, and so forth. Each time it is called, the ring of buffers is cycled so that the *spadbuffer*[0] = *spadbuffer*[1], *spadbuffer*[1] = *spadbuffer*[2], *spadbuffer*[2] = *spadbuffer*[3], *spadbuffer*[3] = *src and **src* = *spadbuffer*[0]. These input pointers point to the first real pixel, not the padded pixel.

kernel points to 10 11-bit coefficients that comprise the horizontal and vertical 5 entry kernel matrices. The first five entries in the matrix correspond to the horizontal filter, the next five to the vertical filter.

LIBRARY

libpip.a

SEE ALSO

c33(3G), c33s(3G), PWConv(3G)

NAME

PWAdd,
PWSub,
PWMul,
PWDiv – add, subtract, multiply and divide scratchpad arrays

SYNOPSIS

PWAdd(dstpw, srcpw, spad1, spad2)
int *dstpw, *srcpw;
pixel *spad1, *spad2;

PWSub(dstpw, srcpw, spad1, spad2)
int *dstpw,*srcpw;
pixel *spad1, *spad2;

PWMul(dstpw, srcpw, spad1, spad2)
int *dstpw, *srcpw;
pixel *spad1, *spad2;

PWDiv(dstpw, srcpw, spad1, spad2)
int *dstpw,*srcpw;
pixel *spad1, *spad2;

DESCRIPTION

These procedures perform image arithmetic on pixel windows. Pixels values are treated as 11-bit fixed point quantities. Therefore, 2048*2048=2048 and 2048/2048=2048. *PWAdd* sets *dstpw += srcpw*; *PWSub* sets *dstpw −= srcpw*; *PWMul* sets *dstpw *= srcpw*; and *PWDiv* sets *dstpw /= srcpw*.

LIBRARY

libpip.a

SEE ALSO

SSArithmetic(3C)

NAME

PWBBox – determine the smallest rectangle that surrounds an image

SYNOPSIS

PWBBox (pw,background,spad)
ChadPW *pw;
pixel background;
pixel *spad;

DESCRIPTION

PWBBox finds the smallest rectangle (bounding box) that surrounds an image in the given pixel window. This can be used to make a smaller pixel window so that subsequent processing is performed on smaller images and takes less time.

The color *background* is used to determine whether the image data is present. If the color equals *background* image data is assumed not to be present; if, on the otherhand, the color is not equal to *background*, image data is assumed to be present. Normally, *background* is set to (0,0,0,0). *background* is passed by value.

The edges of the bounding rectangle are returned in (*xmin, xmax, ymin, ymax*) which are packed into **R0**. *xmin* is stored in the 0th, *xmax* in the 1st, *ymin* in the 2nd, and *ymax* in the 3rd processor. These coordinates are relative to the pixel window. If the entire pixel window contains valid image data, (*xmin,ymin*) will equal (0,0), and (*xmax,ymax*) will equal the width and height of the pixel window, respectively.

spad is a scanline buffer which must equal in size to the maximum of the width and heigth of the pixel window.

LIBRARY

libpip.a

SEE ALSO

PirlBBox (3H)

NAME

PWBoxFilterX,
PWBoxFilterY – convolve pixel window buffer with 1-d pulse (box)

SYNOPSIS

PWBoxFilterX(pw, spad1, spad2, width, highpass)
int *pw;
pixel *spad1, *spad2;
register width, highpass;

PWBoxFilterY(pw, spad1, spad2, width, highpass)
int *pw;
pixel *spad1, *spad2;
register width, highpass;

DESCRIPTION

PWBoxFilterX and *PWBoxFilterY* perform a one-dimensional convolution of the image stored in the pixel window. An image can be convolved with a 2-d pulse function by first convolving in *X*, and then convolving in *Y*.

The convolution is done by summing the center pixel plus the *width* pixels preceding and following it. Therefore, the total width of the pulse is *2*width+1* pixels. In the case of the *X* version, the pixels are summed horizontally; in the case of the *Y* version, vertically.

If the flag *highpass* is non-zero, the result of the convolution is subtracted from the value of the center pixel. This creates a highpass filter.

spad1 and *spad2* are temporary arrays used in scanline processing. They should be equal to at least the xsize of the pixel window.

LIBRARY

libpip.a

SEE ALSO

c33(3C), c55(3C), SSBoxFilter(3C), PWConv(3C)

NAME

PWConvX,
PWConvY – convolve pixel window with a 1-d kernel

SYNOPSIS

```
PWConvX( pw, kernel, spad1, spad2, kernelsize, offset )
int *pw;
pixel *kernel, *spad1, *spad2;
register kernelsize, offset;

PWConvY( pw, kernel, spad1, spad2, kernelsize, offset )
int *pw;
pixel *kernel, *spad1, *spad2;
register kernelsize, offset;
```

DESCRIPTION

PWConvX and *PWConvY* convolve the image in the pixel window with a 1-dimensional kernel. In the *X* version, the kernel extends *kernelsize* pixels along the x axis. In the *Y* version, the kernel is aligned with the y axis. The result is placed in the same pixel window.

Each element in the kernel array is a pixel. Thus, each component can be convolved with different kernel weights.

If the *kernel* values have 11 bits of fraction, the result will also have 11 bits of fraction.

The variable *offset* specifies which entry of the kernel matrix corresponds to the center pixel. If *offset* is 0, the last entry of the kernel is aligned with the pixel being output. If *offset* is width/2, the kernel is centered.

spad1 and *spad2* are two buffers used for scanline processing. *spad2* should be equal to the xsize of the window; *spad1* should be equal to the xsize of the window plus the *kernelsize–1*.

LIBRARY

libpip.a

SEE ALSO

c33(3C), c55(3C), SSConv(3C)

NAME

PWCrc – performs a Cyclic Redundency Check (CRC) on a pixel window

SYNOPSIS

PWCrc(pw,spad)
int *pw;
pixel *spad;

DESCRIPTION

SSCrc computes a CCITT standard CRC value for *pw*.

The spad buffer is equal to the width of the pixel window in pixels.

DIAGNOSTICS

The crc value is returned in **r0**.

LIBRARY

libpip.a

SEE ALSO

SSCrc(3C), PirlCrc(3H), crc(1H), PW(3C), TB(3C)

NAME

PWHistogram – compute the histogram of a pixel window

SYNOPSIS

PWHistogram(pw, histogram, size, component, spad)
int *pw;
pixel *histogram;
register size, component;
pixel *spad;

DESCRIPTION

PWHistogram computes a frequency histogram of one component within a pixel window (*pw*). *size* is the total number of bins in the histogram array. Each entry in the histogram array is a double precision integer (32-bits). Therefore, the histogram array itself occupies 2**size* words of scratchpad memory. *component* is an integer in the range 0-3 that specifies whether the red, green, blue or alpha component is tabulated.

Since pixels are signed numbers, 0 is stored at location *histogram*[*size*/4] and pixel 1 is stored at location *histogram*[*size*/4+*size*/2].

spad is a buffer of pixels used for scanline processing. It should be equal to the xsize of the pixel window.

LIBRARY

libpip.a

SEE ALSO

dhg(3C), SSRange(3C), PWRange(3C)

NAME

PWMap – map a single component through a color table to form a color image

SYNOPSIS

PWMap(pwsrc, dstpw, chan, map, spada, spadb)
PW *pwsrc, *dstpw;
pixel *map;
pixel *spada, *spadb;
int chan;

DESCRIPTION

PWMap creates a color image from a single channel image by using the value in the single channel image as in index into a color table. The given component, *chan*, which must be a number from 0 to 3 (representing red through alpha) of the image stored in the source pixel window, *pwsrc*, are mapped and written to the destination pixel window, *pwdst*.

The map table is actually four tables: TR, TG, TB, TA. The map table should point to the untesselated 4-way value TR[0], TG[0], TB[0], TA[0]. The component value from the source pixel is looked up in each table and then written to the *pwdst*. Note that if the pixels contain negative values (and pixel values may be negative), the table should extend not only forward in scratchpad memory from the map table, but also backwards.

LIBRARY

libpt.a

SEE ALSO

PirlMapComp(3H), PirlMap(3H), PirlMakeMap(3H), PirlCha(3H)

SS4Map(3C), PWMap(3C)

SSRtoRGBALUT(3C), SSGtoRGBALUT(3C), SSBtoRGBALUT(3C), SSAtoRGBALUT(3C)

NAME

PWRange – find the minimum and maximum values in a pixel window

SYNOPSIS

PWRange(pw, min, max, spad)
in *pw;
pixel *min, *max;
pixel *spad;

DESCRIPTION

PWRange finds the minimum and maximum values within a pixel window. These are returned in scratchpad locations pointed to by *min* and *max*. Unlike *SSRange*, *min* and *max* need not be initialized.

spad is a buffer of pixels used for scanline processing. It should be equal to the xsize of the pixel window.

LIBRARY

libpip.a

SEE ALSO

SSRange(3C), PWHistogram(3C)

NAME

PWc33 – convolve pixel window with 3x3 filter

SYNOPSIS

PWc33(pw, kernel, spad1, spad2, spad3, spad4)
int *pw;
int kernel[3][3];
pixel *spad1, *spad2, *spad3, *spad4;

DESCRIPTION

PWc33 convolves images stored in the framebuffer with a 3x3 filter and stores it in the same pixel window.

kernel points to 9 11-bit coefficients that comprise the 3x3 kernel matrix. The first three entries in the matrix correspond to the oldest scanline, the next three to the second oldest scanline and the final three entries to the *src* scanline.

spad1, *spad2*, and *spad3* are temporary arrays used in scanline processing. They should be equal to at least the xsize of the pixel window plus 2. *spad4* should have at least xsize pixels.

LIBRARY

libpip.a

SEE ALSO

PWc33s(3C), c33(3C), c55(3C), SSBoxFilter(3C), PWConv(3C)

NAME

PWc33s – convolve pixel window with 3x3 separable filter

SYNOPSIS

PWc33s(pw, kernel, spad1, spad2, spad3, spad4)
int *pw;
int kernel[6];
pixel *spad1, *spad2, *spad3, *spad4;

DESCRIPTION

PWc33s convolves images stored in the framebuffer with a 3x3 filter and stores it in the same pixel window.

kernel points to 6 11-bit coefficients that comprise the horizontal and vertical 3 entry kernel matrices. The first three entries in the matrix correspond to the horizontal filter, the next three to the vertical filter.

spad1, *spad2*, and *spad3* are temporary arrays used in scanline processing. They should be equal to at least the xsize of the pixel window plus 2. *spad4* should have at least xsize pixels.

LIBRARY

libpip.a

SEE ALSO

PWc33(3C), c33s(3C), c55(3C), SSBoxFilter(3C), PWConv(3C)

NAME

SSAdd,
SSSub,
SSMul,
SSDiv – add, subtract, multiply and divide scratchpad arrays

SYNOPSIS

SSAdd(a, b, c, n)
pixel *a, *b, *c;
register n;

SSSub(a, b, c, n)
pixel *a, *b, *c;
register n;

SSMul(a, b, c, n)
pixel *a, *b, *c;
register n;

SSDiv(a, b, c, n)
pixel *a, *b, *c;
register n

DESCRIPTION

These procedures perform vector arithmetic on arrays of pixels stored in scratchpad memory. Pixels values are treated as 11-bit fixed point quantities. Therefore, 2048*2048=2048 and 2048/2048=2048. The arrays are all *n* pixels long. *SSAdd* sets *c=a+b*; *SSSub* sets *c=a-b*; *SSMul* sets *c=a*b*; and *SSDiv* sets *c=a/b*.

TIMING

SSAdd and *SSSub* take 4 ticks per element. *SSMul* takes 5 ticks per element. *SSDiv* takes approximately 150 ticks (it calls *reciprocal*(3C) at each pixel) per element.

LIBRARY

libpip.a

SEE ALSO

PWArithmetic(3C)

NAME

SSBoxFilter – convolve scratchpad buffer with 1-d pulse (box)

SYNOPSIS

SSBoxFilter(src, dst, n, width, divwidth, highpass)
pixel *src, *dst;
register n, width, divwidth, highpass;

DESCRIPTION

SSBoxFilter performs a one-dimensional convolution between the *src* and a box filter or pulse function. The pulse function is centered at each pixel of the *src* array. The value of each pixel in the *dst* array is equal to the sum of the center pixel plus the *width* pixels preceding and following the center pixel in the scratchpad array. Therefore, the total width of the pulse is *2*width+1* pixels.

If the flag *highpass* is non-zero, the result of the convolution is subtracted from the value of the center pixel. This creates a highpass filter.

After the sum is computed, it is divided by *divwidth.* Normally, *divwidth* is equal to *2*width+1.*

TIMING

The inner loop takes 7 ticks per pixel.

LIBRARY

libpip.a

SEE ALSO

c33(3C), c55(3C), PWBoxFilter(3C), SSConv(3C)

NAME

SSConv, SSConv2, SSConv4 – convolve scratchpad buffer with 1-d kernel

SYNOPSIS

```
SSConv( src, kernel, dst, n, kernelsize )
pixel *src, *kernel, *dst;
register n, kernelsize;

SSConv2( src, dst, n, a, b )
pixel *src, *dst;
register n, a, b;

SSConv4( src, dst, kernel, n )
pixel *src, *dst, *kernel;
register n;
```

DESCRIPTION

SSConv performs a one-dimensional convolution between the *src* and the *kernel.* The *src* is (*n*+*kernelsize*+1) pixels long. The result is placed in *dst*, which must be *n* pixels long. The first result in the destination buffer results from first element of the kernel being aligned with the first entry in the source buffer. The last result in the destination buffer is calculated with the first element of the kernel being aligned with the *n*th element of the source.

Each element in the kernel array is a pixel. Thus, each component can be convolved with different kernel weights.

If both the *src* and *kernel* values have 11 bits of fraction, the *dst* values will also have 11 bits of fraction.

SSConv2 is an optimized verstion for convolutions of length 2. *dst[i]* is equal to *a**src[i] + b*src[i+1]*.

SSConv4 is an optimized version for convolutions of length 4. *dst[i]* is equal to *a*src[i] + b*src[i+1] + c*src[i+2] + d*src[i+3]* where *a*, *b*, *c* and *d* are the coefficients pointed to by *kernel.*

LIBRARY

libpip.a

SEE ALSO

c33(3C), c55(3C), PWConv(3C)

NAME

SSCrc – performs a Cyclic Redundancy Check (CRC) on a scanline in scratchpad

SYNOPSIS

SSCrc(src,crc_value, n)
pixel *src, register crc_value, n;

DESCRIPTION

SSCrc performs a CRC on *n* pixels in scratchpad using the passed crc_value as the initial seed. The crc_value is updated upon completion. The coeficients used for the check are those found in the CCITT standard. The recommended initital seed value is 0xffff.

LIBRARY

libpip.a

SEE ALSO

PWCrc(3C), PirlCrc(3H), crc(1H), PW(3C), TB(3C)

DIAGNOSTICS

The updated crc value is returned in **r0**.

NAME

SSRange – find the minimum and maximum values in a scratchpad array

SYNOPSIS

```
SSRange( src, n, min, max )
pixel *src;
register n;
pixel *min, *max;
```

DESCRIPTION

SSRange tests each component of each pixel in scratchpad memory whether it is less than *min* or greater than *max* and then updates *min* and *max*, respectively. To test for the actual minimum and maximum values stored in a scratchpad array, *min* and *max* should be initialized to 1.5E (3071) and –0.5E (–1024), respectively.

LIBRARY

libpip.a

SEE ALSO

PWRange(3C), PWHistogram(3C)

NAME

idhg – accumulate histogram of input integer array
cdhg – accumulate histogram of input component array

SYNOPSIS

```
idhg(icount,hcount,iptr,hptr)
register icount,hcount;
int *iptr;
double *hptr;

cdhg(icount,hcount,iptr,hptr)
register icount,hcount;
component *iptr;
double *hptr;
```

DESCRIPTION

idhg takes an integer array pointed to by *iptr* and accumulates into a histogam table pointed to by *hptr*. There are *icount* elements of the input array and *hcount* elements of the histogram array. The routine assumes that 11-bit values are being examined; they are rescaled to fit into the *hcount* accumulators of the histogram array. Note that *hptr* points to H[0]; negative indices will accumulate into histogram values previous to H[0]. Input values outside the range [–0.5E,1.5E) will increment values outside the histogram array.

cdhg is very similar, except that the input values are assumed to be from a single component of a scratchpad pixel array.

LIBRARY

libpG.a

NAME

libpm – introduction to Pixar arithmetic library

DESCRIPTION

libpm provides functions for performing common arithmetic operations using the Chap. These include extended precision arithmetic (*xp*(3C)), reciprocals (*reciprocal*(3C)), square roots (*sqrt*(3C)), reciprocals of square roots (*recsqrt*(3C)), random number generation (*rrand*(3C)), and matrix multiplication (*matrix*(3C)).

LIBRARY

/usr/pixar/chap/lib/libpm.a

SEE ALSO

intro(3C), libcolor(3C), libpG(3C), libpip(3C), libpt(3C), libpx(3C)
matrix(3C), reciprocal(3C), recsqrt(3C), rrand(3C), sqrt(3C), xp(3C)

LIST OF FUNCTIONS

Name	*Page*	*Description*
XPXIcopy22	xp(3C)	– double_pixel -> double_pixel
XPabs22	xp(3C)	– abs(double_pixel) -> double_pixel
XPadd222	xp(3C)	– double_pixel+double_pixel -> double_pixel
XPadd333	xp(3C)	– triple_pixel+triple_pixel -> triple_pixel
XPadd444	xp(3C)	– quad_pixel+quad_pixel -> quad_pixel
XPcopy22	xp(3C)	– double_pixel -> double_pixel
XPcopy33	xp(3C)	– triple_pixel -> triple_pixel
XPcopy44	xp(3C)	– quad_pixel -> quad_pixel
XPdiv224	xp(3C)	– double_pixel/double_pixel -> quad_pixel
XPmult112	xp(3C)	– pixel*pixel -> double_pixel
XPmult222	xp(3C)	– pixel*pixel -> double_pixel
XPmult224	xp(3C)	– double_pixel*double_pixel -> quad_pixel
XPneg22	xp(3C)	– -double_pixel -> double_pixel
XPread22	xp(3C)	– double_pixel -> register double_pixel
XPread33	xp(3C)	– triple_pixel -> register triple_pixel
XPread44	xp(3C)	– quad_pixel -> register quad_pixel
XPrec22	xp(3C)	– 1/double_pixel -> double_pixel
XPsub222	xp(3C)	– double_pixel-double_pixel -> double_pixel
XPsub333	xp(3C)	– triple_pixel-triple_pixel -> triple_pixel
XPsub444	xp(3C)	– quad_pixel-quad_pixel -> quad_pixel
XPwrite22	xp(3C)	– register double_pixel -> double_pixel
XPwrite33	xp(3C)	– register triple_pixel -> triple_pixel
XPwrite44	xp(3C)	– register quad_pixel -> quad_pixel
frecsqrt32l	recsqrt(3C)	– compute approximate 4-way reciprocal square root of unsigned 32-bit double-precision fraction
fsqrt32l	sqrt(3C)	– compute approximate 4-way square root of unsigned 32-bit double-precision fraction
isqrt32l	sqrt(3C)	– compute approximate 4-way square root of unsigned 32-bit integer
matmul16	matrix(3C)	– multiply two 4x4 matrices of 16-bit integers
matmul32	matrix(3C)	– multiply two 4x4 matrices of double-precision fractions
matvec32	matrix(3C)	– multiply a double-precision vector list by a double-precision matrix
reciprocal	reciprocal(3C)	– computes 2^16/n of four 16-bit numbers
reciprocal32	reciprocal(3C)	– computes 2^32/n of four non-negative 32-bit numbers
recsqrt16l	recsqrt(3C)	– compute approximate 4-way reciprocal square root of unsigned 16-bit fraction
recsqrt32l	recsqrt(3C)	– compute approximate 4-way reciprocal square root of unsigned 32-bit fraction
rrand	rrand(3C)	– produce 4 random numbers
sqrt16	sqrt(3C)	– compute exact 4-way square root of unsigned 16-bit fraction
sqrt16l	sqrt(3C)	– compute approximate 4-way square root of unsigned 16-bit fraction

NAME

matmul16	– multiply two 4x4 matrices of 16-bit integers
matmul32	– multiply two 4x4 matrices of double-precision fractions
matvec32	– multiply a double-precision vector list by a double-precision matrix

SYNOPSIS

matmul16(m0, m1, m2)
pixel *m0, *m1, *m2;

matmul32(m0_lsp, m0_msp, m1_lsp, m1_msp, m2_lsp, m2_msp)
pixel *m0_lsp, *m0_msp, *m1_lsp, *m1_msp, *m2_lsp, *m2_msp;

matvec32(length, vin_lsp, vin_msp, mat_lsp, mat_msp, vout_lsp, vout_msp)
register length;
pixel *vin_lsp, *vin_msp, *mat_lsp, *mat_msp, *vout_lsp, *vout_msp;

DESCRIPTION

All these routines operate on matrices stored in ''row-column'' order. Double-precision matrices and vector lists are stored in two parts, so that all the *lsp* data is in one array and all the *msp* data in another. This is done to take advantage of the 4-way parallelism of the Chap. Similar considerations cause vector lists to be ''row vectors''; matrices act on these vectors on the right.

matmul16 performs the matrix multiplication: *m2 = m0*m1*, where the matrices are composed of 16-bit integer entries. *m2* may be the same address as *m0* or *m1*.

matmul32 performs the matrix multiplication: *m2 = m0*m1*, where the matrices are composed of 32-bit double-precision fractions. As noted above, the low and high order parts of the entries are stored in separate arrays. *m2* may be the same address as either of the inputs.

matvec32 multiplies the input vector list *vin* on the right by the matrix *mat*, and puts the resulting vectors into the list *vout*. As noted above, the low and high order parts of the entries are stored in separate arrays. The number of vectors processed is a multiple of 4; it is defined to be the smallest multiple of 4 containing *length*. NOTE: When allocating vector lists, make sure they are aligned on 4-vector boundaries.

TIMING

matmul16:	~100 ticks
matmul32:	~550 ticks
matvec32:	~112 ticks/vector

LIBRARY

libpm.a

NAME

reciprocal – computes 2^16/n of four 16-bit numbers
reciprocal32 – computes 2^32/n of four non-negative 32-bit numbers

SYNOPSIS

reciprocal(n, recip)
register n, recip;

reciprocal32(n_lsp, n_msp, reciprocal_lsp, reciprocal_msp)
register n_lsp, n_msp, reciprocal_lsp, reciprocal_msp;

DESCRIPTION

reciprocal computes 2^16/n for four numbers in **r0** and returns the result in **r1**.

An 8-bit approximation is found via a lookup table. Newton's method is used to get 16 bits of precision timing. Execution ranges from 58-71 ticks.

reciprocal32 computes the 2^32/n for four numbers whose least significant parts are in **r0** and most significant parts are in **r1**. The resulting lsps are returned in **r2**, the msps in **r3**.

An 8 bit approximation is found via a lookup table. Newton's method is used to get 32 bits of precision.

TIMING

Execution ranges from 120-194 ticks:

all msps > 0:	122-142 ticks
all msps = 0:	120-142 ticks
some msps > 0, some = 0:	154-194 ticks

LIBRARY

libpm.a

BUGS

The routine only accepts non-negative inputs.

NAME

recsqrt16l – compute approximate 4-way reciprocal square root of unsigned 16-bit fraction
recsqrt32l – compute approximate 4-way reciprocal square root of unsigned 32-bit fraction
frecsqrt32l – compute approximate 4-way reciprocal square root of unsigned 32-bit double-precision fraction

SYNOPSIS

recsqrt16l(x,sx)
register x, sx;

recsqrt32l(x_lsp, x_msp, sx)
register x_lsp, x_msp, sx;

frecsqrt32l(x_lsp, x_msp, sx_lsp, sx_msp)
register x_lsp, x_msp, sx_lsp, sx_msp;

DESCRIPTION

recsqrt16l uses an 8-bit lookup table followed by linear interpolation to compute the approximate reciprocal square root of the unsigned 16-bit integer in *x*. It returns a result in *sx*, such that *x***sx***sx* = (1 << 16). The result is accurate to within 1 part in 10000.

recsqrt32l uses *sqrt16l* to compute an approximate square root of the unsigned 32-bit number contained in *x_lsp* and *x_msp*. It returns a 16-bit result in *sx*. The input should be thought of as a pure unsigned fraction of unity, and the output as a pure unsigned integer (or vice-versa).

frecsqrt32l uses *sqrt16l* to compute an approximate square root of the unsigned 32-bit double-precision fraction in *x_lsp* and *x_msp*. It returns a result in *sx_lsp* and *sx_msp*. Both input and output should be thought of as (16,16) unsigned fractions, that is, each contains 16 bits of integer and 16 fractional bits.

TIMING

The 16-bit routine runs in about 70 ticks; the 32-bit routines in about 100.

LIBRARY

libpm.a

NAME

rrand – produce 4 random numbers

SYNOPSIS

rrand()

saverrand(s56)
int *s56;

restorerrand(s56)
int *s56;

DESCRIPTION

rrand produces four random numbers in **acc**. Knuth's additive random number generator is used. Execution timing is 29 ticks.

saverrand expects a pointer, in *s56*, to 56 pixels of scratchpad space. The current state of the random number generator is stored there. *restorerrand* expects a pointer in **b0** to 56 pixels of scratchpad space into which the random number generator state has been stored. The random number generator is restored to that state.

LIBRARY

libpm.a

DIAGNOSTICS

saverrand returns −1 in **acc** if the random number generator has been corrupted. *restorerrand* returns −1 in **acc** if the 56 locations of scratchpad do not hold a valid random number generator state. Both routines return 0 upon success.

NAME

sqrt16	– compute exact 4-way square root of unsigned 16-bit fraction
sqrt16l	– compute approximate 4-way square root of unsigned 16-bit fraction
fsqrt32l	– compute approximate 4-way square root of unsigned 32-bit double-precision fraction
isqrt32l	– compute approximate 4-way square root of unsigned 32-bit integer

SYNOPSIS

sqrt16(x,sx)
register x, sx;

sqrt16l(x,sx)
register x, sx;

fsqrt32l(x_lsp, x_msp, sx_lsp, sx_msp)
register x_lsp, x_msp, sx_lsp, sx_msp;

isqrt32l(x_lsp, x_msp, sx)
register x_lsp, x_msp, sx;

DESCRIPTION

sqrt16 computes the exact square root of the unsigned 16-bit fraction in **x** and returns a result of the same type in **sx**. It generates one bit at a time by guessing.

sqrt16l uses an 8-bit lookup table followed by linear interpolation to compute the approximate square root of the unsigned 16-bit fraction in **x**. It returns a result of the same type in **r1**. The result is accurate to within 1 part in 10000.

fsqrt32l uses *sqrt16l* to compute an approximate square root of the unsigned 32-bit double-precision fraction in *x_lsp* and *x_msp*. It returns a result of the same type in *sx_lsp* and *sx_msp*.

isqrt32l uses *sqrt16l* to compute an approximate square root of the unsigned 32-bit integer contained in *x_lsp* and *x_msp*. It returns a 16-bit result in *sx*.

TIMING

Execution time of *sqrt16* is about 460 ticks; *sqrt16l*, about 70 ticks. The 32-bit routines require about 90 ticks.

LIBRARY

libpm.a

NAME

XPneg22	– -double_pixel -> double_pixel
XPabs22	– abs(double_pixel) -> double_pixel
XPrec22	– 1/double_pixel -> double_pixel
XPXIcopy22	– double_pixel -> double_pixel
XPcopy22	– double_pixel -> double_pixel
XPcopy33	– triple_pixel -> triple_pixel
XPcopy44	– quad_pixel -> quad_pixel
XPadd222	– double_pixel+double_pixel -> double_pixel
XPadd333	– triple_pixel+triple_pixel -> triple_pixel
XPadd444	– quad_pixel+quad_pixel -> quad_pixel
XPsub222	– double_pixel-double_pixel -> double_pixel
XPsub333	– triple_pixel-triple_pixel -> triple_pixel
XPsub444	– quad_pixel-quad_pixel -> quad_pixel
XPmult112	– pixel*pixel -> double_pixel
XPmult222	– pixel*pixel -> double_pixel
XPmult224	– double_pixel*double_pixel -> quad_pixel
XPdiv224	– double_pixel/double_pixel -> quad_pixel
XPread22	– double_pixel -> register double_pixel
XPread33	– triple_pixel -> register triple_pixel
XPread44	– quad_pixel -> register quad_pixel
XPwrite22	– register double_pixel -> double_pixel
XPwrite33	– register triple_pixel -> triple_pixel
XPwrite44	– register quad_pixel -> quad_pixel

SYNOPSIS

XPneg22(s,t)
base double_pixel *s,*t;

XPabs22(s,t)
base double_pixel *s,*t;

XPrec22(s,t)
base double_pixel *s,*t;

XPcopy22(s,t)
base double_pixel *s,*t;

XPXIcopy22(s,t)
base double_pixel *s;
base int *t;

XPcopy33(s,t)
base triple_pixel *s,*t;

XPcopy44(s,t)
base quad_pixel *s,*t;

XPadd222(s0,s1,t)
base double_pixel *s0,*s1,*t;

XPadd333(s0,s1,t)
base triple_pixel *s0,*s1,*t;

XPadd444(s0,s1,t)
base quad_pixel *s0,*s1,*t;

XPsub222(s0,s1,t)
base double_pixel *s0,*s1,*t;

XPsub333(s0,s1,t)
base triple_pixel *s0,*s1,*t;

XPsub444(s0,s1,t)
base quad_pixel *s0,*s1,*t;

XPmult112(s0,s1,t)
base int *s0,*s1;
base double_pixel *t;

XPmult222(s0,s1,t)
base double_pixel *s0,*s1;
base double_pixel *t;

XPmult224(s0,s1,t)
base double_pixel *s0,*s1;
base quad_pixel *t;

XPdiv224(s0,s1,t)
base double_pixel *s0,*s1;
base quad_pixel *t;

XPread22(s)
base double_pixel *s;

XPread33(s)
base triple_pixel *s;

XPread44(s)
base double_pixel *s;

XPwrite22(t)
base double_pixel *t;

XPwrite33(t)
base triple_pixel *t;

XPwrite44(t)
base quad_pixel *t;

DESCRIPTION

These routines do extended precision arithmetic on pixels (4-way vectors). *Double_pixels* are stored in scratchpad as [rL, gL, bL, aL, rH, gH, bH, aH], so that the four lower-16-bit quantities precede the four higher-16-bit quantities. Triples and quads are stored likewise. The storage must be aligned (like all pixels) to start at a 4-word boundary. Each set of four can be accessed in standard pixel mode addressing.

XPrec22 computes the reciprocal of *s* and writes the result to *t*.

XPXIcopy22 copies the *double_pixel* at *s* into an integer array *t*. [rL, gL, bL, aL, rH, gH, bH, aH] is copied to [rL, rH, gL, gH, bL, bH, aL, aH].

XPcopy22 copies the *double_pixel* at *s* into *t*.

XPcopy33 copies the *triple_pixel* at *s* into *t*.

XPcopy44 copies the *quad_pixel* at *s* into *t*.

XPadd222 adds the *double_pixels s0* and *s1* and puts the sum in *t*.

XPadd333 adds the *triple_pixels s0* and *s1* and puts the sum in *t*.

XPadd444 adds the *quad_pixels s0* and *s1* and puts the sum in *t*.

XPsub222 subtracts the double_pixel at *s1* from the double_pixel at *s0* and puts the difference in *t*.

XPsub333 subtracts the triple_pixel at *s1* from the triple_pixel at *s0* and puts the difference in *t*.

XPsub444 subtracts the quad_pixel at *s1* from the *quad_pixel* at *s0* and puts the difference in *t*.

XPmult112 multiplies the 16-bit quantity at *s0* by the 16-bit quantity at *s1* and puts the product at *t*. The number of fractional bits of the product will be the sum of the number of fractional bits of the inputs.

XPmult224 multiplies the *double_pixel* at *s0* by the *double_pixel* at *s1* and puts the *quad_pixel* product at *t*. The number of fractional bits of the product will be the sum of the number of fractional bits of the inputs.

XPmult222 multiplies the *double_pixel* at *s0* by the *double_pixel* at *s1* and puts only the middle 32 bits of the *quad_pixel* product at *t*. The number of fractional bits of the product will be 16 less than the sum of the number of fractional bits of the inputs.

XPdiv224 divides the *double_pixel* at *s0* by the *double_pixel* at *s1* to produce the *quad_pixel* at *t*. The number of fractional bits of the product will be 32 greater than the difference of the number of fractional bits of the numerator minus denominator.

XPread22 copies the *double_pixel* at *s* to registers (**r1**, **r0**).

XPread33 copies the *triple_pixel* at *s* to registers (**r2**, **r1**, **r0**).

XPread44 copies the *quad_pixel* at *s* to registers (**r3**, **r2**, **r1**, **r0**).

XPwrite22 copies the *double_pixel* stored in registers (**r1**, **r0**) to *t*.

XPwrite33 copies the *triple_pixel* stored in registers (**r2**, **r1**, **r0**) to *t*.

XPwrite44 copies the *quad_pixel* stored in registers (**r3**, **r2**, **r1**, **r0**) to *t*.

LIBRARY

libpm.a

DIAGNOSTICS

These routines never return errors.

These routines never destroy any registers, so there is no need to save the volatile registers before invoking the routines. Only **acc** and **i15** are used.

NAME

libpt – introduction to Chap Pixel Transfer Library

DESCRIPTION

libpt provides a variety of methods for transfering pixels between the framebuffer and the Chap's scratchpad memory (usually referred to as "scratchpad"). Most useful combinations of pixel transfers are accessible through these routines. Each of the routine names is keyed with a letter indicating the type of the source and the destination. The rest of the routine name indicates the direction and function of the routine.

Although the host interface deals in rectangular windows, the Chap code is scan-line oriented. Each of the host routines does multiple calls to the Chap code in order to copy/merge one window. All routines that access the framebuffer, either for reading or writing, use *pixel windows* to specify the area. More detail on windows may be found in "man windows", or in the *Chap Programming Tutorial.*

A prototypical transfer is *SFxCopy.* This routine copies one scanline from the scratchpad (designated by the "S" in the name *SFxCopy*) to the framebuffer memory (the "F"). This routine copies onto a horizontal scanline on the framebuffer (hence the "x"). Similarly, *FySCopy* copies a vertical scanline from the framebuffer to the scratchpad.

The simple *SF* and *FS* transfers are adequate for most operations. Occasionally, a more exotic transfer may significantly decrease computation time. This library has routines to copy scanlines in: reverse order, runlength encoding, individual channels, etc.

Library function names follow these conventions:

Fx	Framebuffer horizontal access
Fy	Framebuffer vertical access
S	Scratchpad (tesselated)
I	Integer array (copies all four channels - RGBA)
C	Channel array (copies only into appropriate channel)
R/G/B/A	Red, green, blue and alpha Channels
Backwards	copies elements in reverse order (last one to first position)

LIBRARY

/usr/pixar/chap/lib/libpt.a

SEE ALSO

intro(3C), libcolor(3C), libpG(3C), libpip(3C), libpm(3C), libpx(3C)

LIST OF FUNCTIONS

Name	*Page*	*Description*
AFxCopy	RGBAFCopy(3C)	– copy component from scratchpad to framebuffer in increasing x order
AFyCopy	RGBAFCopy(3C)	– copy component from scratchpad to framebuffer in increasing y order
AllocTB	TB(3C)	– initializes a tile block in frame buffer memory
BFxCopy	RGBAFCopy(3C)	– copy component from scratchpad to framebuffer in increasing x order
BFyCopy	RGBAFCopy(3C)	– copy component from scratchpad to framebuffer in increasing y order
CCCopy	CICopy(3C)	– copy scratchpad channel array to scratchpad channel array
CFxClear	CFCopy(3C)	– clear frame buffer in increasing x order to a component value
CFxCopy	CFCopy(3C)	– copy component from scratchpad to frame buffer in increasing x order
CFxCopyBackwards	CFCopy(3C)	– copy component from scratchpad to frame buffer

		in decreasing x order
CFyClear	CFCopy(3C)	– clear frame buffer in increasing x order to a component value
CFyCopy	CFCopy(3C)	– copy component from scratchpad to frame buffer in increasing y order
CFyCopyBackwardsy	CFCopy(3C)	– copy component from scratchpad to frame buffer in decreasing y order
CICopy	CICopy(3C)	– copy scratchpad channel array to scratchpad integer array
CRCopy	CRCopy(3C)	– copy scratchpad channel array to runlength array
ClosePV	PW(3C)	– close a pixel volume
ClosePW	PW(3C)	– close a pixel window
DeallocTB	TB(3C)	– deallocates a tile block
FAxCCopy	FCCopy(3C)	– copy scanline from alpha fb channel to spad channel array
FAxCCopyBackwards	FCCopy(3C)	– copy scanline from alpha fb channel backwards to spad channel array
FAxICopy	FICopy(3C)	– copy scanline from alpha fb channel to spad integer array
FAxICopyBackwards	FICopy(3C)	– copy scanline from alpha fb channel backwards to spad integer array
FAyCCopy	FCCopy(3C)	– copy vertical scanline from alpha fb channel to spad channel array
FAyCCopyBackwards	FCCopy(3C)	– copy vertical scanline backwards from alpha fb channel to spad channel array
FAyICopy	FICopy(3C)	– copy vertical scanline from alpha fb channel to spad integer array
FBxCCopy	FCCopy(3C)	– copy scanline from blue fb channel to spad channel array
FBxCCopyBackwards	FCCopy(3C)	– copy scanline from blue fb channel backwards to spad channel array
FBxICopy	FICopy(3C)	– copy scanline from blue fb channel to spad integer array
FBxICopyBackwards	FICopy(3C)	– copy scanline from blue fb channel backwards to spad integer array
FByCCopy	FCCopy(3C)	– copy vertical scanline from blue fb channel to spad channel array
FByCCopyBackwards	FCCopy(3C)	– copy vertical scanline backwards from blue fb channel to spad channel array
FByICopy	FICopy(3C)	– copy vertical scanline from blue fb channel to spad integer array
FByICopyBackwards	FICopy(3C)	– copy vertical scanline backwards from blue fb channel to spad integer array
FByICopyBackwards	FICopy(3C)	– copy vertical scanline backwards from blue fb channel to spad integer array
FFCopy	TBCopy(3C)	– copy a single tile between locations in frame buffer memory
FGxCCopy	FCCopy(3C)	– copy scanline from green fb channel to spad channel array
FGxCCopyBackwards	FCCopy(3C)	– copy scanline from green fb channel backwards to spad channel array
FGxICopy	FICopy(3C)	– copy scanline from green fb channel to spad integer array

FGxICopyBackwards	FICopy(3C)	– copy scanline from green fb channel backwards to spad integer array
FGyCCopy	FCCopy(3C)	– copy vertical scanline from green fb channel to spad channel array
FGyCCopyBackwards	FCCopy(3C)	– copy vertical scanline backwards from green fb channel to spad channel array
FGyICopy	FICopy(3C)	– copy vertical scanline from green fb channel to spad integer array
FGyICopyBackwards	FICopy(3C)	– copy vertical scanline backwards from green fb channel to spad integer array
FRxCCopy	FCCopy(3C)	– copy scanline from red fb channel to spad channel array
FRxCCopyBackwards	FCCopy(3C)	– copy scanline from red fb channel backwards to spad channel array
FRxICopy	FICopy(3C)	– copy scanline from red fb channel to spad integer array
FRxICopyBackwards	FICopy(3C)	– copy scanline from red fb channel backwards to spad integer array
FRyCCopy	FCCopy(3C)	– copy vertical scanline from red fb channel to spad channel array
FRyCCopyBackwards	FCCopy(3C)	– copy vertical scanline backwards from red fb channel to spad channel array
FRyICopy	FICopy(3C)	– copy vertical scanline from red fb channel to spad integer array
FRyICopyBackwards	FICopy(3C)	– copy vertical scanline backwards from red fb channel to spad integer array
FxACopy	FRGBACopy(3C)	– copy scanline from arbitrary fb channel to spad channel array
FxBCopy	FRGBACopy(3C)	– copy scanline from arbitrary fb channel to spad channel array
FxCCopy	FCCopy(3C)	– copy scanline from arbitrary fb channel to spad channel array
FxCCopyBackwards	FCCopy(3C)	– copy scanline from arbitrary fb channel backwards to spad channel array
FxGCopy	FRGBACopy(3C)	– copy scanline from arbitrary fb channel to spad channel array
FxICopy	FICopy(3C)	– copy scanline from arbitrary fb channel to spad integer array
FxICopyBackwards	FICopy(3C)	– copy scanline from arbitrary fb channel backwards to spad integer array
FxRCopy	FRGBACopy(3C)	– copy scanline from arbitrary fb channel to spad channel array
FxSCopy	FSCopy(3C)	– copy partial scanline from frame buffer to scratchpad
FxSCopyBackwards	FSCopy(3C)	– copy partial scanline from frame buffer backwards to scratchpad
FYCopy	FYCopy(3C)	– copy framebuffer to Yapbus
FyACopy	FRGBACopy(3C)	– copy vertical scanline from arbitrary fb channel to spad channel array
FyBCopy	FRGBACopy(3C)	– copy vertical scanline from arbitrary fb channel to spad channel array
FyCCopy	FCCopy(3C)	– copy vertical scanline from arbitrary fb channel to spad channel array

FyCCopyBackwards	FCCopy(3C)	– copy vertical scanline backwards from arbitrary fb channel to spad channel array
FyGCopy	FRGBACopy(3C)	– copy vertical scanline from arbitrary fb channel to spad channel array
FyICopy	FICopy(3C)	– copy vertical scanline from arbitrary fb channel to spad integer array
FyICopyBackwards	FICopy(3C)	– copy vertical scanline backwards from arbitrary fb channel to spad integer array
FyRCopy	FRGBACopy(3C)	– copy vertical scanline from arbitrary fb channel to spad channel array
FySCopy	FSCopy(3C)	– copy partial vertical scanline from frame buffer to scratchpad
FySCopyBackwards	FSCopy(3C)	– copy partial vertical scanline backwards from frame buffer to scratchpad
GFxCopy	RGBAFCopy(3C)	– copy component from scratchpad to frame buffer in increasing x order
GFyCopy	RGBAFCopy(3C)	– copy component from scratchpad to frame buffer in increasing y order
ICCopy	CICopy(3C)	– copy scratchpad integer array to scratchpad integer array
IFxClear	IFCopy(3C)	– clear partial scanline using scratchpad integer value
IFxCopy	IFCopy(3C)	– copy partial scanline from scratchpad integer array to frame buffer
IFxCopyBackwards	IFCopy(3C)	– copy partial scanline backwards from scratchpad integer array to frame buffer
IFyClear	IFCopy(3C)	– clear partial vertical scanline using scratchpad integer value
IFyCopy	IFCopy(3C)	– copy partial vertical scanline from scratchpad integer array to frame buffer
IFyCopyBackwards	IFCopy(3C)	– copy partial vertical scanline backwards from scratchpad integer array to frame buffer
IICopy	CICopy(3C)	– copy scratchpad integer array to scratchpad integer array
InitPV	PW(3C)	– initialize the pixel volume area
InitPW	PW(3C)	– initialize the pixel window area
InitTB	TB(3C)	– cleans out the tile block area
InqPV	PW(3C)	– provide information about an open pixel volume
InqPW	PW(3C)	– provide information about an open pixel window
InqTB	TB(3C)	– gather information on tile block
OpenPV	PW(3C)	– create a pixel volume for frame buffer access
OpenPW	PW(3C)	– create a pixel window for frame buffer access
PW4Map	PW4Map(3C)	– remap 4 components of a pixel window
PWAxb	PWAxb(3C)	– compute new pixel = *a**pixel+*b* for a pixel window.
PWCha	PWCha(3C)	– perform channel arithmetic on the pixels of a pixel window
PWCircularShift	PWShift(3C)	– circular shift pixel window contents in x and/or y
PWClamp	PWClamp(3C)	– clamp pixel between 0.0 (0) and 1.0 (0x800) for a pixel window
PWClear	PWClear(3C)	– clear pixel window to *color*
PWCopy	PWCopy(3C)	– copy the source pixel window to the destination pixel window

PWCopyGeneric	PWCopy(3C)	– PWCopy with user-specified axes, start and direction parameters
PWGeneralSwap	PWSwap(3C)	– general purpose implementation of *PWSwap*.
PWGeneric	PWGeneric(3C)	– call a spad-to-spad routine for each line of a pixel window
PWMerge	PWMerge(3C)	– Merge two pixel windows into a third
PWNot	PWNot(3C)	– subtract pixel value from 1.0 (0x800) for a pixel window
PWShift	PWShift(3C)	– shift pixel window contents in x and/or y
PWShuffle	PWShuffle(3C)	– shuffle components of each pixel for a pixel window.
PWSwap	PWSwap(3C)	– swap the source pixel window and the destination pixel window.
PWTranspose	PWTranspose(3C)	– transpose a pixel window around the diagonal axis.
RFxCopy	RGBAFCopy(3C)	– copy component from scratchpad to frame buffer in increasing x order
RFxCopy	RGBAFCopy(3C)	– copy component from scratchpad to frame buffer in increasing x order
RSCopy	RSCopy(3C)	– copy runlength array to scratchpad pixel array
ReAllocTB	TB(3C)	– reuses a previously allocated tile block
ReOpenPW	PW(3C)	– create a pixel window for frame buffer access
SCCopy, SCClear	SCCopy(3C)	– copy (clear) partial scanline in scratchpad
SFxClear	SFCopy(3C)	– clear partial scanline in frame buffer
SFxCopy	SFCopy(3C)	– copy partial scanline from scratchpad to frame buffer
SFxCopyRGBA, SFxCopyARGB, SFxCopyBARG, SFxCopyGBAR	SFCopy(3C)	– SFxCopy w/ channel rotation
SFxCopyBackwards	SFCopy(3C)	– copy partial scanline backwards from scratchpad to frame buffer
SFyClear	SFCopy(3C)	– clear partial vertical scanline in frame buffer
SFyCopy	SFCopy(3C)	– copy partial vertical scanline from scratchpad to frame buffer
SFyCopyRGBA, SFyCopyARGB, SFyCopyBARG, SFyCopyGBAR	SFCopy(3C)	– SFyCopy w/ channel rotation
SFyCopyBackwards	SFCopy(3C)	– copy partial vertical scanline backwards from scratchpad to frame buffer
SIClear	SICopy(3C)	– clear an integer array to a single channel value
SICopy	SICopy(3C)	– copy channels from a pixel array to integer array
SRCopy	RSCopy(3C)	– copy scratchpad pixel array to runlength array
SS4Map	SS4Map(3C)	– 4-way mapping of scratchpad values using a mapping table
SSAAAAtoAAAA	SSCopyComp(3C)	– Copy one channel to another channel
SSAAAAtoBBBB	SSCopyComp(3C)	– Copy one channel to another channel
SSAAAAtoGGGG	SSCopyComp(3C)	– Copy one channel to another channel
SSAAAAtoRRRR	SSCopyComp(3C)	– Copy one channel to another channel
SSAtoRGBA	SSCopyRGBA(3C)	– Copy one channel from scratchpad to 4 channels
SSAtoRGBALUT	SSCopyRGBA(3C)	– Copy one channel from scratchpad to 4 channels through a color table
SSBBBBtoAAAA	SSCopyComp(3C)	– Copy one channel to another channel
SSBBBBtoBBBB	SSCopyComp(3C)	– Copy one channel to another channel
SSBBBBtoGGGG	SSCopyComp(3C)	– Copy one channel to another channel
SSBBBBtoRRRR	SSCopyComp(3C)	– Copy one channel to another channel

SSBtoRGBA	SSCopyRGBA(3C)	– Copy one channel from scratchpad to 4 channels
SSBtoRGBALUT	SSCopyRGBA(3C)	– Copy one channel from scratchpad to 4 channels through a color table
SSCha	SSCha(3C)	– perform channel arithmetic on the pixels of a pixel window
SSComb	SSComb(3C)	– Combine two images
SSCompare	SSCompare(3C)	– compare scanline pixel buffers in scratchpad
SSCopy, SSClear	SSCopy(3C)	– copy (clear) partial scanline in scratchpad
SSGGGGtoAAAA	SSCopyComp(3C)	– Copy one channel to another channel
SSGGGGtoBBBB	SSCopyComp(3C)	– Copy one channel to another channel
SSGGGGtoGGGG	SSCopyComp(3C)	– Copy one channel to another channel
SSGGGGtoRRRR	SSCopyComp(3C)	– Copy one channel to another channel
SSGtoRGBA	SSCopyRGBA(3C)	– Copy one channel from scratchpad to 4 channels
SSGtoRGBALUT	SSCopyRGBA(3C)	– Copy one channel from scratchpad to 4 channels through a color table
SSMerge	SSMerge(3C)	– merge partial scanline from scratchpad over scratchpad
SSMergeAtop	SSMerge(3C)	– scratchpad to scratchpad merge using ATOP operator
SSMergeIn	SSMerge(3C)	– scratchpad to scratchpad merge using IN operator
SSMergeOut	SSMerge(3C)	– scratchpad to scratchpad merge using OUT operator
SSMergeOver	SSMerge(3C)	– scratchpad to scratchpad merge using OVER operator
SSMergeUnder	SSMerge(3C)	– scratchpad to scratchpad merge using UNDER operator
SSPaint	SSPaint(3C)	– paint partial scanline from scratchpad over scratchpad
SSPaintCopy	SSPaint(3C)	– merge pixels using spad matte
SSPaintOver	SSPaint(3C)	– SSPaint using OVER operator
SSRRRRtoAAAA	SSCopyComp(3C)	– Copy one channel to another channel
SSRRRRtoBBBB	SSCopyComp(3C)	– Copy one channel to another channel
SSRRRRtoGGGG	SSCopyComp(3C)	– Copy one channel to another channel
SSRRRRtoRRRR	SSCopyComp(3C)	– Copy one channel to another channel
SSRtoRGBA	SSCopyRGBA(3C)	– Copy one channel from scratchpad to 4 channels
SSRtoRGBALUT	SSCopyRGBA(3C)	– Copy one channel from scratchpad to 4 channels through a color table
SSShuffleBroadcast	SSShuffle(3C)	– SSCopy, broadcasting single component of src to dst
SSShuffleRot	SSShuffle(3C)	– SSCopy with specified channel rotation
SSShuffleXbar	SSShuffle(3C)	– SSCopy using specified roffset,goffset, boffset, aoffset transform
SSaxb	SSAxb(3C)	– Scale pixels using the formula A*x+B
SYCopy	SYCopy(3C)	– copy scratchpad buffer to Yapbus
SetMaskPW	PW(3C)	– set a pixel window channel mask
YFCopy	YFCopy(3C)	– copy Yapbus to framebuffer
YSCopy	YSCopy(3C)	– copy Yapbus to scratchpad
TBCopy	TBCopy(3C)	– copy between tile blocks in frame buffer memory

NAME

CFxCopy	– copy component from scratchpad to frame buffer in increasing x order
CFyCopy	– copy component from scratchpad to frame buffer in increasing y order
CFxCopyBackwards	– copy component from scratchpad to frame buffer in decreasing x order
CFyCopyBackwards	– copy component from scratchpad to frame buffer in decreasing y order
CFxClear	– clear frame buffer in increasing x order to a component value
CFyClear	– clear frame buffer in increasing y order to a component value

SYNOPSIS

CFxCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

CFyCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

CFxCopyBackwards(pw, src, n, x, y)
int *pw; pixel *src; register n, x, y[, z];

CFyCopyBackwards(pw, src, n, x, y)
int *pw; pixel *src; register n, x, y[, z];

CFxClear(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

CFyClear(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

DESCRIPTION

CFxCopy and *CFyCopy* copy *n* pixel components from scratchpad into frame buffer memory. Individual components of the scratchpad array (starting with *src* and incrementing by 4) are broadcast across all four channels of the frame buffer pixels. The pixel window (pixel volume) *pw* must have previously been opened with *OpenPW* (*OpenPV*); see *PW*(3C). *CFxCopy* copies the *n* pixels from a scratchpad buffer starting at *src* into [*x*, *y*] to [*x*+*n*–1, *y*] of the pixel window. *CFyCopy* copies the *n* pixels from a scratchpad buffer starting at *src* into into [*x*, *y*] to [*x*, *y*+*n*–1] of the pixel window. If the pixel window is a pixel volume, a *z* coordinate is used to specify the slice. Clipping is performed with regard to the *pw*, and the number actually read is returned in **acc**.

Only those channels indicated in the channel mask of the pixel window are written to the frame buffer; the other channels of frame buffer pixels are untouched.

If the pixel window is actually a pixel volume, the *z* value is used to indicate the appropriate window of the volume.

CFxCopyBackwards and *CFyCopyBackwards* reverse the order of the pixels copied.

CFxClear and *CFyClear* clear a scanline (horizontal and vertical, respectively) to the single channel component specified by the *src* pixel.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), IFCopy(3C), SFCopy(3C), FCCopy(3C)

DIAGNOSTICS

–1 is returned in **acc** if an invalid pixel window (pixel volume) is supplied. The number of pixels copied is returned in **acc**.

NAME

CCCopy – copy scratchpad channel array to scratchpad channel array
ICCopy – copy scratchpad integer array to scratchpad channel array
CICopy – copy scratchpad channel array to scratchpad integer array
IICopy – copy scratchpad integer array to scratchpad integer array

SYNOPSIS

CCCopy(src, dst, n)
ICCopy(src, dst, n)
CICopy(src, dst, n)
IICopy(src, dst, n)
pixel *src, *dst;
register n;

DESCRIPTION

These routines copy a single component of a pixel from scratchpad to scratchpad. The source and destin: tion are pixel pointers. A channel array pointer, (indicated by "C" in the title) is incremented by 4 for eac access. An integer array pointer (indicated by "I") is incremented by 1 for each access. If the pix pointer is a multiple of 4, the red component is copied; if the pixel pointer modulo 4 is 1, the green con ponent is copied, etc.

A *CICopy* effectively copies one component of *n* adjacent pixels into consecutive components of memor Likewise, an *ICCopy* copies adjacent components into component positions offset by the destination.

LIBRARY

libpt.a

SEE ALSO

CRCopy(3C), SIClear(3C), SICopy(3C), CFCopy(3C)

DIAGNOSTICS

The number of pixels copied is returned in **acc.**

NAME

CRCopy – copy scratchpad channel array to runlength array

SYNOPSIS

CRCopy(src, dst, n, threshold)
pixel *src, *dst; register n, threshold;

DESCRIPTION

CRCopy copies *n* pixels from a scratchpad array *src* to a runlength array *dst*. A runlength array is a one-bit, single-channel, run-length-encoded array, with a count *n* followed by *n* run lengths. The initial run-length is the number of initial components <= *threshold;* the next is the number of subsequent components > *threshold.* When *threshold* is 0, this partitions arrays into zero and non-zero runs.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), RSCopy(3C).

NAME

FxCCopy	– copy scanline from arbitrary fb channel to spad channel array
FyCCopy	– copy vertical scanline from arbitrary fb channel to spad channel array
FxCCopyBackwards	– copy scanline from arbitrary fb channel backwards to spad channel array
FyCCopyBackwards	– copy vertical scanline backwards from arbitrary fb channel to spad channel a
FRxCCopy	– copy scanline from red fb channel to spad channel array
FRyCCopy	– copy vertical scanline from red fb channel to spad channel array
FRxCCopyBackwards	– copy scanline from red fb channel backwards to spad channel array
FRyCCopyBackwards	– copy vertical scanline backwards from red fb channel to spad channel array
FGxCCopy	– copy scanline from green fb channel to spad channel array
FGyCCopy	– copy vertical scanline from green fb channel to spad channel array
FGxCCopyBackwards	– copy scanline from green fb channel backwards to spad channel array
FGyCCopyBackwards	– copy vertical scanline backwards from green fb channel to spad channel arra
FBxCCopy	– copy scanline from blue fb channel to spad channel array
FByCCopy	– copy vertical scanline from blue fb channel to spad channel array
FBxCCopyBackwards	– copy scanline from blue fb channel backwards to spad channel array
FByCCopyBackwards	– copy vertical scanline backwards from blue fb channel to spad channel array
FAxCCopy	– copy scanline from alpha fb channel to spad channel array
FAyCCopy	– copy vertical scanline from alpha fb channel to spad channel array
FAxCCopyBackwards	– copy scanline from alpha fb channel backwards to spad channel array
FAyCCopyBackwards	– copy vertical scanline backwards from alpha fb channel to spad channel arra

SYNOPSIS

FxCCopy(pw, dst, n, x, y[, z], channelnumber)
int *pw; int *dst; register n, x, y[, z]; index channelnumber;

FyCCopy(pw, dst, n, x, y[, z], channelnumber)
int *pw; int *dst; register n, x, y[, z]; index channelnumber;

FxCCopyBackwards(pw, dst, n, x, y[, z], channelnumber)
int *pw; int *dst; register n, x, y[, z]; index channelnumber;

FyCCopyBackwards(pw, dst, n, x, y[, z], channelnumber)
int *pw; int *dst; register n, x, y[, z]; index channelnumber;

F[R|G|B|A]xCCopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

F[R|G|B|A]yCCopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

F[R|G|B|A]xCCopyBackwards(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

F[R|G|B|A]yCCopyBackwards(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

DESCRIPTION

FxCCopy copies the *n* components from channel *channelnumber* (Red is 0; Green is 1; Blue is 2; Alpha 3) of pixels [*x*,*y*] to [*x*+*n*−1,*y*] of pixel window *pw* into a scratchpad channel array starting at *dst* and incre menting by 4. A channel array is equivalent to a pixel array, except that only one channel is expected to b processed; consecutive channel components are 4 words apart. *FyCCopy* copies the *n* components fro channel *channelnumber* of pixels [*x*,*y*] to [*x*,*y*+*n*−1] of that pixel window into a scratchpad channel arra starting at *dst*. The *FxCCopyBackwards* and *FyCCopyBackwards* routines reverse the order of the pixe copied. The *FRxC, FRyC, FGxC, ..., FAyC* routines copy a specific channel.

The pixel window *pw* must have been previously opened with a call to *OpenPW*, *PW*(3C). If the pix window is a pixel volume, a *z* coordinate is used in every routine to specify the slice. Clipping is pe formed with regard to the *pw*, and the number of pixels actually read is returned in **acc**.

Only those channels indicated in the channel mask of the pixel window can be written to scratchpad.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), FSCopy(3C), FICopy(3C), CFCopy(3C)

DIAGNOSTICS

All routines return −1 in **acc** if an invalid pixel window is supplied.

NAME

FxICopy	– copy scanline from arbitrary fb channel to spad integer array
FyICopy	– copy vertical scanline from arbitrary fb channel to spad integer array
FxICopyBackwards	– copy scanline from arbitrary fb channel backwards to spad integer array
FyICopyBackwards	– copy vertical scanline backwards from arbitrary fb channel to spad integer array
FRxICopy	– copy scanline from red fb channel to spad integer array
FRyICopy	– copy vertical scanline from red fb channel to spad integer array
FRxICopyBackwards	– copy scanline from red fb channel backwards to spad integer array
FRyICopyBackwards	– copy vertical scanline backwards from red fb channel to spad integer array
FGxICopy	– copy scanline from green fb channel to spad integer array
FGyICopy	– copy vertical scanline from green fb channel to spad integer array
FGxICopyBackwards	– copy scanline from green fb channel backwards to spad integer array
FGyICopyBackwards	– copy vertical scanline backwards from green fb channel to spad integer array
FBxICopy	– copy scanline from blue fb channel to spad integer array
FByICopy	– copy vertical scanline from blue fb channel to spad integer array
FBxICopyBackwards	– copy scanline from blue fb channel backwards to spad integer array
FByICopyBackwards	– copy vertical scanline backwards from blue fb channel to spad integer array
FAxICopy	– copy scanline from alpha fb channel to spad integer array
FAyICopy	– copy vertical scanline from alpha fb channel to spad integer array
FAxICopyBackwards	– copy scanline from alpha fb channel backwards to spad integer array
FAyICopyBackwards	– copy vertical scanline backwards from alpha fb channel to spad integer array

SYNOPSIS

FxICopy(pw, dst, n, x, y[, z], channelnumber)
int *pw; int *dst; register n, x, y[, z]; index channelnumber;

FyICopy(pw, dst, n, x, y[, z], channelnumber)
int *pw; int *dst; register n, x, y[, z]; index channelnumber;

FxICopyBackwards(pw, dst, n, x, y[, z], channelnumber)
int *pw; int *dst; register n, x, y[, z]; index channelnumber;

FyICopyBackwards(pw, dst, n, x, y[, z], channelnumber)
int *pw; int *dst; register n, x, y[, z]; index channelnumber;

F[R|G|B|A]xICopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

F[R|G|B|A]yICopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

F[R|G|B|A]xICopyBackwards(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

F[R|G|B|A]yICopyBackwards(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

DESCRIPTION

FxICopy copies the *n* components from channel *channelnumber* of pixels [*x*,*y*] to [*x*+*n*−1,*y*] of pixel window *pw* into a scratchpad integer array starting at *dst* and incrementing by 1. An integer array is like a *component* array, except that it takes one fourth the storage; consecutive integer components are consecutive words in scratchpad. *FyICopy* copies the *n* components from channel *channelnumber* of pixels [*x*,*y*] to [*x*,*y*+*n*−1] of that pixel window into a scratchpad integer array starting at *dst*. The *FxICopyBackwards* and *FyICopyBackwards* routines reverse the order of the pixels copied. The *FRxI*, *FRyI*, *FGxI*, ..., *FAyI* Routines copy a specific channel.

The pixel window *pw* must have been previously opened with a call to *OpenPW*, *PW*(3C). If the pixel window is a pixel volume, a *z* coordinate is used in every routine to specify the slice. Clipping is performed with regard to the *pw*, and the number of pixels actually read is returned in **acc**.

Only those channels indicated in the channel mask of the pixel window can be written to scratchpad.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), FSCopy(3C), FICopy(3C), CFCopy(3C)

DIAGNOSTICS

All routines return −1 in **acc** if an invalid pixel window is supplied.

NAME

FxRCopy – copy scanline from arbitrary fb channel to spad channel array
FxGCopy – copy scanline from arbitrary fb channel to spad channel array
FxBCopy – copy scanline from arbitrary fb channel to spad channel array
FxACopy – copy scanline from arbitrary fb channel to spad channel array
FyRCopy – copy vertical scanline from arbitrary fb channel to spad channel array
FyGCopy – copy vertical scanline from arbitrary fb channel to spad channel array
FyBCopy – copy vertical scanline from arbitrary fb channel to spad channel array
FyACopy – copy vertical scanline from arbitrary fb channel to spad channel array

SYNOPSIS

```
FxRCopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

FxGCopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

FxBCopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

FxACopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

FyRCopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

FyGCopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

FyBCopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];

FyACopy(pw, dst, n, x, y[, z])
int *pw; int *dst; register n, x, y[, z];
```

DESCRIPTION

These procedures copy a pixel from the framebuffer into 4 consecutive component values in scratchpad.

FxRCopy, FxGCopy, FxBCopy, and *FxACopy* copy the *n* pixels [*x*,*y*] to [*x*+*n*–1,*y*] of that pixel window into a scratchpad channel array starting at *dst*. Each pixel is copied to 4 consecutive pixels in the scratchpad array. Each procedure copies to a different component.

FyRCopy, FyGCopy, FyBCopy, and *FyACopy* copy the *n* pixels [*x*,*y*] to [*x*,*y*+*n*–1] of that pixel window into a scratchpad channel array starting at *dst*. Each pixel is copied to 4 consecutive pixels in the scratchpad array. Each procedure copies to a different component.

The pixel window *pw* must have been previously opened with a call to *OpenPW*, *PW*(3C). If the pixel window is a pixel volume, a *z* coordinate is used in every routine to specify the slice. Clipping is performed with regard to the *pw*, and the number of pixels actually read is returned in **acc**.

Only those channels indicated in the channel mask of the pixel window can be written to scratchpad.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), FSCopy(3C), FICopy(3C), CFCopy(3C)

DIAGNOSTICS

All routines return –1 in acc if an invalid pixel window is supplied.

Only those channels indicated in the channel mask of the pixel window can be written to scratchpad.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), FSCopy(3C), FCCopy(3C), IFCopy(3C)

DIAGNOSTICS

All routines return −1 in **acc** if an invalid pixel window is supplied. The number of pixels copied is returned in **acc**.

NAME

FxSCopy	– copy partial scanline from frame buffer to scratchpad
FySCopy	– copy partial vertical scanline from frame buffer to scratchpad
FxSCopyBackwards	– copy partial scanline from frame buffer backwards to scratchpad
FySCopyBackwards	– copy partial vertical scanline backwards from frame buffer to scratchpad

SYNOPSIS

FxSCopy(pw, dst, n, x, y[, z])
int *pw; pixel *dst; register n, x, y[, z];

FySCopy(pw, dst, n, x, y[, z])
int *pw; pixel *dst; register n, x, y[, z];

FxSCopyBackwards(pw, dst, n, x, y[, z])
int *pw; pixel *dst; register n, x, y[, z];

FySCopyBackwards(pw, dst, n, x, y[, z])
int *pw; pixel *dst; register n, x, y[, z];

DESCRIPTION

These routines copy *n* pixels from frame buffer to scratchpad. The pixel window *pw* must have been previously opened with a call to *OpenPW*, *PW*(3C). *FxSCopy* copies the *n* pixels from [*x*,*y*] to [*x*+*n*−1,*y*] of that pixel window into a scratchpad buffer starting at *dst*. *FySCopy* copies the *n* pixels from [*x*,*y*] to [*x*,*y*+*n* of that pixel window into a scratchpad buffer starting at *dst*. Clipping is performed with regard to the *pw*, and the number of pixels actually read is returned in **acc**.

The *FxSCopyBackwards* and *FySCopyBackwards* routines reverse the order of the pixels copied.

If the pixel window is actually a pixel volume, the *z* value is used to indicate the appropriate window of the volume.

Only those channels indicated in the channel mask of the pixel window are written to scratchpad. The other channels of scratchpad pixels are left untouched.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C)

DIAGNOSTICS

All routines return −1 in **acc** if an invalid pixel window is supplied.

NAME

FYCopy

SYNOPSIS

FYCopy(tile, chanmask, dest_addr)
int *tile; register chanmask, dest_addr

DESCRIPTION

This routine copies pixels from the framebuffer to the yapbus. Transfers are tile-based. *tile (b0)* holds the number of the tile to be sent out. *chanmask (r0)* holds the channel mask to be used when reading data from the framebuffer. *dest_addr (r1)* has the receiver address that data is to be sent to, and must be a number from 1 through 15.

Pixels are transmitted at a rate of two pixels per CPU tick. The transmitter priority is set to be the same as the dest_addr. Garbage is sent on the yapbus channels masked off in the channel mask. The same or a more restrictive mask must be used at the receiver.

LIBRARY

libpt.a

SEE ALSO

YFCopy.3c

DIAGNOSTICS

A non-zero value is returned in **acc** if a transmission problem occurs.

NAME

IFxCopy – copy partial scanline from scratchpad integer array to frame buffer
IFyCopy – copy partial vertical scanline from scratchpad integer array to frame buffer
IFxCopyBackwards – copy partial scanline backwards from scratchpad integer array to frame buffer
IFyCopyBackwards – copy partial vertical scanline backwards from scratchpad integer array to frame buffer
IFxClear – clear partial scanline using scratchpad integer value
IFyClear – clear partial vertical scanline using scratchpad integer value

SYNOPSIS

IFxCopy(pw, n, n, x, y[, z])
int *pw; int *n; register n, x, y[, z];

IFyCopy(pw, n, n, x, y[, z])
int *pw; int *n; register n, x, y[, z];

IFxCopyBackwards(pw, n, n, x, y)
int *pw; int *n; register n, x, y[, z];

IFyCopyBackwards(pw, n, n, x, y)
int *pw; int *n; register n, x, y[, z];

IFxClear(pw, n, n, x, y[, z])
int *pw; int *n; register n, x, y[, z];

IFyClear(pw, n, n, x, y[, z])
int *pw; int *n; register n, x, y[, z];

DESCRIPTION

IFxCopy and *IFyCopy* copy ***n*** pixel components from scratchpad into frame buffer. Individual components of the scratchpad array (starting with *n* and incrementing by 1) are broadcast across all four channels of the frame buffer pixels. The pixel window (pixel volume) *pw* must have previously been opened with *OpenPW* (*OpenPV*); see *PW*(3C). *IFxCopy* copies the *n* components from a scratchpad buffer starting at *n* into [*x*, *y*] to [*x*+*n*–1, *y*] of the pixel window. *IFyCopy* copies the *n* components from a scratchpad buffer starting at *n* into [*x*, *y*] to [*x*, *y*+*n*–1] of the pixel window. Clipping is performed with regard to the *pw*, and the number actually read is returned in **acc**.

IFxCopyBackwards and *IFyCopyBackwards* reverse the order of the components copied.

If the pixel window is actually a pixel volume, the *z* value is used to indicate the appropriate window of the volume.

IFxClear and *IFyClear* clear the destination framebuffer scanline (horizontal and vertical, respectively) to the value specified by the single scratchpad integer specified by the source.

Only those channels indicated in the channel mask of the pixel window are written to the frame buffer; the other channels of frame buffer pixels are untouched.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), FICopy(3C), SFCopy(3C), CFCopy(3C)

DIAGNOSTICS

–1 is returned in acc if an invalid pixel window (pixel volume) is supplied.

NAME

OpenPW	– create a pixel window for frame buffer access
OpenPV	– create a pixel volume for frame buffer access
ReOpenPW	– create a pixel window for frame buffer access
InqPW	– provide information about an open pixel window
InqPV	– provide information about an open pixel volume
ClosePW	– close a pixel window
ClosePV	– close a pixel volume
InitPW	– initialize the pixel window area
InitPV	– initialize the pixel volume area
SetMaskPW	– set a pixel window channel mask

SYNOPSIS

int *OpenPW(tb, minx, maxx, miny, maxy)
int *tb; register minx, maxx, miny, maxy;

int *OpenPV(tb, nx, ny, nz)

int *ReOpenPW(tb, pw, minx, maxx, miny, maxy)
int *tb; int *pw; register minx, maxx, miny, maxy; int *tb; register nx, ny, nz;

InqPW(pw)
int *pw;

InqPV(pv)
int *pv;

ClosePW(pw)
int *pw;

ClosePV(pv)
int *pv;

InitPW()

InitPV()

SetMaskPW(pw, channelmask)
int *pw; register channelmask;

DESCRIPTION

A *pixel window* is a logical window into an allocated chunk of frame buffer memory. All frame buffer accesses are offset to an open pixel window and clipped to its bounds. *OpenPW* creates a pixel window inside an allotted tile block *tb*. The bounds are set by the standard window four-tuple. The channel mask is set to indicate all four channels. A pixel window pointer is returned.

All frame buffer accesses are offset to an open pixel volume and clipped to its bounds. *OpenPV* creates a pixel volume inside an allotted tile block *tb*. Each of *nz* windows of the pixel volume is of size (*nx*,*ny*). A pixel volume pointer, which may be used interchangeably with a pixel window pointer, is returned. When a pixel volume pointer is used in place of a pixel window pointer, a *z* value must be supplied to indicate the desired slice.

ReOpenPW reuses the same pixel window with different bounds.

InqPW does the inverse of *OpenPW*, taking a *pw* and returning its *tb* and the window bounds.

InqPV does the inverse of *OpenPV*, taking a *pv* and returning its *tb* and the volume sizes.

ClosePW (*ClosePV*) closes an open pixel window (pixel volume).

InitPW clears out all pixel window structures to offer a fresh start; the routine *InitPV* is also offered.

SetMaskPW associates a channel mask with a pixel window. All further accesses to that window affect only the selected channel(s). The argument *channelmask* is of the same form as the runflag of the Chap, with RGBA indicated by bits 0, 1, 2, 3 respectively.

LIBRARY

libpt.a

SEE ALSO

TB(3C)

DIAGNOSTICS

OpenPW and *ReOpenPW* return −1 in **acc** on failure, 0 on success. Reasons for failure are: *maxx* < *minx*; *maxy* < *miny*; *tb* not valid; *minx* < 0; *miny* < 0; *maxx* or *maxy* too big for this tile block; no more space for window structures.

InqPW and *ClosePW* return −1 in **acc** on failure (invalid *pw*), 0 on success.

NAME

PW4Map – remap 4 components of a pixel window

SYNOPSIS

PW4Map(pwsrc, dstpw, map, spada, spadb)
PW *pwsrc, *dstpw;
pixel *map;
pixel *spada, *spadb;

DESCRIPTION

PW4Map changes the color values of a pixel window according to a function expressed as a map between pixel values in and pixel values out.

The map table is actually four tables: TR, TG, TB, TA. The map table should point to the untesselated 4-way value TR[0], TG[0], TB[0], TA[0]. The 4 components of each pixel in the *srcpw* are used as 4 indices into these 4 tables, are looked up, and then written to the *pwdst*. Note that if the *src* array contains negative values (and pixel values may be negative), the table should extend not only forward in scratchpad memory from the map table, but also backwards.

LIBRARY

libpt.a

SEE ALSO

PirlMapComp(3H), PirlMakeMap(3H), PirlCha(3H)

SS4Map(3C), PWMap(3C)

NAME

PWAxb — compute new pixel = *a*∗pixel+*b* for a pixel window.

SYNOPSIS

PWAxb (pw, a, b, spad)
int ∗pw;
pixel ∗a,*b;
pixel ∗spad;

DESCRIPTION

PWAxb computes a new pixel value by multiplying each pixel component by the appropriate components of *a* and adding *b*. The factors are four-way 11-bit values, where 1.0E (2048). This function is useful for performing simple channel arithmetic.

The spad buffer is equal to the maximum of the x and y directions (in pixels) of the pixel window in pixels.

LIBRARY

libpt.a

SEE ALSO

PirlAxb(3H), PirlArithmetic(3H), SSAxb(3C)

NAME

PWCha – perform channel arithmetic on the pixels of a pixel window

SYNOPSIS

PWCha (pw, inbfr, outbfr, coeffs)
int *pw;
pixel *inbfr, *outbfr;
register coeffs;

DESCRIPTION

PWCha applies a linear transformation to the channel values of each pixel within a specified pixel window, given in **b0** on entry. The transformation is given by a 4x5 array of coefficients in scratchpad memory. A pointer to this array is assumed to be in ALU register **r0** when *PWCha* is called. The function also requires an input and output buffer to be available in scratchpad; pointers to these must be provided in **b1** and **b2**, respectively.

The values in the coefficient matrix are 11-bit pixel values. Multiplication is performed as though the channel values were a homogeneous 5-vector being pre-multiplied by the coefficient matrix: the first five values in the array determine the output red value by summing the products of the four channel values with the first four matrix values, and adding the fifth matrix value to the sum.

LIBRARY

libpt.a

SEE ALSO

SSCha(3C), PirlCha(3H)

ERRORS

The ALU accumulator **acc** has 0 for normal return, –1 for errors.

NAME

PWClamp – clamp pixel between 0.0 (0) and 1.0 (0x800) for a pixel window

SYNOPSIS

```
PWClamp (pw, spad)
int *pw;
pixel *spad;
```

DESCRIPTION

PWClamp clamps each pixel in a pixel window to a minimum of 0.0 (0) and a maximum of 1.0E (2048). This is useful for removing the values outside this range that are occasionally produced by filtering.

The *spad* buffer is equal to the maximum of the x and y directions (in pixels) of the pixel window in pixels.

LIBRARY

libpt.a

SEE ALSO

SSClamp(3C)

NAME

PWClear – clear pixel window to *color*

SYNOPSIS

PWClear (pw,color,spad)
int *pw;
pixel *color;
pixel *spad;

DESCRIPTION

PWClear clears the pixel window to *color*.

The *spad* buffer is equal to the maximum of the x and y directions (in pixels) of the pixel window in pixels.

LIBRARY

libpt.a

SEE ALSO

PirlClear(3H)

NAME

PWCopy – copy the source pixel window to the destination pixel window
PWCopyGeneric – PWCopy with user-specified axes, start and direction parameters

SYNOPSIS

PWCopy (srcpw, dstpw, spad)
int *srcpw;
int *dstpw;
pixel *spad;

PWCopy (srcpw, dstpw, spad,
srcRtn, srcStart, srcInc, srcAxis, dstRtn, dstStart, dstInc, dstAxis)
int *srcpw;
int *dstpw;
pixel *spad;
register srcRtn,srcStart,srcInc,srcAxis,dstRtn,dstStart,dstInc,dstAxis;

DESCRIPTION

PWCopy copies the source pixel window to the destination pixel window. The pixel windows may overlap, although the source pixel window will be overwritten. The spad buffer is equal to the maximum of the x-size of the source pixel window.

PWCopyGeneric copies the source pixel window to the destination pixel window. This routine allows the user to specify several options for greater control of the copying operation. The *srcRtn* is used to extract a scanline from the framebuffer into *spad.* The *srcStart* and *srcInc* parameters specify the starting line, relative to the start of the source pixel window, and the increment (usually 1 or –1) in the direction of *srcAxis* (0 for x direction, 1 for y direction). Similarly, the *dstRtn* is used to copy the *spad* buffer into the framebuffer. *dstStart*, *dstInc*, *dstAxis* have the same semantics as their source counterparts.

The *spad* buffer must be large enough to hold the maximum number of pixels copied from the framebuffer by *srcRtn.*

For programming examples, see the source for *PirlCopy* and *PirlReflect.*

LIBRARY

libpt.a

SEE ALSO

PirlCopy(3H), FSCopy(3C), SFCopy(3C), PWSwap(3C)

ERRORS

Both pixel windows must be the same size and belong to the same Chap.

Both axis specifiers in *PWCopyGeneric* must be the same (either 0 or 1).

NAME

PWGeneric – call a spad-to-spad routine for each line of a pixel window

SYNOPSIS

PWGeneric (pw, routine, spad)
int *pw;
register routine;
pixel *spad;

DESCRIPTION

PWGeneric applies the function *routine* to each line of pixel window. This routine must not use any other registers. This is useful for simple routines, like clamp, which need no external information other than the data in the pixel window.

routine is the address of the routine in the chap.

The spad buffer is equal to the maximum of the x and y directions (in pixels) of the pixel window.

LIBRARY

libpt.a

SEE ALSO

PWNot(3C), PWClamp(3C)

NAME

PWMerge – merge two pixel windows into a third

SYNOPSIS

PWMerge (fgddw, bkgdw, dstdw, fgdspad bkgspad dstspad merger, Lf, Lb, j0k0)
int *fgddw, *bkgdw, *dstdw;
int *fgdspad, *bkgspad, *dstspad;
int merger;
register Lf, Lb, j0k0;

DESCRIPTION

The paper "Compositing Digital Images," *SIGGRAPH '84*, discusses the semantics of merging. Briefly, compositing is performed by combining images using the fourth (alpha) channel in the image as a *matte* giving the opacity of image at each pixel. The assumption is that the interesting information in an image is confined to pixels with non-zero opacity, so that the matte may be used to allow backgrounds to show through, in proportion to the value of alpha.

PWMerge implements the Porter-Duff compositing algegra on a pair of pixel windows, given by *fgddw* and *bkgdw*, writing the composited pixels into *dstdw* (the dimensions of the three pixel windows must be identical, but the pixel windows themselves need not be distinct). The routine requires the addresses of three buffers, corresponding to the three pixel windows and equal in size to a scan line. Pointers to these buffers must be in *fgdspad*, *bkgspad*, and *dstspad.*

The merge operation is performed by the Chap routine whose address is in *merger*. The routine *SSMerge*(3C) performs all of the merge operations described in the Porter-Duff paper, as determined by the parameter *j0k0*, a 4-way register value in **r2**. However, there exist scratchpad-to-scratchpad routines which are optimized for certain of these operations. They may be found in other man pages (see the **SEE ALSO** section below).

The *Lf* and *Lb* parameters give attenuation factors for the channels of the foreground and background images, respectively. Their semantics are also discussed in the paper, as well as in the man page for *PirlMerge*(3H).

LIBRARY

libpt.a

SEE ALSO

"Compositing Digital Images," by Porter and Duff (*SIGGRAPH '84*)
PirlMerge(3H), SSMerge(3C)

BUGS

As discussed in the paper, this style of compositing is susceptible to failure on correlated data, for example when the two input pixel windows depict objects with adjacent edges.

NAME

PWNot – subtract pixel value from 1.0 (0x800) for a pixel window

SYNOPSIS

PWNot (pw,spad)
int *pw;
pixel *spad;

DESCRIPTION

PWNot subtracts the pixel value from 1.0E (2048) for each pixel in the pixel window.

The *spad* buffer is equal to the maximum of the x and y directions (in pixels) of the pixel window.

LIBRARY

libpt.a

SEE ALSO

PirlNot(3H)

NAME

PWShift – shift pixel window contents in x and/or y
PWCircularShift – circular shift pixel window contents in x and/or y

SYNOPSIS

```
PWShift (pw, x, y, spad)
int *pw;
register x,y;
pixel *spad;

PWCircularShift (pw, x, y, spad)
int *pw;
register x, y;
pixel *spad;
```

DESCRIPTION

PWShift shifts the contents of a pixel window in *x* and/or *y*. The pixels shifted outside the pixel window are clipped. The original pixels are retained in the exposed area. X and Y may be positive or negative.

PWCircularShift shifts the contents of a pixel window in *x* and/or *y*. The pixels shifted outside the pixel window are circularly shifted around the edge of the pixel window into the exposed area.

The spad buffer is equal to the maximum of the x and y directions (in pixels) of the pixel window.

LIBRARY

libpt.a

SEE ALSO

PirlShift(3H)

NAME

PWShuffle – shuffle components of each pixel for a pixel window.

SYNOPSIS

PWShuffle (pw, r, g, b, a, spad)
int *pw;
index r, g, b, a;
pixel *spad;

DESCRIPTION

PWShuffle uses the index values in *r, g, b, a* to form a general purpose crossbar for shuffling of pixel components.

The index from zero to three in each of the *r, g, b, a* index registers is the offset from which to draw the source component.

The spad buffer is equal to the maximum of the x and y directions (in pixels) of the pixel window.

LIBRARY

libpt.a

SEE ALSO

PirlShuffle(3H), SSShuffle(3C)

NAME

PWSwap – swap the source pixel window and the destination pixel window.

SYNOPSIS

PWSwap (srcpw, dstpw, spad0, spad1)
pixel *srcpw;
pixel *dstpw;
pixel *spad0,*spad1;

DESCRIPTION

PWSwap swaps the source pixel window and the destination pixel window.

The spad buffers, *spad0* and *spad1*, are equal to the x-size of the pixel window.

LIBRARY

libpt.a

SEE ALSO

PirlSwap(3H), FSCopy(3C), SFCopy(3C)

ERRORS

Both pixel windows must be the same size, not overlap, and belong to the same Chap.

NAME

PWTranspose – transpose a pixel window around the diagonal axis.

SYNOPSIS

PWTranspose (pw, spad)
pixel *pw;
pixel *spad;

DESCRIPTION

PWTranspose transposes a pixel window around its diagonal axis (0,0) to (N,N). Any orientation of the pixel window may be achieved by a combination of transpositions and reflections.

The *spad* buffer is equal to the maximum of the x and y directions (in pixels) of the pixel window.

LIBRARY

libpt.a

SEE ALSO

PirlTranspose(3H), PWGeneralSwap(3C)

ERRORS

The pixel window must be square.

NAME

RFxCopy,
GFxCopy,
BFxCopy,
AFxCopy – copy component from scratchpad to frame buffer in increasing x order
RFyCopy,
GFyCopy,
BFyCopy,
AFyCopy – copy component from scratchpad to frame buffer in increasing y order

SYNOPSIS

RFxCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

GFxCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

BFxCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

AFxCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

RFyCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

GFyCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

BFyCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

AFyCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

DESCRIPTION

These procedures copy 4 consecutive component values, either the R, G, B or A, into a pixel in the frame-buffer.

The pixel window (pixel volume) *pw* must have previously been opened with *OpenPW* (*OpenPV*); see *PW*(3C). *RFxCopy, GFxCopy, BFxCopy,* and *AFxCopy* copy *4*n* component values from a scratchpad buffer starting at *src* into [*x*, *y*] to [*x*+*n*−1, *y*] of the pixel window. *RFyCopy, GFyCopy, BFyCopy,* and *AFyCopy* copy *4*n* pixels from a scratchpad buffer starting at *src* into into [*x*, *y*] to [*x*, *y*+*n*−1] of the pixel window. If the pixel window is a pixel volume, a *z* coordinate is used to specify the slice. Clipping is performed with regard to the *pw*, and the number actually read is returned in **acc**. Note that *n* is the number of pixels written into the frame buffer which is 1/4 the size of the scratchpad array.

Only those channels indicated in the channel mask of the pixel window are written to the frame buffer; the other channels of frame buffer pixels are untouched.

If the pixel window is actually a pixel volume, the *z* value is used

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), FRGBACopy(3C), IFCopy(3C), SFCopy(3C), FCCopy(3C)

DIAGNOSTICS

−1 is returned in **acc** if an invalid pixel window (pixel volume) is supplied. The number of pixels copied is returned in **acc**.

NAME

RSCopy,
SRCopy – copy runlength array to scratchpad pixel array

SYNOPSIS

RSCopy(src, dst)
int *src; pixel *dst;

DESCRIPTION

RSCopy expands a runlength array *src* to a normal scratchpad pixel array *dst*. A runlength array is a one-bit, single-channel, run-length-encoded array, with a count *n* followed by *n* run lengths. The initial run-length is the number of initial zeros; the next is the number of subsequent nonzeros, etc. This routine expands such a 1-bit description into an array of (0,0,0,0) and (2048,2048,2048,2048) pixels.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C)

NAME

SCCopy,
SCClear – copy partial scanline from scratchpad to scratchpad

SYNOPSIS

SCCopy(source, target, n, channelnumber)
pixel *source; component *target; register n, channelnumber;

SCClear(source, target, n, channelnumber)
pixel *source; component *target; register n;

DESCRIPTION

SCCopy copies *n* pixels from a scratchpad pixel array *source* to a scratchpad channel array *target*.

SCClear copies a channel of a pixel *n* times to a *target* channel array without incrementing the *source* pixel pointer. The *target* channel array is effectively cleared to the value of the channel *channelnumber* of the *source* pixel.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C)

DIAGNOSTICS

The number of channels copied is returned in **acc**.

NAME

SFxCopy	– copy partial scanline from scratchpad to frame buffer
SFyCopy	– copy partial vertical scanline from scratchpad to frame buffer
SFxCopyBackwards	– copy partial scanline backwards from scratchpad to frame buffer
SFyCopyBackwards	– copy partial vertical scanline backwards from scratchpad to frame buffer
SFxCopyRGBA,	
SFxCopyARGB,	
SFxCopyBARG,	
SFxCopyGBAR	– SFxCopy w/ channel rotation
SFyCopyRGBA,	
SFyCopyARGB,	
SFyCopyBARG,	
SFyCopyGBAR	– SFyCopy w/ channel rotation
SFxClear	– clear partial scanline in frame buffer
SFyClear	– clear partial vertical scanline in frame buffer

SYNOPSIS

SFxCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

SFyCopy(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

SFxCopyBackwards(pw, src, n, x, y)
int *pw; pixel *src; register n, x, y[, z];

SFyCopyBackwards(pw, src, n, x, y)
int *pw; pixel *src; register n, x, y[, z];

SFxCopyRGBA(pw, src, n, x, y[, z])
SFxCopyARGB(pw, src, n, x, y[, z])
SFxCopyBARG(pw, src, n, x, y[, z])
SFxCopyGBAR(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

SFyCopyRGBA(pw, src, n, x, y[, z])
SFyCopyARGB(pw, src, n, x, y[, z])
SFyCopyBARG(pw, src, n, x, y[, z])
SFyCopyGBAR(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

SFxClear(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

SFyClear(pw, src, n, x, y[, z])
int *pw; pixel *src; register n, x, y[, z];

DESCRIPTION

All of these routines copy pixels from the scratchpad to the framebuffer. Various combinations and directions are supported.

Only those channels indicated in the channel mask of the pixel window are written to the frame buffer; the other channels of frame buffer pixels are untouched.

SFxCopy and *SFyCopy* copy *n* pixels from scratchpad into frame buffer. The pixel window (pixel volume) *pw* must previously been opened with *OpenPW* (*OpenPV*); see *PW*(3C). *SFxCopy* copies the *n* pixels from a scratchpad buffer starting at *src* into [*x*, *y*] to [*x*+*n*–1, *y*] of the pixel window. *SFyCopy* copies the *n* pixels from a scratchpad buffer starting at *src* into into [*x*, *y*] to [*x*, *y*+*n*–1] of the pixel window. If the pixel window is a pixel volume, a *z* coordinate is used to specify the slice. Clipping is performed with regard to the *pw*, and the number actually read is returned in **acc**.

SFxCopyBackwards and *SFyCopyBackwards* reverse the order of the pixels copied.

If the pixel window is actually a pixel volume, the *z* value is used to indicate the appropriate window of the volume.

SFxCopyRGBA, *SFxCopyARGB*, *SFxCopyBARG*, and *SFxCopyGBAR* perform an *SFxCopy*, but rotate the channel components during the copy. *SFxCopyRGBA*, *SFxCopyARGB*, *SFxCopyBARG*, *SFxCopyGBAR* rotate the channel components zero, one, two and three positions, respectively. (*SFxCopyRGBA* is exactly the same as *SFxCopy*)

SFyCopyRGBA, *SFyCopyARGB*, *SFyCopyBARG* and *SFyCopyGBAR* perform the same operations on vertical scanlines.

SFxClear and *SFyClear* do not increment through a src scratchpad array; the frame buffer scanline is cleared to the value of the pixel pointed to by *src*.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), SSCopy(3C), SSShuffle(3C)

DIAGNOSTICS

−1 is returned in **acc** if an invalid pixel window (pixel volume) is supplied. The number of pixels copied is returned in **acc**.

NAME

SICopy – copy channels from a pixel array to integer array
SIClear – clear an integer array to a single channel value

SYNOPSIS

SICopy(src, dst, count, channelnumber)
pixel *src,*dst; register count, channelnumber;

SIClear(src, dst, count, channelnumber)
pixel *src,*dst; register count, channelnumber;

DESCRIPTION

SICopy copies *count* channels from the pixel array pointed to by *src* to the the integer array pointed to by *dst*, using the specified channel number.

SIClear clears the integer array pointed to by *dst* of length *count*, to the single channel value pointed to by *src*, using the specified channel number.

LIBRARY

libpt.a

SEE ALSO

CRCopy(3C)

NAME

SS4Map – 4-way mapping of scratchpad values using a mapping table

SYNOPSIS

SS4Map(src,dst, maptable,n)
pixel *src, *dst, *maptable; register n;

DESCRIPTION

The map table is actually four tables: TR, TG, TB, TA. The map table should point to the untesselated 4-way value TR[0], TG[0], TB[0], TA[0]. The *src* array is used as 4-way indices into the 4-way table to produce the 4-way *dst* array. Note that if the *src* array contains negative values (and pixel values may be negative), the table should extend not only forward in scratchpad memory from the map table, but also backwards.

LIBRARY

libpt.a

NAME

SSaxb – scale pixels using the formula A*x+B

SYNOPSIS

SSaxb(src, dst, n, A, B)
pixel *src, *dst; register n, A, B;

DESCRIPTION

SSaxb copies *n* pixels from a scratchpad *src* to a scratchpad *dst*. Input pixels are multiplied by a four-way 11-bit factor, *A*, and added to a four-way 11-bit term, *B*.

TIMING

The inner loop takes 2 ticks.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), SSCopy(3C)

NAME

SSCha – perform channel arithmetic on the pixels of a pixel window

SYNOPSIS

SSCha(inbfr, outbfr, coeffs, linesize)
pixel *inbfr, *outbfr, *coeffs;
register linesize;

DESCRIPTION

SSCha applies a linear transformation to the channel values of each pixel in the input pixel buffer in scratchpad (pointed to by **b0** upon entry), placing the result in the output buffer pointed to by **b1**. The transformation is applied to the number of pixels in **r0**, and is specified by a 4x5 array of coefficients in scratchpad memory. A pointer to this array is assumed to be in ALU register **b2**.

The values in the coefficient matrix are 11-bit pixel values. Multiplication is performed as though the channel values were a homogeneous 5-vector being pre-multiplied by the coefficient matrix: the first five values in the array determine the output red value by summing the products of the four channel values with the first four matrix values, and adding the fifth matrix value to the sum.

LIBRARY

libpt.a

SEE ALSO

PWCha(3C), PirlCha(3H)

ERRORS

The ALU accumulator **acc** has 0 for normal return, -1 for errors.

NAME

SSComb – Combine two images

SYNOPSIS

SSComb(srca, srcb, dst, n, A, B)
pixel *srca, *srcb, *dst; register n, A, B;

DESCRIPTION

SSComb forms a linear combination of *n* pixels from scratchpad *srca* and *srcb* and writes the result to *dst*. Input pixels from *srca* are multiplied by a four-way 14-bit factor, *A*, and added to to the input pixels from *srcb* multiplied by a four-way 14-bit factor, *B*.

TIMING

The inner loop takes 6 ticks.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C), SSAxb(3C), SSMerge(3C), SSCopy(3C)

NAME

SSCompare　　– compare scanline pixel buffers in scratchpad

SYNOPSIS

SSCompare(src0, src1, n)
pixel *src0, *src1; register n;

DESCRIPTION

SSCompare compares *n* pixels in scratchpad buffers *src 0* and *src 1*.

Each time a comparison fails, **acc** is incremented. This results in the four accumulators holding the number of failed comparisons for the four channels.

LIBRARY

libpt.a

NAME

SSCopy,
SSClear — copy partial scanline from scratchpad to scratchpad

SYNOPSIS

SSCopy(src, dst, n)
pixel *src, *dst; register n;

SSClear(src, dst, n)
pixel *src, *dst; register n;

DESCRIPTION

SSCopy copies *n* pixels from a scratchpad *src* to a scratchpad *dst*.

SSClear copies *n* pixels to a *dst* scratchpad location, without incrementing the *src* pixel pointer. The *dst* pixel array is effectively cleared to the value of the *src* pixel.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C)

DIAGNOSTICS

The number of pixels copied is returned in **acc**.

NAME

SSRRRRtoRRRR, ... – Copy one channel from scratchpad to another channel

SYNOPSIS

SSRRRRtoRRRR(src, dst, n)
SSRRRRtoGGGG(src, dst, n)
SSRRRRtoBBBB(src, dst, n)
SSRRRRtoAAAA(src, dst, n)
SSGGGGtoRRRR(src, dst, n)
SSGGGGtoGGGG(src, dst, n)
SSGGGGtoBBBB(src, dst, n)
SSGGGGtoAAAA(src, dst, n)
SSBBBBtoRRRR(src, dst, n)
SSBBBBtoGGGG(src, dst, n)
SSBBBBtoBBBB(src, dst, n)
SSBBBBtoAAAA(src, dst, n)
SSAAAAtoRRRR(src, dst, n)
SSAAAAtoGGGG(src, dst, n)
SSAAAAtoBBBB(src, dst, n)
SSAAAAtoAAAA(src, dst, n)
pixel *src, *dst; register n;

DESCRIPTION

These procedures copy *4*n* values from the specified channel of the *src* pixel array to the destination pixel array. The source and the destination should be aligned to a multiple of 4. This is an optimized version of *CCCopy*.

LIBRARY

libpt.a

SEE ALSO

SSCopyRGBA(3C), SSCopyRGBALUT(3C), CICopy(3C)

DIAGNOSTICS

The number of pixels copied is returned in **acc**.

NAME

SSRtoRGBA,
SSGtoRGBA,
SSBtoRGBA,
SSAtoRGBA – Copy one channel from scratchpad to 4 channels

SYNOPSIS

SSRtoRGBA(src, dst, n)
SSGtoRGBA(src, dst, n)
SSBtoRGBA(src, dst, n)
SSAtoRGBA(src, dst, n)
pixel *src, *dst; register n;

DESCRIPTION

These procedures copy *4*n* values from the specified channel of the *src* pixel array to the destination pixel array. Each value is replicated in all 4 components of the of the destination array. The source and destination arrays should be aligned to a multiple of 4. This is an optimized version of *CCCopy.*

LIBRARY

libpt.a

SEE ALSO

SSCopyComp(3C), SSCopyRGBALUT(3C), CICopy(3C)

DIAGNOSTICS

The number of pixels copied is returned in **acc.**

NAME

SSRtoRGBALUT,
SSGtoRGBALUT,
SSBtoRGBALUT,
SSAtoRGBALUT – Copy one channel from scratchpad to 4 channels through a color table

SYNOPSIS

SSRtoRGBALUT(src, dst, n, table)
SSGtoRGBALUT(src, dst, n, table)
SSBtoRGBALUT(src, dst, n, table)
SSAtoRGBALUT(src, dst, n, table)
pixel *src, *dst; register n;
pixel table[];

DESCRIPTION

These procedures copy *n* values from the specified channel of the *src* pixel array to the destination pixel array. Each value from the source array is looked up in the 4-way color table *map* to yield the component values of the result.

The map table is actually four tables: TR, TG, TB, TA. The map table should point to the untesselated 4-way value TR[0], TG[0], TB[0], TA[0]. The address passed should be the address of the 0th element in this table. If the source array contains negative values, the table should extend backward.

The source and destination arrays should be aligned to a multiple of 4.

LIBRARY

libpt.a

SEE ALSO

PirlMapComp(3H)

PWMap(3C)

SSCopyComp(3C), SSCopyRGBA(3C), CICopy(3C)

NAME

SSMerge	– merge partial scanline from scratchpad over scratchpad
SSMergeIn	– scratchpad to scratchpad merge using IN operator
SSMergeOut	– scratchpad to scratchpad merge using OUT operator
SSMergeOver	– scratchpad to scratchpad merge using OVER operator
SSMergeAtop	– scratchpad to scratchpad merge using ATOP operator
SSMergeUnder	– scratchpad to scratchpad merge using UNDER operator

SYNOPSIS

SSMerge(Fptr, Bptr, Tptr, JKptr, n, Lf, Lb)
pixel *Fptr, *Bptr, *Tptr, *JKptr; register n, Lf, Lb;

SSMergeIn(Fptr, Bptr, Tptr, n, Lf, Lb)
pixel *Fptr, *Bptr, *Tptr; register n, Lf, Lb;

SSMergeOut(Fptr, Bptr, Tptr, n, Lf, Lb)
pixel *Fptr, *Bptr, *Tptr; register n, Lf, Lb;

SSMergeOver(Fptr, Bptr, Tptr, n, Lf, Lb)
pixel *Fptr, *Bptr, *Tptr; register n, Lf, Lb;

SSMergeAtop(Fptr, Bptr, Tptr, n, Lf, Lb)
pixel *Fptr, *Bptr, *Tptr; register n, Lf, Lb;

SSMergeUnder(Fptr, Bptr, Tptr, n, Lf, Lb)
pixel *Fptr, *Bptr, *Tptr; register n, Lf, Lb;

DESCRIPTION

SSMerge merges *n* pixels from one scratchpad location with another, writing the result into a third scratchpad buffer, *Tptr*. The compositing expression is

*Lf**F op *Lb**B.

The equation being computed is

Tptr = (j0 + j1**Lb*[a]*B[a])**Lf**F + (k0 + k1**Lf*[a]*F[a])**Lb**B

where F and B are the pixel structs associated with *Fptr* and *Bptr*, *Lf* and *Lb* are 4-way coefficients for the foreground and background, and *x*[a] designates the alpha component of *x*.

JKptr points to four words in scratchpad holding {j0, j1, k0, k1} (these are all 11-bit quantities). Sample settings are {1.0E, 0, 1.0E, –1.0E} for **Over**, {0, 1.0E, 0, 0} for **In**.

The current implementation of the general merge takes about forty ticks per pixel. The shorthand operators can reduce this to as little as six ticks per pixel. Routines are written to optimize code for these operators: **Over**, **Out**, **Atop**, and **Under**. These operations may be accessed using the shorthand calls as described in the synopsis.

The following equations are implemented for each shorthand operator, using the same notation as the general merge.

SSMergeIn: Tptr = *Lf**F**Lb*[a]*B[a]

SSMergeOut: Tptr = *Lf**F*(1 – *Lb*[a]*B[a])

SSMergeOver: Tptr = *Lf**F + (1 – F[a])**Lb**B

SSMergeAtop: Tptr = (*Lb*[a]*B[a])**Lf**F + (1 - *Lf*[a]*F[a])**Lb**B

SSMergeUnder: Tptr = (1 - *Lb*[a]*B[a])**Lf**F + *Lb**B

LIBRARY

libpt.a

SEE ALSO

SSCopy(3C),
Compositing Digital Images, by Porter and Duff.

NAME

SSPaint – paint partial scanline from scratchpad over scratchpad
SSPaintOver – SSPaint using OVER operator
SSPaintCopy – merge pixels using spad matte

SYNOPSIS

SSPaint(Fptr, Bptr, Tptr, matte, JKptr, count, La, Fincr, matteincr)
pixel *Fptr, *Bptr, *Tptr, *matte, *JKptr; register count, La;
index Fincr, matteincr;

SSPaintOver(Fptr, Bptr, Tptr, matte, count, La, Fincr, matteincr)
pixel *Fptr, *Bptr, *Tptr, *matte; register count, La;
index Fincr, matteincr;

SSPaintCopy(Fptr, Bptr, Tptr, matte, count, La, Fincr, matteincr)
pixel *Fptr, *Bptr, *Tptr, *matte; register count, La;
index Fincr, matteincr;

DESCRIPTION

SSPaint paints *count* foreground pixels over *count* background pixels with regard to an independent *matte*, writing the result into a *Tptr* scratchpad buffer. The compositing expression being calculated is

((F op B) IN A) PLUS (B OUT A).

The equation being computed is therefore

$$Tptr = (j0 + j1*B[a])*a*La*F + (1 - a*La*(1 - k0 - k1*F[a]))*B$$

where F and B are the pixel structs associated with *Fptr* and *Bptr*, *La* is a coefficient for the independent matte *a*, and *x*[a] designates the alpha component of *x*.

JKptr points to four words in scratchpad holding {j0, j1, k0, k1} (these are all 11-bit quantities). Sample settings are {1.0E, 0, 1.0E, –1.0E} for **Over**, {0, 1.0E, 0, 0} for **In**.

SSPaintOver paints *count* pixels from scratchpad OVER frame buffer. The equation computed is

$$Tptr = a*La*F + (1 - a*La*F[a])*B,$$

where F and B are the pixel structs associated with *Fptr* and *Bptr*, *La* is a coefficient for the independent matte *a*, and *x*[a] designates the alpha component of *x*.

SSPaintCopy merges *count* foreground pixels from the scratchpad *Fptr* inside the scratchpad matte with *count* background pixels from the scratchpad *Bptr* outside the scratchpad matte into the target scratchpad *Tptr*.

$$Tptr = a*La*F + (1 - a*La)*B$$

where F and B are the pixel structs associated with *Fptr* and *Bptr*, *La* is a coefficient for the independent matte *a*, and *x*[a] designates the alpha component of *x*.

LIBRARY

libpt.a

SEE ALSO

SSCopy(3C)
Compositing Digital Images, by Porter and Duff.

NAME

SSShuffleBroadcast – SSCopy, broadcasting single component of src to dst
SSShuffleXbar – SSCopy using specified roffset,goffset, boffset, aoffset transform
SSShuffleRot – SSCopy with specified channel rotation

SYNOPSIS

SSShuffleRot(src, dst, count, rotation)
pixel *src,*dst; register count; index rotation;

SSShuffleBroadcast(src, dst, count, component)
pixel *src,*dst; register count; index component;

SSShuffleXbar(src, dst, count, roffset, goffset, boffset, aoffset)
pixel *src,*dst; register count; index roffset,goffset,boffset,aoffset;

DESCRIPTION

SSShuffleRot copies from scratchpad to scratchpad using the specified channel rotation *rotation*; *count* pixels are copied.

0 RGBA->RGBA (copy)
1 RGBA->GBAR
2 RGBA->BARG
3 RGBA->ARGB

SSShuffleBroadcast copies only the specified component from scratchpad to scratchpad:

0 R
1 G
2 B
3 A

count components are copied.

SSShuffleXbar forms a general purpose crossbar for movement of scratchpad components into the scratchpad destination. The index registers hold offsets, 0 for *roffset* to 3 to *aoffset*, for the destinations of the individual components of the pixel. For example, the *roffset* component of the src pixel is copied to *dst address* + *roffset*; *count* pixels are copied. Thus, this procedure can perform the function of the two previous procedures. However, the other two are faster.

LIBRARY

libpt.a

SEE ALSO

SFCopy(3C)

DIAGNOSTICS

The number of pixels copied is returned in **acc.**

NAME

SYCopy

SYNOPSIS

SYCopy(buf, cnt, dest_addr, priority)
int *buf; register cnt, dest_addr, priority

DESCRIPTION

This routine copies pixels from the scratchpad to the yapbus. *buf (b0)* points to the base of the scratchpad buffer to be sent. *cnt (r0)* holds the number of pixels to be sent out, and should be a multiple of 16. *dest_addr (r1)* has the receiver address that data is to be sent to, and must be a number from 1 through 15. *priority (r2)* has the transmitter priority level, and must be a number from 0 through 15.

Pixels are transmitted at a rate of two pixels per CPU tick.

LIBRARY

libpt.a

SEE ALSO

YSCopy.3c

DIAGNOSTICS

A non-zero value is returned in **acc** if a transmission problem occurs.

NAME

AllocTB	– initializes a tile block in frame buffer memory
ReAllocTB	– reuses a previously allocated tile block
InqTB	– gather information on tile block
DeallocTB	– deallocates a tile block
InitTB	– cleans out the tile block area

SYNOPSIS

int * AllocTB(firsttile, tilewidth, tileheight)
register firsttile, tilewidth, tileheight;

int * ReAllocTB(tb, firsttile, tilewidth, tileheight)
int *tb; register firsttile, tilewidth, tileheight;

InqTB(tb)
int *tb;

DeallocTB(tb)
int *tb;

InitTB()

DESCRIPTION

A tile block is a linear array of 32x32 pixel tiles in frame buffer memory. The tile block data structure lets the Chap and video controller agree on a rectangular allocation of the linear memory. *AllocTB* initializes a tile block for subsequent creation of pixel windows. Note that, in spite of the routine name, this **does not allocate the space** – presumably that has been done in the host. *firsttile* is the first of the linear tiles (upper left corner); *tilewidth* is the number of tiles across the rectangular block; *tileheight* is the number of tiles down the rectangular block; the tile block pointer is returned.

ReAllocTB reuses a previously allocated tile block. *firsttilerm*, *tilewidth*, and *tileheight* have the same meanings as in *AllocTB*.

InqTB does the inverse of *AllocTB*, taking a *tb* and returning *firsttile*, *tilewidth*, and *tileheight*.

DeallocTB deallocates an open tile block.

InitTB clears out all tile block structures to offer a fresh start.

LIBRARY

libpt.a

SEE ALSO

PW(3C)

DIAGNOSTICS

AllocTB and *ReAllocTB* return –1 in **acc** on failure (no more space for tile blocks) and 0 on success.

DeallocTB and *InqTB* return –1 in **acc** on failure (invalid *tb*), and 0 on success.

NAME

TBCopy – copy between tile blocks in frame buffer memory
FFCopy – copy a single tile between locations in frame buffer memory

SYNOPSIS

#include <chap/pbus.h>

TBCopy(srctb, dsttb, chanmask)
int *srctb, *dsttb; register channelmask;

FFCopy(srctile, dsttile, channelmask)
int srctile, dsttile; register channelmask;

DESCRIPTION

The routines *TBCopy* and *FFCopy* perform a fast copy of whole tiles of image memory without using scratchpad buffers. *FFCopy* copies a single tile to another location in image memory, where the tiles are referred to by number. Somewhat friendlier is *TBCopy*, which copies an entire tile block (as discussed in TB(3C)), taking as arguments a source and destination tile block. No checking is performed to assure that the two tile blocks are of like size.

The *channelmask* argument to both routines may be used to restrict the copy to a subset of the pixel channels. The bit masks *PBUSCSR_RED*, *PBUSCSR_GREEN*, *PBUSCSR_BLUE* and *PBUSCSR_ALPHA*, defined in <chap/pbus.h>, are bitwise-or'ed to specify the requisite channels. No crossbar-like interchannel copying is supported.

LIBRARY

libpt.a

SEE ALSO

PW(3C), TB(3C)

NAME

YFCopy

SYNOPSIS

YFCopy(tile, chanmask, addr)
int *tile; register chanmask, addr

DESCRIPTION

This routine copies pixels from the yapbus to the framebuffer. *tile (b0)* holds the framebuffer tile number to be filled in from the yapbus. *chanmask (r0)* holds the channel mask to be used when transferring data to the framebuffer. *addr (r1)* has the receiver address that is to be used, and must be a number from 1 through 15.

Pixels are received at a rate of one pixel per CPU tick. Tiles are filled in using 32 pixel X access.

LIBRARY

libpt.a

SEE ALSO

FYCopy.3c

DIAGNOSTICS

A non-zero value is returned in acc if a transmission problem occurs.

NAME

YSCopy

SYNOPSIS

YSCopy(buf, cnt, dest_addr, priority)
int *buf; register cnt, dest_addr

DESCRIPTION

This routine copies pixels from the yapbus to the scratchpad. *buf (b0)* points to the base of the scratchpad buffer to be filled. *cnt (r0)* holds the number of pixels to be received, and should be a multiple of 16. *dest_addr (r1)* has the receiver address to be used, and must be a number from 1 through 15.

Pixels are received at a peak rate of one pixel per CPU tick.

LIBRARY

libpt.a

SEE ALSO

SYCopy.3c

DIAGNOSTICS

A non-zero value is returned in acc if the transmitter disappears after starting a transfer.

NAME

libpx – introduction to Pixar image transformation library

DESCRIPTION

libpx contains routines to geometrically transform images. There are procedures to change the size of an image using linear, quadratic or cubic interpolation. A procedure exists to decrease the size of an image. Other procedures can be used to rotate and warp images.

LIBRARY

/usr/pixar/chap/lib/libpx.a

SEE ALSO

intro(3C), libcolor(3C), libpG(3C), libpip(3C), libpm(3C), libpt(3C)

LIST OF FUNCTIONS

Name	*Page*	*Description*
PWResize	PWResize(3C)	– resize source pixel window to destination pixel window
PWShear	PWShear(3C)	– Shear a pixel window
SSHalve	SSHalve(3C)	– average 2 scanlines down to one of half size
hd1	SSScale(3C)	– use no filter to scale down horizontally
hd2	SSScale(3C)	– use linear filter to scale down horizontally
hd4	SSScale(3C)	– use cubic filter to scale down horizontally
hu1	SSScale(3C)	– use no filter to scale up horizontally
hu2	SSScale(3C)	– use linear filter to scale up horizontally
hu4	SSScale(3C)	– use cubic filter to scale up horizontally
setmag1table	SSScale(3C)	– set up filter coefficients for subsequent hu1 or vu1
setmag2table	SSScale(3C)	– set up filter coefficients for subsequent hu2 or vu2
setmag4table	SSScale(3C)	– set up filter coefficients for subsequent hu4 or vu4
setmin1table	SSScale(3C)	– set up filter coefficients for subsequent hd1 or vd1
setmin2table	SSScale(3C)	– set up filter coefficients for subsequent hd2 or vd2
setmin4table	SSScale(3C)	– set up filter coefficients for subsequent hd4 or vd4
stwarp	stwarp(3C)	– warp source to target
stwarptable	stwarptable(3C)	– initialize quadradic warping table
vd1	SSScale(3C)	– use no filter to scale down vertically
vd2	SSScale(3C)	– use linear filter to scale down vertically
vd4	SSScale(3C)	– use cubic filter to scale down vertically
vu1	SSScale(3C)	– use no filter to scale up vertically
vu2	SSScale(3C)	– use linear filter to scale up vertically
vu4	SSScale(3C)	– use cubic filter to scale up vertically

The routine *hu4* loops *Ow* times, one for each output pixel, using the *count table* to indicate which source pixels contribute to that output pixel, and using the *coefficient table* to recover the weighting factors for each source pixel contribution.

The inner loop takes 11 ticks per output pixel. The total time is approximately (11**Ow*+40) ticks.

The *hd4* routine takes an input scanline of *Iw+6* pixels and produces an output scanline of *Ow* pixels. The *Iw* pixels of the input scanline are normally padded by 3 null pixels on each end. The *coefficient table* and *count table* must have been produced by the *setmin4table* routine. The *hd4* routine leaves these tables untouched. The value *Ow* is approximately floor[5+*scale**(*Iw*–1)]; the input value passed in **r1** should be the value of *Ow* returned by *setmin4table*.

The routine *hd4* loops for each of the *Iw* input pixels, using the *count table* to indicate which target pixels get contributions from that source pixel, and using the *coefficient table* to recover the weighting factors for each source pixel contribution.

The inner loop takes 17 ticks per input pixel. The total time is approximately (17**Iw*+40) ticks.

The 2I vu4 routine takes an input scanline of *width* pixels and produces output scanlines. The routine is intended for interleaved use with a horizontal scaling routine (e.g., *hu4*, *hd4*) for the second pass of two dimensional scaling without the use of an intermediate picture buffer. The only intermediate storage is pointed to by *scanlineptr*. This is an array of 3 scanline pointers, pointing to the last three source scanlines.

To effect this interleaved 2D scaling, the *vu4* routine returns several values. The base registers pointing to the *coefficient table* and *count table* are incremented, and their values at the end of one call to *vu4* should be the same as their input values at the start of the next call. On output, **r1** holds a flag indicating whether the target scanline has been completed. On output, **r2** holds a flag indicating whether another input scanline is needed. On output, **r3** holds a flag indicating whether the output scanline must be zeroed before reuse. In fact, *vu4* always produces an output scanline (**r1=1**), and never requires it to be zeroed (**r3=0**). These outputs are done this way for compatibility with *vd4*. Thus **r1** = and **r3** = 0.

The *coefficient table* and *count table* for *vu4* must have been produced by the *setmag4table* routine. The *vu4* routine leaves these tables untouched. The value *width* is presumably the output width of the horizontal scaling process.

The routine loops for each of the *width* output pixels, using this last input and three previous ones to contribute to the current pixel of the target scanline, and using the *coefficient table* to recover the weighting factors for each contribution.

The inner loop takes 15 ticks per output pixel. The total time is approximately (15**Ow*+50) ticks.

The *vd4* routine takes an input scanline of *width* pixels and possibly produces an output scanline. The routine is intended for interleaved used with a horizontal scaling routine (e.g., *hu4*, *hd4*) for the second pass of two dimensional scaling without the use of an intermediate picture buffer. The only intermediate storage in pointed to by *scanlineptr*. This is array of 3 scanline pointers, pointing to the next three target scanlines to be produced.

To effect this interleaved 2D scaling, the *vd4* routine returns several values. The base registers pointing to the *coefficient table* and *count table* are incremented along, and their values at the end of one call to *vd4* should be the same as their input values at the start of the next call. On output, **r1** holds a flag indicating whether the target scanline has been completed. It is possible that more input scanlines are necessary before this output can be finished. On output, **r2** holds a flag indicating whether another input scanline is needed. On output, **r3** holds a flag indicating whether the output scanline must be zeroed before reuse. In fact, *vd4* always needs another input scanline (**r2=1**). These outputs are done this way for compatibility with *vu4*.

The *coefficient table* and *count table* for *vd4* must have been produced by the *setmin4table* routine. The *vd4* routine leaves these tables untouched. The value *width* is presumably the output width of the horizontal scaling process.

The routine loops for each of the *width* output pixels, using this last input scanline to contribute to the current pixel of the target scanline and the next three after that, and using the *coefficient table* to recover the weighting factors for each contribution.

The inner loop takes 16 ticks per output pixel. The total time is approximately (16**Ow*+50) ticks.

The *setmag4table* routine sets up filter coefficients specifically for the *hu4* and *vu4* routines. Two table are filled: the *counttable* holds a count of the number of output pixels that map back into each of the *Iw*+1 intervals of the input scanline; the *coefftable* holds sets of four filter coefficients for each of those output pixels. An output pixel is computed by summing the products of the four filter coefficients with the four consecutive input pixels that straddle the current interval.

Filter coefficients come in sets of four because the filter (see filtertable.s) spans four pixels. The current filter of choice is a Catmull-Rom basis function.

The *setmag4table* routine takes 25 ticks for each output pixel.

The *setmin4table* routine sets up filter coefficients specifically for the *hd4* routine. Two tables are filled: the *counttable* holds a count of the number of input pixels that map into each of the *Ow*+1 intervals of the output scanline; the *coefftable* holds sets of four filter coefficients for each of those output pixels. An input pixel contributes to four consecutive output pixels which straddle the current interval by multiplying it by this set of four filter coefficients. An output pixel is accumulated with as many input contributions as necessary.

Filter coefficients come in sets of four because the filter (see filtertable.s) spans four pixels. The current filter of choice is a Catmull-Rom basis function.

The *setmin4table* routine takes 27 ticks for each input pixel.

LIBRARY

libpx.a

DIAGNOSTICS

Both the *setmag4table* and *setmin4table* routines return negative values in **acc** if the scale is out of the [0,1.0E] range.

NAME

stwarp – warp source to target

SYNOPSIS

```
stwarp(inputwidth, outputwidth, outputoffset, src, dst, hptr)
register inputwidth, outputwidth, outputoffset;
pixel *src, *dst;
double *hptr;
```

DESCRIPTION

stwarp takes an input scanline pointed to by *iptr* and an array of targets, double precision values pointed to by *hptr,* which indicate where each of the input pixels maps into the output scanline. The routine then distorts the incoming pixel values as appropriate to create pixel values for the output pixel array pointed to by *dst.* The routine is given the length of the input array *inputwidth* and the length *outputwidth* and offset *outputoffset* of the output array.

The routine will write a double precision number immediately preceding and immediately after the input array of targets, so the input array must allow for 2 words of padding at each end. The routine also expects a padded input scanline, though nothing will be written into the padding. The calling program should fill the padding with whatever value, most likely (0,0,0,0), is the appropriate border color. Note that in each case, the pointer to the array points beyond the initial padding.

The idea here is that an input array represents pixel values at the discrete sample points (0.5, 1.5, 2.5, ... *inputwidth* –.5). Each of these *inputwidth* samples will be mapped to a target space according to the target array (t(0.5), t(1.5), t(2.5), ..., t(*inputwidth* –.5)). The assumption at the endpoints is that t(0) = *outputoffset* and *outputwidth+outputoffset* = t(*inputwidth*).

Using linear interpolation where the targets are far apart and area averaging where the targets are close together, the input is squashed and stretched to produce an anti-aliased output array.

LIBRARY

libpx.a

NAME

stwarptable – initialize quadratic warping table

SYNOPSIS

stwarptable(width, height, y0, y1, x0, x1, hptr)
register width, height, y0, y1, x0, x1;
double *hptr;

DESCRIPTION

stwarptable produces an array of targets for subsequent use by the routine *stwarp*.

This routine relates to the math used for the *SIGGRAPH '85* warping demo, so a little background is appropriate. The intended interaction is to move a point (*x0*, *y0*) of the source picture to a point (*x1*, *y1*) of the target picture while all perimeter points of the window stay fixed. Assume the window goes from 0 to w in x and from 0 to h in y.

Consider the x pass. We have some function f(x) = a+b*x+c*x*x (0 <= x <= w) at each line y which warps quadratically in x. This is specified by {f(0)=0; f(w)=w; f(i)=j;}; in other words, the right and left stay fixed while i moves to j.

So what are i and j for any line y? They are completely determined by the need to warp point *x0* to point *x1* at line *y0*. This meets the first half of our goal of mapping (*x0*, *y0*) to (*x1*, *y1*). Thus, for 0 <= y <= h, i(y) = *x0*; j(y) = d+e*y+f*y*y constrained by {j(0)=*x0*; j(*y0*)=*x1*; j(h)=*x0*}; in other words, at the top and bottom the point *x0* stays put, while *x0* moves to *x1* at line *y0*.

The y pass is similar, except that now we guarantee that *y0* moves to *y1* at column *x1*.

The routine *stwarptable* accepts the picture *width* and *height*, with the understanding that *x0* moves to *x1* on line *y0*. The routine then takes the line y and produces an array of output locations (double precision) in which each of the *width* input pixels maps to in the output image.

For the y pass of and two-pass method for warping, the x's and y's must naturally be reversed.

LIBRARY

libpx.a

NAME

PWResize – resize source pixel window to destination pixel window

SYNOPSIS

```
PWResize (srcpw, dstpw, hcoefftable, hcounttable, vcoefftable, vcounttable,
          vinput, voutput, rest, hinput, hextent, vextent, how, hj0, vow, vj0)
int *srcpw, *dstpw;
int *hcoefftable,*hcounttable;
int *vcoefftable,*vcounttable;
index pixel **vinput,**voutput;
index pixel **rest;
index pixel *hinput;          /* horizontal input buffer for FxSCopy */
register hextent,vextent;     /* horizontal/vertical filter extents (2 or 4) */
register how,hj0;             /* output width/offset from set[min,mag]table */
register vow,vj0;             /* output width/offset from set[min,mag]table */
```

DESCRIPTION

PWResize is an assembly code call for resizing an image. This routine calls the *SSScale*(3C) routines in libpx.a for horizontal and vertical scaling of the image. The routines are fast and need no intermediate off-screen framebuffer storage, unlike *PWRotate*, but they are somewhat complicated to use.

This routine needs several temporary areas of scratchpad memory.

The coefficient and counttables are produced by the set[min,mag][2,4]table.s routines. These routines are explained in *SSScale*(3C). Each of these scaling table routines takes integer and fractional scales and produces a magnification (minification) coefficient table, a count table, and two additional values. The caller must have allocated enough space for these tables, usually via Chad.

The first variable is the modified output width, representing the actual number of pixels to write, some of which will be clipped by the pixel window. The scaling routine produces inaccurate values toward the edges of the window, and this slightly higher value makes sure these values will lie outside the pixel window. The second variable is offset, a negative value, which is added to the starting pixel position in the scanline, so the valid pixels start at the zeroth position. After calling the appropriate set[min,mag][2,4]table routine, these values are saved, then passed to *PWResize*, as the horizontal output width (*how*), the horizontal offset (*hj0*), the vertical output width (*vow*), and the vertical offset (*vj0*).

The vertical scaling parameters may need several additional scanlines of scratchpad memory in order to create output scanlines during resizing. The scanline buffers are reused by the vertical scaling routine. The reuse of buffers is controlled by the scaling routine, so it is only necessary to allocate the buffers and start it off.

The number of input scanline buffers and out scanline buffers is dependent on the filter size used, and the direction of scaling.

To handle overlapping windows, *PWResize* might swap the sense of "h" and "v," and use vertical scanlines. This means that the buffers must be allocated to hold the maximum of the horizontal and vertical widths.

If *maxow* is the maximum of *how* and *vow*, and *maxiw* is the maximum of *hiw* and *viw*, the buffers should be: *vinput* a pointer to a pointer to *maxow* pixels; *voutput* a pointer to a pointer to *maxow* pixels; *rest* a table of three pointers, each to *maxow* pixels (see below); and *hinput* a pointer to *maxiw* pixels.

WARNING: The next paragraph will be infinitely clearer if you read and understand the *SSScale*(3C) manual page in this section.

Consider the case of a filter size of four, and vertical scale greater than one (vu4). This routine takes four input scanlines and creates one output scanline. Since *PWResize* requires pointers to pointers to scanlines, a small buffer area must be allocated. Space for each of these scanlines, input and output, must be allocated, as well as the other areas described in this paragraph. Notice that the *vinput*, *voutput* and *rest* parameters are pointers to pointers to pixels. The easiest thing is to allocate a spad of one word for the input, output, and three words for the *rest* buffer. The pointer to the input scanline, (after horizontal scaling), is placed in the input word. This now a pointer to the pointer to the input scanline. The pointer to the output scanline, (produced by vertical scaling), is placed in the output word. This now a pointer to the pointer to the output scanline. The additional three input scanline pointers are placed in the rest buffer. The address of the input scanline before horizontal scaling is placed in *hinput*.

For scaling down (vd4), the *rest* buffer stores pointers to the additional output scanlines. Instead of placing the pointers to the additional input scanlines in the *rest* buffer, the pointers of the output scanlines are placed in the *rest* buffer.

LIBRARY

libpx.a

SEE ALSO

PirlResize(3H)

ERRORS

Both pixel windows must belong to the same chap.

NAME

PWShear – Shear a pixel window

SYNOPSIS

#include <chad.h>
#include <pirl.h>
PWShear4, PWShear2

parameters:
b0 – srcpw
b1 – dstpw
b2 – bufferA
b3 – bufferB
b4 – bufferC
b5 – bufferD
r0 – scalef (fractional part (16-bits))
r1 – scalei (integer part)
r2 – offsetf (fractional part (16-bits))
r3 – offseti (integer part)
r4 – incrementf (fractional part (16-bits))
r5 – incrementi (integer part)
r6 – access mode
r7 – clr
r8 – width
r9 – height

DESCRIPTION"

PWShear4 and *PWShear2* shear a source pixel window and places the result in a destination pixel window. *PWShear4* uses a cubic resampling filter with a filter extent of 4. *PWShear2* uses a linear resampling filter with a filter extent of 2.

srcpw defines the source pixel window to shear.
dstpw defines the the destination pixel window.

Four scratchpad buffers are required. Their sizes must be at least as large as the following:
bufferA – 4 * (input_width+6) words
bufferB – 4 * (scale*(input_width+3)+1) words
bufferC – filter_extent * (max(scale, 1)*(input_width+3)+1) words
bufferD – max(scale, 1)*(input_width+3)+1 words
scalef, scalei specify a scale factor for resizing each scanline

Additional parameters are:
offsetf, offseti specify the offset for the first destination scanline.
incrementf, incrementi specifes the incremental offset for each additional destination scanline.
access specifies the scanline access directions for the src and dst.
access must be one of the following defined options:

```
#define XIN_XOUT               0
#define XBACKWARDSIN_XOUT      1
#define YIN_XOUT               2
#define YBACKWARDSIN_XOUT      3
#define YIN_YOUT               4
#define YBACKWARDSIN_YOUT      5
#define XIN_YOUT               6
#define XBACKWARDSIN_YOUT      7
```

clr is a flag wich specifies whether shear should clear out its borders.
iw, ih specify the input width and height of
the src window with respect to

the access mode specified (above).

FILES

/usr/pixar/chap/src/lib/libpx/pwshear.s

LIBRARY

libpx.a

SEE ALSO

Rotate(1), PirlRotate(3h), PirlAffine(3h), PirlShear(3c)

NAME

SSHalve – average 2 scanlines down to one of half size

SYNOPSIS

SSHalve(src1, src2, dst, n)
pixel *src1, *src2, *dst; register n;

DESCRIPTION

SSHalve takes two scanlines *src1* and *src2* of length *2n* pixels and averages them down to one scanline *dst* of length *n* pixels. The computation is simply

$$dst[i] = (src1[2i] + src1[2i+1] + src2[2i] + src2[2i+1] + 2)/4$$

for i between 0 and $n-1$.

LIBRARY

libpx.a

SEE ALSO

SSCopy(3C)

NAME

hu4	– Use cubic filter to scale up horizontally
hd4	– Use cubic filter to scale down horizontally
vu4	– Use cubic filter to scale up vertically
vd4	– Use cubic filter to scale down vertically
setmag4table, setmin4table	– set up filter coefficients for subsequent hd4 or vd4
hu2	– Use linear filter to scale up horizontally
hd2	– Use linear filter to scale down horizontally
vu2	– Use linear filter to scale up vertically
vd2	– Use linear filter to scale down vertically
setmag2table, setmin2table	– set up filter coefficients for subsequent hd2 or vd2
hu1	– Use no filter to scale up horizontally
hd1	– Use no filter to scale down horizontally
vu1	– Use no filter to scale up vertically
vd1	– Use no filter to scale down vertically
setmag1table, setmin1table	– set up filter coefficients for subsequent hd1 or vd1

SYNOPSIS

```
hu4(src, dst, coefftable, counttable, Iw, Ow)
pixel *src, *dst, *coefftable, *counttable;
register Iw, Ow;

hd4(src, dst, coefftable, counttable, Iw, Ow)
pixel *src, *dst, *coefftable, *counttable;
register Iw, Ow;

vu4(srcptr, dstptr, coefftable, counttable, width, scanlineptr)
pixel **srcptr, **dstptr, *coefftable, *counttable;
register width;
index pixel *scanlineptr[3];

vd4(srcptr, dstptr, coefftable, counttable, width, scanlineptr)
pixel **srcptr, **dstptr, *coefftable, *counttable;
register width;
index pixel *scanlineptr[3];

setmag4table(coefftable, counttable, Iw, scale, reciprocal, outputoffset)
pixel *coefftable, *counttable;
register Iw, scale, reciprocal;
register double outputoffset;

setmin4table(coefftable, counttable, Iw, reciprocal, scale, outputoffset)
pixel *coefftable, *counttable;
register Iw, reciprocal, scale;
register double outputoffset;

hu2(src, dst, coefftable, counttable, Iw, Ow)
pixel *src, *dst, *coefftable, *counttable;
register Iw, Ow;

hd2(src, dst, coefftable, counttable, Iw, Ow)
pixel *src, *dst, *coefftable, *counttable;
register Iw, Ow;
```

vu2(srcptr, dstptr, coefftable, counttable, width, scanlineptr)
pixel **srcptr, **dstptr, *coefftable, *counttable;
register width;
index pixel *scanlineptr[1];

vd2(srcptr, dstptr, coefftable, counttable, width, scanlineptr)
pixel **srcptr, **dstptr, *coefftable, *counttable;
register width;
index pixel *scanlineptr[1];

setmag2table(coefftable, counttable, Iw, scale, reciprocal, outputoffset)
pixel *coefftable, *counttable;
register Iw, scale, reciprocal;
register double outputoffset;

setmin2table(coefftable, counttable, Iw, reciprocal, scale, outputoffset)
pixel *coefftable, *counttable;
register Iw, reciprocal, scale;
register double outputoffset;

hu1(src, dst, coefftable, counttable, Iw, Ow)
pixel *src, *dst, *coefftable, *counttable;
register Iw, Ow;

hd1(src, dst, coefftable, counttable, Iw, Ow)
pixel *src, *dst, *coefftable, *counttable;
register Iw, Ow;

vu1(srcptr, dstptr, coefftable, counttable, width)
pixel **srcptr, **dstptr, *coefftable, *counttable;
register width;

vd1(srcptr, dstptr, coefftable, counttable, width)
pixel **srcptr, **dstptr, *coefftable, *counttable;
register width;

setmag1table(coefftable, counttable, Iw, scale, reciprocal, outputoffset)
pixel *coefftable, *counttable;
register Iw, scale, reciprocal;
register double outputoffset;

setmin1table(coefftable, counttable, Iw, reciprocal, scale, outputoffset)
pixel *coefftable, *counttable;
register Iw, reciprocal, scale;
register double outputoffset;

DESCRIPTION

The scaling routines come in three different varieties depending on how the filtering is done. Some use 4-pixel wide (cubic) filter tables; some use 2-pixel wide (linear) filter tables; some use 1-pixel wide (jaggy) filter tables. For each set, there are routines that create the filter coefficients (e.g., *setmag4table*), routines that stretch a single input scanline to a single output scanline for horizontal scaling (e.g., *hu4*), and routines that accept multiple continugous input scanlines to produce a single output scanline for vertical scaling (e.g., *vu4*).

The *hu4* routine takes an input scanline of *Iw*+6 pixels and produces an output scanline of *Ow* pixels. The *Iw* pixels of the input scanline are normally padded by 3 null pixels on each end. The *coefficient table* and *count table* must have been produced by the *setmag4table* routine. The *hu4* routine leaves these tables untouched. The value *Ow* is approximately floor[*scale**(*Iw*+3)]; the input value passed in **r1** should be the value of *Ow* returned by *setmag4table*.

NAME

chap – Pixar Chap graphics device interface

SYNOPSIS

/dev/chap*

DESCRIPTION

The *chap* interface provides access to a Pixar Chap processor (and any associated framebuffer). The device interface is actually part of the Dumi device driver, *dumi*(4), and need not be separately configured. Up to eight Chaps may be supported on a single Dumi; they are assigned minor sub-device numbers 1-8.

In normal use, a Chap device is opened and its diagnostic registers are mapped into the process's address space with an *mmap*(2) system call. The file *<pixardev/chapreg.h>* contains a definition of the registers. The registers start at logical offset 0 in the special file. In addition to the diagnostic registers, the *chap* interface supports a number of *ioctl* commands, described below.

Only one user may use a *chap* device at any time, though the user may utilize multiple processes. This locking policy is imposed at the time an *mmap* call is made to map the diagnostic registers into the process's address space. This allows unrelated processes to perform *ioctl* calls without being interfered with by the locking protocol.

A Chap is autoconfigured by probing the diagnostic registers at boot time. If the driver is successful in halting the processor (poking CSR_HALT into the csr), the driver presumes the Chap is present on the Sysbus.

The *chap* interface has two important features not commonly found in other device drivers. Chap interrupts are transformed into signals, and resource allocation requests are tracked for the instruction, scratchpad, and framebuffer memories. The latter facility may, optionally, be provided directly to programs running in a Chap.

CHAP INTERRUPTS

Chap user and breakpoint interrupts are automatically translated into signals by the driver. To enable delivery of an interrupt, the CHAPIOSSIG *ioctl* must be used. This request takes a pointer to a *chapsig* structure specifying an interrupt type (user or breakpoint), a signal to translate the interrupt into, and a process id or process group to which the signal should be delivered (a process group is specified with a negative value). When the specified interrupt is received from the Chap, the associated signal is delivered to the process id/group. The previous state for the interrupt is returned by the *ioctl* call. A zero process id or signal may be used to disable delivery of an interrupt. Signal delivery is automatically revoked when noone is using the Chap.

The CHAPIOGSIG *ioctl* returns the current state of the interrupt specified in the passed *chapsig* structure.

The *chapsig* structure and associated definitions are found in *<pixardev/chapioctl.h>*.

MEMORY MANAGEMENT

The *chap* interface implements a first-fit memory management structure for Chap instruction, scratchpad, and framebuffer memories. Programs executing on the host or Chap may allocate or free memory, or request an allocation at a specific address.

Prior to any allocation requests, the resource maps must be allocated and initialized with a CHAPIOSCONF request. This *ioctl* call takes a pointer to a *chapconf* structure which, among other things, contains the number of entries to be allocated for each resource map. A program may retrieve the information stored in this structure with a CHAPIOGCONF request.

With the resource maps initialized, the following requests are available:

CHAPIOALLOC Allocate space in a resource map. Size and resource map parameters are specified in a *chapalloc* structure. Maps are identified as CALLOC_FB (framebuffer memory), CALLOC_RAM (instruction memory), or CALLOC_SPAD (scratchpad memory). Framebuffer requests are specified in *tiles* (32x32 pixel blocks of memory); instruction RAM requests are specified in *instructions* (96-bit locations); and scratchpad requests are specified in *pixels* (4 16-bit words). Similarly, addresses are in the above units.

CHAPIOFREE Return space previously allocated. The *chapalloc* structure must specify the map address, and size of the block of memory to free.

CHAPIOGET Allocate space at a specific address. The parameters are as for CHAPIOALLOC bu with an address specified as well.

CHAPRESET Reset a resource map, or maps, to their default state (everything free). An intege parameter indicates a specific map, one of CALLOC_FB, CALLOC_RAM, c CALLOC_SPAD, or, for all maps, CALLOC_ALL.

CHAPIOGETMAP Retrieve the contents of the specified resource map. The *chapmap* structure passe as a parameter specifies the map and a place in which the data should be stored.

As mentioned previously, memory allocation requests may come either from programs executing on th host, or from programs executing on a Chap. In the case of the latter, requests are submitted by placin parameters in Sysbus shared data registers and posting a user interrupt. The three requests currently su ported, and their parameters, are:

Request	Map	Address	Size
CHAP_ALLOC	sysbus<RMAP>		sysbus<RSIZE>
CHAP_FREE	sysbus<RMAP>	sysbus<RADDR>	sysbus<RSIZE>
CHAP_GET	sysbus<RMAP>	sysbus<RADDR>	sysbus<RSIZE>

All requests are placed in sysbus<RCMD>. Synchronization is effected by setting sysbus<RRESULT> to impossible value (commonly −2) prior to posting an interrupt, then waiting for the register to change valu Errors are signaled by returning a −1 value. The file *<pixar/mman.h>* contains definitions for use in impl menting the protocol as well as several macros that may be used in Chap assembly code to carry out t requests; see also *mman*(3C).

Finally, there are several *ioctl* requests related to Chap memory management facilities.

CHAPIOMMAN Enable/disable intercepting of user interrupts for interpretation as memory manag ment requests. Before the device driver will interpret any user interrupts as co mand requests, this call must be made to enable service. An integer parame should be set to a non-zero value to enable service, setting it to zero disables servic

CHAPIOCLEAR Clear the resource allocation maps of any resources allocated by programs runni on the Chap. The device driver tracks allocations from Chap programs; this call c be used to flush all such requests from one or all resource maps. The map specified as the third parameter, as in the CHAPIORESET request.

FILES

/dev/chap* Chap special files

SEE ALSO

dumi(4), chconfig(8)

DIAGNOSTICS

chap%d on dumi%d at %x%s. The specified Chap was configured. The address specified is where diagnostic registers were found in the host's address space. If the configuration was forced for diagno purposes (i.e., the device was attached even though the Chap didn't actually respond), the mess **(forced)** will be displayed.

chap: bad map arg, %d. A user interrupt from a Chap was received and interpreted as a mem management request, but the resource map specified was bogus.

%s: rmap ovflo, lost [%d,%d). A resource map overflowed as the result of an allocation request. results when a map is configured too small and/or allocation requests badly fragment the allocation n The indicated map is displayed as well as the segment which could not be placed back in the map.

segment is lost until the map is reset or the system is rebooted.

BUGS

Since the close routine gets called only on *last* close of the device, signals may be erroneously delivered to an unsuspecting process. For this reason, benign signals are highly recommended, e.g., SIGIO.

The framebuffer allocation maps should not be on a per-Chap basis, but instead on a per-framebuffer basis (when multiple Chaps share a single framebuffer); this requires more intimate knowledge of the Pixar hardware configuration than is currently possible.

NAME

dumi – Pixar Dumi device interface

SYNOPSIS

device dumi0 at mb0 csr 0xa0000 priority 2

DESCRIPTION

The *dumi* driver provides access to a Pixar Dumi device and to the associated devices on the Sysbus. Th minor device encoding specifies the devices attached to a Dumi. Minor device 0 is the Dumi itself wi each Dumi having 16 minor devices (i.e., minor devices 0-15 are on dumi0, 16-31 on dumi1, etc.). Sul devices are encoded as follows:

Minor	Device	Description	Number
0	dumi	Dumi controller	1
1-8	chap	Chap processor	8
9-12	video	video controller	4
13	mctrl	memory controller	1
15	db	disk buffer (disk window)	1

When a Dumi device is opened, its interface registers may be mapped, via virtual memory, into a us process's address space with the *mmap*(2) system call (address 0 is always the base of the device's regi ters). This allows the user process very high bandwidth to the device with no system call overhead.

The Dumi register definitions are found in the include file *<pixardev/dumireg.h>*.

The driver imposes a single-user locking policy on all devices. That is, each device may have only o user at any one time, though a user may have multiple processes sharing a device. This locking policy implemented at the time a process tries to map the device's associated interface registers into its virtu address space via *mmap*. Unfortunately, due to limitations in the design of the system code implementi *mmap*, it is not possible for a program to distinguish "device in use" errors from other potential errors c might encounter in using *mmap*.

FILES

/dev/dumi* device special files

SEE ALSO

chap(4), db(4), mctrl(4), video(4)

DIAGNOSTICS

Sub-device-specific diagnostics are described under each sub-device's manual entry.

BUGS

A user process could possibly cause infinite interrupts, bringing things to a crawl. Currently the disk bu device is not supported.

NAME

mctrl – Pixar memory controller device interface

SYNOPSIS

/dev/mctrl*

DESCRIPTION

The *mctrl* interface provides access to a Pixar memory controller. The device interface is actually a part of the Dumi device driver, *dumi*(4), and need not be separately configured. Only one memory controller is supported on a single Dumi; the minor sub-device number is 13.

In normal use an *mctrl* device is opened and its interface registers mapped, via virtual memory, into a process's address space with the *mmap*(2) system call (address 0 is always the base of the memory controller's registers). This allows the process very high bandwidth to the device, with no system call overhead.

Only one user may use an *mctrl* device at any time, though the user may utilize multiple processes. This locking policy is imposed at the time an *mmap* call is made to map the registers into the process's address space.

A memory controller is autoconfigured by probing the registers at boot time. If the driver is successful in initializing the controller (poking the MCCSR_REFX, MCCSR_REFY, MCCSR_REFRD, and MCCSR_REF16 bits into the csr), the driver presumes the memory controller is present on the Sysbus.

The memory controller register definitions are normally found in the include file *<pixardev/mctrlreg.h>*.

FILES

/dev/mctrl* memory controller device special files

SEE ALSO

dumi(4), mctrl(8)

DIAGNOSTICS

mctrl on dumi%d at %x%s. The specified memory controller was configured. The address specified is where the register bank was found in the host's address space. If the configuration was forced for diagnostic purposes (i.e., the device was attached even though the memory controller didn't actually respond), the message **(forced)** will be displayed.

NAME

video – Pixar video controller device interface

SYNOPSIS

/dev/video*

DESCRIPTION

The *video* interface provides access to a Pixar video controller. The device interface is actually a part of the Dumi device driver, *dumi*(4), and need not be separately configured. Up to four video controllers may be supported on a single Dumi; they are assigned minor sub-device numbers 9-12.

In normal use, a *video* device is opened and its interface registers are mapped, via virtual memory, into a process's address space with the *mmap*(2) system call (address 0 is always the base of the memory controller's registers). This allows the process very high bandwidth to the device with no system call overhead.

Only one user may use a *video* device at any time, though the user may utilize multiple processes. This locking policy is imposed at the time an *mmap* call is made to map the registers into the process's address space.

A video controller is autoconfigured by probing the registers at boot time. If the driver is successful in reading the controller's csr, it presumes the controller is present on the Sysbus.

The video controller register definitions are normally found in the include file *<pixardev/videoreg.h>*.

FILES

/dev/video* video controller device special files

SEE ALSO

dumi(4), video(1)

DIAGNOSTICS

video%d on dumi%d at %x%s. The specified video controller was configured. The address specified is where the register bank was found in the host's address space. If the configuration was forced for diagnostic purposes (i.e., the device was attached even though the video controller didn't actually respond), the message **(forced)** will be displayed.

NAME

chap.out – Chap assembler and link editor output

SYNOPSIS

#include <pixar/reloc.h>

DESCRIPTION

chap.out is the output file of the assembler *chas*(1) and the link editor *chld*(1). The latter makes *chap.out* if there were no errors and no unresolved external references. Layout information as given in the include file is:

```
/*
 * Header prepended to each chap.out file.
 */
struct exec {
        long      a_magic;  /* magic number */
        unsigned  a_text;   /* size of text segment */
        unsigned  a_data;   /* size of initialized data */
        unsigned  a_bss;    /* size of uninitialized data */
        unsigned  a_syms;   /* size of symbol table */
        unsigned  a_entry;  /* entry point */
        unsigned  a_trsize; /* size of text relocation */
        unsigned  a_drsize; /* size of data relocation */
};

#define  CHAPMAGIC      0420/* chap binary */
/*
 * Macros that take exec structures as arguments and tell whether
 * the file has a reasonable magic number or offsets to text | symbols | strings.
 */
#define  N_BADMAG(x) \
   (((x).a_magic)!=OMAGIC && ((x).a_magic)!=NMAGIC && ((x).a_magic)!=ZMAGIC)

#define  N_TXTOFF(x) \
        ((x).a_magic==ZMAGIC ? PAGSIZ : sizeof (struct exec))
#define N_SYMOFF(x) \
        (N_TXTOFF(x) + (x).a_text+(x).a_data + (x).a_trsize+(x).a_drsize)
#define  N_STROFF(x) \
        (N_SYMOFF(x) + (x).a_syms)
/*
 * Macros which take exec structures as arguments and tell where the
 * various pieces will be loaded.
 */
#define N_TXTADDR(x) TXTRELOC
#define N_DATADDR(x) \
        (((x).a_magic==OMAGIC)? (N_TXTADDR(x)+(x).a_text) \
        : (SEGSIZ+((N_TXTADDR(x)+(x).a_text-1) & ~SEGRND)))
#define N_BSSADDR(x)  (N_DATADDR(x)+(x).a_data)
```

The *chap.out* file has five sections: a header, the program text and data, relocation information, a symbol table and a string table (in that order). The last three may be omitted if the program was loaded with the –s option of *chld*.

In the header, the sizes of each section are given in bytes. The size of the header is not included in any of the other sizes.

When a *chap.out* file is downloaded, two logical segments are set up: the text segment and the data segment (with uninitialized data, which starts off as all 0, following initialized data). The header is not loaded with the text segment. The macros N_TXTADDR, N_DATADDR, and N_BSSADDR give the core addresses at which the text, data, and bss segments, respectively, will be loaded.

After the header in the file, the text, data, text relocation data relocation, symbol table and string table follow in that order. The text begins just after the header. The N_TXTOFF macro returns this absolute file position when given the name of an exec structure as argument. The symbol table follows all this; its position is computed by the N_SYMOFF macro. Finally, the string table immediately follows the symbol table at a position easily gotten using N_STROFF. The first 4 bytes of the string table are not used for string storage, but rather contain the size of the string table; this size INCLUDES the 4 bytes.

RELOCATION

The value of a byte in the text or data that is not a portion of a reference to an undefined external symbol is exactly that value in memory when the file is executed. If a byte in the text or data involves a reference to an undefined external symbol, as indicated by the relocation information, then the value stored in the file is an offset from the associated external symbol. When the file is processed by the link editor, and the external symbol becomes defined, the value of the symbol is added to the bytes in the file.

If relocation information is present, it amounts to eight bytes per relocatable datum as in the following structure:

```
/*
 * Format of a relocation datum.
 */
struct relocation_info {
        int         r_address;          /* address which is relocated */
        unsigned r_symbolnum:24,  /* local symbol ordinal */
                    r_pcrel:1,          /* always 0 */
                    r_length:2,         /* always 1=word */
                    r_extern:1,         /* does not include value of sym referenced */
                    :4;                 /* nothing, yet */
};
```

There is no relocation information if a_trsize+a_drsize==0. If r_extern is 0, then r_symbolnum is actually a n_type for the relocation (i.e., N_TEXT meaning relative to segment text origin.)

SYMBOL TABLE

The layout of a symbol table entry and the principal flags that distinguish symbol types are given in the include file as follows:

```
/*
 * Format of a symbol table entry.
 */
struct nlist {
        union {
            char        *n_name; /* for use when in-memory */
            long        n_strx;    /* index into file string table */
        } n_un;
        unsigned char n_type;    /* type flag, i.e., N_TEXT etc; see below */
        char             n_other;
        short            n_desc;    /* see <stab.h> */
        unsigned         n_value;   /* value of this symbol (or adb offset) */
};
#define n_hash          n_desc      /* used internally by ld */
```

```
/*
 * Simple values for n_type.
 */
#define N_UNDF    0x0     /* undefined */
#define N_ABS     0x2     /* absolute */
#define N_TEXT    0x4     /* text */
#define N_DATA    0x6     /* data */
#define N_BSS     0x8     /* bss */
#define N_QUAL    0x10    /* qualifier */
#define N_COMM    0x12    /* common (internal to chld) */
#define N_FN      0x1f    /* file name symbol */
#define N_PATCH   0x20    /* patch refs (internal to dynamic loader) */

#define N_EXT     01      /* external bit, or'ed in */
#define N_TYPE    0x1e    /* mask for all the type bits */
```

In the *chap.out* file, a symbol's n_un.n_strx field gives an index into the string table. An n_strx value of 0 indicates that no name is associated with a particular symbol table entry. The field n_un.n_name can refer to the symbol name only if the program sets this up using n_strx and appropriate data from the string table. Because of the union in the nlist declaration, it is impossible in C to statically initialize such a structure.

If a symbol's type is undefined external, and the value field is non-zero, the symbol is interpreted by the loader *chld* as the name of a common region whose size is indicated by the value of the symbol.

SEE ALSO

chas(1), chld(1), chnm(1)

NAME

chapsym – Chap runtime symbol table

SYNOPSIS

#include <pixar/chapdiag.h>

DESCRIPTION

Each Chap on a host has a file containing information about code and data currently loaded in the machine. This information, together with the symbols defined by the resident programs, constitutes the ''runtime symbol table'' for the Chap.

The symbol table files' header has the following definition:

```
/*
 * This header is present at the
 * front of all symbol table files.
 */
typedef struct symheader {
        long    sh_magic; /* magic identifier */
        u_short sh_syms;  /* number of symbols */
        u_short sh_refs;  /* number of symbol references */
        long    sh_strsize; /* size of string table */
        long    sh_pad[8]; /* for future expansion */
} SymHeader;

#define CHAPSYMMAGIC      0x030959

#define S_BADMAG(h)    ((h).sh_magic != CHAPSYMMAGIC)
#define S_SYMOFF(h)    (sizeof (SymHeader))
#define S_REFOFF(h)    (S_SYMOFF(h) + (h).sh_syms * sizeof (LoadSym))
#define S_STROFF(h)    (S_REFOFF(h) + (h).sh_refs * sizeof (SymRef))
```

Symbol table files have four sections: a header, the symbols, information about references to symbols, an a string table (in that order).

In the header, the size of the symbol section is given in *symbols*, the size of the references section is give in *references*, and the size of the string table in *bytes*. The size of the header is not included in any of th other sizes.

SYMBOLS

The layout of a symbol entry closely follows that used in Chap object files, *chap.out*(5). In particular, th principal flag values that distinguish symbol types must be identical. A symbol entry is given in the incluc file as follows:

```
/*
 * Beware, this structure must be compatible
 * with a struct nlist when on disk.
 */
typedef union loadsym {
        struct ondisk {                 /* symbols as it's stored on disk */
                long    lsu_strx;       /* index into string table */
                u_char  lsu_type;
                u_char  lsu_other;
                u_short lsu_desc;
                long    lsu_value;
                u_short lsu_nrefs;
                long    lsu_refx;       /* index into reference table */
```

```
        } lsu_ondisk;
        struct segsym {                     /* segment information */
                long    lsu_strx;           /* file name */
                u_char  lsu_type;           /* should always be N_FN */
                u_char  lsu_other;
                u_short lsu_hash;
                u_short lsu_tbase;          /* base of code segment */
                u_short lsu_tsize;          /* size of code segment */
                u_short lsu_dbase;          /* base of data segment */
                u_short lsu_dsize;          /* size of data segment */
                u_short lsu_refcnt;         /* reference count on file */
                u_short lsu_pad;            /* reserved for future expansion */
        } lsu_segsym;
        struct incore {                     /* in-core version of symbol */
                char    *lsu_name;          /* symbol name */
                u_char  lsu_type;           /* type a la struct nlist */
                u_char  lsu_other;
                u_short lsu_hash;           /* part of hash scheme */
                long    lsu_value;          /* symbol's value */
                u_short lsu_nrefs;          /* # references to symbol */
                union   symref *lsu_refs;   /* reference list */
        } lsu_incore;
} LoadSym;

/* i'm so lazy... */
#define ls_strx   lsu_ondisk.lsu_strx
#define ls_refx   lsu_ondisk.lsu_refx

#define ls_name   lsu_incore.lsu_name
#define ls_type   lsu_incore.lsu_type
#define ls_value  lsu_incore.lsu_value
#define ls_nrefs  lsu_incore.lsu_nrefs
#define ls_hash   lsu_incore.lsu_hash
#define ls_refs   lsu_incore.lsu_refs

#define ls_tbase  lsu_segsym.lsu_tbase
#define ls_tsize  lsu_segsym.lsu_tsize
#define ls_dbase  lsu_segsym.lsu_dbase
#define ls_dsize  lsu_segsym.lsu_dsize

/*
 * Simple values for ls_type.
 */
#define N_UNDF    0x0    /* undefined */
#define N_ABS     0x2    /* absolute */
#define N_TEXT    0x4    /* text */
#define N_DATA    0x6    /* data */
#define N_BSS     0x8    /* bss */
#define N_QUAL    0x10   /* qualifier */
#define N_COMM    0x12   /* common (internal to chld) */
#define N_FN      0x1f   /* file name symbol */
#define N_PATCH   0x20   /* patch refs (internal to dynamic loader) */

#define N_EXT     01     /* external bit, or'ed in */
```

```
#define N_TYPE    0x1e    /* mask for all the type bits */
```

In the symbol table file, a symbol's ls_strx field gives an index into the string table. An ls_strx value of 0 indicates that no name is associated with a particular symbol table entry. The field ls_name refers to the symbol name only if the program sets this up using ls_strx and appropriate data from the string table.

Similarly, a symbol's ls_refx field gives an index into the reference table. The ls_nrefs field specifies the number of references associated with a symbol (see below). The field ls_refs can be used to refer to the symbol's references only if the program sets this up using ls_refx and the appropriate data from the reference table.

The symbols section of the symbol table is segmented by file. The start of a file is delimited by a symbol with type N_FN. File name symbols have a special format symbol table entry containing a description of resources associated with the loaded file's text and data segments (bss is converted to data at the time the file is relocated and loaded into a Chap).

When a file is unloaded from a Chap but references still exist to symbols defined in the file, the file's symbol table "segment" is preserved to allow entries to remain for the undefined symbols. By convention, segments of this type are assigned a file name of "*unloaded*". All symbols in such a segment must be undefined. Unloaded segments are discarded from the symbol table when there are no longer references to symbols contained in the segment.

Finally, the in-core version of the symbol table may contain symbol entries with an ls_name entry of 0 These entries are the result of symbols which have been deleted but can not be purged until the symbol table is written to disk and read back in again; they should be ignored.

REFERENCE LISTS

Each reference to an external symbol in a file processed by the dynamic loader results in a *reference* entry in the symbol table. References are tracked to allow segments of a program to be loaded piece-wise, with each new symbol's definition/deletion resulting in validation/invalidation of references to the symbol.

A reference list is a segmented linked-list of reference entries, each of which is described as:

```
/*
 * Symbol references are kept in the
 * symbol table as a list of segments
 * (for dynamic expansion).
 */
#define NREFSEG     8               /* must be pow2 */
#define NREFMASK  (NREFSEG-1)

typedef union symref {
        struct {
                u_char  sru_seg;            /* segment */
                u_char  sru_pad;
                u_short sru_loc;            /* location in spad/iram of reference */
        } sru;
        union {
                union   symref *sru_next;  /* next block of references */
                off_t   sru_refx;
        } srun;
} SymRef;

#define sr_seg  sru.sru_seg
#define sr_loc  sru.sru_loc
#define sr_pad  sru.sru_pad
#define sr_next srun.sru_next
```

```
#define sr_refx srun.sru_refx
```

A symbol table entry points to a linked-list of reference blocks, each NREFSEG in length. For each reference block, the last entry is always reserved for a pointer to another block of references. The reference itself specifies the segment, sr_seg, in which the reference resides and the location in scratchpad or instruction memory (in words for scratchpad, instructions for instruction memory).

SEE ALSO

charm(1), ChapOpen(3H), ChapLoad(3H), chap.out(5)

BUGS

The segmented nature of the symbol table is unwieldy when it comes to deleting symbols and/or expanding segments.

NAME

fbdefs,
lfbdefs,
fbpath,
lfbpath – framebuffer description definitions

SYNOPSIS

setenv FBDEFS "fbdef:fbdef:..."
setenv LFBDEFS "lfbdef:lfbdef:..."
setenv FBPATH "fbname:fbname:..."
setenv LFBPATH "lfbname:lfbname:..."

DESCRIPTION

Physical frame buffer definitions (*fbdefs*) and logical frame buffer definitions (*lfbdefs*) are definitions that assign names to rectangular areas of Pixar image memory. An *fbdef* describes a *tile block* of picture memory to be accessed by a particular chap. An *lfbdef* describes a logical window mapping, or *pixel window*, into a tile block.

The environment variables FBPATH and LFBPATH are colon-separated lists of names of *fbdefs* and *lfbdefs*, respectively. These lists define reference orderings of physical and logical framebuffers. The first name in the FBPATH may be referred to as FB0, the second as FB1, etc. Names in the LFBPATH may be referenced as LFB0, LFB1, etc. The names defined in FBDEFS are appended to the FBPATH in the order in which they were defined, followed by a default fbdef. The LFBPATH is appended with the names in LFBDEFS, followed by the names in the FBPATH.

An *fbdef* has the following syntax:

name= [sizex sizey [starttile [device]]]

where:

name is the name of the framebuffer being defined
sizex is the width in pixels of the framebuffer
sizey is the height in pixels of the framebuffer
starttile is the first tile to use for the tile block
device is the name of the pixar/chap to use. Devices are referenced with the prefix **pxr** followed by a number **N. N** div 8 defines which pixar card cage to use. **N** mod 8 defines which chap to use. (*ex: pxr17 refers to chap 1 in card cage 2*)

Any optional arguments not defined are given the values of the default fbdef. The *default fbdef* is: pxr0 = 1024 4096 0 pxr0.

An *lfbdef* has the following syntax:

[[name=] fbname] [xmin xmax ymin ymax]

where:

name is the name of the logical framebuffer being defined

fbname is the name of the fbdef, the lfbdef is being defined

within. The fbname may be a previously defined fbdef or a device name such as *pxr2*. The latter type of name uses a default fbdef on the specified device.

xmin,xmax, *ymin,ymax,* is the window within the physical framebuffer to use as a logical framebuffer. If this is the only argument given for the lfbdef, then fbdef **FB0** is used as the fbdef.

SEE ALSO

PW(3C), TB(3C), FbGetDef(3H)

```
setenv CHAPLIBPATH "lib1:lib2:lib3:..."
```

NAME

Diagnostic – Pixar system diagnostc check

SYNOPSIS

/usr/pixar/diag/bin/Diagnostic

DESCRIPTION

Diagnostic provides a menu driven interface to the PIXAR hardware test programs. The menu allows the operator to test major sections of the PIXAR Image Computer to the board level. Options also exist to report the PIXAR device configuration, and support the testing of multiple PIXAR Image Computers attached to a single host.

The *Diagnostic* program attempts to determine the PIXAR Image Computer configuration, initializes all sections of the computer, and then tests the selected subsystems of the computer. Normally, a single pass test of the complete system (Selection 1) takes about 15 minutes. If failures occur, board level failure reports are printed to the operator's terminal.

FILES

/usr/pixar/diag/bin/*
/usr/pixar/diag/ucode/*

SEE ALSO

chap(4)

NAME

chconfig – Chap configuration tool

SYNOPSIS

chconfig [**–i** *iram-size*] [**–s** *spad-size*] [**–f** *fb-size*] [**–k** *stack-depth*] [**–c** *component-width*] [**–I** *imap-size*] [**–S** *smap-size*] [**–F** *fmap-size*] [**–a**] [device]

DESCRIPTION

chconfig is used to set or view the system's idea of the hardware configuration of a Chap. This configuration information is maintained by the system and may be interrogated by programs so it may be written in a hardware-independent fashion. *chconfig* is normally run at boot time for each Chap on a system from the file "/etc/rc.local" with the **–a** flag. This causes *chconfig* to "autoconfigure" the Chap's characteristics by running various tests intended to deduce the appropriate values for memory size, stack depth, etc.

Other options to *chconfig* allow individual parameters to be set. If any of these parameters are set in conjunction with the **–a** flag, *chconfig* will use the specified parameters instead of the values it would normally use. The **–i**, **–s**, and **–f** options set the size of *instruction RAM*, *scratchpad memory*, and the associated *framebuffer*, respectively. Each size parameter is expressed in the units of the appropriate resource: instructions for instruction RAM, 4 word pixels for scratchpad memory, and 32x32 word tile blocks for framebuffer memory. The Chap's stack depth and the framebuffer's component width may be specified with the **–k** and **–c** flags (component width is specified in bits).

In addition to the basic hardware characteristics, *chconfig* also defines the size of the resource allocation maps used by the system to keep track of memory allocation in each Chap and framebuffer. Three maps exist for each Chap-framebuffer pair: instruction RAM, scratchpad memory, and framebuffer memory. The size of each allocation map defines the number of hunks of non-contiguous free memory available at any one time. In normal operation, the maps are constantly being compacted, so this value normally reflects the maximum "fragmentation" allowed. If the map is too small to keep track of all the free memory for a particular resource, it discards part of the available memory. To reclaim the lost memory, the allocation maps must be reset. The **–I**, **–S**, and **–F** flags specify the size of the allocation maps for instruction RAM, scratchpad memory, and framebuffer memory, respectively. If the map sizes are not manually specified, *chconfig* allocates 250 entries to each map.

The default device for *chconfig* is /dev/chap0; this may be changed by specifying a trailing device name on the command line.

If *chconfig* is invoked without options, it prints the current configuration.

NOTES

In calculating the size of the framebuffer, the code loaded overwrites the first 32 pixels of each tile block.

FILES

/dev/chap0	default Chap device
/usr/pixar/chap/bin/config.ucode	Chap code used for autoconfiguration

SEE ALSO

chap(4G)

NAME

mctrl – set/clear options of a memory controller

SYNOPSIS

mctrl [*command*] ...

DESCRIPTION

mctrl is a simple program used to peek and poke registers on the memory controller. If no arguments are given, *mctrl* prints the contents of the control status register. Arguments are interpreted as commands and processed one at a time as follows:

halt, –halt
Set/clear the halt bit in the csr.

aoen, –aoen
Set/clear the address output enable bit in the csr.

refen, –refen
Set/clear the refresh enable bit in the csr.

refinh, –refinh
Clear/set the refresh enable bit in the csr.

refx, –refx
Set/clear the refresh X access bit in the csr.

refy, –refy
Set/clear the refresh Y access bit in the csr.

refrd, –refrd
Set/clear the refresh read bit in the csr.

refwr, –refwr
Clear/set the refresh read bit in the csr.

ref16, –ref16
Set/clear the refresh 16 bit in the csr.

ref32, –ref32
Clear/set the refresh 16 bit in the csr.

The remaining arguments are treated either as *peek* or *poke* requests for memory controller registers, or as a bit definition for a register (where appropriate). A peek is specified by a register name alone; a poke by a register name followed by a hexadecimal value. To use a bit definition, the register name must precede the name of the bit (see the usage message for more information). The register names are:

csr	control status register
mips	MIPS meter register
step	single step register
req	iobus request register
addr0	iobus address (low) register
addr1	iobus address (high) register
addr2	iobus device select register
addr_load	iobus readback load register
req_a0	iobus grant pal (low) register
req_a1	iobus grant pal (high) register
req_a2	iobus grant enable register
res0	reservation table 0 register
res1	reservation table 1 register
res2	reservation table 2 register
res3	reservation table 3 register
vbus0	Vbus sequencer 0 register

vbus0	Vbus sequencer 1 register
pbus0	Pbus sequencer 0 register
pbus0	Pbus sequencer 1 register
mem0	memory address 0 register
mem1	memory address 1 register
mem2	memory address 2 register

mctrl catches faults generated by peeks and pokes on the Sysbus and prints the message ''Bus error''.

SEE ALSO

dumi(4), mctrl(4)

1. Overview

(1) This document describes the installation of PIXAR software on SUN Unix version 3.2. Refer to the SUN manuals entitled "Installing UNIX" and "Release 3.2 Manual", for details of installing Unix on a new system or upgrading an old system.

(2) If you are upgrading an old system, make a set of level zero dump tapes of all filesystems before beginning the install. These tapes will provide a safety net in the event of a serious problem.

(3) Where additional detail is desired additional documentation for the non-PIXAR postion of the install is available in the SUN documents entitled "System Administration", and "Writing Device Drivers". The second document will be especially valuable if non-standard device configurations are required.

(4) SUN release 3.2 is being included with PIXAR software release 1.2. New systems are shipped with the normal set of SUN manuals, older systems are shipped with the SUN Release 3.2 manual.

(5) PIXAR recommendeds that installations running older SUN releases convert immediately to 3.2, before installing PIXAR release 1.2. However, should you temporarily decide to install PIXAR release 1.2 first and convert to SUN 3.2 at a later date, see the cautionary notes at the end of this document under "Older SUN Releases".

2. Setup Prerequisites

(1) The Sun system must report

Self Test Completed successfully.

when system power is first turned on and the monitor quick self-test is run.

(2) The disk formatting operation must have been completed correctly with no uncorrectable errors. If the disk is already formatted and is not to be re-formatted, the disk test must be run. If the system is equipped with a Xylogics 450 or Xylogics 451 disk controller, the dmatest must be run for 10 minutes. No failures should be detected.

The diag program is booted from the monitor using the command line:

```
> b stand/diag
```

Refer to DIAG(8S) in the Maintenance Commands section of the System Internals Manual for the Sun UNIX System.

(3) The disk should be built up with the current version of Sun Release 3.2 software, from the Sun Operating System installation tapes. See SUN's document entitled "Installing Unix".

(4) The system should be equipped with a 1/4" streaming cartridge tape drive, or have network access to a system with such a tape drive.

3. General Description

The software release installation procedure consists of three general parts.

(1) Software loading from tapes onto the disk.

(2) Software installation.

(3) Software turn-on and operational checks.

4. Software Loading

The PIXAR Software Release 1.2 is supplied on several magnetic tapes. The following description of the first two tapes is subject to change without notice. Tape 1 contains the Chap and host libraries, host utility and applications programs, manual pages, and the tutorial programs. This tape requires at least 13 Megabytes of disk space. Tape 2 contains the demonstration programs, and requires 11 Megabytes of disk space.

4.1. Cartridge Tape Drive

If the system is equipped with a cartridge tape drive, software can be loaded onto disk as follows:

(1) Put tape 1 of 2 into the tape drive.

(2) Log in on the Sun Workstation as root.

(3) At the UNIX command prompt, '# ', type the following command line:

```
tar pxvfb /dev/rst0 256
```

(4) Once the tape has been read in, remove the tape from the tape drive.

(5) Put tape 2 of 2 into the tape drive.

(6) At the UNIX command prompt, '# ', type the following command line:

```
tar pxvfb /dev/rst0 256
```

(7) Once the tape has been read in, remove the tape from the tape drive.

If software is to be loaded over the net, using a remote system's tape drive, proceed as follows:

(1) Put tape 1 of 2 into the tape drive of the remote system. This system will be referred to as host 'remote' throughout this procedure.

(2) Log in on the Sun Workstation on which the software is to be installed as root.

(3) At the UNIX command prompt, '# ', type the following command line:

```
rsh remote dd if=/dev/rst0 bs=128k | tar pxvfb - 256
```

This command line tells the system named 'remote' to read the tape in 128 Kbyte blocks, and sent the output over the net to the 'tar' program. The '-' argument to tar tells tar to read it's input from the pipe instead of a tape.

(4) Once the tape has been read in, remove the tape from the tape drive.

(5) Put tape 2 of 2 into the tape drive of the remote system.

(6) At the UNIX command prompt, '# ', type the following command line:

```
rsh remote dd if=/dev/rst0 bs=128k | tar pxvfb - 256
```

(7) Once the tape has been read in, remove the tape from the tape drive.

4.2. 1/2 Inch Magtape Drives

The 1/2 " tapes supplied are recorded at 1600 BPI using the 'tar' program and the raw magtape interface. The tape blocking factor is 20, resulting in 10 Kilobyte tape blocks.

If the system is equipped with a 1/2 " magtape drive, software can be loaded onto disk as follows:

(1) Mount tape 1 of 2 on the tape drive.

(2) Log in on the Sun Workstation as root.

(3) At the UNIX command prompt, '# ', type the following command line:

```
tar pxvfb /dev/rmt8 20
```

(4) Once the tape has been read in, rewind and remove the tape from the tape drive.

(5) Put tape 2 of 2 into the tape drive.

(6) At the UNIX command prompt, '# ', type the following command line:

```
tar pxvfb /dev/rmt8 20
```

(7) Once the tape has been read in, rewind and remove the tape from the tape drive.

If software is to be loaded over the net, using a remote system's tape drive, proceed as follows:

(1) Mount tape 1 of 2 on the tape drive of the remote system. This system will be referred to as host 'remote' throughout this procedure.

(2) Log in on the Sun Workstation on which the software is to be installed as root.

(3) At the UNIX command prompt, '# ', type the following command line:

```
rsh remote dd if=/dev/rmt8 bs=10k | tar pxvfb - 20
```

This command line tells the system named 'remote' to read the tape in 10 Kbyte blocks, and sent the output over the net to the 'tar' program. The '-' argument to tar tells tar to read it's input from the pipe instead of a tape.

(4) Once the tape has been read in, rewind and remove the tape from the tape drive.

(5) Mount tape 2 of 2 on the tape drive of the remote system.

(6) At the UNIX command prompt, '# ', type the following command line:

```
rsh remote dd if=/dev/rmt8 bs=10k | tar pxvfb - 20
```

(7) Once the tape has been read in, rewind and remove the tape from the tape drive.

5. Software Installation

Now that the software has been loaded onto the Sun Workstation disk, it must be set up for use. This setup consists of making a backup of the PIXAR software, installing a kernel with the PIXAR device drivers, setting up the device files, and adding a few commands to the /etc/rc.local file to initialize the PIXAR computer cage when the host is started.

The general installation procedure is as follows:

(1) Make a backup tape using the customer's tape drive, as in the section "Make a Backup".

(2) Determine if the default PIXAR UNIX kernel can be used. If the customer does not currently have a customized UNIX kernel, the default PIXAR UNIX kernel can be used. Proceed as in the section "Default Installation".

(3) If a custom kernel is needed, proceed as in the section "Custom Kernel Installation". If the customer is adding his own device drivers to the SUN UNIX kernel, a custom kernel must be made.

(4) If more than one DUMI interface and PIXAR Image Computer are to be connected to the host, do the default or custom installation as appropriate, and then add the additional device entries as described in the section "PIXAR Device Installation".

(5) If multiple Chap boards or video boards are installed in the customer's PIXAR Image Computer, do the default or custom installation as appropriate, and then add the additional device entries as described in the section "PIXAR Device Installation".

5.1. Make a Backup

Make a backup tape using the customer's Sun Workstation tape drive. Using the customer's tape drive minimizes the chance of making an unreadable backup tape due to mechanical and electrical variations from one tape drive to another. The backup operation will take about 30 minutes.

The customer may also want to perform an incremental dump of the /usr filesystem at this time.

5.1.1. Cartridge Tape Drives

If the system is equipped with a cartridge tape drive, software can be backed up to tape as follows:

(1) Put the tape into the tape drive. Note that the tape should be 450 feet or longer and rated at 10,000 FTPI (8000 BPI). The backup will require about 24 Megabytes of tape storage.

(2) Log in on the Sun Workstation as root.

(3) At the UNIX command prompt, '# ', type the following command line:

```
tar crvfb /dev/rst0 256 /usr/pixar
```

This tape may be used to load the PIXAR software on to the system at a later date.

If software is to be backed up over the net, using a remote system's tape drive, proceed as follows:

(1) Put the tape into the tape drive of the remote system. This system will be referred to as host 'remote' throughout this procedure.

(2) Log in on the Sun Workstation on which the software is to be backed up as root.

(3) At the UNIX command prompt, '# ', type the following command line:

```
tar crfb - 256 /usr/pixar | rsh remote dd of=/dev/rst0 bs=128k
```

5.1.2. 1/2 Inch Magtape Drives

If the system is equipped with a 1/2 " magtape drive, software can be backed up to tape as follows:

(1) Mount a blank tape on the tape drive. The tape will be written at 1600 BPI with a blocking factor of 20. About 24 Megabytes will be written to the tape. A 2400 foot tape reel is recommended.

(2) Log in on the Sun Workstation as root.

(3) At the UNIX command prompt, '# ', type the following command line:

```
tar crvfb /dev/rmt8 20 /usr/pixar
```

This tape may be used to load the PIXAR software on to the system at a later date.

If software is to be backed up over the net, using a remote system's tape drive, proceed as follows:

(1) Mount the tape on the tape drive of the remote system. This system will be referred to as host 'remote' throughout this procedure.

(2) Log in on the Sun Workstation on which the software is to be backed up as root.

(3) At the UNIX command prompt, '# ', type the following command line:

```
tar crfb - 20 /usr/pixar | rsh remote dd of=/dev/rmt8 bs=10k
```

5.2. Default Installation

A default system may be configured by running the shell script

```
/usr/pixar/sys/DEFAULT
```

The default system consists of a Sun 3 computer, one DUMI interface, and one PIXAR Image Computer. A modem may be attached to Serial Port B on the Sun 3 computer.

The default device configuration supports up to three Chaps and two video boards in the PIXAR card cage. Refer to "PIXAR Device Installation" for information on adding additional device entries.

5.3. Custom Kernel Installation

A custom kernel is only needed if the customer is not using the Sun 3 GENERIC kernel. The default PIXAR kernel at /usr/pixar/sys/vmunix is the Sun 3 GENERIC kernel with the PIXAR device driver added. This default PIXAR kernel is capable of supporting multiple PIXAR Image Computers and DUMI interface boards. Refer to "PIXAR Device Installation" for details on adding additional device entries.

Follow these steps to build a custom UNIX kernel for a Sun 3 Workstation. Please be careful.

(1) cd /sys/OBJ

(2) Copy the PIXAR device driver object module to /sys/OBJ:

```
cp /usr/pixar/sys/pixar.obj /sys/OBJ/pixar.o
```

(3) cd ../conf

(4) Edit "files.sun3" and append the following line to the file:

```
pixardev/pixar.c        optional dumi device-driver
```

The file doesn't actually exist, but this makes config write a makefile that uses /sys/OBJ/pixar.o.

(5) List the files in this directory and look at the various configurations files (the ones in all capital letters). Decide (or ask) if the customer has already customized a configuration. If not, "cp GENERIC PIXAR" and "mkdir ../PIXAR". Now edit the appropriate configuration file (PIXAR or the customer's file) and append this line:

For Multibus Suns (Sun 2):

```
device  dumi0 at mb0 csr 0xa0000 priority 2
```

For VME Suns (Sun 3):

```
device  dumi0 at vme24d16 ? csr 0xea0000 priority 2 vector dumiintr
```

(6) Configure the custom kernel by typing the command:

```
/etc/config PIXAR
```

If using a configuration file other than PIXAR, use that file as the argument to config.

(7) cd ../sun

(8) Edit "conf.c" and add the following lines just before the line that says

```
extern  int     ttselect(),   seltrue();
```

This code is duplicated in /usr/pixar/sys/pixar_conf.c

```
#include "dumi.h"
#if NDUMI > 0
int     dumiopen(), dumiclose(), dumiioctl(), dumimmap();
#else
#define dumiopen        nodev
#define dumiclose       nodev
#define dumiioctl       nodev
#define dumimmap        nodev
#endif
```

Add the following lines after a similar set of lines that define device number 34:

```
{
dumiopen,       dumiclose,      nodev,          nodev,          /*35
dumiioctl,      nodev,          nulldev,        0,
seltrue,        dumimmap,
},
```

CAUTION

These lines are position dependent and must really come after the 34th entry in the file. The numbered comment (e.g. "/*35*/") serves as a reminder of this. If the customer already has a device at position 35 place the lines for the dumi at the end of the list and change the comment to be the next higher number. Remember this number, it will be needed as a parameter to /usr/pixar/sys/MAKEDEV when making the devices in "/dev" later on.

(9) Move to the configuration directory that was set up earlier.

```
cd ../PIXAR
```

(10) Unfortunately, some hand copying of include files is also required. First, copy all the files from /usr/pixar/include/pixardev into /sys/pixardev. This provides the bulk of the include files needed for the kernel. In addition, create /usr/include/pixar as a symbolic link to /usr/pixar/include/pixar. The reason for this is buried in history, and later releases will not require this symbolic link.

(11) Type the command 'make'. The kernel will now be compiled and loaded.

(12) Make the devices in /dev by using the following commands. If the major device number is to be 35, just type:

```
/usr/pixar/sys/MAKEDEV
```

If the major device number is something other than 35, just type:

```
/usr/pixar/sys/MAKEDEV XXX
```

Where XXX is the desired major device number.

(13) Install the new kernel in the root directory, by entering the following commands. If the customer's configuration file name is something other than PIXAR, use that name instead of PIXAR as the /sys subdirectory.

```
cd /
mv vmunix vmunix.works
mv /sys/PIXAR/vmunix /
```

(14) To initialize the PIXAR system automatically on a reboot, add the following two lines to /etc/rc.local:

```
/usr/pixar/host/bin/chconfig -a -k 32 /dev/chap0
/usr/pixar/host/bin/video -file /dev/video0 -version 1
```

These two lines are also found in /usr/pixar/sys/rc.local for your convenience.

(15) "fastboot" the system. If all goes well you should see the following:

```
Multibus systems:
dumi0 at mbmem a0000 pri 2
chap0 on dumi0 at a0800
video0 on dumi0 at a4800
mctrl on dumi0 at a6800
dw on dumi0 at a8000

VME systems:
dumi0 at vme24d16 ea0000 vec 0xc8
chap0 on dumi0 at ffea0800
video0 on dumi0 at ffea4800
mctrl on dumi0 at ffea6800
```

If the PIXAR isn't turned on you won't see the entries for chap0 and video0.

(16) If the new kernel fails to boot, get back to the monitor, using the <PF1><A> key combination. Boot the good kernel that you saved as /vmunix.works from the monitor prompt.

```
> b /vmunix.works
```

Carefully review the configuration changes that were made. Make sure that no errors are reported when configuring or making the new kernel, and try again.

5.4. PIXAR Device Installation

The PIXAR device driver supports multiple PIXAR Image Computers, as well as multiple Chap and video boards in each computer. To use all of this hardware, device entries must be made in the /dev directory. For a system with a single PIXAR Image Computer, these /dev/entries are created by running /usr/pixar/sys/MAKEDEV. If your system has more than one PIXAR Image Computer, read the following paragraphs, which explain the PIXAR naming conventions needed for using

/dev/dumi0	0	0	DUMI Interface Board
/dev/chap0	0	1	Chap 0
/dev/chap1	0	2	Chap 1
/dev/chap2	0	3	Chap 2
/dev/video0	0	9	Video Board 0
/dev/video1	0	10	Video Board 1
/dev/mctrl0	0	13	Memory Controller
/dev/diskw0	0	15	Disk Window
/dev/dumi8	1	16	DUMI Interface Board
/dev/chap8	1	17	Chap 0
/dev/chap9	1	18	Chap 1
/dev/chap10	1	19	Chap 2
/dev/video8	1	25	Video Board 0
/dev/video9	1	26	Video Board 1
/dev/mctrl8	1	29	Memory Controller
/dev/diskw8	1	31	Disk Window
/dev/dumi16	2	32	DUMI Interface Board
/dev/chap16	2	33	Chap 0
/dev/chap17	2	34	Chap 1
/dev/chap18	2	35	Chap 2
/dev/video16	2	41	Video Board 0
/dev/video17	2	42	Video Board 1
/dev/mctrl16	2	45	Memory Controller
/dev/diskw16	2	47	Disk Window

Table 1. Device naming conventions.

the the UNIX mknod command.

A device naming convention exists which should be followed closely, in order to avoid problems with future equipment and support. The device name consists of a terse descriptive name followed by a number.

The number following the device name indicates which PIXAR Image Computer contains that device. Blocks of 8 numbers are assigned to each computer. Thus, on a host system with multiple PIXAR Image Computers, /dev/chap0 refers to the first Chap in the first PIXAR, and /dev/chap8 refers to the first Chap in the second PIXAR.

The minor device number is used as an argument to the 'mknod' UNIX command. This number identifies the device to the operating system. Note that the shell script /usr/pixar/sys/MAKEDEV will set up entries for one PIXAR Image Computer with one Chap and one Video Board. All other devices must be made using 'mknod'.

Mknod makes a special file. The first argument is the name of the entry. The second argument is 'c' for all PIXAR devices. The last two arguments specify the major device type (usually 35) and the minor device.

```
/etc/mknod name c major minor
```

If a custom kernel was made, use the major device number that the PIXAR device driver was placed at. Refer to Table 1 for the minor device number.

6. Software Turn-On

Once the software has been installed in a system, the complete package should be verified to be operational. This verification consists of running the diagnostics and the demo programs on a system which has just been powered up.

(1) The system should initially be shut down, with no power applied.

(2) Apply power to the Sun Workstation and the PIXAR Image Computer. Start up the Sun Workstation running UNIX in multi-user mode.

(3) Log in to the Sun Workstation.

(4) Type the command:

```
/usr/pixar/host/bin/pixinit
```

Note that on a multi-PIXAR system, the commands in the

pixinit shell script will be run individually for each PIXAR attached to the host system.

(5) Type the command:

```
/usr/pixar/diag/bin/Diagnostic
```

(6) Select Option 1 from the menu, and press <RETURN>. The diagnostics should run without any errors and return to the menu.

(7) Type the following commands:

```
cd /usr/pixar/demo
Demo
```

This runs the Demo program. The Demo program will not work under suntools.

(8) Run each of the demo programs. No error messages should occur.

Note that the message 'silo overflow' may appear when starting the FFT Demo. This is harmless, and is usually due to mouse motion having occurred before a program was ready to use the mouse. Exit the FFT demo by clicking the left and right mouse buttons twice.

(9) That's it! If no problems have occurred, the software installation is done.

7. Installation Notes

This section contains assorted installation notes on how to take care of software problems that may occur in setting up the system at a customer's site.

7.1. Ethernet On/Off

If no Ethernet service is to be supplied at the customer site, the Sun Workstation should be told that the network is unavailable. If this is not done, cryptic error messages about device 'ie0' will keep appearing on the console.

Add the following command line to the file /etc/rc.local:

```
/etc/ifconfig ie0 down
```

This will tell the Sun Operating System that the network is not to be used, every time the system is rebooted. Refer to

the Sun Programmer's Manual IFCONFIG(8C) for details.

7.2. Second Ethernet Controller

SUN VME systems with a second ethernet controller will have an address overlap problem with the dumi hardware registers. This requires a different switch setting for the VME adaptor and a change to the config file. Contact PIXAR customer support if your system has a second ethernet controller.

7.3. Modem Support

A shell script is supplied to turn on hardware carrier control for a Sun 3 Serial Port B with attached Hayes modem. This allows automatic dialout using a Hayes modem and the 'tip' program, and supports the Hayes auto-answer function for dialup lines.

The DEFAULT shell script automatically sets up modem operation and related support for tip and uucp. Manual installation is described below.

7.3.1. Software Installation

If the PIXAR software is installed without using the DEFAULT shell script, the modem software setup may be done as follows:

(1) The PIXAR Release 1.2 software must be loaded and installed before proceeding.

(2) Log in as root. Change directories to /usr/pixar/sys

```
# cd /usr/pixar/sys
```

(3) Type the command 'uu_install'.

```
# uu_install
```

(4) Type the command 'modem on'.

```
# modem on
```

(5) Re-boot the system, using 'fastboot'

The 'uu_install' shell script has installed UUCP support, and a working /etc/remote file. The old /etc/remote file is saved at '/etc/remote.old'. The 'modem' script has patched /vmunix to enable modem control on Serial Port B.

8. Hardware Installation

Hardware installation consists of connecting the modem to Serial Port B, and setting up the modem switches.

(1) Plug the RS-232C cable into Serial Port B on the back of the Sun CPU board.

(2) Connect the other end of the cable to the back of the Hayes Smartmodem 1200.

(3) Remove the end cap from the front of the modem.

(4) Check the switch settings and set as follows:

1	Down	Software DTR
2	Down	Numeric result codes.
3	Down	Result codes displayed.
4	Down	Do not echo commands.
5	Up	Auto-answer on.
6	Up	Automatic Carrier Detect.
7	Up	Setting for RJ11 single-line phone.
7	Down	Setting for RJ12 or RJ13 multi-line phone.
8	Down	Enable Command Recognition.
9	Up	Bell-212 Compatibility.
10	Up	Bell-212 Compatibility.

(5) Plug in the modem and turn on power.

(6) Test the system by using 'tip' to dial a local computer system. The /etc/remote file is configured to dial PIXAR automatically, as follows:

```
% tip Pixar
% tip pixar
```

8.1. Older SUN Releases

SUN release 3.2 is included with PIXAR version 1.2 and PIXAR recommends that you install SUN 3.2 prior to installing PIXAR 1.2. If this is not possible, use the files with suffix "_3.0" in /usr/pixar/sys in place of the normal ones.

For PIXAR device driver:

```
./DEFAULT_3.0
./MAKEDEV_3.0
./pixar_conf.c_3.0
./rc.local_3.0
./vmunix_3.0
./tablet_3.0
```

For PIXAR tablet line discipline:

```
./tablet/vmunix_3.0
./tablet/tty_conf.o_3.0
./tablet/tty_tb.o_3.0
```

merrell@flywheel

tape1.list

Mon Jul 13 08:50:21 1987

lw / Fluoride

```
Fluoride   flywheel:merrell   Job: tape1.list   Date: Mon Jul 13 08:50:21 1987
Fluoride   flywheel:merrell   Job: tape1.list   Date: Mon Jul 13 08:50:21 1987
Fluoride   flywheel:merrell   Job: tape1.list   Date: Mon Jul 13 08:50:21 1987
Fluoride   flywheel:merrell   Job: tape1.list   Date: Mon Jul 13 08:50:21 1987
Fluoride   flywheel:merrell   Job: tape1.list   Date: Mon Jul 13 08:50:21 1987
```

```
rwxr-xr-x  0/10      0 Dec  4 17:58 1986 /usr/pixar/
rw-r--r--  0/10     41 Dec  2 16:47 1986 /usr/pixar/Version
r--r--r--  0/10   1719 Nov 26 18:09 1986 /usr/pixar/Makefile
rwxrwxrwx  0/10      0 Dec  4 16:57 1986 /usr/pixar/bin/
rwxr-xr-x  0/10  25600 Nov 24 17:25 1986 /usr/pixar/bin/dep
rwxr-xr-x  0/10    886 Nov 24 17:25 1986 /usr/pixar/bin/depend
rwxr-xr-x  0/10      0 Dec  4 16:57 1986 /usr/pixar/chap/
r--r--r--  0/10   1756 Nov 24 17:23 1986 /usr/pixar/chap/Makefile
rwxrwxrwx  0/10      0 Dec  4 16:57 1986 /usr/pixar/chap/bin/
r--r--r--  0/7   2769 Dec  2 21:13 1986 /usr/pixar/chap/bin/config.ucode
rwxrwxrwx  0/10      0 Dec  4 16:57 1986 /usr/pixar/chap/lib/
rw-r--r--  0/7  16830 Dec  2 21:03 1986 /usr/pixar/chap/lib/libchad.a
rw-r--r--  0/7  13040 Dec  2 21:03 1986 /usr/pixar/chap/lib/libcolor.a
rw-r--r--  0/7   4368 Dec  2 21:04 1986 /usr/pixar/chap/lib/libpG.a
rw-r--r--  0/7  36536 Dec  2 21:05 1986 /usr/pixar/chap/lib/libpip.a
rw-r--r--  0/7 135968 Dec  3 14:33 1986 /usr/pixar/chap/lib/libpt.a
r--r--r--  0/7  24290 Dec  2 21:10 1986 /usr/pixar/chap/lib/libpicio.a
rw-r--r--  0/7  52590 Dec  2 21:12 1986 /usr/pixar/chap/lib/libpx.a
rw-r--r--  0/7  23606 Dec  2 21:13 1986 /usr/pixar/chap/lib/libpm.a
rwxr-xr-x  0/10      0 Dec  4 16:59 1986 /usr/pixar/diag/
r--r--r--  0/10   1690 Nov 24 17:22 1986 /usr/pixar/diag/Makefile
rwxrwxrwx  0/10      0 Dec  4 16:59 1986 /usr/pixar/diag/bin/
rwxr-xr-x  0/10 327680 Dec  2 20:46 1986 /usr/pixar/diag/bin/fbtest
rwxr-xr-x  0/10 327680 Dec  2 20:47 1986 /usr/pixar/diag/bin/pcmtest
rwxr-xr-x  0/10 327680 Dec  2 20:47 1986 /usr/pixar/diag/bin/spadtest
rwxr-xr-x  0/10 335872 Dec  2 20:48 1986 /usr/pixar/diag/bin/iramtest
rwxr-xr-x  0/10 311296 Dec  2 20:48 1986 /usr/pixar/diag/bin/fbex
rwxr-xr-x  0/10  57344 Dec  2 20:49 1986 /usr/pixar/diag/bin/poke
rwxr-xr-x  0/10  90112 Dec  2 20:49 1986 /usr/pixar/diag/bin/mvideo
rwxr-xr-x  0/10 376832 Dec  2 20:51 1986 /usr/pixar/diag/bin/chaptest
rwxr-xr-x  0/10 196608 Dec  2 20:52 1986 /usr/pixar/diag/bin/twinkle
rwxr-xr-x  0/10 212992 Dec  2 20:53 1986 /usr/pixar/diag/bin/bt
rwxr-xr-x  0/10 204800 Dec  2 20:54 1986 /usr/pixar/diag/bin/lt
rwxr-xr-x  0/10 204800 Dec  2 20:55 1986 /usr/pixar/diag/bin/cm
rwxr-xr-x  0/10 221184 Dec  2 20:56 1986 /usr/pixar/diag/bin/lb
rwxr-xr-x  0/10 221184 Dec  2 20:57 1986 /usr/pixar/diag/bin/vramps
rwxr-xr-x  0/10 221184 Dec  2 20:57 1986 /usr/pixar/diag/bin/vcbars
rwxr-xr-x  0/10  57344 Dec  2 20:58 1986 /usr/pixar/diag/bin/mctest
rwxr-xr-x  0/10  57344 Dec  2 20:59 1986 /usr/pixar/diag/bin/dumitest
rwxr-xr-x  0/10  32768 Dec  2 20:59 1986 /usr/pixar/diag/bin/epoch
rwxr-xr-x  0/10 188416 Dec  2 21:00 1986 /usr/pixar/diag/bin/power
rwxr-xr-x  0/10  32768 Dec  2 21:00 1986 /usr/pixar/diag/bin/elapsed
rwxr-xr-x  0/10 188416 Dec  2 21:01 1986 /usr/pixar/diag/bin/note
rwxr-xr-x  0/10  32768 Dec  2 21:01 1986 /usr/pixar/diag/bin/temp
rwxr-xr-x  0/10 188416 Dec  2 21:01 1986 /usr/pixar/diag/bin/err
rwxr-xr-x  0/10  49152 Dec  2 21:01 1986 /usr/pixar/diag/bin/timeout
rwxr-xr-x  0/10  40960 Dec  2 21:02 1986 /usr/pixar/diag/bin/verify
rwxr-xr-x  0/10 180224 Dec  2 21:02 1986 /usr/pixar/diag/bin/pixscan
rwxr-xr-x  0/10  12631 Dec  2 21:02 1986 /usr/pixar/diag/bin/Diagnostic
rwxrwxrwx  0/10      0 Dec  4 16:59 1986 /usr/pixar/diag/ucode/
rwxr-xr-x  0/10  16855 Dec  2 20:52 1986 /usr/pixar/diag/ucode/fbtest.ucode
rwxr-xr-x  0/10   8982 Dec  2 20:52 1986 /usr/pixar/diag/ucode/fbex.ucode
rwxr-xr-x  0/10  28759 Dec  2 20:52 1986 /usr/pixar/diag/ucode/nlayer.ucode
rwxr-xr-x  0/10   5266 Dec  2 20:52 1986 /usr/pixar/diag/ucode/spad.ucode
rwxr-xr-x  0/10   3052 Dec  2 20:52 1986 /usr/pixar/diag/ucode/chap.ucode
rwxrwxrwx  0/10      0 Dec  4 16:59 1986 /usr/pixar/diag/doc/
rwxrwxrwx  0/10      0 Dec  4 16:59 1986 /usr/pixar/diag/man/
rwxrwxrwx  0/10      0 Dec  4 16:59 1986 /usr/pixar/diag/man/man1/
rwxrwxrwx  0/10      0 Dec  4 16:59 1986 /usr/pixar/diag/man/man3/
rwxrwxrwx  0/10      0 Dec  4 16:59 1986 /usr/pixar/diag/man/man8/
rwxr-xr-x  0/10      0 Dec  4 17:00 1986 /usr/pixar/doc/
r--r--r--  0/10   1075 Nov 25 19:15 1986 /usr/pixar/doc/Makefile
rwxr-xr-x  0/10      0 Dec  4 16:59 1986 /usr/pixar/doc/tutorial/
rwxrwxrwx  0/10      0 Dec  4 17:09 1986 /usr/pixar/doc/tutorial/pirl/
r--r--r--  0/10 163840 Nov 25 19:15 1986 /usr/pixar/doc/tutorial/pirl/andre.pic
r--r--r--  0/10    499 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/fill.c
```

```
r--r--r--  0/10 573440 Nov 25 19:15 1986 /usr/pixar/doc/tutorial/pirl/genesis.pic
r--r--r--  0/10    892 Nov 25 19:29 1986 /usr/pixar/doc/tutorial/pirl/Makefile
r--r--r--  0/10   1458 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/lines.c
r--r--r--  0/10    665 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/getpic.c
r--r--r--  0/10    130 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/lazybum.c
r--r--r--  0/10   1356 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/merge.c
r--r--r--  0/10   1542 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/plaster.c
r--r--r--  0/10    587 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/savepic.c
r--r--r--  0/10    303 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/skinny.c
r--r--r--  0/10    325 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/skinny2.c
r--r--r--  0/10    192 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/testpat.c
r--r--r--  0/10    923 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/testpat2.c
r--r--r--  0/10    409 Nov 25 19:22 1986 /usr/pixar/doc/tutorial/pirl/wrong.c
rwxrwxrwx  0/10      0 Dec  4 17:09 1986 /usr/pixar/doc/tutorial/chap/
r--r--r--  0/10   2866 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/Makefile
r--r--r--  0/10   1121 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/chdbdemo.s
r--r--r--  0/10   1489 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/chvdr.s
r--r--r--  0/10   1768 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/contour.c
r--r--r--  0/10   1573 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/cursor.c
r--r--r--  0/10   1607 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/dbdemo.c
r--r--r--  0/10   1883 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample7s.c
r--r--r--  0/10   1815 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample1.s
r--r--r--  0/10   2293 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample2.s
r--r--r--  0/10   2332 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample2a.s
r--r--r--  0/10   2767 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample3.s
r--r--r--  0/10   1559 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample3s.c
r--r--r--  0/10   3154 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample4.s
r--r--r--  0/10   2695 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample4a.s
r--r--r--  0/10   1634 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample4s.c
r--r--r--  0/10   1181 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/foo.make
r--r--r--  0/10   1730 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample5s.c
r--r--r--  0/10   2092 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/sample6s.c
r--r--r--  0/10   1315 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/vdr.c
r--r--r--  0/10   1614 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/videmo.c
r--r--r--  0/10   3181 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/foo.s
r--r--r--  0/10   1541 Nov 25 19:18 1986 /usr/pixar/doc/tutorial/chap/mkprimes.c
rwxrwxrwx  0/10      0 Dec  4 17:00 1986 /usr/pixar/doc/macros/
rw-rw-r--  0/10    311 Dec  1 20:23 1986 /usr/pixar/doc/macros/README
r--r--r--  0/10      4 Dec  1 20:23 1986 /usr/pixar/doc/macros/endmacs
r--r--r--  0/10   2453 Dec  1 20:23 1986 /usr/pixar/doc/macros/macros
r--r--r--  0/10    298 Dec  1 20:23 1986 /usr/pixar/doc/macros/toc.sed
rwxrwxrwx  0/10      0 Dec  4 17:00 1986 /usr/pixar/doc/reference/
r--r--r--  0/10  20403 Dec  1 20:24 1986 /usr/pixar/doc/reference/charm.ms
r--r--r--  0/10  52177 Dec  1 20:24 1986 /usr/pixar/doc/reference/chas.ms
r--r--r--  0/10  26039 Dec  1 20:24 1986 /usr/pixar/doc/reference/comp.ms
r--r--r--  0/10  11075 Dec  1 20:24 1986 /usr/pixar/doc/reference/desc.ms
r--r--r--  0/10   8650 Dec  1 20:24 1986 /usr/pixar/doc/reference/format.ms
r--r--r--  0/10  12112 Dec  1 20:24 1986 /usr/pixar/doc/reference/pbusprog.ms
r--r--r--  0/10   1535 Dec  1 20:24 1986 /usr/pixar/doc/reference/tbl.ms
rwxrwxrwx  0/10    921 Dec  1 20:24 1986 /usr/pixar/doc/reference/Makefile
rw-rw-r--  0/10    189 Dec  1 20:24 1986 /usr/pixar/doc/reference/charmlet
rwxr-xr-x  0/10      0 Dec  4 17:05 1986 /usr/pixar/host/
r--r--r--  0/10   1822 Nov 24 17:04 1986 /usr/pixar/host/Makefile
rwxrwxrwx  0/10      0 Dec  4 17:01 1986 /usr/pixar/host/lib/
rw-r--r--  0/10   7632 Dec  2 18:57 1986 /usr/pixar/host/lib/libport.a
rw-r--r--  0/10  54864 Dec  2 19:00 1986 /usr/pixar/host/lib/libaa.a
rw-r--r--  0/10  66624 Dec  2 19:00 1986 /usr/pixar/host/lib/libaa.i.a
rw-r--r--  0/10 764318 Dec  3 14:41 1986 /usr/pixar/host/lib/libpixar.a
rw-r--r--  0/10  80574 Dec  2 19:34 1986 /usr/pixar/host/lib/librG.a
rw-r--r--  0/10 129092 Dec  2 19:38 1986 /usr/pixar/host/lib/libchad.a
rw-r--r--  0/10 587822 Dec  2 19:58 1986 /usr/pixar/host/lib/libpirl.a
rw-r--r--  0/10 463186 Dec  2 20:06 1986 /usr/pixar/host/lib/libpicio.a
rw-r--r--  0/10  10644 Dec  2 20:06 1986 /usr/pixar/host/lib/libcolr.a
rwxrwxrwx  0/10      0 Dec  4 17:05 1986 /usr/pixar/host/bin/
rwxr-xr-x  0/10 352256 Dec  3 16:53 1986 /usr/pixar/host/bin/charm
rwxr-xr-x  0/10 212992 Dec  3 16:56 1986 /usr/pixar/host/bin/loop
```

```
rwxr-xr-x  0/10 212992 Dec  3 17:01 1986 /usr/pixar/host/bin/chas
rwxr-xr-x  0/10    827 Dec  3 17:01 1986 /usr/pixar/host/bin/gamma
rwxr-xr-x  0/10   1924 Dec  3 17:01 1986 /usr/pixar/host/bin/pixinit
rwxr-xr-x  0/10  40960 Dec  3 16:08 1986 /usr/pixar/host/bin/chc
rwxr-xr-x  0/10  40960 Dec  3 16:08 1986 /usr/pixar/host/bin/chnm
rwxr-xr-x  0/10  32768 Dec  3 16:08 1986 /usr/pixar/host/bin/chsize
rwxr-xr-x  0/10  40960 Dec  3 16:10 1986 /usr/pixar/host/bin/chranlib
rwxr-xr-x  0/10 188416 Dec  3 16:11 1986 /usr/pixar/host/bin/chcmp
rwxr-xr-x  0/10 204800 Dec  3 16:14 1986 /usr/pixar/host/bin/chconfig
rwxr-xr-x  0/10  65536 Dec  3 16:15 1986 /usr/pixar/host/bin/chd
rwxr-xr-x  0/10 196608 Dec  3 16:16 1986 /usr/pixar/host/bin/chload
rwxr-xr-x  0/10 196608 Dec  3 16:18 1986 /usr/pixar/host/bin/chmap
rwxr-xr-x  0/10  65536 Dec  3 16:19 1986 /usr/pixar/host/bin/dumi
rwxr-xr-x  0/10  73728 Dec  3 16:22 1986 /usr/pixar/host/bin/mctrl
rwxr-xr-x  0/10 212992 Dec  3 16:24 1986 /usr/pixar/host/bin/video
rwxr-xr-x  0/10  73728 Dec  3 16:25 1986 /usr/pixar/host/bin/chld
rwxr-xr-x  0/10 516096 Dec  3 16:26 1986 /usr/pixar/host/bin/gt
rwxr-xr-x  0/10 516096 Dec  3 16:28 1986 /usr/pixar/host/bin/sv
rwxr-xr-x  0/10 204800 Dec  3 16:28 1986 /usr/pixar/host/bin/gtinfo
rwxr-xr-x  0/10 450560 Dec  3 16:29 1986 /usr/pixar/host/bin/tool
rwxr-xr-x  0/10 393216 Dec  3 16:30 1986 /usr/pixar/host/bin/clr
rwxr-xr-x  0/10 409600 Dec  3 16:31 1986 /usr/pixar/host/bin/cbars
rwxr-xr-x  0/10 409600 Dec  3 16:32 1986 /usr/pixar/host/bin/clamp
rwxr-xr-x  0/10 417792 Dec  3 16:33 1986 /usr/pixar/host/bin/guide
rwxr-xr-x  0/10 434176 Dec  3 16:34 1986 /usr/pixar/host/bin/perm
rwxr-xr-x  0/10 409600 Dec  3 16:35 1986 /usr/pixar/host/bin/blur
rwxr-xr-x  0/10 409600 Dec  3 16:36 1986 /usr/pixar/host/bin/conv
rwxr-xr-x  0/10 409600 Dec  3 16:38 1986 /usr/pixar/host/bin/hg
rwxr-xr-x  0/10 401408 Dec  3 16:39 1986 /usr/pixar/host/bin/merge
rwxr-xr-x  0/10 409600 Dec  3 16:40 1986 /usr/pixar/host/bin/cha
rwxr-xr-x  0/10 409600 Dec  3 16:41 1986 /usr/pixar/host/bin/ramp
rwxr-xr-x  0/10 409600 Dec  3 16:42 1986 /usr/pixar/host/bin/scale
rwxr-xr-x  0/10 409600 Dec  3 16:44 1986 /usr/pixar/host/bin/copy
rwxr-xr-x  0/10 409600 Dec  3 16:44 1986 /usr/pixar/host/bin/resize
rwxr-xr-x  0/10 417792 Dec  3 16:45 1986 /usr/pixar/host/bin/rotate
rwxr-xr-x  0/10 393216 Dec  3 16:46 1986 /usr/pixar/host/bin/crc
rwxr-xr-x  0/10 409600 Dec  3 16:47 1986 /usr/pixar/host/bin/see
rwxrwxrwx  0/10      0 Dec  4 17:05 1986 /usr/pixar/host/etc/
r--r--r--  0/10    858 Nov 25 14:32 1986 /usr/pixar/host/etc/rpacerrors
rwxrwxrwx  0/10      0 Dec  4 17:06 1986 /usr/pixar/include/
rwxrwxrwx  0/10      0 Dec  4 17:05 1986 /usr/pixar/include/pixar/
rwxrwxrwx  0/10      0 Dec  4 17:05 1986 /usr/pixar/include/pixar/chap/
r--r--r--  0/10   2759 Dec  2 18:20 1986 /usr/pixar/include/pixar/chap/mman.h
r--r--r--  0/10   1134 Dec  2 18:20 1986 /usr/pixar/include/pixar/chap/pbus.h
r--r--r--  0/10   2105 Dec  2 18:20 1986 /usr/pixar/include/pixar/chap/pbusreg.h
r--r--r--  0/10   1802 Dec  2 18:20 1986 /usr/pixar/include/pixar/chap/pw.h
r--r--r--  0/10   2799 Dec  2 18:20 1986 /usr/pixar/include/pixar/chap/yapbusreg.h
r--r--r--  0/10  11813 Dec  2 18:19 1986 /usr/pixar/include/pixar/alu.h
r--r--r--  0/10   8014 Dec  2 18:19 1986 /usr/pixar/include/pixar/chap.h
r--r--r--  0/10  10619 Dec  2 18:19 1986 /usr/pixar/include/pixar/chapdiag.h
r--r--r--  0/10   1658 Dec  2 18:19 1986 /usr/pixar/include/pixar/chapdefines.h
r--r--r--  0/10   2317 Dec  2 18:19 1986 /usr/pixar/include/pixar/chaperrno.h
r--r--r--  0/10   1984 Dec  2 18:20 1986 /usr/pixar/include/pixar/diag.h
r--r--r--  0/10   3797 Dec  2 18:20 1986 /usr/pixar/include/pixar/reloc.h
r--r--r--  0/10   1322 Dec  2 18:20 1986 /usr/pixar/include/pixar/ucalls.h
r--r--r--  0/10   2261 Dec  2 18:20 1986 /usr/pixar/include/pixar/video.h
r--r--r--  0/10   1442 Dec  3 14:13 1986 /usr/pixar/include/pixar/pixar.h
r--r--r--  0/10   2155 Nov 25 14:32 1986 /usr/pixar/include/pixar/rpacmd.h
r--r--r--  0/10   2175 Nov 25 14:32 1986 /usr/pixar/include/pixar/charpac.h
r--r--r--  0/10    717 Nov 24 17:42 1986 /usr/pixar/include/pixar/table.h
r--r--r--  0/10   1012 Nov 24 17:42 1986 /usr/pixar/include/pixar/yap.h
r--r--r--  0/10   1913 Dec  2 18:20 1986 /usr/pixar/include/pixar/chapedge.h
r--r--r--  0/10   9597 Dec  2 18:20 1986 /usr/pixar/include/pixar/chapmult.h
rwxrwxrwx  0/10      0 Dec  4 17:05 1986 /usr/pixar/include/pixardev/
r--r--r--  0/10   3455 Dec  2 18:20 1986 /usr/pixar/include/pixardev/chapioctl.h
r--r--r--  0/10   3223 Dec  2 18:20 1986 /usr/pixar/include/pixardev/chapreg.h
```

```
r--r--r--  0/10   3403 Dec  2 18:20 1986 /usr/pixar/include/pixardev/videoreg.h
r--r--r--  0/10   1789 Dec  3 14:13 1986 /usr/pixar/include/pixardev/dumireg.h
r--r--r--  0/10   6785 Dec  3 14:13 1986 /usr/pixar/include/pixardev/mctrlreg.h
r--r--r--  0/10   5783 Dec  3 14:13 1986 /usr/pixar/include/pixardev/yumireg.h
r--r--r--  0/10   1793 Dec  3 14:13 1986 /usr/pixar/include/pixardev/yumioctl.h
rwxrwxrwx  0/10      0 Dec  4 17:05 1986 /usr/pixar/include/port/
rwxrwxrwx  0/10      0 Dec  4 17:05 1986 /usr/pixar/include/port/sys/
r--r--r--  0/10   1276 Dec  2 18:56 1986 /usr/pixar/include/port/sys/uio.h
r--r--r--  0/10   1183 Dec  2 18:56 1986 /usr/pixar/include/port/filestuff.h
r--r--r--  0/10   2063 Dec  2 19:00 1986 /usr/pixar/include/aarg.h
r--r--r--  0/10   1082 Dec  2 19:00 1986 /usr/pixar/include/std.h
r--r--r--  0/10   1472 Dec  2 19:00 1986 /usr/pixar/include/aarg.i.h
r--r--r--  0/10   9488 Dec  2 19:34 1986 /usr/pixar/include/chad.h
r--r--r--  0/10   2239 Dec  2 19:34 1986 /usr/pixar/include/Chad.h
r--r--r--  0/10    877 Dec  2 19:58 1986 /usr/pixar/include/cbars.h
r--r--r--  0/10   4025 Dec  2 19:58 1986 /usr/pixar/include/pirl.h
r--r--r--  0/10   1306 Dec  2 19:58 1986 /usr/pixar/include/merge.h
r--r--r--  0/10   3276 Dec  2 18:22 1986 /usr/pixar/include/rpacemu.h
r--r--r--  0/10   3194 Dec  2 18:22 1986 /usr/pixar/include/picio.h
r--r--r--  0/10   2164 Dec  2 18:22 1986 /usr/pixar/include/picture.h
r--r--r--  0/10    905 Dec  2 20:06 1986 /usr/pixar/include/colr.h
r--r--r--  0/10   1516 Dec  2 19:31 1986 /usr/pixar/include/pixeldef.h
r--r--r--  0/10   2717 Dec  2 18:22 1986 /usr/pixar/include/rpac.h
r--r--r--  0/10   4300 Dec  2 18:22 1986 /usr/pixar/include/rpacmacs.h
r--r--r--  0/10   2371 Nov 24 17:42 1986 /usr/pixar/include/fbregs.h
r--r--r--  0/10   2056 Nov 24 17:57 1986 /usr/pixar/include/aarg.globals.h
r--r--r--  0/10   1699 Nov 24 17:57 1986 /usr/pixar/include/pirlxform.h
r--r--r--  0/10   2175 Nov 24 17:57 1986 /usr/pixar/include/charpac.h
r--r--r--  0/10   1577 Nov 24 17:57 1986 /usr/pixar/include/rpacerrors.h
r--r--r--  0/10   2410 Nov 24 17:57 1986 /usr/pixar/include/rpacinter.h
r--r--r--  0/10   2155 Nov 24 17:57 1986 /usr/pixar/include/rpacmd.h
r--r--r--  0/10    915 Nov 24 17:57 1986 /usr/pixar/include/rpacpixar.h
r--r--r--  0/10   1196 Nov 24 17:57 1986 /usr/pixar/include/loop.h
r--r--r--  0/10   2209 Nov 24 17:57 1986 /usr/pixar/include/screen.h
r--r--r--  0/10    967 Nov 24 17:57 1986 /usr/pixar/include/cpu.h
r--r--r--  0/10   2774 Nov 24 17:57 1986 /usr/pixar/include/piccode.h
r--r--r--  0/10   2539 Nov 24 17:57 1986 /usr/pixar/include/alutab.h
r--r--r--  0/10   3897 Nov 24 17:57 1986 /usr/pixar/include/chas.h
r--r--r--  0/10   1890 Nov 24 17:57 1986 /usr/pixar/include/datapath.h
r--r--r--  0/10   2239 Nov 24 17:57 1986 /usr/pixar/include/eval.h
r--r--r--  0/10   2132 Nov 24 17:57 1986 /usr/pixar/include/io.h
r--r--r--  0/10  12889 Nov 24 17:57 1986 /usr/pixar/include/ops.h
r--r--r--  0/10   1349 Nov 24 17:57 1986 /usr/pixar/include/seg.h
r--r--r--  0/10   2048 Nov 24 17:57 1986 /usr/pixar/include/symbols.h
r--r--r--  0/10   2285 Nov 24 17:57 1986 /usr/pixar/include/trees.h
r--r--r--  0/10  11813 Nov 24 17:57 1986 /usr/pixar/include/alu.h
r--r--r--  0/10   8014 Nov 24 17:57 1986 /usr/pixar/include/chap.h
r--r--r--  0/10   3797 Nov 24 17:57 1986 /usr/pixar/include/reloc.h
r--r--r--  0/10   1161 Nov 24 18:24 1986 /usr/pixar/include/chaptest.h
r--r--r--  0/10   1786 Nov 24 18:24 1986 /usr/pixar/include/macros.h
r--r--r--  0/10   1730 Nov 24 18:24 1986 /usr/pixar/include/vdiag.h
r--r--r--  0/10   1129 Dec  2 19:31 1986 /usr/pixar/include/coloraarg.h
r--r--r--  0/10   1127 Dec  2 19:31 1986 /usr/pixar/include/constants.h
r--r--r--  0/10   3397 Dec  2 19:31 1986 /usr/pixar/include/gfxtypes.h
r--r--r--  0/10   1415 Dec  2 19:31 1986 /usr/pixar/include/fbaarg.h
r--r--r--  0/10   1482 Dec  2 19:31 1986 /usr/pixar/include/fbdefs.h
r--r--r--  0/10   1340 Dec  2 19:31 1986 /usr/pixar/include/math.h
r--r--r--  0/10   1988 Dec  2 19:31 1986 /usr/pixar/include/pixwin.h
r--r--r--  0/10    995 Dec  2 19:31 1986 /usr/pixar/include/random.h
r--r--r--  0/10   1317 Dec  2 19:31 1986 /usr/pixar/include/environ.h
rwxrwxrwx  0/10      0 Dec  4 17:06 1986 /usr/pixar/lib/
rwxrwxrwx  0/10      0 Dec  4 17:54 1986 /usr/pixar/sys/
rwxr-xr-x  0/10 607793 Dec  1 23:30 1986 /usr/pixar/sys/vmunix
r-xr-xr-x  0/10   1736 Dec  3 12:52 1986 /usr/pixar/sys/DEFAULT
r-xr-xr-x  0/10   1736 Dec  3 12:52 1986 /usr/pixar/sys/DEFAULT_3.0
r-xr-xr-x  0/10   1432 Dec  3 12:52 1986 /usr/pixar/sys/MAKEDEV
```

```
r-xr-xr-x  0/10   1451 Dec  3 12:52 1986 /usr/pixar/sys/MAKEDEV_3.0
r--r--r--  0/10   1751 Dec  3 13:02 1986 /usr/pixar/sys/Makefile
rw-r--r--693/10  37394 Dec  3 16:21 1986 /usr/pixar/sys/README
rw-r--r--  0/10  10266 Dec  1 23:30 1986 /usr/pixar/sys/pixar.obj
rw-r--r--  0/10  10177 Dec  1 23:30 1986 /usr/pixar/sys/pixar.obj_3.0
r--r--r--  0/10    987 Dec  3 13:02 1986 /usr/pixar/sys/pixar_conf.c
r--r--r--  0/10    984 Dec  1 23:29 1986 /usr/pixar/sys/pixar_conf.c_3.0
r--r--r--  0/10   2133 Dec  1 23:29 1986 /usr/pixar/sys/rc.local
r--r--r--  0/10   2133 Dec  1 23:29 1986 /usr/pixar/sys/rc.local_3.0
rwxrwxrwx  0/10      0 Dec  4 17:09 1986 /usr/pixar/sys/tablet/
rwxr-xr-x  0/10 607793 Dec  1 23:29 1986 /usr/pixar/sys/tablet/vmunix
r--r--r--  0/10    487 Dec  1 23:29 1986 /usr/pixar/sys/tablet/MAKEDEV
rw-r--r--  0/10   1420 Dec  1 23:29 1986 /usr/pixar/sys/tablet/README
r--r--r--  0/10   2121 Dec  1 23:29 1986 /usr/pixar/sys/tablet/tablet.h
r--r--r--  0/10   2133 Dec  1 23:29 1986 /usr/pixar/sys/tablet/tty_conf.c
rw-r--r--  0/10   7633 Dec  1 23:29 1986 /usr/pixar/sys/tablet/tty_tb.c
rwxr-xr-x  0/10 546235 Dec  1 23:29 1986 /usr/pixar/sys/tablet/vmunix_3.0
rw-r--r--  0/10   2361 Dec  1 23:29 1986 /usr/pixar/sys/tablet/tty_conf.o_3.0
rw-r--r--  0/10   3881 Dec  1 23:29 1986 /usr/pixar/sys/tablet/tty_tb.o_3.0
rwxrwxrwx  0/10      0 Dec  4 17:09 1986 /usr/pixar/sys/tablet/test/
rwxr-xr-x  0/10  32768 Dec  1 23:29 1986 /usr/pixar/sys/tablet/test/tablet
rw-r--r--  0/10   1807 Dec  1 23:29 1986 /usr/pixar/sys/tablet/test/tablet.c
rwxr-xr-x  0/10  32768 Dec  1 23:29 1986 /usr/pixar/sys/tablet/test/tb
rw-r--r--  0/10   2493 Dec  1 23:29 1986 /usr/pixar/sys/tablet/test/tb.c
rw-r--r--  0/10    378 Dec  1 23:29 1986 /usr/pixar/sys/tablet/test/README
rwxr-xr-x  0/10 540680 Dec  1 23:29 1986 /usr/pixar/sys/vmunix_3.0
r-xr-xr-x  0/10   4395 Dec  3 12:34 1986 /usr/pixar/sys/modem
r-xr-xr-x  0/10   3566 Dec  3 12:41 1986 /usr/pixar/sys/uu_install
rwxrwxrwx  0/10      0 Dec  4 17:08 1986 /usr/pixar/man/
rwxrwxrwx  0/10      0 Dec  4 17:07 1986 /usr/pixar/man/cat7/
rwxr-xr-x  0/10      0 Dec  4 17:07 1986 /usr/pixar/man/cat1/
rwxr-xr-x  0/10      0 Dec  4 17:07 1986 /usr/pixar/man/cat3/
rwxrwxrwx  0/10      0 Dec  4 17:07 1986 /usr/pixar/man/cat4/
rwxrwxrwx  0/10      0 Dec  4 17:07 1986 /usr/pixar/man/cat5/
rwxrwxrwx  0/10      0 Dec  4 17:07 1986 /usr/pixar/man/cat8/
rwxrwxrwx  0/10      0 Dec  4 17:07 1986 /usr/pixar/man/man1/
r--r--r--  0/10   5848 Dec  3 19:09 1986 /usr/pixar/man/man1/intro.1
r--r--r--  0/10  17838 Dec  3 19:09 1986 /usr/pixar/man/man1/charm.1
r--r--r--  0/10   1835 Dec  3 19:09 1986 /usr/pixar/man/man1/chas.1
r--r--r--  0/10   3009 Dec  3 19:09 1986 /usr/pixar/man/man1/chc.1
r--r--r--  0/10   1607 Dec  3 19:09 1986 /usr/pixar/man/man1/chcmp.1
r--r--r--  0/10   1751 Dec  3 19:09 1986 /usr/pixar/man/man1/chd.1
r--r--r--  0/10   5473 Dec  3 19:09 1986 /usr/pixar/man/man1/chld.1
r--r--r--  0/10   1995 Dec  3 19:09 1986 /usr/pixar/man/man1/chload.1
r--r--r--  0/10   3038 Dec  3 19:09 1986 /usr/pixar/man/man1/chmap.1
r--r--r--  0/10   1975 Dec  3 19:09 1986 /usr/pixar/man/man1/chnm.1
r--r--r--  0/10   1323 Dec  3 19:09 1986 /usr/pixar/man/man1/chranlib.1
r--r--r--  0/10   1034 Dec  3 19:09 1986 /usr/pixar/man/man1/chsize.1
r--r--r--  0/10   1770 Dec  3 19:09 1986 /usr/pixar/man/man1/dumi.1
r--r--r--  0/10   2271 Dec  3 19:08 1986 /usr/pixar/man/man1/blur.1
r--r--r--  0/10   2658 Dec  3 19:08 1986 /usr/pixar/man/man1/cbars.1
r--r--r--  0/10   2712 Dec  3 19:09 1986 /usr/pixar/man/man1/cha.1
r--r--r--  0/10    743 Dec  3 19:09 1986 /usr/pixar/man/man1/clamp.1
r--r--r--  0/10   1753 Dec  3 19:09 1986 /usr/pixar/man/man1/clr.1
r--r--r--  0/10   1560 Dec  3 19:09 1986 /usr/pixar/man/man1/conv.1
r--r--r--  0/10   3071 Dec  3 19:09 1986 /usr/pixar/man/man1/copy.1
r--r--r--  0/10   1089 Dec  3 19:09 1986 /usr/pixar/man/man1/crc.1
r--r--r--  0/10   1117 Dec  3 19:09 1986 /usr/pixar/man/man1/gamma.1
r--r--r--  0/10   2583 Dec  3 19:09 1986 /usr/pixar/man/man1/gt.1
r--r--r--  0/10   1362 Dec  3 19:09 1986 /usr/pixar/man/man1/gtinfo.1
r--r--r--  0/10   1427 Dec  3 19:09 1986 /usr/pixar/man/man1/guide.1
r--r--r--  0/10   1135 Dec  3 19:09 1986 /usr/pixar/man/man1/hg.1
r--r--r--  0/10   3433 Dec  3 19:09 1986 /usr/pixar/man/man1/loop.1
r--r--r--  0/10   3813 Dec  3 19:09 1986 /usr/pixar/man/man1/merge.1
r--r--r--  0/10   2580 Dec  3 19:09 1986 /usr/pixar/man/man1/perm.1
r--r--r--  0/10   1269 Dec  3 19:09 1986 /usr/pixar/man/man1/pixinit.1
```

```
r--r--r--  0/10   2587 Dec  3 19:09 1986 /usr/pixar/man/man1/ramp.1
r--r--r--  0/10   3040 Dec  3 19:09 1986 /usr/pixar/man/man1/resize.1
r--r--r--  0/10   3031 Nov 28 13:29 1986 /usr/pixar/man/man1/rotate.1
r--r--r--  0/10   2705 Dec  3 19:09 1986 /usr/pixar/man/man1/scale.1
r--r--r--  0/10   2097 Dec  3 19:09 1986 /usr/pixar/man/man1/see.1
r--r--r--  0/10   2724 Dec  3 19:09 1986 /usr/pixar/man/man1/sv.1
r--r--r--  0/10   2764 Dec  3 19:09 1986 /usr/pixar/man/man1/tool.1
r--r--r--  0/10   2174 Dec  3 19:09 1986 /usr/pixar/man/man1/video.1
rwxrwxrwx  0/10      0 Dec  4 17:08 1986 /usr/pixar/man/man3/
r--r--r--  0/10    265 Dec  3 19:53 1986 /usr/pixar/man/man3/docu
r--r--r--  0/10    237 Dec  3 19:53 1986 /usr/pixar/man/man3/errors
r--r--r--  0/10   6245 Dec  3 20:01 1986 /usr/pixar/man/man3/ChadAlloc.3
r--r--r--  0/10   1615 Dec  3 19:56 1986 /usr/pixar/man/man3/ChadBegin.3
r--r--r--  0/10   3580 Nov 28 13:31 1986 /usr/pixar/man/man3/linetest.c
r--r--r--  0/10   4329 Dec  3 20:02 1986 /usr/pixar/man/man3/ChadWrite.3
r--r--r--  0/10   4232 Dec  3 19:11 1986 /usr/pixar/man/man3/intro.3c
r--r--r--  0/10   1089 Dec  3 19:11 1986 /usr/pixar/man/man3/SSClamp.3c
r--r--r--  0/10   2315 Dec  3 19:11 1986 /usr/pixar/man/man3/XYZ2rgb.3c
r--r--r--  0/10   1278 Dec  3 19:11 1986 /usr/pixar/man/man3/libcolor.3c
r--r--r--  0/10   2364 Dec  3 19:12 1986 /usr/pixar/man/man3/rgb2XYZ.3c
r--r--r--  0/10   2561 Dec  3 19:12 1986 /usr/pixar/man/man3/rgb2xyY.3c
r--r--r--  0/10   1516 Dec  3 19:12 1986 /usr/pixar/man/man3/libpg.3c
r--r--r--  0/10   1690 Dec  3 19:12 1986 /usr/pixar/man/man3/mman.3c
r--r--r--  0/10   3208 Dec  3 19:12 1986 /usr/pixar/man/man3/stack.3c
r--r--r--  0/10   2039 Dec  3 19:09 1986 /usr/pixar/man/man3/C33.3c
r--r--r--  0/10   2008 Dec  3 19:09 1986 /usr/pixar/man/man3/C33s.3c
r--r--r--  0/10   2086 Dec  3 19:09 1986 /usr/pixar/man/man3/C55s.3c
r--r--r--  0/10   1514 Dec  3 19:10 1986 /usr/pixar/man/man3/PWArithmetic.3c
r--r--r--  0/10   1546 Dec  3 19:10 1986 /usr/pixar/man/man3/PWBBox.3c
r--r--r--  0/10   1899 Dec  3 19:10 1986 /usr/pixar/man/man3/PWBoxFilter.3c
r--r--r--  0/10   2032 Dec  3 19:10 1986 /usr/pixar/man/man3/PWConv.3c
r--r--r--  0/10    993 Dec  3 19:10 1986 /usr/pixar/man/man3/PWCrc.3c
r--r--r--  0/10   1626 Dec  3 19:10 1986 /usr/pixar/man/man3/PWHistogram.3c
r--r--r--  0/10   1352 Dec  3 19:10 1986 /usr/pixar/man/man3/PWMap.3c
r--r--r--  0/10   1176 Dec  3 19:10 1986 /usr/pixar/man/man3/PWRange.3c
r--r--r--  0/10   1461 Dec  3 19:10 1986 /usr/pixar/man/man3/PWc33.3c
r--r--r--  0/10   1444 Dec  3 19:10 1986 /usr/pixar/man/man3/PWc33s.3c
r--r--r--  0/10   1623 Dec  3 19:11 1986 /usr/pixar/man/man3/SSArithmetic.3c
r--r--r--  0/10   1639 Dec  3 19:11 1986 /usr/pixar/man/man3/SSBoxFilter.3c
r--r--r--  0/10   2197 Dec  3 19:11 1986 /usr/pixar/man/man3/SSConv.3c
r--r--r--  0/10   1168 Dec  3 19:11 1986 /usr/pixar/man/man3/SSCrc.3c
r--r--r--  0/10   1198 Dec  3 19:11 1986 /usr/pixar/man/man3/SSRange.3c
r--r--r--  0/10   1658 Dec  3 19:11 1986 /usr/pixar/man/man3/dhg.3c
r--r--r--  0/10   4104 Dec  3 19:12 1986 /usr/pixar/man/man3/libpip.3c
r--r--r--  0/10   3666 Dec  3 19:12 1986 /usr/pixar/man/man3/libpm.3c
r--r--r--  0/10   2656 Dec  3 19:12 1986 /usr/pixar/man/man3/matrix.3c
r--r--r--  0/10   1739 Dec  3 19:12 1986 /usr/pixar/man/man3/reciprocal.3c
r--r--r--  0/10   2297 Dec  3 19:12 1986 /usr/pixar/man/man3/recsqrt.3c
r--r--r--  0/10   1491 Dec  3 19:12 1986 /usr/pixar/man/man3/rrand.3c
r--r--r--  0/10   2274 Dec  3 19:12 1986 /usr/pixar/man/man3/sqrt.3c
r--r--r--  0/10   6519 Dec  3 19:12 1986 /usr/pixar/man/man3/xp.3c
r--r--r--  0/10   3340 Dec  3 19:09 1986 /usr/pixar/man/man3/CFCopy.3c
r--r--r--  0/10   1861 Dec  3 19:09 1986 /usr/pixar/man/man3/CICopy.3c
r--r--r--  0/10   1244 Dec  3 19:09 1986 /usr/pixar/man/man3/CRCopy.3c
r--r--r--  0/10   4624 Dec  3 19:10 1986 /usr/pixar/man/man3/FCCopy.3c
r--r--r--  0/10   4642 Dec  3 19:10 1986 /usr/pixar/man/man3/FICopy.3c
r--r--r--  0/10   2831 Dec  3 19:10 1986 /usr/pixar/man/man3/FRGBACopy.3c
r--r--r--  0/10   2415 Dec  3 19:10 1986 /usr/pixar/man/man3/FSCopy.3c
r--r--r--  0/10   1446 Dec  3 19:10 1986 /usr/pixar/man/man3/FYCopy.3c
r--r--r--  0/10   3226 Dec  3 19:10 1986 /usr/pixar/man/man3/IFCopy.3c
r--r--r--  0/10   3736 Dec  3 19:10 1986 /usr/pixar/man/man3/PW.3c
r--r--r--  0/10   1039 Dec  3 19:10 1986 /usr/pixar/man/man3/PW4Map.3c
r--r--r--  0/10   1196 Dec  3 19:10 1986 /usr/pixar/man/man3/PWAxb.3c
r--r--r--  0/10   1788 Dec  3 19:10 1986 /usr/pixar/man/man3/PWCha.3c
r--r--r--  0/10   1107 Dec  3 19:10 1986 /usr/pixar/man/man3/PWClamp.3c
r--r--r--  0/10    954 Dec  3 19:10 1986 /usr/pixar/man/man3/PWClear.3c
```

```
r--r--r--  0/10   2495 Dec  3 19:10 1986 /usr/pixar/man/man3/PWCopy.3c
r--r--r--  0/10   1243 Dec  3 19:10 1986 /usr/pixar/man/man3/PWGeneric.3c
r--r--r--  0/10   2850 Dec  3 19:10 1986 /usr/pixar/man/man3/PWMerge.3c
r--r--r--  0/10    967 Dec  3 19:10 1986 /usr/pixar/man/man3/PWNot.3c
r--r--r--  0/10   1501 Dec  3 19:10 1986 /usr/pixar/man/man3/PWShift.3c
r--r--r--  0/10   1203 Dec  3 19:10 1986 /usr/pixar/man/man3/PWShuffle.3c
r--r--r--  0/10   1129 Dec  3 19:10 1986 /usr/pixar/man/man3/PWSwap.3c
r--r--r--  0/10   1161 Dec  3 19:10 1986 /usr/pixar/man/man3/PWTranspose.3c
r--r--r--  0/10   3040 Dec  3 19:11 1986 /usr/pixar/man/man3/RGBAFCopy.3c
r--r--r--  0/10   1204 Dec  3 19:11 1986 /usr/pixar/man/man3/RSCopy.3c
r--r--r--  0/10   1398 Dec  3 19:11 1986 /usr/pixar/man/man3/SCCopy.3c
r--r--r--  0/10   4306 Dec  3 19:11 1986 /usr/pixar/man/man3/SFCopy.3c
r--r--r--  0/10   1332 Dec  3 19:11 1986 /usr/pixar/man/man3/SICopy.3c
r--r--r--  0/10   1208 Dec  3 19:11 1986 /usr/pixar/man/man3/SS4Map.3c
r--r--r--  0/10   1012 Dec  3 19:11 1986 /usr/pixar/man/man3/SSAxb.3c
r--r--r--  0/10   1738 Dec  3 19:11 1986 /usr/pixar/man/man3/SSCha.3c
r--r--r--  0/10   1102 Dec  3 19:11 1986 /usr/pixar/man/man3/SSComb.3c
r--r--r--  0/10   1211 Dec  3 19:11 1986 /usr/pixar/man/man3/SSCopy.3c
r--r--r--  0/10    992 Dec  3 19:11 1986 /usr/pixar/man/man3/SSCompare.3c
r--r--r--  0/10   1198 Dec  3 19:11 1986 /usr/pixar/man/man3/SSCopyComp.3c
r--r--r--  0/10    867 Dec  3 19:11 1986 /usr/pixar/man/man3/SSCopyRGBA.3c
r--r--r--  0/10   1242 Dec  3 19:11 1986 /usr/pixar/man/man3/SSCopyRGBALUT.3c
r--r--r--  0/10   3385 Dec  3 19:11 1986 /usr/pixar/man/man3/SSMerge.3c
r--r--r--  0/10   3160 Dec  3 19:11 1986 /usr/pixar/man/man3/SSPaint.3c
r--r--r--  0/10   2314 Dec  3 19:11 1986 /usr/pixar/man/man3/SSShuffle.3c
r--r--r--  0/10   1325 Dec  3 19:11 1986 /usr/pixar/man/man3/SYCopy.3c
r--r--r--  0/10   2507 Dec  3 19:11 1986 /usr/pixar/man/man3/TB.3c
r--r--r--  0/10   1836 Dec  3 19:11 1986 /usr/pixar/man/man3/TBCopy.3c
r--r--r--  0/10   1262 Dec  3 19:11 1986 /usr/pixar/man/man3/YFCopy.3c
r--r--r--  0/10   1232 Dec  3 19:11 1986 /usr/pixar/man/man3/YSCopy.3c
r--r--r--  0/10  18374 Dec  3 19:12 1986 /usr/pixar/man/man3/libpt.3c
r--r--r--  0/10   5627 Dec  3 19:10 1986 /usr/pixar/man/man3/PWResize.3c
r--r--r--  0/10   2781 Nov 28 13:31 1986 /usr/pixar/man/man3/PWShear.3c
r--r--r--  0/10   1092 Dec  3 19:11 1986 /usr/pixar/man/man3/SSHalve.3c
r--r--r--  0/10  12670 Dec  3 19:11 1986 /usr/pixar/man/man3/SSScale.3c
r--r--r--  0/10   2915 Dec  3 19:12 1986 /usr/pixar/man/man3/libpx.3c
r--r--r--  0/10   2437 Dec  3 19:12 1986 /usr/pixar/man/man3/stwarp.3c
r--r--r--  0/10   2477 Dec  3 19:12 1986 /usr/pixar/man/man3/stwarptable.3c
r--r--r--  0/10   2769 Dec  3 19:11 1986 /usr/pixar/man/man3/intro.3
r--r--r--  0/10   1458 Dec  3 20:01 1986 /usr/pixar/man/man3/ChadErrReport.3
r--r--r--  0/10   4842 Dec  3 20:01 1986 /usr/pixar/man/man3/ChadFrame.3
r--r--r--  0/10   1937 Dec  3 20:02 1986 /usr/pixar/man/man3/ChadGo.3
r--r--r--  0/10    524 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapAbus.3
r--r--r--  0/10   1021 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapAlu.3
r--r--r--  0/10   4501 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapArchive.3
r--r--r--  0/10    604 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapBpt.3
r--r--r--  0/10    828 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapConfig.3
r--r--r--  0/10    834 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapInst.3
r--r--r--  0/10   4461 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapLoad.3
r--r--r--  0/10   1922 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapLoadGo.3
r--r--r--  0/10   4178 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapMMan.3
r--r--r--  0/10    915 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapMbus.3
r--r--r--  0/10   1754 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapOpen.3
r--r--r--  0/10   1044 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapRam.3
r--r--r--  0/10   3476 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapReg.3
r--r--r--  0/10    555 Dec  3 19:09 1986 /usr/pixar/man/man3/ChapReset.3
r--r--r--  0/10   2023 Dec  3 19:10 1986 /usr/pixar/man/man3/ChapRun.3
r--r--r--  0/10    984 Dec  3 19:10 1986 /usr/pixar/man/man3/ChapSbus.3
r--r--r--  0/10   2354 Dec  3 19:10 1986 /usr/pixar/man/man3/ChapSpad.3
r--r--r--  0/10   1462 Dec  3 19:10 1986 /usr/pixar/man/man3/ChapStack.3
r--r--r--  0/10   1195 Dec  3 19:10 1986 /usr/pixar/man/man3/ChapSym.3
r--r--r--  0/10   1595 Dec  3 19:10 1986 /usr/pixar/man/man3/ChapWait.3
r--r--r--  0/10    562 Dec  3 19:10 1986 /usr/pixar/man/man3/ChapXbar.3
r--r--r--  0/10   1478 Dec  3 19:10 1986 /usr/pixar/man/man3/DbOpen.3
r--r--r--  0/10   1356 Dec  3 19:10 1986 /usr/pixar/man/man3/DumiOpen.3
r--r--r--  0/10   1477 Dec  3 19:10 1986 /usr/pixar/man/man3/MctrlOpen.3
```

```
r--r--r--  0/10   1040 Dec  3 19:10 1986 /usr/pixar/man/man3/PicClose.3
r--r--r--  0/10   4468 Dec  3 19:10 1986 /usr/pixar/man/man3/PicCreat.3
r--r--r--  0/10   3336 Dec  3 19:10 1986 /usr/pixar/man/man3/PicDecode.3
r--r--r--  0/10   2898 Dec  3 19:10 1986 /usr/pixar/man/man3/PicEncode.3
r--r--r--  0/10   4724 Dec  3 19:10 1986 /usr/pixar/man/man3/PicFrame.3
r--r--r--  0/10   3165 Dec  3 19:10 1986 /usr/pixar/man/man3/PicLabel.3
r--r--r--  0/10   1666 Dec  3 19:10 1986 /usr/pixar/man/man3/PicRead.3
r--r--r--  0/10   1926 Dec  3 19:10 1986 /usr/pixar/man/man3/PirlArithmetic.3
r--r--r--  0/10   2608 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlAxb.3
r--r--r--  0/10   1703 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlBBox.3
r--r--r--  0/10   2705 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlBegin.3
r--r--r--  0/10   2162 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlBoxFilter.3
r--r--r--  0/10   1399 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlCbars.3
r--r--r--  0/10   1408 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlCha.3
r--r--r--  0/10   1403 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlClamp.3
r--r--r--  0/10   1249 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlClear.3
r--r--r--  0/10   2180 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlConvolve.3
r--r--r--  0/10   2048 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlConvolve3x3.3
r--r--r--  0/10   2874 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlCopy.3
r--r--r--  0/10   1420 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlCrc.3
r--r--r--  0/10   1948 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlDisplay.3
r--r--r--  0/10   1392 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlErrReport.3
r--r--r--  0/10   1863 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlFrame.3
r--r--r--  0/10   2045 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlGetBuf.3
r--r--r--  0/10   2221 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlGetPic.3
r--r--r--  0/10   2174 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlGetRaster.3
r--r--r--  0/10   1803 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlHistogram.3
r--r--r--  0/10   8567 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlLine.3
r--r--r--  0/10   2282 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlMakeMap.3
r--r--r--  0/10   2480 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlMap.3
r--r--r--  0/10    979 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlMapComp.3
r--r--r--  0/10   4752 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlMerge.3
r--r--r--  0/10   1370 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlNewPW.3
r--r--r--  0/10   1267 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlNot.3
r--r--r--  0/10   1573 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlRamp.3
r--r--r--  0/10   1340 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlRange.3
r--r--r--  0/10   1549 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlReflect.3
r--r--r--  0/10   1986 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlResize.3
r--r--r--  0/10   2409 Dec  1 20:01 1986 /usr/pixar/man/man3/PirlRotate.3
r--r--r--  0/10   1506 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlSetChannelMask.3
r--r--r--  0/10   2054 Dec  1 20:01 1986 /usr/pixar/man/man3/PirlShear.3
r--r--r--  0/10   1824 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlShift.3
r--r--r--  0/10   1607 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlShuffle.3
r--r--r--  0/10   1406 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlSwap.3
r--r--r--  0/10   1773 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlSweep.3
r--r--r--  0/10   1495 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlTranspose.3
r--r--r--  0/10   1659 Dec  3 19:11 1986 /usr/pixar/man/man3/PirlZoom.3
r--r--r--  0/10   1428 Dec  3 19:11 1986 /usr/pixar/man/man3/VideoCmap.3
r--r--r--  0/10   2592 Dec  3 19:11 1986 /usr/pixar/man/man3/VideoCursor.3
r--r--r--  0/10   2551 Dec  3 19:11 1986 /usr/pixar/man/man3/VideoDisplay.3
r--r--r--  0/10   2003 Dec  3 19:11 1986 /usr/pixar/man/man3/VideoFormat.3
r--r--r--  0/10   1624 Dec  1 20:01 1986 /usr/pixar/man/man3/VideoOpen.3
r--r--r--  0/10   2280 Dec  3 19:11 1986 /usr/pixar/man/man3/fbgetdef.3
r--r--r--  0/10   2580 Dec  3 19:11 1986 /usr/pixar/man/man3/getdevs.3
r--r--r--  0/10   7787 Dec  3 19:11 1986 /usr/pixar/man/man3/libchad.3
r--r--r--  0/10   6936 Dec  3 19:12 1986 /usr/pixar/man/man3/libpicio.3
r--r--r--  0/10  10940 Dec  3 19:12 1986 /usr/pixar/man/man3/libpirl.3
r--r--r--  0/10   7795 Dec  3 19:12 1986 /usr/pixar/man/man3/libpixar.3
rwxrwxrwx  0/10      0 Dec  4 17:08 1986 /usr/pixar/man/man4/
r--r--r--  0/10   8404 Dec  3 19:16 1986 /usr/pixar/man/man4/chap.4
r--r--r--  0/10   2159 Dec  3 19:16 1986 /usr/pixar/man/man4/dumi.4
r--r--r--  0/10   1928 Dec  3 19:16 1986 /usr/pixar/man/man4/mctrl.4
r--r--r--  0/10   1819 Dec  3 19:16 1986 /usr/pixar/man/man4/video.4
rwxrwxrwx  0/10      0 Dec  4 17:08 1986 /usr/pixar/man/man5/
r--r--r--  0/10   6432 Dec  3 19:16 1986 /usr/pixar/man/man5/chap.out.5
r--r--r--  0/10   7916 Dec  3 19:16 1986 /usr/pixar/man/man5/chapsym.5
```

```
rwxrwxrwx   0/10      0 Dec  4 17:08 1986 /usr/pixar/man/man7/
r--r--r--   0/10   3059 Dec  3 19:16 1986 /usr/pixar/man/man7/fbdefs.7
rwxrwxrwx   0/10      0 Dec  4 17:08 1986 /usr/pixar/man/man8/
r--r--r--   0/10    997 Dec  3 19:17 1986 /usr/pixar/man/man8/Diagnostic.8
r--r--r--   0/10   3217 Dec  3 19:17 1986 /usr/pixar/man/man8/chconfig.8
r--r--r--   0/10   2450 Dec  3 19:17 1986 /usr/pixar/man/man8/mctrl.8
rw-r--r--   0/10   2063 Dec  3 19:34 1986 /usr/pixar/man/pixar
rwxrwxrwx999/10      0 Dec  4 17:11 1986 /usr/pixar/fs/
rw-rw-rw-   0/10     74 Dec  4 17:11 1986 /usr/pixar/fs/.cshrc
rw-rw-rw-   0/10    148 Dec  4 17:11 1986 /usr/pixar/fs/.login
rw-r--r--   0/0  25771 Dec  4 17:58 1986 /usr/pixar/Verify.bin
```

merrell@flywheel

tape2.list

Mon Jul 13 09:20:58 1987

lw / Fluoride

```
Fluoride   flywheel:merrell   Job: tape2.list   Date: Mon Jul 13 09:20:58 1987
```


```
Fluoride   flywheel:merrell   Job: tape2.list   Date: Mon Jul 13 09:20:58 1987
Fluoride   flywheel:merrell   Job: tape2.list   Date: Mon Jul 13 09:20:58 1987
Fluoride   flywheel:merrell   Job: tape2.list   Date: Mon Jul 13 09:20:58 1987
Fluoride   flywheel:merrell   Job: tape2.list   Date: Mon Jul 13 09:20:58 1987
```

```
rw-r--r--  0/0   35797 Dec  5 21:04 1986 /usr/pixar/Verify.Gen.Src
rw-r--r--  0/10     41 Dec  2 13:47 1986 /usr/pixar/Version
rwxr-xr-x  0/10      0 Dec  5 20:54 1986 /usr/pixar/chap/
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/chap/src/
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/chap/src/lib/
rwxrwxrwx  0/10      0 Dec  5 20:54 1986 /usr/pixar/chap/src/lib/libcolor/
r--r--r--  0/10   1395 Dec  2 15:53 1986 /usr/pixar/chap/src/lib/libcolor/Makefile
r--r--r--  0/10   3934 Dec  2 15:26 1986 /usr/pixar/chap/src/lib/libcolor/rgb2xyY.s
r--r--r--  0/10  41466 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libcolor/xyY2dens.
r--r--r--  0/10   7812 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libcolor/dens2rgb.
r--r--r--  0/10   2557 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libcolor/rgb2XYZ.s
r--r--r--  0/10   2521 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libcolor/XYZ2rgb.s
r--r--r--  0/10   2720 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libcolor/mx3.s
rw-r--r--  0/10  12878 Dec  2 18:03 1986 /usr/pixar/chap/src/lib/libcolor/libcolor.
rwxrwxrwx  0/10      0 Dec  5 20:54 1986 /usr/pixar/chap/src/lib/libpG/
r--r--r--  0/10   2073 Dec  2 15:53 1986 /usr/pixar/chap/src/lib/libpG/Makefile
r--r--r--  0/10   2562 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpG/saveb.s
r--r--r--  0/10   2564 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpG/savei.s
r--r--r--  0/10   5210 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpG/saver.s
r--r--r--  0/10   2841 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpG/savev.s
r--r--r--  0/10   1994 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpG/stack.s
r--r--r--  0/10   8901 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpG/line.s
r--r--r--  0/10   6290 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpG/text.vs
r--r--r--  0/10   2047 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpG/dufftable.s
r--r--r--  0/10    717 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpG/table.h
rw-r--r--  0/10   4138 Dec  2 18:03 1986 /usr/pixar/chap/src/lib/libpG/libpG.a
rwxrwxrwx  0/10      0 Dec  5 20:54 1986 /usr/pixar/chap/src/lib/libpt/
r--r--r--  0/10   6695 Dec  3 11:26 1986 /usr/pixar/chap/src/lib/libpt/Makefile
r--r--r--  0/10   1842 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/sic.s
r--r--r--  0/10   2492 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/crc.s
r--r--r--  0/10   2360 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/rsc.s
r--r--r--  0/10   1335 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssclamp.s
r--r--r--  0/10   1564 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwclear.s
r--r--r--  0/10   2046 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ss4map.s
r--r--r--  0/10   1950 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssc.s
r--r--r--  0/10   2444 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssm.s
r--r--r--  0/10   2277 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssmi.s
r--r--r--  0/10   2773 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssmo.s
r--r--r--  0/10   2138 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ccc.s
r--r--r--  0/10   3488 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssp.s
r--r--r--  0/10   2445 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/sspo.s
r--r--r--  0/10   2044 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwaxb.s
r--r--r--  0/10   1289 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/sscmp.s
r--r--r--  0/10   2338 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssmout.s
r--r--r--  0/10   2710 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/fcc.s
r--r--r--  0/10   2029 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssaxb.s
r--r--r--  0/10   2374 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/sspc.s
r--r--r--  0/10  16522 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/fsc.s
r--r--r--  0/10  10097 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pw.s
r--r--r--  0/10  14009 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/sfc.s
r--r--r--  0/10   2339 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssmu.s
r--r--r--  0/10   2273 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwcha.s
r--r--r--  0/10   3495 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/sscha.s
r--r--r--  0/10   1100 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwclamp.s
r--r--r--  0/10   4268 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwcopy.s
r--r--r--  0/10   2220 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwgeneric.s
r--r--r--  0/10   2933 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwmerge.s
r--r--r--  0/10   1090 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwnot.s
r--r--r--  0/10   4275 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwshift.s
r--r--r--  0/10   2221 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwshuffle.s
r--r--r--  0/10   3108 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwswap.s
r--r--r--  0/10   2044 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwtranspose.
r--r--r--  0/10   1233 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssnot.s
r--r--r--  0/10   2678 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/sscomb.s
r--r--r--  0/10   3127 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ssma.s
r--r--r--  0/10   2954 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/shuffle.s
```

```
r--r--r--  0/10   2737 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/ffc.s
r--r--r--  0/10   2042 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pwmapc.s
r--r--r--  0/10   3987 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/sscompc.s
r--r--r--  0/10   1012 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/yap.h
r--r--r--  0/10   1803 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpt/pw4map.s
rw-r--r--  0/10 131976 Dec  3 11:32 1986 /usr/pixar/chap/src/lib/libpt/libpt.a
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/chap/src/lib/libpx/
r--r--r--  0/10   2847 Dec  2 15:54 1986 /usr/pixar/chap/src/lib/libpx/Makefile
r--r--r--  0/10   3622 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/hd4.s
r--r--r--  0/10   3599 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/vu4.s
r--r--r--  0/10   4839 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/vd4.s
r--r--r--  0/10   2031 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/hu1.s
r--r--r--  0/10   4194 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/mag1table.s
r--r--r--  0/10   3044 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/hu2.s
r--r--r--  0/10   3082 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/hd2.s
r--r--r--  0/10   5674 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/mag2table.s
r--r--r--  0/10   3947 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/min2table.s
r--r--r--  0/10   3865 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/min1table.s
r--r--r--  0/10   3273 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/vd2.s
r--r--r--  0/10   2967 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/vu2.s
r--r--r--  0/10   1964 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/vd1.s
r--r--r--  0/10   5346 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/setstwarptab
r--r--r--  0/10   1914 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/vu1.s
r--r--r--  0/10   7930 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/stwarp.s
r--r--r--  0/10   1903 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/ssshalve.s
r--r--r--  0/10   4422 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/min4table.s
r--r--r--  0/10   5927 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/mag4table.s
r--r--r--  0/10   1322 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/makereciptab
r--r--r--  0/10   1342 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/makerec16_25
r--r--r--  0/10   4656 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/makefilter4t
r--r--r--  0/10   3526 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/hu4.s
r--r--r--  0/10   3152 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/rotxx.s
r--r--r--  0/10   3126 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/rotyx.s
r--r--r--  0/10   3135 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/rotyy.s
r--r--r--  0/10   3486 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/pwshear.s
r--r--r--  0/10  10067 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/pwresize.s
r--r--r--  0/10   3121 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/shearxx.s
r--r--r--  0/10   3120 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/shearxy.s
r--r--r--  0/10   3117 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/shearyx.s
r--r--r--  0/10   3125 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/shearyy.s
r--r--r--  0/10   2815 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpx/pwshear_clr.
r--r--r--  0/10   2850 Dec  2 18:10 1986 /usr/pixar/chap/src/lib/libpx/rec16_256.s
r--r--r--  0/10  48267 Dec  2 18:11 1986 /usr/pixar/chap/src/lib/libpx/filter4table
rw-r--r--  0/10  51782 Dec  2 18:12 1986 /usr/pixar/chap/src/lib/libpx/libpx.a
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/chap/src/lib/libpip/
r--r--r--  0/10   2688 Dec  2 15:53 1986 /usr/pixar/chap/src/lib/libpip/Makefile
r--r--r--  0/10   2603 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/dhg.s
r--r--r--  0/10   2269 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/ssconv.s
r--r--r--  0/10   2595 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwc33.s
r--r--r--  0/10   2707 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwc33s.s
r--r--r--  0/10   2757 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwconv.s
r--r--r--  0/10   1655 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwdiv.s
r--r--r--  0/10   1944 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwhg.s
r--r--r--  0/10   1655 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwmul.s
r--r--r--  0/10   1534 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwrange.s
r--r--r--  0/10   1655 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwsub.s
r--r--r--  0/10   1319 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/ssadd.s
r--r--r--  0/10   1699 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/ssdiv.s
r--r--r--  0/10   1343 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/ssmul.s
r--r--r--  0/10   1163 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/ssrange.s
r--r--r--  0/10   1319 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/sssub.s
r--r--r--  0/10   1656 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwadd.s
r--r--r--  0/10   3044 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwboxfilter
r--r--r--  0/10   2393 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/ssboxfilter
r--r--r--  0/10   6290 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/c55s.s
r--r--r--  0/10   4898 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/c33s.s
```

```
r--r--r--  0/10   4665 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/c33.s
r--r--r--  0/10   1365 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/sscrc.s
r--r--r--  0/10   1790 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/pwcrc.s
r--r--r--  0/10   2641 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/bbox.s
r--r--r--  0/10   2280 Dec  2 15:27 1986 /usr/pixar/chap/src/lib/libpip/ssconv2.s
r--r--r--  0/10   3039 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpip/ssconv4.s
r--r--r--  0/10   4247 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpip/edge.s
r--r--r--  0/10  31257 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpip/pwline.s
rw-r--r--  0/10  35900 Dec  2 18:05 1986 /usr/pixar/chap/src/lib/libpip/libpip.a
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/chap/src/lib/libpm/
r--r--r--  0/10   2238 Dec  2 15:54 1986 /usr/pixar/chap/src/lib/libpm/Makefile
r--r--r--  0/10   6813 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/rec15_256.s
r--r--r--  0/10   5146 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/reciprocal.s
r--r--r--  0/10   6354 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/recsqrt16l.s
r--r--r--  0/10   3998 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/recsqrt32l.s
r--r--r--  0/10   3027 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/rrand.s
r--r--r--  0/10   1677 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/sqrt16.s
r--r--r--  0/10   5361 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/sqrt16l.s
r--r--r--  0/10   3999 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/sqrt32l.s
r--r--r--  0/10   8255 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/xp2.s
r--r--r--  0/10   5509 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/xp4.s
r--r--r--  0/10   1620 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/matvec32.s
r--r--r--  0/10   4865 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/matmul32.s
r--r--r--  0/10   2321 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/makerrlist.c
r--r--r--  0/10   1510 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/matmul16.s
r--r--r--  0/10   4340 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/fsqrt.s
r--r--r--  0/10   9910 Dec  2 15:29 1986 /usr/pixar/chap/src/lib/libpm/radialdist.s
r--r--r--  0/10   2045 Dec  2 18:12 1986 /usr/pixar/chap/src/lib/libpm/rrlist.s
rw-r--r--  0/10  22736 Dec  2 18:13 1986 /usr/pixar/chap/src/lib/libpm/libpm.a
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/chap/src/lib/libchad/
r--r--r--  0/10   2497 Dec  2 15:52 1986 /usr/pixar/chap/src/lib/libchad/Makefile
r--r--r--  0/10   3764 Dec  2 15:26 1986 /usr/pixar/chap/src/lib/libchad/newgetw.s
r--r--r--  0/10   6573 Dec  2 15:26 1986 /usr/pixar/chap/src/lib/libchad/newcmds.s
r--r--r--  0/10   6181 Dec  2 15:26 1986 /usr/pixar/chap/src/lib/libchad/newloop.s
r--r--r--  0/10   2321 Dec  2 15:26 1986 /usr/pixar/chap/src/lib/libchad/pxlio.s
r--r--r--  0/10  13004 Dec  2 15:26 1986 /usr/pixar/chap/src/lib/libchad/chad.s
r--r--r--  0/10   2371 Dec  2 15:26 1986 /usr/pixar/chap/src/lib/libchad/fbregs.h
rw-r--r--  0/10  16202 Dec  2 18:03 1986 /usr/pixar/chap/src/lib/libchad/libchad.a
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/chap/src/lib/libpicio/
r--r--r--  0/10   1754 Dec  2 15:53 1986 /usr/pixar/chap/src/lib/libpicio/Makefile
r--r--r--  0/10   8036 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpicio/duff8To12
r--r--r--  0/10  50488 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpicio/duff12To8
r--r--r--  0/10  27978 Dec  2 15:28 1986 /usr/pixar/chap/src/lib/libpicio/picio.s
rw-r--r--  0/10  24290 Dec  2 18:10 1986 /usr/pixar/chap/src/lib/libpicio/libpicio.
r--r--r--  0/10   1838 Nov 24 14:23 1986 /usr/pixar/chap/src/lib/Makefile
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/chap/src/bin/
r--r--r--  0/10   2615 Dec  2 15:54 1986 /usr/pixar/chap/src/bin/Makefile
r--r--r--  0/10   5696 Dec  2 15:29 1986 /usr/pixar/chap/src/bin/config.s
rwxr-xr-x  0/10   2769 Dec  2 18:13 1986 /usr/pixar/chap/src/bin/config
r--r--r--  0/10   1723 Nov 24 14:23 1986 /usr/pixar/chap/src/Makefile
rwxr-xr-x  0/10      0 Dec  5 20:58 1986 /usr/pixar/demo/
r--r--r--  0/10   1831 Dec  3 13:24 1986 /usr/pixar/demo/Makefile
rwxr-xr-x  0/10      0 Dec  5 20:57 1986 /usr/pixar/demo/lib/
rw-r--r--  0/10 143206 Dec  3 13:16 1986 /usr/pixar/demo/lib/libfbtool.a
rw-r--r--  0/10  38268 Dec  3 13:16 1986 /usr/pixar/demo/lib/cube.a
rwxr-xr-x  0/10  32950 Dec  3 13:16 1986 /usr/pixar/demo/lib/fht
rwxr-xr-x  0/10   1251 Dec  3 13:16 1986 /usr/pixar/demo/lib/blits.ucode
rwxr-xr-x  0/10  48340 Dec  3 13:16 1986 /usr/pixar/demo/lib/t.out.d
rwxr-xr-x  0/10      8 Dec  3 13:16 1986 /usr/pixar/demo/lib/t.stop
rwxr-xr-x  0/10      0 Dec  5 20:58 1986 /usr/pixar/demo/src/
rwxr-xr-x  0/10      0 Dec  5 20:57 1986 /usr/pixar/demo/src/fft/
r--r--r--  0/10   3729 Dec  3 13:16 1986 /usr/pixar/demo/src/fft/fftdemo.c
r--r--r--  0/10   1528 Dec  3 13:16 1986 /usr/pixar/demo/src/fft/fht.h
r--r--r--  0/10   5660 Dec  3 13:16 1986 /usr/pixar/demo/src/fft/convolve.s
r--r--r--  0/10  17045 Dec  3 13:16 1986 /usr/pixar/demo/src/fft/fft.s
r--r--r--  0/10  15031 Dec  3 13:16 1986 /usr/pixar/demo/src/fft/io.s
```

```
r--r--r--  0/10   6904 Dec  3 13:16 1986 /usr/pixar/demo/src/fft/main.s
r--r--r--  0/10  17034 Dec  3 13:16 1986 /usr/pixar/demo/src/fft/math.s
r--r--r--  0/10   2004 Dec  3 13:16 1986 /usr/pixar/demo/src/fft/perm_table.s
r--r--r--  0/10   4328 Dec  3 13:17 1986 /usr/pixar/demo/src/fft/trig_table.s
r--r--r--  0/10   2975 Dec  3 13:17 1986 /usr/pixar/demo/src/fft/Makefile
rwxr-xr-x  0/10  32950 Dec  3 13:17 1986 /usr/pixar/demo/src/fft/fht
rwxr-xr-x  0/10 876544 Dec  3 13:17 1986 /usr/pixar/demo/src/fft/fftdemo
rwxr-xr-x  0/10      0 Dec  5 20:58 1986 /usr/pixar/demo/src/fbtool/
rwxr-xr-x  0/10      0 Dec  5 20:57 1986 /usr/pixar/demo/src/fbtool/cube/
r--r--r--  0/10   1973 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/CubeIcon
r--r--r--  0/10   3370 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/Makefile
r--r--r--  0/10  11986 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/cubetool.c
r--r--r--  0/10   3588 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/rottable.c
r--r--r--  0/10   2437 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/ax.s
r--r--r--  0/10   6174 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/axial.s
r--r--r--  0/10   2384 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/cor.s
r--r--r--  0/10  10275 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/coronal.s
r--r--r--  0/10   7342 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/drawcube.s
r--r--r--  0/10   2444 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/sag.s
r--r--r--  0/10  10211 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/sagital.s
r--r--r--  0/10   3480 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/sscompc.s
rw-r--r--  0/10  38268 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/cube.a
r--r--r--  0/10   2292 Dec  3 13:18 1986 /usr/pixar/demo/src/fbtool/cube/cuberamp.c
rwxr-xr-x  0/10 188416 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/cube/cuberamp
rwxr-xr-x  0/10      0 Dec  5 20:58 1986 /usr/pixar/demo/src/fbtool/libfbtool/
r--r--r--  0/10   3270 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/libfbtool/Makef
r--r--r--  0/10   6629 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/libfbtool/item.
r--r--r--  0/10   2658 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/libfbtool/confi
r--r--r--  0/10   4370 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/libfbtool/fbtat
r--r--r--  0/10   1791 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/libfbtool/test.
r--r--r--  0/10   1971 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/libfbtool/FBIcc
r--r--r--  0/10  11857 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/libfbtool/fbtoc
r--r--r--  0/10   3209 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/libfbtool/sampl
rw-r--r--  0/10 142016 Dec  3 13:19 1986 /usr/pixar/demo/src/fbtool/libfbtool/libfk
rwxr-xr-x  0/10      0 Dec  5 20:58 1986 /usr/pixar/demo/src/fbtool/include/
r--r--r--  0/10   1805 Dec  3 13:20 1986 /usr/pixar/demo/src/fbtool/include/fbt_imp
r--r--r--  0/10   2791 Dec  3 13:20 1986 /usr/pixar/demo/src/fbtool/include/fbtattr
r--r--r--  0/10   2353 Dec  3 13:20 1986 /usr/pixar/demo/src/fbtool/include/fbtool.
r--r--r--  0/10  23175 Dec  3 13:20 1986 /usr/pixar/demo/src/fbtool/include/video.g
r--r--r--  0/10   1125 Dec  3 13:20 1986 /usr/pixar/demo/src/fbtool/include/rottabl
rwxr-xr-x  0/10      0 Dec  5 20:58 1986 /usr/pixar/demo/src/fbtool/magloop/
r--r--r--  0/10   1971 Dec  3 13:20 1986 /usr/pixar/demo/src/fbtool/magloop/MLIcon
r--r--r--  0/10  10381 Dec  3 13:20 1986 /usr/pixar/demo/src/fbtool/magloop/magloop
r--r--r--  0/10   2807 Dec  3 13:20 1986 /usr/pixar/demo/src/fbtool/magloop/Makefil
rwxr-xr-x  0/10      0 Dec  5 20:58 1986 /usr/pixar/demo/src/fbtool/video/
rw-r--r--  0/10    193 Dec  3 13:22 1986 /usr/pixar/demo/src/fbtool/video/ClockCur
r--r--r--  0/10   2823 Dec  3 13:22 1986 /usr/pixar/demo/src/fbtool/video/Makefile
r--r--r--  0/10   1970 Dec  3 13:22 1986 /usr/pixar/demo/src/fbtool/video/VIcon
r--r--r--  0/10  11467 Dec  3 13:22 1986 /usr/pixar/demo/src/fbtool/video/vtool.c
r--r--r--  0/10   1527 Dec  3 13:22 1986 /usr/pixar/demo/src/fbtool/video/vt.c
r--r--r--  0/10   1932 Dec  3 13:22 1986 /usr/pixar/demo/src/fbtool/fbt_merge.c
r--r--r--  0/10   2293 Dec  3 13:22 1986 /usr/pixar/demo/src/fbtool/Makefile
rwxr-xr-x  0/101048576 Dec  3 13:22 1986 /usr/pixar/demo/src/fbtool/fbt_merge
rwxr-xr-x  0/10      0 Dec  5 20:58 1986 /usr/pixar/demo/src/treestuff/
r--r--r--  0/10      8 Dec  3 13:22 1986 /usr/pixar/demo/src/treestuff/t.stop
r--r--r--  0/10   1309 Dec  3 13:22 1986 /usr/pixar/demo/src/treestuff/Makefile
r--r--r--  0/10   4932 Dec  3 13:22 1986 /usr/pixar/demo/src/treestuff/blit.c
r--r--r--  0/10  48340 Dec  3 13:22 1986 /usr/pixar/demo/src/treestuff/t.out.d
r--r--r--  0/10   2158 Dec  3 13:22 1986 /usr/pixar/demo/src/treestuff/blits.s
rwxr-xr-x  0/10   1251 Dec  3 13:22 1986 /usr/pixar/demo/src/treestuff/blits.ucode
rwxr-xr-x  0/10 311296 Dec  3 13:22 1986 /usr/pixar/demo/src/treestuff/blit
r--r--r--  0/10   1746 Dec  3 13:22 1986 /usr/pixar/demo/src/Makefile
r--r--r--  0/10   3892 Dec  3 13:22 1986 /usr/pixar/demo/src/Demo
rwxr-xr-x  0/10      0 Dec  5 20:58 1986 /usr/pixar/demo/bin/
rwxr-xr-x  0/101048576 Dec  3 13:23 1986 /usr/pixar/demo/bin/fbt_merge
rwxr-xr-x  0/101048576 Dec  3 13:23 1986 /usr/pixar/demo/bin/videotool
```

```
rwxr-xr-x  0/101048576 Dec  3 13:23 1986 /usr/pixar/demo/bin/cubetool
rwxr-xr-x  0/101048576 Dec  3 13:24 1986 /usr/pixar/demo/bin/magloop
rwxr-xr-x  0/10 876544 Dec  3 13:24 1986 /usr/pixar/demo/bin/fftdemo
rwxr-xr-x  0/10   3892 Dec  3 13:24 1986 /usr/pixar/demo/bin/Demo
rwxr-xr-x  0/10 311296 Dec  3 13:24 1986 /usr/pixar/demo/bin/blit
rwxr-xr-x  0/10 188416 Dec  3 13:24 1986 /usr/pixar/demo/bin/cuberamp
rwxr-xr-x  0/10      0 Dec  5 20:55 1986 /usr/pixar/host/
rwxr-xr-x  0/10      0 Dec  5 20:57 1986 /usr/pixar/host/src/
rwxrwxrwx  0/10      0 Dec  5 20:57 1986 /usr/pixar/host/src/lib/
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/host/src/lib/libG/
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/host/src/lib/libG/profiled/
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/host/src/lib/libaa/
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/host/src/lib/libaa/profiled/
r--r--r--559/10   4692 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aa_save.c
r--r--r--559/10   5840 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aaetof.c
r--r--r--559/10   3862 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aahelp.c
r--r--r--559/10   3049 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aais.c
r--r--r--559/10   8566 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aaparse.c
r--r--r--559/10   5330 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aarg.c
r--r--r--559/10  14150 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aarg.i.c
r--r--r--559/10   1490 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aawin.c
r--r--r--559/10   2063 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aarg.h
r--r--r--559/10   2056 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aarg.globals
r--r--r--559/10   1472 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aarg.i.h
r--r--r--559/10   2912 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/Makefile
r--r--r--559/10  13989 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aarg.3
r--r--r--559/10  13908 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aarg.i.3
r--r--r--559/10   6682 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aachkarg.c
r--r--r--559/10   3165 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/aaonoff.c
r--r--r--559/10   1082 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libaa/std.h
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/host/src/lib/libchad/
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/host/src/lib/libchad/profiled/
r--r--r--559/10   4513 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libchad/Makefile
r--r--r--559/10   2467 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libchad/chadram.c
r--r--r--559/10   9488 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libchad/chad.h
r--r--r--559/10   7408 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libchad/chad.c
r--r--r--559/10   2239 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libchad/Chad.h
r--r--r--559/10  17884 Dec  5 12:53 1986 /usr/pixar/host/src/lib/libchad/chadalloc.
r--r--r--559/10   5804 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libchad/nvideo.c
r--r--r--559/10  11329 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libchad/chadio.c
r--r--r--559/10   7409 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libchad/chaddevs.c
r--r--r--559/10   5251 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libchad/chaddevsca
r--r--r--559/10   4137 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libchad/ncursor.c
r--r--r--559/10  10214 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libchad/font.h
rwxrwxrwx  0/10      0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpicio/
rwxrwxrwx  0/10      0 Dec  5 20:55 1986 /usr/pixar/host/src/lib/libpicio/profiled/
r--r--r--559/10   7972 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/Makefile
r--r--r--559/10   2209 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpicio/screen.h
r--r--r--559/10    967 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/cpu.h
r--r--r--559/10   2774 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpicio/piccode.h
r--r--r--559/10   3276 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/rpacemu.h
r--r--r--559/10   3194 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpicio/picio.h
r--r--r--559/10   2164 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpicio/picture.h
r--r--r--559/10   1214 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/PD.c
r--r--r--559/10   1167 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/Pclose.c
r--r--r--559/10   5040 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/Pcreat.c
r--r--r--559/10   1579 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/Pfind.c
r--r--r--559/10   2486 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/Popen.c
r--r--r--559/10   3359 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/Preadbuff
r--r--r--559/10   6729 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/Preadfb.c
r--r--r--559/10   3539 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/Pwritebuf
r--r--r--559/10   2966 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/Pwritefb.
r--r--r--559/10  17092 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/dectile.c
r--r--r--559/10  22526 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/dectilefb
r--r--r--559/10   1437 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/deczero.c
r--r--r--559/10  21121 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/enctile.c
```

```
r--r--r--559/10   1871 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/encfb.c
r--r--r--559/10  17017 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/encfb8bit
r--r--r--559/10  17496 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/encfb12bi
r--r--r--559/10   3527 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/histogram
r--r--r--559/10   5974 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/picbyte.c
r--r--r--559/10   4761 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/piclabel.
r--r--r--559/10   9796 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/pixar.c
r--r--r--559/10  11887 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpicio/pixarfr.c
r--r--r--559/10   8152 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/Pputframe
r--r--r--559/10   3380 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/Pgetframe
r--r--r--559/10  22711 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/dectilefr
r--r--r--559/10   1897 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/encfr.c
r--r--r--559/10  17038 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/encfr8bit
r--r--r--559/10  17536 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/encfr12bi
r--r--r--559/10   8489 Dec  5 12:54 1986 /usr/pixar/host/src/lib/libpicio/Plerpfb.c
r--r--r--559/10   4300 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpicio/rpacmacs.
r--r--r--559/10   2717 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpicio/rpac.h
r--r--r--559/10   1736 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/mkduff.c
r--r--r--559/10  26573 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpicio/duff.c
rwxr-xr-x  0/10  32768 Dec  2 17:02 1986 /usr/pixar/host/src/lib/libpicio/mkduff
r--r--r--559/10   7293 Dec  5 12:55 1986 /usr/pixar/host/src/lib/libpicio/wops.c
rwxrwxrwx  0/10      0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpirl/
rwxrwxrwx  0/10      0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpirl/profiled/
r--r--r--559/10  17032 Dec  5 19:48 1986 /usr/pixar/host/src/lib/libpirl/Makefile
r--r--r--559/10   3335 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/begin.c
r--r--r--559/10   5658 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/cbars.c
r--r--r--559/10   2414 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/copy.c
r--r--r--559/10   1790 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/swap.c
r--r--r--559/10   1429 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/clear.c
r--r--r--559/10   1586 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/transpose.
r--r--r--559/10   2225 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/reflect.c
r--r--r--559/10   2106 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/cha.c
r--r--r--559/10   1799 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/pw.c
r--r--r--559/10   2408 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/shift.c
r--r--r--559/10   3300 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/map.c
r--r--r--559/10   4801 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/merge.c
r--r--r--559/10   1285 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/mask.c
r--r--r--559/10   1815 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/shuffle.c
r--r--r--559/10   1404 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/clamp.c
r--r--r--559/10   1395 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/not.c
r--r--r--559/10   2783 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/axb.c
r--r--r--559/10   3465 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/convolve1d
r--r--r--559/10   2291 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/c33.c
r--r--r--559/10   2425 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/c33s.c
r--r--r--559/10   3021 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/boxfilter.
r--r--r--559/10   1840 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/add.c
r--r--r--559/10   1849 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/sub.c
r--r--r--559/10   1850 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/mul.c
r--r--r--559/10   1842 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/div.c
r--r--r--559/10   1225 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/getframe.c
r--r--r--559/10   1226 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/putframe.c
r--r--r--559/10   2131 Dec  5 12:56 1986 /usr/pixar/host/src/lib/libpirl/hg.c
r--r--r--559/10   1991 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/range.c
r--r--r--559/10   3057 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/sweep.c
r--r--r--559/10   1763 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/ramp.c
r--r--r--559/10   2546 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/error.c
r--r--r--559/10    877 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/cbars.h
r--r--r--559/10   8568 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/resize.c
r--r--r--559/10   1564 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/crc.c
r--r--r--559/10   1850 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/buf.c
r--r--r--559/10   2151 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/getsv.c
r--r--r--559/10   1548 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/mapc.c
r--r--r--559/10   2082 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/bbox.c
r--r--r--559/10   4258 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/getraster.
r--r--r--559/10   5833 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/fbinq.c
r--r--r--559/10   4025 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/pirl.h
```

```
r--r--r--559/10    1306 Dec  5 12:57 1986 /usr/pixar/host/src/lib/libpirl/merge.h
r--r--r--559/10    1699 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/pirlxform.
r--r--r--559/10    3267 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/prexform.c
r--r--r--559/10   14694 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/pirlline.c
r--r--r--559/10    8065 Dec  5 19:48 1986 /usr/pixar/host/src/lib/libpirl/affine.c
r--r--r--559/10    1927 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/pirlmapcom
r--r--r--559/10    4913 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/scale.c
r--r--r--559/10     748 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpirl/mapn.c
rwxrwxrwx  0/10       0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpixar/
rwxrwxrwx  0/10       0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpixar/chap/
rwxrwxrwx  0/10       0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpixar/chap/prof
r--r--r--559/10   29016 Dec  5 12:58 1986 /usr/pixar/host/src/lib/libpixar/chap/Make
r--r--r--559/10    2399 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/afil
r--r--r--559/10    1195 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/allc
r--r--r--559/10    1199 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/allc
r--r--r--559/10    1201 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/allc
r--r--r--559/10   19048 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/arch
r--r--r--559/10    1203 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/clrb
r--r--r--559/10    2406 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/cont
r--r--r--559/10    2006 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/dump
r--r--r--559/10    2333 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/dump
r--r--r--559/10   25298 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/dyna
r--r--r--559/10    2070 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/errl
r--r--r--559/10    1768 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/errc
r--r--r--559/10    1529 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/exec
r--r--r--559/10    3387 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/fill
r--r--r--559/10    1192 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/free
r--r--r--559/10    1196 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/free
r--r--r--559/10    1198 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/free
r--r--r--559/10    1111 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/getc
r--r--r--559/10    1225 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/getf
r--r--r--559/10    1279 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/getm
r--r--r--559/10    1229 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/getr
r--r--r--559/10    1248 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/run.
r--r--r--559/10    1231 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/gets
r--r--r--559/10    1719 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/halt
r--r--r--559/10    2842 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/load
r--r--r--559/10    1425 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/load
r--r--r--559/10    2006 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/load
r--r--r--559/10    3507 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/load
r--r--r--559/10    1082 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/mman
r--r--r--559/10    5476 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/open
r--r--r--559/10    1613 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1781 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1463 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1402 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1650 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1438 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1914 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1798 Dec  5 12:59 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1157 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1835 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1526 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    2073 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    2801 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    2511 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/read
r--r--r--559/10    1594 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/rese
r--r--r--559/10    1094 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/rese
r--r--r--559/10    1503 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/runa
r--r--r--559/10    1201 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/setb
r--r--r--559/10    1111 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/setc
r--r--r--559/10    1612 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/step
r--r--r--559/10   20898 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/symt
r--r--r--559/10    2287 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/wait
r--r--r--559/10    1930 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
r--r--r--559/10    1446 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
```

```
r--r--r--559/10   1869 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
r--r--r--559/10   1354 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
r--r--r--559/10   1880 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
r--r--r--559/10   1956 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
r--r--r--559/10   1496 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
r--r--r--559/10   2330 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
r--r--r--559/10   3016 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
r--r--r--559/10   2107 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/writ
r--r--r--559/10   2103 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/load
r--r--r--559/10   1268 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/flag
r--r--r--559/10    993 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/sett
r--r--r--559/10   9429 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/mall
r--r--r--559/10   3468 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/chap
r--r--r--559/10   3223 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/chap
r--r--r--559/10  11813 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/alu.
r--r--r--559/10   8014 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/chap
r--r--r--559/10   1658 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/chap
r--r--r--559/10  10619 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/chap
r--r--r--559/10   3797 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/relo
r--r--r--559/10   2317 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/chap
r--r--r--559/10   2759 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/mman
r--r--r--559/10   2105 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/pbus
r--r--r--559/10   1984 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/diag
r--r--r--559/10   2799 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/yapb
r--r--r--559/10   1134 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/pbus
r--r--r--559/10   1802 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/pw.h
r--r--r--559/10   1322 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/ucal
r--r--r--559/10   1763 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/wrun
r--r--r--559/10   1320 Dec  5 13:00 1986 /usr/pixar/host/src/lib/libpixar/chap/envi
r--r--r--559/10   1913 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/chap
r--r--r--559/10   9597 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/chap/chap
rwxrwxrwx   0/10      0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpixar/video/
rwxrwxrwx   0/10      0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpixar/video/pro
rwxrwxrwx   0/10      0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--   0/10   1771 Nov 24 17:05 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--   0/10   1641 Nov 24 17:05 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--   0/10   1924 Nov 24 17:05 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--   0/10   1503 Nov 24 17:05 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--   0/10   1483 Nov 24 17:05 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--   0/10   1891 Nov 24 17:05 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--559/10   5994 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/Mak
r--r--r--559/10   1546 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/cmg
r--r--r--559/10   2070 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/cms
r--r--r--559/10   2212 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--559/10   2563 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--559/10   1281 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--559/10   1364 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--559/10   1800 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--559/10   1161 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/get
r--r--r--559/10   2222 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/get
r--r--r--559/10   2585 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/pos
r--r--r--559/10   1259 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/set
r--r--r--559/10   3770 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/vop
r--r--r--559/10   1713 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/zoo
r--r--r--559/10  10446 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/gac
r--r--r--559/10   2053 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/cur
r--r--r--559/10   2261 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/vid
r--r--r--559/10   3403 Dec  5 13:01 1986 /usr/pixar/host/src/lib/libpixar/video/vid
rwxrwxrwx   0/10      0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libpixar/profiled/
rw-r--r--   0/10  57114 Dec  3 11:40 1986 /usr/pixar/host/src/lib/libpixar/profiled/
r--r--r--559/10   5185 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/Makefile
r--r--r--559/10   2334 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/disk.c
r--r--r--559/10   2864 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/dumi.c
r--r--r--559/10   2904 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/mctrl.c
r--r--r--559/10   1442 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/pixar.h
r--r--r--559/10   1789 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/dumireg.h
```

```
r--r--r--559/10   6785 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/mctrlreg.
r--r--r--559/10   3174 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/yumi.c
r--r--r--559/10   1793 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/yumioctl.
r--r--r--559/10   5783 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libpixar/yumireg.h
rwxrwxrwx  0/10      0 Dec  5 20:57 1986 /usr/pixar/host/src/lib/libcolr/
rwxrwxrwx  0/10      0 Dec  5 20:56 1986 /usr/pixar/host/src/lib/libcolr/profiled/
r--r--r--559/10   2413 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libcolr/Makefile
r--r--r--559/10   2684 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libcolr/RgbToHsv.c
r--r--r--559/10    905 Dec  5 13:02 1986 /usr/pixar/host/src/lib/libcolr/colr.h
rwxrwxrwx  0/10      0 Dec  5 20:57 1986 /usr/pixar/host/src/lib/librG/
rwxrwxrwx  0/10      0 Dec  5 20:57 1986 /usr/pixar/host/src/lib/librG/profiled/
r--r--r--559/10   3452 Dec  5 13:03 1986 /usr/pixar/host/src/lib/librG/Makefile
r--r--r--559/10  13885 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/fbdefs.c
r--r--r--559/10   6197 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/drand.c
r--r--r--559/10   1415 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/fbaarg.h
r--r--r--559/10   1482 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/fbdefs.h
r--r--r--559/10   3514 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/isqrt.c
r--r--r--559/10   3186 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/random.c
r--r--r--559/10    995 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/random.h
r--r--r--559/10   3491 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/rrand.c
r--r--r--559/10   1057 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/rrand.h
r--r--r--559/10   1514 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/wallinterval
r--r--r--559/10   2542 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/aa_fb.c
r--r--r--559/10   1968 Dec  5 13:02 1986 /usr/pixar/host/src/lib/librG/aa_setcolor.
r--r--r--559/10   1317 Dec  5 13:03 1986 /usr/pixar/host/src/lib/librG/environ.h
r--r--r--559/10   1129 Dec  5 13:03 1986 /usr/pixar/host/src/lib/librG/coloraarg.h
r--r--r--559/10   1127 Dec  5 13:03 1986 /usr/pixar/host/src/lib/librG/constants.h
r--r--r--559/10   1340 Dec  5 13:03 1986 /usr/pixar/host/src/lib/librG/math.h
r--r--r--559/10   3397 Dec  5 13:03 1986 /usr/pixar/host/src/lib/librG/gfxtypes.h
r--r--r--559/10   1988 Dec  5 13:03 1986 /usr/pixar/host/src/lib/librG/pixwin.h
r--r--r--559/10   2625 Dec  5 13:03 1986 /usr/pixar/host/src/lib/librG/LineDraw.c
r--r--r--559/10   1516 Dec  5 13:03 1986 /usr/pixar/host/src/lib/librG/pixeldef.h
rwxrwxrwx  0/10      0 Dec  5 20:57 1986 /usr/pixar/host/src/lib/libport/
rwxrwxrwx  0/10      0 Dec  5 20:57 1986 /usr/pixar/host/src/lib/libport/profiled/
r--r--r--559/10   2683 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/Makefile
r--r--r--559/10   1410 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/ffs.c
r--r--r--559/10   1361 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/flock.c
r--r--r--559/10   1041 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/fork.c
r--r--r--559/10   1100 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/getpagesiz
r--r--r--559/10    920 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/random.c
r--r--r--559/10   1884 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/readv.c
r--r--r--559/10   1502 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/rename.c
r--r--r--559/10   1119 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/setlinebuf
r--r--r--559/10   1275 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/uio.h
r--r--r--559/10   1296 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/valloc.c
r--r--r--559/10   1182 Dec  5 13:03 1986 /usr/pixar/host/src/lib/libport/filestuff.
r--r--r--559/10   1991 Dec  5 13:04 1986 /usr/pixar/host/src/lib/Makefile
rwxr-xr-x  0/10      0 Dec  5 20:57 1986 /usr/pixar/host/src/bin/
rwxrwxrwx  0/10      0 Dec  5 20:57 1986 /usr/pixar/host/src/bin/loop/
r--r--r--559/10   2285 Dec  5 13:06 1986 /usr/pixar/host/src/bin/loop/Makefile
r--r--r--559/10   8169 Dec  5 13:06 1986 /usr/pixar/host/src/bin/loop/loop.c
r--r--r--559/10   1196 Dec  5 13:06 1986 /usr/pixar/host/src/bin/loop/loop.h
r--r--r--559/10   2903 Dec  5 13:06 1986 /usr/pixar/host/src/bin/loop/nap.c
r--r--r--559/10   4415 Dec  5 13:06 1986 /usr/pixar/host/src/bin/loop/poll.c
r--r--r--559/10  27378 Dec  5 17:20 1986 /usr/pixar/host/src/bin/Makefile
r--r--r--559/10   3600 Dec  5 13:07 1986 /usr/pixar/host/src/bin/MxMatrix.h
r--r--r--559/10   7642 Dec  5 13:07 1986 /usr/pixar/host/src/bin/blur.c
r--r--r--559/10   3332 Dec  5 13:06 1986 /usr/pixar/host/src/bin/cbars.c
r--r--r--559/10   5437 Dec  5 13:07 1986 /usr/pixar/host/src/bin/clamp.c
r--r--r--559/10   5317 Dec  5 13:07 1986 /usr/pixar/host/src/bin/clr.c
r--r--r--559/10   7264 Dec  5 13:06 1986 /usr/pixar/host/src/bin/conv.c
r--r--r--559/10   6827 Dec  5 13:07 1986 /usr/pixar/host/src/bin/copy.c
r--r--r--559/10   2990 Dec  5 13:07 1986 /usr/pixar/host/src/bin/crc.c
r--r--r--559/10   2483 Dec  5 13:07 1986 /usr/pixar/host/src/bin/cursor.h
r--r--r--559/10    827 Dec  5 13:06 1986 /usr/pixar/host/src/bin/gamma.sh
r--r--r--559/10   6723 Dec  5 13:06 1986 /usr/pixar/host/src/bin/gt.c
```

```
r--r--r--559/10  10494 Dec  5 13:06 1986 /usr/pixar/host/src/bin/gtinfo.c
r--r--r--559/10   4847 Dec  5 13:07 1986 /usr/pixar/host/src/bin/guide.c
r--r--r--559/10   9062 Dec  5 13:06 1986 /usr/pixar/host/src/bin/hg.c
r--r--r--559/10   9432 Dec  5 13:06 1986 /usr/pixar/host/src/bin/malloc.c
r--r--r--559/10  10785 Dec  5 13:06 1986 /usr/pixar/host/src/bin/mctrl.c
r--r--r--559/10   8398 Dec  5 13:07 1986 /usr/pixar/host/src/bin/merge.c
r--r--r--559/10   9210 Dec  5 13:06 1986 /usr/pixar/host/src/bin/perm.c
r--r--r--559/10   1924 Dec  5 13:07 1986 /usr/pixar/host/src/bin/pixinit.sh
r--r--r--559/10  11085 Dec  5 13:07 1986 /usr/pixar/host/src/bin/ramp.c
r--r--r--559/10   6911 Dec  5 13:07 1986 /usr/pixar/host/src/bin/resize.c
r--r--r--559/10   7012 Dec  5 13:07 1986 /usr/pixar/host/src/bin/rotate.c
r--r--r--559/10   8594 Dec  5 13:06 1986 /usr/pixar/host/src/bin/scale.c
r--r--r--559/10   6615 Dec  5 13:07 1986 /usr/pixar/host/src/bin/see.c
r--r--r--559/10   8899 Dec  5 13:06 1986 /usr/pixar/host/src/bin/sv.c
r--r--r--559/10  14178 Dec  5 13:07 1986 /usr/pixar/host/src/bin/tool.c
r--r--r--559/10   8685 Dec  5 13:06 1986 /usr/pixar/host/src/bin/video.c
r--r--r--559/10  23153 Dec  5 13:07 1986 /usr/pixar/host/src/bin/video.gammamaps.h
```

merrell@flywheel

tape3.list

Mon Jul 13 08:56:20 1987

lw / Fluoride

```
Fluoride  flywheel:merrell  Job: tape3.list  Date: Mon Jul 13 08:56:20 1987
Fluoride  flywheel:merrell  Job: tape3.list  Date: Mon Jul 13 08:56:20 1987
Fluoride  flywheel:merrell  Job: tape3.list  Date: Mon Jul 13 08:56:20 1987
Fluoride  flywheel:merrell  Job: tape3.list  Date: Mon Jul 13 08:56:20 1987
Fluoride  flywheel:merrell  Job: tape3.list  Date: Mon Jul 13 08:56:20 1987
```

```
rwxr-xr-x  0/10       0 Dec  4 18:16 1986 /usr/pixar/
rwxr-xr-x  0/10       0 Dec  4 18:06 1986 /usr/pixar/demo/
rwxr-xr-x  0/10       0 Dec  4 18:14 1986 /usr/pixar/demo/pix/
rw-r--r--  0/10     135 Dec  3 17:00 1986 /usr/pixar/demo/pix/README
r--r--r--  0/107340032 Dec  3 16:09 1986 /usr/pixar/demo/pix/antenna.half
r--r--r--  0/1012722176 Dec  3 16:14 1986 /usr/pixar/demo/pix/awb.loop
r--r--r--  0/101990656 Dec  3 16:09 1986 /usr/pixar/demo/pix/fft.screen
rw-r--r--  0/0    217 Dec  4 18:16 1986 /usr/pixar/Verify.pic1
```

merrell@flywheel

tape4.list

Mon Jul 13 08:19:32 1987

lw / Fluoride

```
Fluoride   flywheel:merrell   Job: tape4.list   Date: Mon Jul 13 08:19:32 1987
```


```
Fluoride   flywheel:merrell   Job: tape4.list   Date: Mon Jul 13 08:19:32 1987
Fluoride   flywheel:merrell   Job: tape4.list   Date: Mon Jul 13 08:19:32 1987
Fluoride   flywheel:merrell   Job: tape4.list   Date: Mon Jul 13 08:19:32 1987
Fluoride   flywheel:merrell   Job: tape4.list   Date: Mon Jul 13 08:19:32 1987
```

```
rwxr-xr-x  0/10       0 Dec  4 18:32 1986 /usr/pixar/
rwxr-xr-x  0/10       0 Dec  4 18:21 1986 /usr/pixar/demo/
rwxr-xr-x  0/10       0 Dec  4 18:27 1986 /usr/pixar/demo/pix/
rw-r--r--  0/10     135 Dec  3 17:00 1986 /usr/pixar/demo/pix/README
r--r--r--  0/102547712 Dec  3 16:14 1986 /usr/pixar/demo/pix/1984
r--r--r--  0/1012673024 Dec  3 16:11 1986 /usr/pixar/demo/pix/fruit.4M
rwxr-xr-x  0/10       0 Dec  4 18:30 1986 /usr/pixar/demo/pix/trees/
r--r--r--  0/10 802816 Dec  3 16:14 1986 /usr/pixar/demo/pix/trees/trees.comp
r--r--r--  0/104792320 Dec  3 16:16 1986 /usr/pixar/demo/pix/trees/trees.elm
r--r--r--  0/10 696320 Dec  3 16:16 1986 /usr/pixar/demo/pix/trees/trees.bckgrnd
rw-r--r--  0/0    339 Dec  4 18:31 1986 /usr/pixar/Verify.pic2
```