

ROM Kernel Manual

Volume 2

Appendix A

Routine Summaries

This appendix contains Unix-like summaries for the routines that are built into the Amiga ROM (or kickstart) software, as well as summaries of routines in disk-loadable libraries. The debug library documentation is included here as well.

These documentation files are organized on a functional basis. For example, things pertinent to Exec are listed under "exec.library/someDocFile". Things pertinent to graphics are listed under "graphics.library/someDocFile".

The tutorial sections of this manual give you information that shows how these routines relate to each other and the prerequisites for calling them.

Most routines are listed as part of a library of routines. Before you can use a routine within your program, you must assure that the library is opened. This is explained fully in the "Libraries" chapter of this manual but it bears repeating here. You open a library by using the **OpenLibrary()** function. The call takes the form:

```
struct LibBase *LibBase;  
LibBase = OpenLibrary("library.name",version);
```

where:

library.name

is a string that describes the name of the library you wish to open.

version

is the version number of the library that you wish to have opened. A value of 0 says give me any version. A value of 31, for example (latest as of this writing) says open specifically version 31 of this library or greater if 31 is not available.

If the library is disk-resident, it gets loaded and initialized. The **OpenLibrary()** returns the address of the library base, which you must assign to a specific variable. It is through this means that your program links into the library-specific interface code that is contained in amiga.lib.

The following is a list of the names of the libraries that are currently part of the Amiga software and the corresponding names of the library base pointers associated with them:

| Library Name | Library Base Pointer Name |
|-------------------------|---------------------------|
| exec.library | ExecBase |
| clist.library | ClistBase |
| graphics.library | GfxBase |
| layers.library | LayersBase |
| intuition.library | IntuitionBase |
| mathfp.library | MathBase |
| mathtrans.library | MathTransBase |
| mathieeedoubbas.library | MathIeeeDoubBasBase |
| dos.library | DosBase |
| translator.library | TranslatorBase |
| icon.library | IconBase |
| diskfont.library | DiskfontBase |
| ramlib.library | --- |

(not useful to C language)

Example:

```
#include "graphics/gfx.h"
struct GfxBase *GfxBase;
GfxBase = OpenLibrary("graphics.library",0);
if(GfxBase == NULL) exit(NO_GRAPHICS_LIBRARY_FOUND);
```

Note: If your program is coming up through the normal startup code (see the Workbench chapter of this manual) the exec.library and dos.library are already opened for you. Thus you needn't open them yourself.

The reason for this code are

1. C, when calling a routine, takes the parameters for the routine and pushes them onto the stack. For example:

```
x = Routine(parmA, parmB);
```

Then it calls a routine named "_Routine" (adds an underscore to the head of the routine name).

2. The underlying ROM (or disk based) code usually expects its parameters to be passed in registers rather than on the stack. This is to make it truly general purpose (does not impose a particular stack frame), and more efficient for assembly language coding.

Therefore, the interface code at **_Routine**, in turn, saves the contents of registers the routine will use and pulls parameters off of the stack and jams them into

registers, finally passing control directly to the actual starting location of the routine itself.

The linker needs the library base location since it is through a jump-with-offset from a machine register that the **_Routine** entry point is found. (The Amiga uses a relocating loader in AmigaDOS, so you can never be sure exactly where a library of routines is located. However once the system has loaded a library, it knows how and where to find it and gives you a way to use the library's routines.)

The following shows typical interface code linked to your program from `amiga.lib`:

```
xref _LibBase      ;Library base name is defined in
                  ;user's file, this code gets linked
                  ;to user's program, get the value
                  ;from there when library is opened.

xdef _Routine
                  ;make _Routine name external,
                  ;visible to linker.

_Routine:
  move.l    A6,-(sp)      ;save register(s)
  move.l    8(sp),A0/A1   ;copy parms A and B to regs.
  move.l    _LibBase,A6   ;load library base address
  jsr      _LVORoutine(A6);go to real routine
  move.l    (sp)+,A6      ;restore registers
  rts
```

where **_LVORoutine** is a value representing the offset, within the library, at which the "real" routine (the one that expects parameters in registers) is located.

When you are finished using a library, at the end of your program, you should close it, using the `CloseLibrary` function as follows:

```
CloseLibrary(LibBase);
```

If the system is running out of memory and needs to free up space, it can check the library-accessors field for various libraries and for those whose accessors value is zero, it can retrieve the memory that the library had used.

| | |
|-----------------|------------------------|
| abs | |
| AddAnimOb | mathffp library |
| AddBob | graphics library[gels] |
| AddDevice | graphics library[gels] |
| AddFont | exec library |
| AddFreeList | graphics library[text] |
| AddGadget | icon library |
| AddHead | intuition library |
| AddIntServer | exec library |
| AddLibrary | exec library |
| AddPort | exec library |
| AddResource | exec library |
| AddTail | exec library |
| AddTask | exec library |
| AddVSprite | exec library |
| Allocate | graphics library[gels] |
| AllocCList | exec library |
| AllocEntry | clist library |
| AllocMem | exec library |
| AllocRaster | exec library |
| AllocRemember | graphics library |
| AllocSignal | intuition library |
| AllocTrap | exec library |
| AllocWBOject | exec library |
| AndRectRegion | icon library |
| Animate | graphics library |
| AreaDraw | graphics library[gels] |
| AreaEnd | graphics library |
| AreaMove | graphics library |
| arnd | graphics library |
| AskFont | mathlink_lib library |
| AskSoftStyle | graphics library[text] |
| AutoRequest | graphics library[text] |
| AvailFonts | intuition library |
| AvailMem | diskfont library |
| BeginRefresh | exec library |
| BeginUpdate | intuition library |
| BehindLayer | layers library |
| BltBitMap | layers library |
| BltClear | graphics library[text] |
| BltPattern | graphics library |
| BltTemplate | graphics library |
| BuildSysRequest | graphics library[text] |
| BumpRevision | intuition library |
| Cause | icon library |
| CEND | exec library |
| ChangeSprite | graphics library |
| CheckIO | graphics library |
| CINIT | exec library |
| ClearDMRequest | graphics library |
| ClearEOL | intuition library |
| ClearMenuStrip | graphics library[text] |
| ClearPointer | intuition library |
| ClearRegion | intuition library |
| ClearScreen | graphics library |
| Close | graphics library[text] |
| | dos library |

| | |
|--------------------|------------------------|
| Close | translator library |
| CloseDevice | exec library |
| CloseFont | graphics library[text] |
| CloseLibrary | exec library |
| CloseScreen | intuition library |
| CloseWindow | intuition library |
| CloseWorkBench | intuition library |
| CMOVE | graphics library |
| ColdReset | exec library |
| ConcatCList | clist library |
| CopyCList | clist library |
| CopySBitMap | graphics library |
| CreateBehindLayer | layers library |
| CreateDir | dos library |
| CreatePort | exec_support library |
| CreateProc | dos library |
| CreateStdIO | exec_support library |
| CreateTask | exec_support library |
| CreateUpfrontLayer | layers library |
| CurrentDir | dos library |
| CurrentTime | intuition library |
| CWAIT | graphics library |
| DateStamp | dos library |
| dbf | mathlink_lib library |
| Deallocate | exec library |
| Delay | dos library |
| DeleteFile | dos library |
| DeleteLayer | layers library |
| DeletePort | exec_support library |
| DeleteStdIO | exec_support library |
| DeleteTask | exec_support library |
| DeviceProc | dos library |
| DisownBlitter | graphics library |
| DisplayAlert | intuition library |
| DisplayBeep | intuition library |
| DisposeLayerInfo | layers library |
| DisposeRegion | graphics library |
| DoCollision | graphics library[gels] |
| DoIO | exec library |
| DoubleClick | intuition library |
| Draw | graphics library |
| DrawBorder | intuition library |
| DrawGList | graphics library[gels] |
| DrawImage | intuition library |
| DupLock | dos library |
| EndRefresh | intuition library |
| EndRequest | intuition library |
| EndUpdate | layers library |
| Enqueue | exec library |
| Examine | dos library |
| execute | dos library |
| Exit | dos library |
| ExNext | dos library |
| faddi | dos library |
| FattenLayerInfo | mathfp library |
| fcmpi | layers library |
| | mathfp library |

| | |
|-------------------|-------------------------|
| fdivi | mathffp library |
| fflti | mathffp library |
| FindName | exec library |
| FindPort | exec library |
| FindTask | exec library |
| FindToolType | icon library |
| Flood | graphics library |
| FlushCList | clist library |
| fmuli | mathffp library |
| fnegi | mathffp library |
| fpa | mathlink_lib library |
| fpbcd | mathlink_lib library |
| FreeCList | clist library |
| FreeColorMap | graphics library |
| FreeCopList | graphics library |
| FreeCprList | graphics library |
| FreeDiskObject | icon library |
| FreeEntry | exec library |
| FreeFreeList | icon library |
| FreeGBuffers | graphics library[gels] |
| FreeMem | exec library |
| FreeRaster | graphics library |
| FreeRemember | intuition library |
| FreeSignal | exec library |
| FreeSprite | graphics library |
| FreeSysRequest | intuition library |
| FreeTrap | exec library |
| FreeVPortCopLists | graphics library |
| FreeWBOobject | icon library |
| fsubi | mathffp library |
| ftsti | mathffp library |
| GetCC | exec library |
| GetCLBuf | clist library |
| GetCLChar | clist library |
| GetCLWord | clist library |
| GetColorMap | graphics library |
| GetDefPrefs | intuition library |
| GetDiskObject | icon library |
| GetGBuffers | graphics library[gels] |
| GetIcon | icon library |
| GetMsg | exec library |
| GetPrefs | intuition library |
| GetRGB4 | graphics library |
| GetSprite | graphics library |
| GetWBOobject | icon library |
| IEEEDPAbs | mathieeedoubbas library |
| IEEEDPAdd | mathieeedoubbas library |
| IEEEDPComp | mathieeedoubbas library |
| IEEEDPDiv | mathieeedoubbas library |
| IEEEDPFlt | mathieeedoubbas library |
| IEEEPML | mathieeedoubbas library |
| IEEEPNeg | mathieeedoubbas library |
| IEEEPSub | mathieeedoubbas library |
| IEEEPTst | mathieeedoubbas library |
| IncrCLMark | clist library |
| Info | dos library |

| | |
|--------------------|------------------------|
| InitArea | graphics library |
| InitBitMap | graphics library |
| InitCLPool | clist library |
| InitGels | graphics library[gels] |
| InitGMasks | graphics library[gels] |
| InitLayers | layers library |
| InitMasks | graphics library[gels] |
| InitRastPort | graphics library |
| InitRequester | intuition library |
| InitStruct | exec library |
| InitTmpRas | graphics library |
| InitView | graphics library |
| InitVPort | graphics library |
| Input | dos library |
| Insert | exec library |
| IntuiTextLength | intuition library |
| IoErr | dos library |
| IsInteractive | dos library |
| ItemAddress | intuition library |
| LoadRGB4 | graphics library |
| LoadSeg | dos library |
| LoadView | graphics library |
| Lock | dos library |
| LockLayer | layers library |
| LockLayerInfo | layers library |
| LockLayerRom | graphics library |
| LockLayers | layers library |
| MakeLibrary | exec library |
| MakeScreen | intuition library |
| MakeVPort | graphics library |
| MarkCList | clist library |
| MatchToolValue | icon library |
| ModifyIDCMP | intuition library |
| ModifyProp | intuition library |
| Move | graphics library |
| MoveLayer | layers library |
| MoveLayerInFrontOf | layers library |
| MoveScreen | intuition library |
| MoveSprite | graphics library |
| MoveWindow | intuition library |
| MrgCop | graphics library |
| NewLayerInfo | layers library |
| NewList | exec_support library |
| NewRegion | graphics library |
| OffGadget | intuition library |
| OffMenu | intuition library |
| OnGadget | intuition library |
| OnMenu | intuition library |
| Open | dos library |
| Open | translator library |
| OpenDevice | exec library |
| OpenDiskFont | diskfont library |
| OpenFont | graphics library[text] |
| OpenLibrary | exec library |
| OpenResource | exec library |
| OpenScreen | intuition library |

| | |
|----------------|------------------------|
| OpenWindow | intuition library |
| OpenWorkBench | intuition library |
| OrRectRegion | graphics library |
| Output | dos library |
| OwnBlitter | graphics library |
| ParentDir | dos library |
| PeekCLMark | clist library |
| PolyDraw | graphics library |
| PrintIText | intuition library |
| PutCLBuf | clist library |
| PutCLChar | clist library |
| PutCLWord | clist library |
| PutDiskObject | icon library |
| PutIcon | icon library |
| PutMsg | exec library |
| PutWBObject | icon library |
| QBlit | graphics library |
| QBSBlit | graphics library |
| Read | dos library |
| ReadPixel | graphics library |
| RectFill | graphics library |
| RefreshGadgets | intuition library |
| RemakeDisplay | intuition library |
| RemDevice | exec library |
| RemFont | graphics library[text] |
| RemHead | exec library |
| RemIBob | graphics library[gels] |
| RemIntServer | exec library |
| RemLibrary | exec library |
| Remove | exec library |
| RemoveGadget | intuition library |
| RemPort | exec library |
| RemResource | exec library |
| RemTail | exec library |
| RemTask | exec library |
| RemVSprite | graphics library[gels] |
| Rename | dos library |
| ReplyMsg | exec library |
| ReportMouse | intuition library |
| Request | intuition library |
| RethinkDisplay | intuition library |
| ScreenToBack | intuition library |
| ScreenToFront | intuition library |
| ScrollLayer | layers library |
| ScrollRaster | graphics library |
| ScrollVPort | graphics library |
| Seek | dos library |
| SendIO | exec library |
| SetAPen | graphics library |
| SetBPen | graphics library |
| SetCollision | graphics library[gels] |
| SetComment | dos library |
| SetDMRequest | intuition library |
| SetDrMd | graphics library |
| SetExcept | exec library |
| SetFont | graphics library[text] |

| | |
|--------------------------|------------------------|
| SetFunction | exec library |
| SetIntVector | exec library |
| SetMenuStrip | intuition library |
| SetPointer | intuition library |
| SetProtection | dos library |
| SetRast | graphics library |
| SetRGB4 | graphics library |
| SetSignal | exec library |
| SetSoftStyle | graphics library[text] |
| SetSR | exec library |
| SetTaskPri | exec library |
| SetWindowTitles | intuition library |
| ShowTitle | intuition library |
| Signal | exec library |
| SizeCList | clist library |
| SizeLayer | layers library |
| SizeWindow | intuition library |
| SortGLList | graphics library[gels] |
| SPAbs | mathffp library |
| SPAcos | mathtrans library |
| SPAdd | mathffp library |
| SPAsin | mathtrans library |
| SPAtan | mathtrans library |
| SPCmp | mathffp library |
| SPCos | mathtrans library |
| SPCosh | mathtrans library |
| SPDiv | mathffp library |
| SPExp | mathtrans library |
| SPFieee | mathtrans library |
| SPFlt | mathffp library |
| SplitCList | clist library |
| SPLog | mathtrans library |
| SPLog10 | mathtrans library |
| SPMul | mathffp library |
| SPNeg | mathffp library |
| SPPow | mathtrans library |
| SPSin | mathtrans library |
| SPSincos | mathtrans library |
| SPSinh | mathtrans library |
| SPSqrt | mathtrans library |
| SPSub | mathffp library |
| SPTanh | mathtrans library |
| SPTieee | mathtrans library |
| SPTst | mathffp library |
| SubCList | clist library |
| SumLibrary | exec library |
| SuperState | exec library |
| SwapBitsRastPortClipRect | layers library |
| SyncSBitMap | graphics library |
| tan | mathtrans library |
| Text | graphics library[text] |
| TextLength | graphics library[text] |
| ThinLayerInfo | layers library |
| Translate | translator library |
| UnGetCLChar | clist library |
| UnGetCLWord | clist library |

| | |
|-----------------|-------------------|
| UnLoadSeg | dos library |
| UnLock | dos library |
| UnlockLayer | layers library |
| UnlockLayerInfo | layers library |
| UnlockLayerRom | graphics library |
| UnlockLayers | layers library |
| UnPutCLChar | clist library |
| UnPutCLWord | clist library |
| UpfrontLayer | layers library |
| UserState | exec library |
| VBeamPos | graphics library |
| ViewAddress | intuition library |
| ViewPortAddress | intuition library |
| Wait | exec library |
| WaitBlit | graphics library |
| WaitBOVP | graphics library |
| WaitForChar | dos library |
| WaitIO | exec library |
| WaitPort | exec library |
| WaitTOF | graphics library |
| WBenchToBack | intuition library |
| WBenchToFront | intuition library |
| WhichLayer | layers library |
| WindowLimits | intuition library |
| WindowToBack | intuition library |
| WindowToFront | intuition library |
| Write | dos library |
| WritePixel | graphics library |
| XorRectRegion | graphics library |

| | |
|-------------|-------------------------------|
| ABS | clib macro [macros.h] |
| abs | lattice macro [stdio.h] |
| BNDRYOFF | graphics macro [gfxmacros.h] |
| CEND | graphics macro [gfxmacros.h] |
| CINIT | graphics macro [gfxmacros.h] |
| clearerr | lattice macro [stdio.h] |
| CMOVE | graphics macro [gfxmacros.h] |
| CWAIT | graphics macro [gfxmacros.h] |
| feof | lattice macro [stdio.h] |
| ferror | lattice macro [stdio.h] |
| fflush | lattice macro [stdio.h] |
| fileno | lattice macro [stdio.h] |
| FOREVER | intuition macro [intuition.h] |
| getc | lattice macro [stdio.h] |
| getchar | lattice macro [stdio.h] |
| InitAnimate | graphics macro [gels.h] |
| isalnum | lattice macro [ctype.h] |
| isalpha | lattice macro [ctype.h] |
| isascii | lattice macro [ctype.h] |
| iscntrl | lattice macro [ctype.h] |
| iscsym | lattice macro [ctype.h] |
| iscsymf | lattice macro [ctype.h] |
| isdigit | lattice macro [ctype.h] |
| isgraph | lattice macro [ctype.h] |
| islower | lattice macro [ctype.h] |
| isprint | lattice macro [ctype.h] |
| ispunct | lattice macro [ctype.h] |
| isspace | lattice macro [ctype.h] |
| isupper | lattice macro [ctype.h] |
| isxdigit | lattice macro [ctype.h] |
| ITEMNUM | intuition macro [intuition.h] |
| MAX | clib macro [macros.h] |
| max | lattice macro [stdio.h] |
| MENUNUM | intuition macro [intuition.h] |
| MIN | clib macro [macros.h] |
| min | lattice macro [stdio.h] |
| NOT | intuition macro [intuition.h] |
| ObjAlloc | workbench macro [workbench.h] |
| putc | lattice macro [stdio.h] |
| putchar | lattice macro [stdio.h] |
| RASSIZE | graphics macro [gfx.h] |
| RemBob | graphics macro [gels.h] |
| rewind | lattice macro [stdio.h] |
| SetAfPt | graphics macro [gfxmacros.h] |
| SetDrPt | graphics macro [gfxmacros.h] |
| SetOPen | graphics macro [gfxmacros.h] |
| SetWrMsk | graphics macro [gfxmacros.h] |
| SHIFTITEM | intuition macro [intuition.h] |
| SHIFTMENU | intuition macro [intuition.h] |
| SHIFTSUB | intuition macro [intuition.h] |
| SIGN | intuition macro [intuition.h] |
| STREQ | workbench macro [workbench.h] |
| SUBNUM | intuition macro [intuition.h] |
| TMAlloc | workbench macro [workbench.h] |
| toascii | lattice macro [ctype.h] |
| TOBB | graphics macro [gfx.h] |

tolower
toupper

lattice macro[ctype.h]
lattice macro[ctype.h]

TABLE OF CONTENTS

mathfp.library/_LVOSPabs
mathfp.library/_LVOSPComp
mathfp.library/_LVOSFDiv
mathfp.library/_LVOSPFix
mathfp.library/_LVOSPFIt
mathfp.library/_LVOSPMul
mathfp.library/_LVOSPNeg
mathfp.library/_LVOSPSub
mathfp.library/_LVOSPtst
mathfp.library/_LVSPAdd
mathieeedoubbas.library/_LVOIEEDPabs
mathieeedoubbas.library/_LVOIEEDPadd
mathieeedoubbas.library/_LVOIEEDPCmp
mathieeedoubbas.library/_LVOIEEDPDiv
mathieeedoubbas.library/_LVOIEEDPFIt
mathieeedoubbas.library/_LVOIEEDPMul
mathieeedoubbas.library/_LVOIEEDPNeg
mathieeedoubbas.library/_LVOIEEDPSub
mathieeedoubbas.library/_LVOIEEDPTst
mathtrans.library/_LVOSPacos
mathtrans.library/_LVOSPasin
mathtrans.library/_LVOSPatan
mathtrans.library/_LVOSPCos
mathtrans.library/_LVOSPCosh
mathtrans.library/_LVOSPExp
mathtrans.library/_LVOSPFieee
mathtrans.library/_LVOSFLog
mathtrans.library/_LVOSFLog10
mathtrans.library/_LVOSFPow
mathtrans.library/_LVOSFSin
mathtrans.library/_LVOSFsinCos
mathtrans.library/_LVOSFSinh
mathtrans.library/_LVOSFSqrt
mathtrans.library/_LVOSFTan
mathtrans.library/_LVOSFTanh
mathtrans.library/_LVOSFTieee

mathfp.library/_LVOSPabs mathfp.library/_LVOSPabs

NAME

abs - obtain the absolute value of the fast floating point number

SYNOPSIS

```

MOVE.L ARG,D0
LEA _LVOSPabs,A0
ADD.L _MathBase,A0
JSR (A0)

```

FUNCTION

Accepts a floating point number and returns the absolute value of said number

INPUTS

D0 - floating point number

RESULT

D0 - floating point absolute value of input register (D0)

BUGS

None

SEE ALSO

abs, SPabs

mathfp.library/_JVOFCmp mathfp.library/_JVOFCmp

NAME

_JVOFCmp - compares two floating point numbers and sets appropriate condition codes

SYNOPSIS

MOVE.L ARG1,D0
MOVE.L ARG2,D1
LEA _JVOFCmp,A0
ADD.L _MathBase,A0
JSR (A0)

FUNCTION

Accepts two floating point numbers and returns the condition codes set to indicate the result of said comparison

INPUTS

D1 - floating point number (arg 1)
D0 - floating point number (arg 2)

RESULT

Condition codes set to reflect the following branches:

GT - arg 2 > arg 1
GE - arg 2 >= arg 1
EQ - arg 2 = arg 1
NE - arg 2 != arg 1
LT - arg 2 < arg 1
LE - arg 2 <= arg 1

Also sets D0 = +1 => arg1 > arg2
 = -1 => arg1 < arg2
 = 0 => arg1 = arg2

BUGS

None

SEE ALSO

fcmp1, SFCmp

mathfp.library/_JVOFDiv mathfp.library/_JVOFDiv

NAME

_JVOFDiv - divide two floating point numbers

SYNOPSIS

MOVE.L ARG1,D0
MOVE.L ARG2,D1
LEA _JVOFDiv,A0
ADD.L _MathBase,A0
JSR (A0)

FUNCTION

Accepts two floating point numbers and returns the arithmetic division of said numbers

INPUTS

D1 - floating point number (arg 1)
D0 - floating point number (arg 2)

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

fdiv1, SFDiv

mathfp.library/_JVOSEFFix mathfp.library/_JVOSEFFix

NAME

_JVOSEFFix - convert fast floating point number to integer

SYNOPSIS

MOVE.L ARG,D0
LEA _JVOSEFFix,A0
ADD.L _MathBase,A0
JSR (A0)

FUNCTION

Accepts a floating point number and returns the truncated integer portion of said number

INPUTS

D0 - floating point number

RESULT

D0 - signed integer number

BUGS

None

SEE ALSO

ffixi, SPFix

mathfp.library/_JVOSEFFlt mathfp.library/_JVOSEFFlt

NAME

_JVOSEFFlt - convert integer number to fast floating point

SYNOPSIS

MOVE.L ARG,D0
LEA _JVOSEFFlt,A0
ADD.L _MathBase,A0
JSR (A0)

FUNCTION

Accepts an integer and returns the converted floating point result of said number

INPUTS

D0 - signed integer number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

fflti, SPFlt

mathfp.library/_JVOSEPMul mathfp.library/_JVOSEPMul

NAME

_JVOSEPMul - multiply two floating point numbers

SYNOPSIS

```

MOVE.L ARG1,D0
MOVE.L ARG2,D1
LEA   _JVOSEPMul,A0
ADD.L _MathBase,A0
JSR   (A0)

```

FUNCTION

Accepts two floating point numbers and returns the arithmetic multiplication of said numbers

INPUTS

D1 - floating point number (arg 1)
D0 - floating point number (arg 2)

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

fmul, SPMul

mathfp.library/_JVOSEPNeg mathfp.library/_JVOSEPNeg

NAME

_JVOSEPNeg - negate the supplied floating point number

SYNOPSIS

```

MOVE.L ARG,D0
LEA   _JVOSEPNeg,A0
ADD.L _MathBase,A0
JSR   (A0)

```

FUNCTION

Accepts a floating point number and returns the value of said number after having been subtracted from 0.0

INPUTS

D0 - floating point number

RESULT

D0 - floating point negation of input register (D0)

BUGS

None

SEE ALSO

fnegl, SPNeg

mathfp.library/_JVOFSSub mathfp.library/_JVOFSSub

NAME

_JVOFSSub - subtract two floating point numbers

SYNOPSIS

```

MOVE.L ARG1,D0
MOVE.L ARG2,D1
LEA   _JVOFSSub,A0
ADD.L _MathBase,A0
JSR   (A0)

```

FUNCTION

Accepts two floating point numbers and returns the arithmetic subtraction of said numbers

INPUTS

D1 - floating point number (arg 1)
D0 - floating point number (arg 2)

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

fsubi, SFSUB

mathfp.library/_JVOESTst mathfp.library/_JVOESTst

NAME

_JVOESTst - compares a fast floating point number against the value zero (0.0) and sets the appropriate condition codes

SYNOPSIS

```

MOVE.L ARG,D1
LEA   _JVOESTst,A0
ADD.L _MathBase,A0
JSR   (A0)

```

FUNCTION

Accepts a floating point number and returns the condition codes set to indicate the result of a comparison against the value of zero (0.0)

INPUTS

D1 - floating point number

RESULT

Condition codes set to reflect the following branches:

```

EQ - argument = 0.0
NE - argument != 0.0
PL - argument >= 0.0
MI - argument < 0.0

```

```

Also sets D0 = +1 => arg > 0.0
              = -1 => arg < 0.0
              = 0 => arg = 0.0

```

BUGS

None

SEE ALSO

ftsti, SPTst

mathfp.library/_LVSPPAdd mathfp.library/_LVSPPAdd

NAME

_LVSPPAdd - add two floating point numbers

SYNOPSIS

```

MOVE.L ARG1,D0
MOVE.L ARG2,D1
LEA   _LVSPPAdd,A0
ADD.L _MathBase,A0
JSR   (A0)

```

FUNCTION

Accepts two floating point numbers and returns the arithmetic sum of said numbers

INPUTS

D1 - floating point number (arg 1)
D0 - floating point number (arg 2)

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

faddi, SPAdd

mathieeedubbas.library/_LVOIEEEDPAbs

NAME

_LVOIEEEDPAbs - obtain the absolute value of the IEEE double precision floating point number

SYNOPSIS

```

MOVE.L _MathIeeeDoubBasBase,A6
MOVEM.L ARG,D0/D1
JSR   _LVOIEEEDPAbs(A6)

```

FUNCTION

Accepts an IEEE D.P. floating point number and returns the absolute value of said number.

INPUTS

D0/D1 - IEEE double precision floating point number

RESULT

D0/D1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

IEEEDPAbs

mathieeeoubbas.library/_JVOIEEEDPAdd

NAME

_JVOIEEEDPAdd - add two IEEE double precision floating point numbers

SYNOPSIS

MOVE.L _MathIeeeDoubBasBase,A6
MOVEM.L ARG1,D0/D1
MOVEM.L ARG2,D2/D3
JSR _JVOIEEEDPAdd(A6)

FUNCTION

Accepts two IEEE D.P. floating point numbers and returns the arithmetic sum of said numbers.

INPUTS

D0/D1 - IEEE double precision floating point number
D2/D3 - IEEE double precision floating point number

RESULT

D0/D1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

IEEEDPAdd

mathieeeoubbas.library/_JVOIEEEDPCmp

NAME

_JVOIEEEDPCmp - compares two IEEE D.P. floating point numbers and returns a relative value indicator

SYNOPSIS

MOVE.L _MathIeeeDoubBasBase,A6
MOVEM.L ARG1,D0/D1
MOVEM.L ARG2,D2/D3
JSR _JVOIEEEDPCmp(A6)

FUNCTION

Accepts two IEEE double precision floating point numbers and returns the OCR and the integer functional result as an indicator of the result of said comparison.

INPUTS

D0/D1 - IEEE double precision floating point number
D2/D3 - IEEE double precision floating point number

RESULT

Condition codes set to reflect the following branches:

LT - ARG1 < ARG2 (D0 = -1)
GT - ARG1 > ARG2 (D0 = +1)
ELSE - ARG1 = ARG2 (D0 = 0)

BUGS

None

SEE ALSO

IEEEDPCmp

mathieeedoubbas.library/_LVOIEEEDDiv

NAME

_LVOIEEEDDiv - divide two IEEE double precision floating point numbers

SYNOPSIS

MOVE.L _MathIeeeDoubBasBase,A6
MOVEM.L ARG1,D0/D1
MOVEM.L ARG2,D2/D3
JSR _LVOIEEEDPAdd(A6)

FUNCTION

Accepts two IEEE double precision floating point numbers and returns the arithmetic division of said numbers.

INPUTS

D0/D1 - IEEE double precision floating point number
D2/D3 - IEEE double precision floating point number

RESULT

D0/D1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

IEEEDDiv

mathieeedoubbas.library/_LVOIEEEDPflt

NAME

_LVOIEEEDPflt - convert integer number to IEEE D.P. floating point

SYNOPSIS

MOVE.L _MathIeeeDoubBasBase,A6
MOVE.L ARG,D0
JSR _LVOIEEEDPflt(A6)

FUNCTION

Accepts an integer and returns the converted IEEE double precision floating point result of said number.

INPUTS

D0 - signed integer number

RESULT

D0/D1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

IEEEDPflt

mathIeesdoubbas.library/_LVOIEEEDF*1

NAME

_LVOIEEEDF*1 - multiply two IEEE double precision floating point numbers

SYNOPSIS

MOVE.L _MathIeesdoubBasBase,A6
MOVEM.L ARG1,D0/D1
MOVEM.L ARG2,D2/D3
JSR _LVOIEEEDF*1(A6)

FUNCTION

Accepts two IEEE D.P. floating point numbers and returns the arithmetic multiplication of said numbers.

INPUTS

D0/D1 - IEEE double precision floating point number
D2/D3 - IEEE double precision floating point number

RESULT

D0/D1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

IEEEDF*1

mathIeesdoubbas.library/_LVOIEEEDFNeg

NAME

_LVOIEEEDFNeg - negate the supplied IEEE double precision floating point number

SYNOPSIS

MOVE.L _MathIeesdoubBasBase,A6
MOVEM.L ARG,D0/D1
JSR _LVOIEEEDFNeg(A6)

FUNCTION

Accepts an IEEE D.P. floating point number and returns the value of said number after having been subtracted from 0.0

INPUTS

D0/D1 - IEEE double precision floating point number

RESULT

D0/D1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

IEEEDFNeg

mathieesdoubbas.library/_JVOIEEDPSub

NAME

_JVOIEEDPSub - subtract two IEEE double precision floating point numbers

SYNOPSIS

```

MOVE.L _MathIeesDoubBasBase,A6
MOVEM.L ARG1,D0/D1
MOVEM.L ARG2,D2/D3
JSR _JVOIEEDPAdd(A6)

```

FUNCTION

Accepts two IEEE D.P. floating point numbers and returns the arithmetic subtraction of said numbers.

INPUTS

D0/D1 - IEEE double precision floating point number
D2/D3 - IEEE double precision floating point number

RESULT

D0/D1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

IEEEDPSub

mathieesdoubbas.library/_JVOIEEDPTst

NAME

_JVOIEEDPTst - compares an IEEE D.P. floating point number against the value 0.0 and returns a relative value indicator

SYNOPSIS

```

MOVE.L _MathIeesDoubBasBase,A6
MOVEM.L ARG,D0/D1
JSR _JVOIEEDPTst(A6)

```

FUNCTION

Accepts an IEEE double precision floating point number and returns the OCR and the integer functional result as an indicator of the result of comparison against the value 0.0.

NOTE: Using number directly within parenthesis to generate in-line code is much more efficient.

INPUTS

D0/D1 - IEEE double precision floating point number

RESULT

Condition codes set to reflect the following branches:

```

LT - ARG < 0.0 (D0 = -1)
GT - ARG > 0.0 (D0 = +1)
ELSE - ARG = 0.0 (D0 = 0)

```

BUGS

None

SEE ALSO

IEEEDPTst

mathtrans.library/_LVOSPACos mathtrans.library/_LVOSPACos

NAME

_LVOSPACos - obtain the arccosine of the floating point number

SYNOPSIS

MOVE.L ARC,D0
LEA _LVOSPACos,A0
ADD.L _MathTransBase,A0
JSR (A0)

FUNCTION

Accepts a floating point number representing the cosine of an angle and returns the value of said angle in radians

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPACos

mathtrans.library/_LVOSPAsin mathtrans.library/_LVOSPAsin

NAME

_LVOSPAsin - obtain the arcsine of the floating point number

SYNOPSIS

MOVE.L ARC,D0
LEA _LVOSPAsin,A0
ADD.L _MathTransBase,A0
JSR (A0)

FUNCTION

Accepts a floating point number representing the sine of an angle and returns the value of said angle in radians

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPAsin

mathtrans.library/_JVOSEPatan mathtrans.library/_JVOSEPatatan

NAME

_JVOSEPatatan - obtain the arctangent of the floating point number

SYNOPSIS

```

MOVE.L ARG,D0
LEA    _JVOSEPatatan,A0
ADD.L  _MathTransBase,A0
JSR    (A0)

```

FUNCTION

Accepts a floating point number representing the tangent of an angle and returns the value of said angle in radians

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPatan

mathtrans.library/_JVOSECos mathtrans.library/_JVOSEPCos

NAME

_JVOSEPCos - obtain the cosine of the floating point number

SYNOPSIS

```

MOVE.L ARG,D0
LEA    _JVOSEPCos,A0
ADD.L  _MathTransBase,A0
JSR    (A0)

```

FUNCTION

Accepts a floating point number representing an angle in radians and returns the cosine of said angle

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPCos

mathtrans.library/_JVOSEPCosh mathtrans.library/_JVOSEPCosh

NAME

_JVOSEPCosh - obtain the hyperbolic cosine of the floating point number

SYNOPSIS

MOVE.L ARG,D0
LEA _JVOSEPCosh,A0
ADD.L _MathTransBase,A0
JSR (A0)

FUNCTION

Accepts a floating point number representing an angle in radians and returns the hyperbolic cosine of said angle

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPCosh

mathtrans.library/_JVOSEPEXP mathtrans.library/_JVOSEPEXP

NAME

_JVOSEPEXP - obtain the exponent (e**X) of the floating point number

SYNOPSIS

MOVE.L ARG,D0
LEA _JVOSEPEXP,A0
ADD.L _MathTransBase,A0
JSR (A0)

FUNCTION

Accepts a floating point number and returns e raised to the input numbers power

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPExp

mathtrans.library/_JVOSEFieee mathtrans.library/_JVOSEFieee

NAME

_JVOSEFieee - convert an IEEE standard number to EFP format

SYNOPSIS

MOVE.L ARG,D0
LEA _JVOSEFieee,A0
ADD.L _MathTransBase,A0
JSR (A0)

FUNCTION

Accepts an IEEE standard format number and returns the same number, only converted into Motorola fast floating point format

INPUTS

D0 - floating point number (IEEE STD format)

RESULT

D0 - floating point number (Motorola EFP format)

BUGS

None

SEE ALSO

SPEFieee

mathtrans.library/_JVOSEFLog

mathtrans.library/_JVOSEFLog

NAME

_JVOSEFLog - obtain the natural logarithm of the floating point number

SYNOPSIS

MOVE.L ARG,D0
LEA _JVOSEFLog,A0
ADD.L _MathTransBase,A0
JSR (A0)

FUNCTION

Accepts a floating point number and returns the natural logarithm (base e) of said number

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPEFLog

mathtrans.library/_JVOSFLog10 mathtrans.library/_JVOSFLog10

NAME

_JVOSFLog10 - obtain the napaian logarithm (base 10) of the floating point nu

SYNOPSIS

MOVE.L ARG,D0
LEA _JVOSFLog10,A0
ADD.L MathTransBase,A0
JSR (A0)

FUNCTION

Accepts a floating point number and returns the napaian logarithm (base 10) of said number

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SFLog10

mathtrans.library/_JVOSFPow mathtrans.library/_JVOSFPow

NAME

_JVOSFPow - obtain the exponentiation of two FFP numbers

SYNOPSIS

MOVE.L ARG1,D0
MOVE.L ARG2,D1
LEA _JVOSFPow,A0
ADD.L MathTransBase,A0
JSR (A0)

FUNCTION

Accepts two (2) floating point numbers and returns the result of ARG1 raised to the ARG2 power

INPUTS

D1 - floating point number (arg 1)
D0 - floating point number (arg 2)

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SFPow

mathtrans.library/_JVOFPSIn mathtrans.library/_JVOFPSIn

NAME

_JVOFPSIn - obtain the sine of the floating point number

SYNOPSIS

```

MOVE.L ARG,D0
LEA _JVOFPSIn,A0
ADD.L _MathTransBase,A0
JSR (A0)

```

FUNCTION

Accepts a floating point number representing an angle in radians and returns the sine of said angle

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPSIn

mathtrans.library/_JVOFPSIncos mathtrans.library/_JVOFPSIncos

NAME

_JVOFPSIncos - obtain the sine & cosine of the FFP number

SYNOPSIS

```

MOVE.L ARG1,D1
MOVE.L ARG2,D0
LEA _JVOFPSIncos,A0
ADD.L _MathTransBase,A0
JSR (A0)

```

FUNCTION

Accepts a floating point number representing an angle in radians and returns both the sine & cosine of said angle

INPUTS

D0 - floating point number

D1 - address of cosine result

RESULT

D0 - floating point number (sine)
(D1) - floating point number (cosine)

BUGS

None

SEE ALSO

SPSIncos

mathtrans.library/_JVOFSinh mathtrans.library/_JVOFSinh

NAME

_JVOFSinh - obtain the hyperbolic sine of the floating point number

SYNOPSIS

```

MOVE.L ARG,D0
LEA _JVOFSinh,A0
ADD.L _MathTransBase,A0
JSR (A0)

```

FUNCTION

Accepts a floating point number representing an angle in radians and returns the hyperbolic sine of said angle

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPSinh

mathtrans.library/_JVOFSqrt mathtrans.library/_JVOFSqrt

NAME

sqrt - obtain the square root of the floating point number

SYNOPSIS

```

MOVE.L ARG,D0
LEA _JVOFSqrt,A0
ADD.L _MathTransBase,A0
JSR (A0)

```

FUNCTION

Accepts a floating point number and returns the square root of said number

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPSqrt

mathtrans.library/_JVOSPTan mathtrans.library/_JVOSPTan

NAME

_JVOSPTan - obtain the tangent of the floating point number

SYNOPSIS

```

MOVE.L ARG,D0
LEA    _JVOSPTan,A0
ADD.L  _MathTransBase,A0
JSR    (A0)

```

FUNCTION

Accepts a floating point number representing an angle in radians and returns the tangent of said angle

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPTan

mathtrans.library/_JVOSPTanh mathtrans.library/_JVOSPTanh

NAME

_JVOSPTanh - obtain the hyperbolic tangent of the floating point number

SYNOPSIS

```

MOVE.L ARG,D0
LEA    _JVOSPTanh,A0
ADD.L  _MathTransBase,A0
JSR    (A0)

```

FUNCTION

Accepts a floating point number representing an angle in radians and returns the hyperbolic tangent of said angle

INPUTS

D0 - floating point number

RESULT

D0 - floating point number

BUGS

None

SEE ALSO

SPTanh

mathtrans.library/_IWOSEPTieee mathtrans.library/_IWOSEPTieee

NAME

_IWOSEPTieee - convert an FFP number to IEEE standard format

SYNOPSIS

```
MOVE.L ARG,D0
LEA _IWOSEPTieee,A0
ADD.L _MathTransBase,A0
JSR (A0)
```

FUNCTION

Accepts a Motorola fast floating point number and returns the same number, only converted into IEEE standard format

INPUTS

D0 - floating point number (Motorola FFP format)

RESULT

D0 - floating point number (IEEE STD format)

BUGS

None

SEE ALSO

SPTieee

TABLE OF CONTENTS

mathfp.library/abs
 mathfp.library/faddl
 mathfp.library/fcmp1
 mathfp.library/fdiv1
 mathfp.library/ffit1
 mathfp.library/fmul1
 mathfp.library/fneg1
 mathfp.library/fsubi
 mathfp.library/ftst1
 mathfp.library/SPAbs
 mathfp.library/SPAdd
 mathfp.library/SPComp
 mathfp.library/SPDiv
 mathfp.library/SPFit
 mathfp.library/SPMul
 mathfp.library/SPNeg
 mathfp.library/SPSub
 mathfp.library/SPtst
 mathieeedoubbas.library/IEEDPAbS
 mathieeedoubbas.library/IEEDPAdd
 mathieeedoubbas.library/IEEDPCmp
 mathieeedoubbas.library/IEEDPDiv
 mathieeedoubbas.library/IEEDPFit
 mathieeedoubbas.library/IEEDPMul
 mathieeedoubbas.library/IEEDPNeg
 mathieeedoubbas.library/IEEDPSub
 mathieeedoubbas.library/IEEDPTst
 mathlink.lib.lib/arnd
 mathlink.lib.lib/dbf
 mathlink.lib.lib/fpa
 mathlink.lib.lib/fpbcd
 mathtrans.library/SPAcos
 mathtrans.library/SPAsin
 mathtrans.library/SPAtan
 mathtrans.library/SPCos
 mathtrans.library/SPCosh
 mathtrans.library/SPExp
 mathtrans.library/SPFieee
 mathtrans.library/SPLog
 mathtrans.library/SPLog10
 mathtrans.library/SPPow
 mathtrans.library/SPSin
 mathtrans.library/SPSincos
 mathtrans.library/SPSinh
 mathtrans.library/SPSqrt
 mathtrans.library/SPTanh
 mathtrans.library/SPTieee
 mathtrans.library/tan

mathfp.library/abs

mathfp.library/abs

NAME

abs - obtain the absolute value of the fast floating point number

C USAGE

fnum2 = abs(fnum1);
D0

FUNCTION

Accepts a floating point number and returns the absolute value of said number. Note that this function is called by compiler generated code, not by a user generated function call.

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point absolute value of fnum1

BUCKS

None

SEE ALSO

SPAbs, _JVOSEPAbs

mathffp.library/fadd1 mathffp.library/fadd1

NAME

fadd1 - add two floating point numbers

C USAGE

fnum3 = fnum1 + fnum2;
D1 D0

FUNCTION

Accepts two floating point numbers and returns the arithmetic sum of said numbers. Note that this function is called by compiler generated code, not by a user generated function call.

INPUTS

fnum1 - floating point number
fnum2 - floating point number

RESULT

fnum3 - floating point number

BUGS

None

SEE ALSO

SPAdd, _JAVOSPAAdd

mathffp.library/fcmp1 mathffp.library/fcmp1

NAME

fcmp1 - compares two floating point numbers and sets appropriate condition codes

C USAGE

if (fnum1 <= fnum2) {...}
D1 D0

FUNCTION

Accepts two floating point numbers and returns the condition codes set to indicate the result of said comparison. Note that this function is called by compiler generated code, not by a user generated function call.

INPUTS

fnum1 - floating point number
fnum2 - floating point number

RESULT

Condition codes set to reflect the following branches:

- GT - fnum2 > fnum1
- GE - fnum2 >= fnum1
- EQ - fnum2 = fnum1
- NE - fnum2 != fnum1
- LT - fnum2 < fnum1
- LE - fnum2 <= fnum1

BUGS

None

SEE ALSO

SPComp, _JAVOSPComp

mathfp.library/fdivi mathfp.library/fdivi

NAME

fdivi - divide two floating point numbers

C USAGE

fnum3 = fnum1 / fnum2;
D1 D0

FUNCTION

Accepts two floating point numbers and returns the arithmetic division of said numbers. Note that this function is called by compiler generated code, not by a user generated function call.

INPUTS

fnum1 - floating point number
fnum2 - floating point number

RESULT

fnum3 - floating point number

BUGS

None

SEE ALSO

SPDiv, _JVOSEFDiv

mathfp.library/fflti mathfp.library/fflti

NAME

fflti - convert integer number to fast floating point
fflti - convert integer number to fast floating point

C USAGE

fnum = (FLOAT) inum;
D0

FUNCTION

Accepts an integer and returns the converted floating point result of said number. Note that this function is called by compiler generated code, not by a user generated function call.

INPUTS

inum - signed integer number

RESULT

fnum - floating point number

BUGS

None

SEE ALSO

SPflt, _JVOSEFFlt

mathffp.library/fmul1 mathffp.library/fmul1

NAME

fmul1 - multiply two floating point numbers

C USAGE

fnum3 = fnum1 * fnum2;
D1 D0

FUNCTION

Accepts two floating point numbers and returns the arithmetic multiplication of said numbers. Note that this function is called by compiler generated code, not by a user generated function call.

INPUTS

fnum1 - floating point number
fnum2 - floating point number

RESULT

fnum3 - floating point number

BUGS

None

SEE ALSO

SPMul, _IJOSEPMul

mathffp.library/fneg1 mathffp.library/fneg1

NAME

fneg1 - negate the supplied floating point number

C USAGE

fnum2 = -fnum1;
D0

FUNCTION

Accepts a floating point number and returns the value of said number after having been subtracted from 0.0. Note that this function is called by compiler generated code, not by a user generated function call.

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point negation of fnum1

BUGS

None

SEE ALSO

SPNeg, _IJOSEPNeg

mathfp.library/fsubi mathfp.library/fsubi

NAME

fsubi - subtract two floating point numbers

C USAGE

fnum3 = fnum1 - fnum2;
D1 D0

FUNCTION

Accepts two floating point numbers and returns the arithmetic subtraction of said numbers. Note that this function is called by compiler generated code, not by a user generated function call.

INPUTS

fnum1 - floating point number
fnum2 - floating point number

RESULT

fnum3 - floating point number

BUGS

None

SEE ALSO

SFSub, _JAVOSFSub

mathfp.library/ftst1 mathfp.library/ftst1

NAME

ftst1 - compares a fast floating point number against the value zero (0.0) and sets the appropriate condition codes

C USAGE

if (ifnum) {...}
D1

FUNCTION

Accepts a floating point number and returns the condition codes set to indicate the result of a comparison against the value of zero (0.0). Note that this function is called by compiler generated code, not by a user generated function call.

INPUTS

fnum - floating point number

RESULT

Condition codes set to reflect the following branches:

- EQ - fnum = 0.0
- NE - fnum != 0.0
- PL - fnum >= 0.0
- MI - fnum < 0.0

BUGS

None

SEE ALSO

SFTst, _JAVOSFTst

mathfp.library/SPAbs mathfp.library/SPAbs

NAME

SPAbs - obtain the absolute value of the fast floating point number

C USAGE

frum2 = SPAbs (frum1);
D0

FUNCTION

Accepts a floating point number and returns the absolute value of said number.

INPUTS

frum1 - floating point number

RESULT

frum2 - floating point absolute value of frum1

BUGS

None

SEE ALSO

_JVOSEPAbs, abs

mathfp.library/SPAdd mathfp.library/SPAdd

NAME

SPAdd - add two floating point numbers

C USAGE

frum3 = SPADD (frum1, frum2);
D1 D0

FUNCTION

Accepts two floating point numbers and returns the arithmetic sum of said numbers.

INPUTS

frum1 - floating point number
frum2 - floating point number

RESULT

frum3 - floating point number

BUGS

None

SEE ALSO

_JVOSEPAAdd, fadd1

mathfp.library/SFOmp mathfp.library/SFOmp

NAME

SFOmp - compares two floating point numbers and sets appropriate condition codes

C USAGE

if (SFOmp(fnum1, fnum2)) { ... }
D1 D0

FUNCTION

Accepts two floating point numbers and returns the condition codes set to indicate the result of said comparison. Additionally, the integer functional result is returned to indicate the result of said comparison.

INPUTS

fnum1 - floating point number
fnum2 - floating point number

RESULT

Condition codes set to reflect the following branches:

- GT - fnum2 > fnum1
- GE - fnum2 >= fnum1
- EQ - fnum2 = fnum1
- NE - fnum2 != fnum1
- LT - fnum2 < fnum1
- LE - fnum2 <= fnum1

Integer functional result as:

- +1 => fnum1 > fnum2
- 1 => fnum1 < fnum2
- 0 => fnum1 = fnum2

BUGS

None

SEE ALSO

_JVOSEmp, fcmp1

mathfp.library/SFDiv mathfp.library/SFDiv

NAME

SFDiv - divide two floating point numbers

C USAGE

fnum3 = SFDiv(fnum1, fnum2);
D1 D0

FUNCTION

Accepts two floating point numbers and returns the arithmetic division of said numbers.

INPUTS

fnum1 - floating point number
fnum2 - floating point number

RESULT

fnum3 - floating point number

BUGS

None

SEE ALSO

_JVOSEmp, fdiv1

mathfp.library/SFFlt mathfp.library/SFFlt

NAME

SFFlt - convert integer number to fast floating point

C USAGE

fnum = SFFlt(inum);
D0

FUNCTION

Accepts an integer and returns the converted floating point result of said number.

INPUTS

inum - signed integer number

RESULT

fnum - floating point number

BUGS

None

SEE ALSO

_LVOSFFlt, fflt1

mathfp.library/SFmul mathfp.library/SFmul

NAME

SFmul - multiply two floating point numbers

C USAGE

fnum3 = SFmul(fnum1, fnum2);
D1 D0

FUNCTION

Accepts two floating point numbers and returns the arithmetic multiplication of said numbers.

INPUTS

fnum1 - floating point number
fnum2 - floating point number

RESULT

fnum3 - floating point number

BUGS

None

SEE ALSO

_LVOSFmul, fmul1

mathffp.library/SPNeg mathffp.library/SPNeg

NAME

SPNeg - negate the supplied floating point number

C USAGE

fnum2 = SPNeg(fnum1);
D0

FUNCTION

Accepts a floating point number and returns the value of said number after having been subtracted from 0.0

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point negation of fnum1

BUGS

None

SEE ALSO

_LVOSPNeg, fneg1

mathffp.library/SFSub mathffp.library/SFSub

NAME

SFSub - subtract two floating point numbers

C USAGE

fnum3 = SFSub(fnum1, fnum2);
D1 D0

FUNCTION

Accepts two floating point numbers and returns the arithmetic subtraction of said numbers.

INPUTS

fnum1 - floating point number

fnum2 - floating point number

RESULT

fnum3 - floating point number

BUGS

None

SEE ALSO

_LVOSFSub, fsub1

mathfp.library/SPTst

mathfp.library/SPTst

NAME

SPTst - compares a fast floating point number against the value zero (0.0) and sets the appropriate condition codes

C USAGE

```
if (!(SPTst(fnum))) { ... }
DI
```

FUNCTION

Accepts a floating point number and returns the condition codes set to indicate the result of a comparison against the value of zero (0.0). Additionally, the integer functional result is returned.

INPUTS

fnum - floating point number

RESULT

Condition codes set to reflect the following branches:

- EQ - fnum = 0.0
- NE - fnum != 0.0
- PL - fnum >= 0.0
- MI - fnum < 0.0

Integer functional result as:

- +1 => fnum > 0.0
- 1 => fnum < 0.0
- 0 => fnum = 0.0

BUCS

None

SEE ALSO

_JVOSPTst, ftst1

mathieeedoubbas.library/IEEDPabs

mathieeedoubbas.library/IEEDPabs

NAME

IEEDPabs - obtain the absolute value of the IEEE double precision floating point number

C USAGE

```
fnum1 = IEEDPabs(fnum2);
D0/D1
```

FUNCTION

Accepts an IEEE D.P. floating point number and returns the absolute value of said number.

INPUTS

fnum2 - IEEE double precision floating point number

RESULT

fnum1 - IEEE double precision floating point number

BUCS

None

SEE ALSO

_JVOIEEDPabs

mathieeeoubbas.library/IEEEFPAdd mathieesoubbas.library/IEEEFPAdd

NAME

IEEEFPAdd - add two IEEE double precision floating point numbers

C USAGE

fnum1 = IEEEFPAdd(fnum2, fnum3);
D0/D1 D2/D3

FUNCTION

Accepts two IEEE D.P. floating point numbers and returns the arithmetic sum of said numbers.

INPUTS

fnum2 - IEEE double precision floating point number
fnum3 - IEEE double precision floating point number

RESULT

fnum1 - IEEE double precision floating point number

BUCS

None

SEE ALSO

_LVOIEEEFPAdd

mathieesoubbas.library/IEEEFPComp mathieesoubbas.library/IEEEFPComp

NAME

IEEEFPComp - compares two IEEE D.P. floating point numbers and returns a relative value indicator

C USAGE

if (IEEEFPComp(fnum1, fnum2)) {...}
D0/D1 D2/D3

FUNCTION

Accepts two IEEE double precision floating point numbers and returns the OCR and the integer functional result as an indicator of the result of said comparison.

INPUTS

fnum1 - IEEE double precision floating point number
fnum2 - IEEE double precision floating point number

RESULT

Condition codes set to reflect the following branches:

LT - fnum1 < fnum2 (Functional Result = -1)
GT - fnum1 > fnum2 (Functional Result = +1)
ELSE - fnum1 = fnum2 (Functional Result = 0)

BUCS

None

SEE ALSO

_LVOIEEEFPComp

mathieeeedubbas.library/IEEEPDIV mathieeeedubbas.library/IEEEPDIV

NAME

IEEEPDIV - divide two IEEE double precision floating point numbers

C USAGE

fnum = IEEEPMa1(fnum2, fnum3);
D0/D1 D2/D3

FUNCTION

Accepts two IEEE double precision floating point numbers and returns the arithmetic division of said numbers.

INPUTS

fnum2 - IEEE double precision floating point number
fnum3 - IEEE double precision floating point number

RESULT

fnum1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

_LVOIEEEDPDIV

mathieeeedubbas.library/IEEEDFIT mathieeeedubbas.library/IEEEDFIT

NAME

IEEEDFIT - convert integer number to IEEE D.P. floating point

C USAGE

fnum = IEEEDFIT(inum);
D0/D1 D0

FUNCTION

Accepts an integer and returns the converted IEEE double precision floating point result of said number.

INPUTS

inum - signed integer number

RESULT

fnum - IEEE double precision floating point number

BUGS

None

SEE ALSO

_LVOIEEEDFIT

mathieeedoubbas.library/IEEEDFMul mathieeedoubbas.library/IEEEDFMul

NAME

IEEEDFMul - multiply two IEEE double precision floating point numbers

C USAGE

fnum1 = IEEEDFMul(fnum2, fnum3);
D0/D1 D2/D3

FUNCTION

Accepts two IEEE D.P. floating point numbers and returns the arithmetic multiplication of said numbers.

INPUTS

fnum2 - IEEE double precision floating point number
fnum3 - IEEE double precision floating point number

RESULT

fnum1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

_LVOIEEEDFMul

mathieeedoubbas.library/IEEEDFNeg

mathieeedoubbas.library/IEEEDFNeg

NAME

IEEEDFNeg - negate the supplied IEEE double precision floating point number

C USAGE

fnum1 = IEEEDFNeg(fnum2);
D0/D1

FUNCTION

Accepts an IEEE D.P. floating point number and returns the value of said number after having been subtracted from 0.0

INPUTS

fnum2 - IEEE double precision floating point number

RESULT

fnum1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

_LVOIEEEDFNeg

mathleesedoubbas.library/IEEEDFSUB mathleesedoubbas.library/IEEEDFSUB

NAME

IEEEDFSUB - subtract two IEEE double precision floating point numbers

C USAGE

frum1 = IEEEDFSUB(frum2, frum3);
D0/D1 D2/D3

FUNCTION

Accepts two IEEE D.P. floating point numbers and returns the arithmetic subtraction of said numbers.

INPUTS

frum2 - IEEE double precision floating point number
frum3 - IEEE double precision floating point number

RESULT

frum1 - IEEE double precision floating point number

BUGS

None

SEE ALSO

_LVOIEEEDFSUB

mathleesedoubbas.library/IEEEDTST mathleesedoubbas.library/IEEEDTST

NAME

IEEEDTST - compares an IEEE D.P. floating point number against the value 0.0 and returns a relative value indicator

C USAGE

if (IEEEDTST(fnum)) {...}
D0/D1

FUNCTION

Accepts an IEEE double precision floating point number and returns the OCK and the integer functional result as an indicator of the result of comparison against the value 0.0.

NOTE: Using number directly within parenthesis to generate in-line code is much more efficient.

INPUTS

fnum - IEEE double precision floating point number

RESULT

Condition codes set to reflect the following branches:

LT - fnum < 0.0 (Functional Result = -1)
GT - fnum > 0.0 (Functional Result = +1)
ELSE - fnum = 0.0 (Functional Result = 0)

BUGS

None

SEE ALSO

_LVOIEEEDTST

mathlink_lib.lib/arrnd mathlink_lib.lib/arrnd

NAME

arrnd - ASCII round of the provided floating point string

C USAGE

arrnd(place, exp, &string[0]);

FUNCTION

Accepts an ASCII string representing an FFP floating point number, the binary representation of the exponent of said floating point number and the number of places to round to. A rounding process is initiated, either to the left or right of the decimal place and the result placed back at the input address defined by &string[0].

INPUTS

place - integer value representing number of decimal places to round to
exp - integer value representing exponent value of the ASCII string
&string[0] - address where rounded ASCII string is to be placed

RESULT

&string[0] - rounded ASCII string

BUGS

None

SEE ALSO

N/A

mathlink_lib.lib/dbf

NAME

dbf - convert FFP dual-binary number to FFP format

C USAGE

fnrm = dbf(exp, mant);

FUNCTION

Accepts a dual-binary format (described below) floating point number and converts it to an FFP format floating point number. The dual-binary format is defined as:

exp bit 16 = sign (0=>positive, 1=>negative)
exp bits 15-0 = binary integer representing the base ten (10) exponent
mant = binary integer mantissa

INPUTS

exp - binary integer representing sign and exponent
mant - binary integer representing the mantissa

RESULT

fnrm - converted FFP floating point format number

BUGS

None

SEE ALSO

N/A

mathlink_lib.lib/dbf

mathlink_lib.lib/fpa mathlink_lib.lib/fpa

NAME

fpa - convert fast floating point variable into ASCII string equivalent

C USAGE

exp = fpa(fnum, &string[0]);

FUNCTION

Accepts an FFP floating point number and the address of the ASCII string where it's converted output is to be stored. The number is converted to an ASCII string in C format and stored at the address provided. Additionally, the base ten (10) exponent in binary form is returned as the functional value.

INPUTS

fnum - floating point number
&string[0] - address for output of converted ASCII character string

RESULT

&string[0] - converted ASCII character string
exp - Integer exponent value in binary form

BUGS

None

SEE ALSO

N/A

mathlink_lib.lib/fpbcd mathlink_lib.lib/fpbcd

NAME

fpbcd - convert FFP floating point number to BCD format

C USAGE

fpbcd(fnum, &string[0]);

FUNCTION

Accepts a floating point number and the address where the converted BCD data is to be stored. The FFP number is converted and stored at the specified address in an ASCII form in accordance with the following format:

MMMM S E S B

Where: M = Four bytes of BCD, each with two (2) digits of the mantissa (8 digits)
S = Sign of mantissa (0x00 = positive, 0xFF = negative)
E = BCD byte for two (2) digit exponent
S = Sign of exponent (0x00 = positive, 0xFF = negative)
B = One (1) byte binary two's complement representation of the exponent

INPUTS

fnum - floating point number
&string[0] - address where converted BCD data is to be placed

RESULT

&string[0] - converted BCD data

BUGS

None

SEE ALSO

N/A

mathtrans.library/SPAcos

mathtrans.library/SPAcos

NAME

SPAcos - obtain the arccosine of the floating point number

SYNOPSIS

fnum2 = SPAcos (fnum1);
D0

FUNCTION

Accepts a floating point number representing the cosine of an angle and returns the value of said angle in radians

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_LVOSPAcos

mathtrans.library/SPAasin

mathtrans.library/SPAasin

NAME

SPAasin - obtain the arcsine of the floating point number

SYNOPSIS

fnum2 = SPAasin (fnum1);
D0

FUNCTION

Accepts a floating point number representing the sine of an angle and returns the value of said angle in radians

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_LVOSPAasin

mathtrans.library/SPAtan mathtrans.library/SPAtan

NAME

SPAtan - obtain the arctangent of the floating point number

SYNOPSIS

fnum2 = SPAtan(fnum1);
D0

FUNCTION

Accepts a floating point number representing the tangent of an angle and returns the value of said angle in radians

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_LVOSEAtan

mathtrans.library/SFCos mathtrans.library/SFCos

NAME

SFCos - obtain the cosine of the floating point number

SYNOPSIS

fnum2 = SFCos(fnum1);
D0

FUNCTION

Accepts a floating point number representing an angle in radians and returns the cosine of said angle

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_LVOSEFCos

mathtrans.library/SPCosh mathtrans.library/SPCosh

NAME

SPCosh - obtain the hyperbolic cosine of the floating point number

SYNOPSIS

fnum2 = SPCosh (fnum1);
D0

FUNCTION

Accepts a floating point number representing an angle in radians and returns the hyperbolic cosine of said angle

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUCS

None

SEE ALSO

_LVOSPCosh

mathtrans.library/SPExp mathtrans.library/SPExp

NAME

SPExp - obtain the exponent (e^{**X}) of the floating point number

SYNOPSIS

fnum2 = SPExp (fnum1);
D0

FUNCTION

Accepts a floating point number and returns e raised to the input numbers power

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUCS

None

SEE ALSO

_LVOSPExp

mathtrans.library/SFFieee

mathtrans.library/SFFieee

NAME

SFFieee - convert an IEEE standard number to FFP format

SYNOPSIS

fnum = SFFieee(ieeeenum);
D0

FUNCTION

Accepts an IEEE standard format number and returns the same number, only converted into Motorola fast floating point format

INPUTS

ieeeenum - floating point number (IEEE STD format)

RESULT

fnum - floating point number (Motorola FFP format)

BUGS

None

SEE ALSO

_JAVOSFFieee

mathtrans.library/SFLog

mathtrans.library/SFLog

NAME

SFLog - obtain the natural logarithm of the floating point number

SYNOPSIS

fnum2 = SFLog(fnum1);
D0

FUNCTION

Accepts a floating point number and returns the natural logarithm (base e) of said number

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_JAVOSFLog

mathtrans.library/SFLog10

mathtrans.library/SFLog10

NAME

SFLog10 - obtain the naperian logarithm (base 10) of the floating point number

SYNOPSIS

fnum2 = SFLog10(fnum1);
D0

FUNCTION

Accepts a floating point number and returns the naperian logarithm (base 10) of said number

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_LVOSELog10

mathtrans.library/SFPow

mathtrans.library/SFPow

NAME

SFPow - obtain the exponentiation of two EFP numbers

SYNOPSIS

fnum3 = SFPow(fnum1, fnum2);
D1 D0

FUNCTION

Accepts two (2) floating point numbers and returns the result of fnum1 raised to the fnum2 power

INPUTS

fnum1 - floating point number

fnum2 - floating point number

RESULT

fnum3 - floating point number

BUGS

None

SEE ALSO

_LVOSEFPow

mathtrans.library/SPSin mathtrans.library/SPSin

NAME

SPSin - obtain the sine of the floating point number

SYNOPSIS

fnum2 = SPSIn(fnum1);
D0

FUNCTION

Accepts a floating point number representing an angle in radians and returns the sine of said angle

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_JVOSPSIn

mathtrans.library/SPSincos

mathtrans.library/SPSincos

NAME

SPSincos - obtain the sine & cosine of the FFP number

SYNOPSIS

fnum3 = SPSincos(fnum1, &fnum2);
D1 D0

FUNCTION

Accepts a floating point number representing an angle in radians and returns both the sine & cosine of said angle

INPUTS

fnum1 - floating point number
&fnum2 - address of cosine result

RESULT

fnum2 - floating point number (cosine)
fnum3 - floating point number (sine)

BUGS

None

SEE ALSO

_JVOSPSincos

mathtrans.library/SFSinh

mathtrans.library/SFSinh

NAME

SFSinh - obtain the hyperbolic sine of the floating point number

SYNOPSIS

fnum2 = SFSinh (fnum1);
D0

FUNCTION

Accepts a floating point number representing an angle in radians and returns the hyperbolic sine of said angle

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_JVOSFSinh

mathtrans.library/SFSqrt

mathtrans.library/SFSqrt

NAME

SFSqrt - obtain the square root of the floating point number

SYNOPSIS

fnum2 = SFSqrt (fnum1);
D0

FUNCTION

Accepts a floating point number and returns the square root of said number

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_JVOSFSqrt

mathtrans.library/SPTanh

mathtrans.library/SPTanh

NAME

SPTanh - obtain the hyperbolic tangent of the floating point number

SYNOPSIS

fnum2 = SPTanh(fnum1);
D0

FUNCTION

Accepts a floating point number representing an angle in radians and returns the hyperbolic tangent of said angle

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_LJVSPTanh

mathtrans.library/SPTieee

mathtrans.library/SPTieee

NAME

SPTieee - convert an FFP number to IEEE standard format

SYNOPSIS

ieeeenum = SPTieee(fnum);

FUNCTION

Accepts a Motorola fast floating point number and returns the same number, only converted into IEEE standard format

INPUTS

fnum - floating point number (Motorola FFP format)

RESULT

ieeeenum - floating point number (IEEE STD format)

BUGS

None

SEE ALSO

_LJVSPTieee

mathtrans.library/tan mathtrans.library/tan

NAME

SPTan - obtain the tangent of the floating point number

SYNOPSIS

fnum2 = SPTan(fnum1);
D0

FUNCTION

Accepts a floating point number representing an angle in radians and returns the tangent of said angle

INPUTS

fnum1 - floating point number

RESULT

fnum2 - floating point number

BUGS

None

SEE ALSO

_LWOSPTan

TABLE OF CONTENTS

clist.library/AllocCList
 clist.library/ConcatCList
 clist.library/CopyCList
 clist.library/FlushCList
 clist.library/FreeCList
 clist.library/GetCLBuf
 clist.library/GetCLChar
 clist.library/GetCLWord
 clist.library/IncrCLMark
 clist.library/InitCLPool
 clist.library/MarkCLList
 clist.library/PeekCLMark
 clist.library/PutCLBuf
 clist.library/PutCLChar
 clist.library/PutCLWord
 clist.library/SizeCList
 clist.library/SplitCList
 clist.library/UnGetCLChar
 clist.library/UnGetCLWord
 clist.library/UnPutCLChar
 clist.library/UnPutCLWord

clist.library/AllocCList

clist.library/AllocCList

NAME AllocCList - allocate and initialize a clist

SYNOPSIS
clist = AllocCList(cIPool)
D0 A1

FUNCTION
Get a descriptor that can be used to reference a clist. The clist described is empty. Clists that are no longer in use must be explicitly closed with FreeCList in order to free all their memory: an empty clist still consumes clist pool resources.

INPUTS
cIPool - A clist pool that has already been initialized.

RESULTS
clist - a longword descriptor for a clist that can be used for clist functions.

EXCEPTIONS
if clist is negative, no space was available for a new clist.

NOTES
This function is implicitly performed by BuffToCL.

clist.library/ConcatCList

clist.library/ConcatCList

NAME ConcatCList - concatenate two character lists

SYNOPSIS
error = ConcatCList (sourceCList, destCList)
A0 A1

FUNCTION
Exhaust the contents of the sourceCList onto the end of the destCList. The resulting destCList is the concatenation of the original destCList and sourceCList; the resulting sourceCList is empty.

INPUTS
sourceCList - The cList descriptor used to manage the source character list.
destCList - The cList descriptor used to manage the destination character list.

RESULT
error - An error code that, if non-zero, indicates the cList pool associated with the destCList had an out of memory condition during the concatenation process.

clist.library/CopyCList

clist.library/CopyCList

NAME CopyCList - copy a cList to a new cList

SYNOPSIS
cList = CopyCList (cList)
D0 A0

FUNCTION
Copy a cList non-destructively into a new cList, created by this operation in the same clPool.

INPUTS
cList - The cList descriptor used to manage the original character list.

RESULTS
cList - a longword descriptor for a cList that can be used for cList functions, and contains the same contents as the original cList.

EXCEPTIONS
if cList is negative, not enough space was available for the new cList.

clist.library/FlushClist

clist.library/FlushClist

NAME FlushClist - clear a character list

SYNOPSIS
FlushClist(clist)
A0

FUNCTION ensure that the clist is empty.

INPUTS
clist -

The clist header used to manage this character list,
as returned by AllocClist or StrToCl.

RESULTS

clist.library/FreeClist

clist.library/FreeClist

NAME FreeClist - free a clist

SYNOPSIS
FreeClist(clist)
A0

FUNCTION Release the clist descriptor and any resources it uses.
References to the clist are no longer valid.

INPUTS
clist -

a descriptor for a clist that is no longer to be used.

NOTES This function is implicitly performed by ClToBuf.

cclist.library/GetClistBuf

cclist.library/GetClistBuf

NAME GetClistBuf - convert a character list to contiguous data

SYNOPSIS
length = GetClistBuf(cclist, buffer, maxLength)
D0 A0 A1 D1

FUNCTION
Move the cclist data into the block of memory pointed to by buffer. Exhaust the character list. If a non-destructive peek at the character list is desired, use SubCL. If the cclist will no longer be used, remember to FreeClist.

INPUTS
cclist - The cclist descriptor used to manage this character list, as returned by AllocClist.
buffer -

maxLength - A pointer for the byte data from the character list. The maximum size of buffer.

RESULTS
length - the number of bytes copied into buffer. This is never greater than maxLength.

EXCEPTIONS
if cclist was bigger than maxLength, the cclist is not empty.

cclist.library/GetClistChar

cclist.library/GetClistChar

NAME GetClistChar - get a byte from the beginning of a character list

SYNOPSIS
byte = GetClistChar(cclist)
D0 A0

FUNCTION
Get a byte from the beginning of the character list described by the cclist.

INPUTS
cclist - The cclist header used to manage this character list, as returned by AllocClist or StrioCL.

RESULTS
byte - The byte from the beginning of the character list. If no data is available, the upper three bytes are set (longword is -1).

clist.library/GetCLWord

clist.library/GetCLWord

NAME GetCLWord - get a word from the beginning of a character list

SYNOPSIS
word = GetCLWord(cList)
D0 A0

FUNCTION
Get a word from the beginning of the character list described
by the cList.

INPUTS
cList -
The cList header used to manage this character list,
as returned by AllocCList or StrToCL.

RESULTS
word
The word from the beginning of the character list.
If no data is available, the upper two bytes are set
(longword is -1). Partial words (1 byte) are not
returned.

clist.library/IncrCLMark

clist.library/IncrCLMark

NAME IncrCLMark - increment a cList mark to the next position

SYNOPSIS
error = IncrCLMark(cList)
D0 A0

FUNCTION
Increment a mark for cList operations to mark the next byte
in the cList.

INPUTS
cList -
a longword descriptor for a cList that can be used
for cList functions.

RESULTS
error -
non-zero if the next offset is not in the cList

EXCEPTIONS
If error is non-zero, the request asked to move the mark
beyond the end of the cList, and the mark is invalid.

clist.library/InitCLPool

clist.library/InitCLPool

NAME InitCLPool - initialize a cllist pool

SYNOPSIS
error = InitCLPool(cIPool, size)
D0 A0 D0

FUNCTION
Initialize a block of memory for use as a pool for cllist nodes. This involves setting up a header structure and building a free list of all the nodes.

INPUTS
cIPool - The data area that is to be used as the character list pool for the cllist operations.
size - The size of the pool, in bytes. Cllist pools are limited to 16M bytes.

RESULTS
error - If the cllist pool provided is so small that not even pool management memory will fit, this is set to non-zero.

clist.library/MarkCLList

clist.library/MarkCLList

NAME MarkCLList - mark a position in a cllist

SYNOPSIS
error = MarkCLList(cList, offset)
A0 D0

FUNCTION
Mark the cllist for index operations by specifying a byte offset into the cllist. Note that only one mark is retained by each cllist. If the byte to which the mark refers is subsequently manipulated, the mark will become invalid.

INPUTS
cList - a longword descriptor for a cllist that can be used for cllist functions.
offset - a byte offset into the cllist. The first byte in the cllist is at offset zero. This value should not be greater than (SizeCLList-1).

RESULTS
error - non-zero if the offset is not in the cllist

EXCEPTIONS
If the offset is more than the length of the cllist, the mark is invalid.

cllist.library/PeekCLMark

cllist.library/PeekCLMark

NAME PeekCLMark - peek at the byte in the cllist at the mark

SYNOPSIS
byte = PeekCLMark(cList)
D0 A0

FUNCTION
Return the byte value at the mark in the character list associated with the mark.

INPUTS
cList - a longword descriptor for a cllist that can be used for cllist functions.

RESULTS
byte - the byte at the mark in the cllist.

cllist.library/PutCLBuf

cllist.library/PutCLBuf

NAME PutCLBuf - convert contiguous data into a character list

SYNOPSIS
error = PutCLBuf(cList, buffer, length)
D0 A0 A1 D1

FUNCTION
Append the contents of the data buffer to a character list. The buffer data remains intact.

INPUTS
cList - The cllist descriptor used to manage this character list, as returned by AllocCLlist.
buffer - A pointer to byte data used to initialize the character list.
length - The number of bytes of data in the buffer.

RESULTS
error - non-zero indicates the number of bytes not added.

cllist.library/PutCLChar

cllist.library/PutCLChar

NAME PutCLChar - add a byte to the end of a character list

SYNOPSIS
error = PutCLChar(cList, byte)
D0 A0 D0

FUNCTION
Add a byte to the end of the character list described by the cList.

INPUTS
cList -
The cList header used to manage this character list,
as returned by AllocCList or StrToCL.
byte -
The byte to add to the end of the character list

RESULTS
error - non-zero indicates the byte could not be added

cllist.library/PutCLWord

cllist.library/PutCLWord

NAME PutCLWord - add a word to the end of a character list

SYNOPSIS
error = PutCLWord(cList, word)
D0 A0 D0

FUNCTION
Add a word to the end of the character list described by the cList.

INPUTS
cList -
The cList header used to manage this character list,
as returned by AllocCList or StrToCL.
word -
The word to add to the end of the character list

RESULTS
error - non-zero indicates the number of bytes not added.
Partial words are not added, so error is always zero
or two.

clist.library/SizeCList

clist.library/SizeCList

NAME SizeCList - get the number of bytes in a character list

SYNOPSIS
bytes = SizeCList(cList)
D0 A0

FUNCTION
Inquire the number of characters in cList.

INPUTS
cList - The cList header used to manage this character list,
as returned by AllocCList or StrToCl.

RESULTS
bytes - the number of bytes in cList.

clist.library/SplitCList

clist.library/SplitCList

NAME SplitCList - split a cList

SYNOPSIS
tailCList = SplitCList(cList)
D0 A0

FUNCTION
Split a cList into two cLists. The original cList will contain the head of the cList up to but not including the mark (obtained via the MarkCList command). A new cList will be created and returned containing the bytes associated with the mark thru the end of the original cList.

INPUTS
cList - a longword descriptor for a cList that can be used for cList functions.

RESULTS
tailCList - a longword descriptor for a cList that contains the tail end of the original cList.

EXCEPTIONS
if there is not enough memory to build the new cList, or the mark is invalid tailCList is negative.

cllist.library/SubCLlist

cllist.library/SubCLlist

NAME SubCLlist - copy a substring from a cllist

SYNOPSIS
cllist = SubCLlist(cllist, index, length)
D0 A0 D0 D1

FUNCTION
Copy a substring of the cllist into a new cllist created by this operation. Start at offset index into the character list, and copy for length bytes. The source cllist is not altered.

INPUTS
cllist - The cllist descriptor used to manage this character list, as returned by NewCLlist or StrToCL.
index - The offset in the character list to start copying the substring from. An index of 0 is the first character in the cllist.
length - The number of bytes to copy.

RESULTS
cllist - a longword descriptor for a cllist that can be used for cllist functions.

EXCEPTIONS
if cllist is negative, not enough space was available for the new cllist.
if the substring does not exist for the index and length specified, the resulting cllist will be shorter than expected.

cllist.library/UnGetCLChar

cllist.library/UnGetCLChar

NAME UnGetCLChar - add a byte to the beginning of a character list

SYNOPSIS
error = UnGetCLChar(cllist, byte)
D0 A0 D0

FUNCTION
Add a byte to the beginning of the character list described by the cllist.

INPUTS
cllist - The cllist header used to manage this character list, as returned by AllocCLlist or StrToCL.
byte - The byte to add to the beginning of the character list

RESULTS
error - non-zero indicates the byte could not be added

clist.library/UnGetCLMord

clist.library/UnGetCLMord

NAME UnGetCLMord - add a word to the beginning of a character list

SYNOPSIS
error = UnGetCLMord(cList, word)
D0 A0 D0

FUNCTION
Add a word to the beginning of the character list described by the cList.

INPUTS
cList -
The cList header used to manage this character list, as returned by AllocCList or StrToCL.
word -
The word to add to the beginning of the character list

RESULTS
error -
non-zero indicates the number of bytes not added. Partial words are not added, so error is always zero or two.

clist.library/UnPutCLChar

clist.library/UnPutCLChar

NAME UnPutCLChar - get a byte from the end of a character list

SYNOPSIS
byte = UnPutCLChar(cList)
D0 A0

FUNCTION
Get a byte from the end of the character list described by the cList.

INPUTS
cList -
The cList header used to manage this character list, as returned by AllocCList or StrToCL.

RESULTS
byte -
The byte from the end of the character list. If no data is available, the upper three bytes are set (longword is -1).

TABLE OF CONTENTS

diskfont.library/AvailFonts
diskfont.library/OpenDiskFont

diskfont.library/AvailFonts

diskfont.library/AvailFonts

NAME

AvailFonts - build an array of all fonts in memory / on disk

SYNOPSIS

error = AvailFonts (buffer, bufBytes, types);
 A0 D0 D1

FUNCTION

AvailFonts fills a user supplied buffer with the structure, described below, that contains information about all the fonts available in memory and/or on disk. Those fonts available on disk need to be loaded into memory and opened via OpenDiskFont, those already in memory are accessed via OpenFont. The TextAttr structure required by the open calls is part of the information AvailFonts supplies.

INPUTS

buffer - memory to be filled with struct AvailFontsHeader followed by an array of AvailFonts elements, which contains entries for the available fonts and their names.

bufBytes - the number of bytes in the buffer

types - AFF_MEMORY is set to search memory for fonts to fill - the structure, AFF_DISK is set to search the disk for fonts to fill the structure. Both can be specified.

RESULTS

buffer - filled with struct AvailFontsHeader followed by the AvailFonts elements. There will be duplicate entries for fonts found both in memory and on disk, differing only by type. The existence of a disk font in the buffer indicates that it exists as an entry in a font contents file -- the underlying font file has not been checked for validity, thus an OpenDiskFont of it may fail.

error - if non-zero, this indicates the number of bytes needed for AvailFonts in addition to those supplied. Thus structure elements were not returned because of insufficient bufBytes.

diskfont.library/OpenDiskFont diskfont.library/OpenDiskFont

NAME

OpenDiskFont - load and get a pointer to a disk font.

SYNOPSIS

font = OpenDiskFont(textAttr)
D0 A0

FUNCTION

This function finds the font with the specified textAttr on disk, loads it into memory, and returns a pointer to the font that can be used in subsequent SetFont and CloseFont calls. It is important to match this call with a corresponding CloseFont call for effective management of font memory.

If the font is already in memory, the copy in memory is used. The disk copy is not reloaded.

INPUTS

textAttr - a TextAttr structure that describes the text font attributes desired.

EXCEPTIONS

D0 is zero if the desired font cannot be found.

TABLE OF CONTENTS

Exec/AddDevice
 Exec/AddHead
 Exec/AddIntServer
 Exec/AddLibrary
 Exec/AddPort
 Exec/AddResource
 Exec/AddTail
 Exec/AddTask
 Exec/Allocate
 Exec/AllocEntry
 Exec/AllocMem
 Exec/AllocSignal
 Exec/AllocTrap
 Exec/AvailMem
 Exec/Cause
 Exec/CheckIO
 Exec/CloseDevice
 Exec/CloseLibrary
 Exec/ColdReset
 Exec/Deallocate
 Exec/DoIO
 Exec/Enqueue
 Exec/FindName
 Exec/FindPort
 Exec/FindTask
 Exec/FreeEntry
 Exec/FreeMem
 Exec/FreeSignal
 Exec/FreeTrap
 Exec/GetOC
 Exec/GetMsg
 Exec/InitStruct
 Exec/Insert
 Exec/MakeLibrary
 Exec/OpenDevice
 Exec/OpenLibrary
 Exec/OpenResource
 Exec/PutMsg
 Exec/RemDevice
 Exec/RemHead
 Exec/RemIntServer
 Exec/RemLibrary
 Exec/Remove
 Exec/RemPort
 Exec/RemResource
 Exec/RemTask
 Exec/ReplyMsg
 Exec/SendIO
 Exec/SetExcept
 Exec/SetFunction
 Exec/SetIntVector

Exec/SetSignal
 Exec/SetSR
 Exec/SetTaskPri
 Exec/Signal
 Exec/SumLibrary
 Exec/SuperState
 Exec/UserState
 Exec/Wait
 Exec/WaitIO
 Exec/WaitPort

Exec/AddDevice

Exec/AddDevice

NAME AddDevice -- add a device to the system

SYNOPSIS
AddDevice(device)
A1

FUNCTION
This function adds a new device to the system, making it available to everyone. The device should be ready to be called at this time.

INPUTS
device - pointer to a properly initialized device node

SEE ALSO
RemDevice

Exec/AddHead

Exec/AddHead

NAME AddHead -- insert node at the head of a list

SYNOPSIS
AddHead(list, node)
A0 A1

FUNCTION
Add a node to the head of a doubly linked list.

INPUTS
list - a pointer to the target list header
node - the node to insert at head

Exec/AddIntServer

Exec/AddIntServer

NAME AddIntServer -- add an interrupt server to the system

SYNOPSIS AddIntServer (intNum, interrupt)
D0-0:4 A1

FUNCTION

This function adds a new interrupt server to a given server chain. The node is located on the chain in a priority dependent position. Higher priority nodes will be serviced first.

If this server is the first one, interrupt will be enabled on this chain.

Servers are called with the following register conventions:

- D0 - scratch
- D1 - scratch
- A0 - scratch
- A1 - server data segment pointer (scratch)
- A5 - jump vector register (scratch)
- A6 - library base pointer (scratch)

all other registers - must be preserved

INPUTS

intNum - the Portia interrupt bit (0..14)
interrupt - pointer to an interrupt server node

SEE ALSO RemIntServer

Exec/AddLibrary

Exec/AddLibrary

NAME AddLibrary -- add a library to the system

SYNOPSIS AddLibrary(library)
A1

FUNCTION

This function adds a new library to the system making it available to everyone. The library should be ready to be called at this time. It will be added to the system library name list, and the checksum on the library entries will be calculated.

INPUTS

library - pointer to a properly initialized library structure

SEE ALSO

RemLibrary

Exec/AddPort Exec/AddPort

NAME AddPort -- add a message port to the system

SYNOPSIS
AddPort (port)
A1

FUNCTION
This function attaches a message port structure to the system's message port list. The name and priority fields of the port structure should be initialized prior to calling this function. If the user does not require the name and priority fields, they should be initialized to zero. As with the name field in other system list items, the name is useful when more than one task needs to rendezvous on at port.

INPUTS
port - pointer to a message port

SEE ALSO
RemPort, FindName

Exec/AddResource

Exec/AddResource

NAME AddResource -- add a resource to the system

SYNOPSIS
AddResource (resource)
A1

FUNCTION
This function adds a new resource to the system and makes it available to other users. The resource should be ready to be called at this time.

INPUTS
resource - pointer to a properly initialized resource node

SEE ALSO
RemResource

Exec/AddTail

Exec/AddTail

NAME AddTail -- append node to tail of a list

SYNOPSIS
AddTail(list, node)
A0 A1

FUNCTION
Add a node to the tail of a doubly linked list.

INPUTS
list - a pointer to the target list header
node - the node to insert at tail

Exec/AddTask

Exec/AddTask

NAME AddTask -- add a task to the system

SYNOPSIS
AddTask(task, initialPC, finalPC)
A1 A2 A3

FUNCTION
Add a task to the system.

Certain fields of the task control block must be initialized and a minimal stack should be allocated prior to calling this function.

This function will temporarily use space from the new task's stack for the task's initial set of registers. This space is allocated starting at the SPREG location specified in the task control block (not from SPUPPER). This means that a task's stack may contain static data put there prior to its execution. This is useful for providing initialized global variables or some tasks may want to use this space for passing the task its initial arguments.

A task's initial registers are set to zero (except the PC).

INPUTS
task - pointer to the task control block
initialPC - the initial entry point
finalPC - the finalization code entry point. If zero, the system will use a general finalizer. This pointer is placed on the stack as if it were the outermost return address.

SEE ALSO
RemTask

Exec/Allocate

Exec/Allocate

NAME Allocate -- allocate a block of memory

SYNOPSIS

```
memoryBlock = allocate(freeList, byteSize)
D0          A0          D0
```

FUNCTION

This function is used to allocate blocks of memory from a given free memory pool. It will return the first free block that is greater than or equal to the requested size.

All blocks, whether free or allocated, will be block aligned; hence, all allocation sizes are rounded up to the next block even value (e.g. the minimum allocation resolution is 8 bytes).

This function, when used in conjunction with a private free list, can be used to manage an application's internal data memory.

INPUTS

freeList - points to the memory list header
byteSize - the size of the desired block in bytes

RESULT

memoryBlock - a pointer to the just allocated free block.

If there are no free regions large enough to satisfy the request, return zero. If the amount of requested memory is invalid, return zero.

EXCEPTIONS

If the free list is corrupt, the system will panic.

SEE ALSO

Deallocate

Exec/AllocEntry

Exec/AllocEntry

NAME AllocEntry -- allocate many regions of memory

SYNOPSIS

```
memList = AllocEntry(memList)
D0          A0
```

FUNCTION

This routine takes a memList structure and allocates enough memory to hold the required memory as well as a MemList structure to keep track of it. These MemList structures may be linked together in a task control block to keep track of the total memory usage of this task.

INPUTS

memList -- A memList structure filled in with memEntry structures.

RESULTS

memList -- A different memList filled in with the actual memory allocated, and their sizes.

If enough memory cannot be obtained, then the requirements of the failed allocation are returned and bit 31 is set.

EXAMPLES

The user wants five regions of 2, 4, 8, 16, and 32 bytes in size with requirements of MEME_CLEAR, MEME_PUBLIC, MEME_CHIP.OR.MEME_CLEAR, MEME_FAST.OR.MEME_CLEAR, and MEME_PUBLIC.OR.MEME_CLEAR respectively. The following code fragment would do that:

MemListDecl:

```
DS.B LN_SIZE          * reserve space for list node
DC.W 5                * number of entries
DC.L MEME_CLEAR
DC.L 2                * entry #0
DC.L MEME_PUBLIC
DC.L 4                * entry #1
DC.L MEME_CHIP.OR.MEME_CLEAR * entry #2
DC.L 8                * entry #3
DC.L MEME_FAST.OR.MEME_CLEAR * entry #4
DC.L 16
DC.L MEME_PUBLIC.OR.MEME_CLEAR * entry #5
DC.L 32
```

start:

```
LEA MemListDecl,A0
CALLLIB _LVOAllocEntry,A5
```

```
BCLR.L #31,D0
BEQ.S success
```

----- Type of memory that we failed on is in D0

Exec/AllocMem

Exec/AllocMem

NAME AllocMem -- allocate memory given certain requirements

SYNOPSIS

memoryBlock = AllocMem(byteSize, requirements)
D0 D1 D1-0:31

FUNCTION

This is the memory allocator to be used by system code and applications. It provides a means of specifying whether the allocation should be made in a memory area accessible to the chips, or accessible to shared system code.

The proper allocation of memory is necessary for system code that needs to be compatible with memory mapped systems.

Memory is allocated based on the "requirements" listed. The rule is that (requirements & attributes) == requirements for any particular memory block.

AllocMem will try all memory spaces until one is found with the requested attributes and room for the memory request.

INPUTS

byteSize - the size of the desired block in bytes
This number is rounded up to the next larger block size for the actual allocation.
requirements - (still in flux)
(see IA_Structs for bit definitions)

MEMB_PUBLIC:

memory must not be mapped, swapped, or otherwise made non-addressable.
ALL MEMORY THAT IS REFERENCED VIA INTERRUPTS AND/OR BY OTHER TASKS MUST BE EITHER PUBLIC OR LOCKED INTO MEMORY!
This includes both code and data.

MEMB_CHIP:

Only certain parts of memory are reachable by the special chip sets' DMA circuitry. Anything that will use on-chip DMA must be in memory with this attribute. DMA includes screen memory, things that are blitted, audio blocks, raw disc buffers, etc.

MEMB_FAST:

This is non-chip memory. It is possible for the processor to get locked out of chip memory under certain conditions. If one cannot accept these delays, then one should use FAST memory (by default the system will allocate from FAST memory first anyway).

MEMB_CLEAR: The memory will be initialized to all zeros.

RESULT

memoryBlock - a pointer to the allocated free block.

If there are no free regions large enough to satisfy the request (or if the amount of requested memory is invalid), return zero.

EXAMPLES

AllocMem(321, MEMB_CHIP) - private chip memory

AllocMem(25, MEMB_PUBLIC) - a "public" system structure that does not require chip memory.

EXCEPTIONS

If the free list is corrupt, the system will panic.

SEE ALSO

AllocAbs, FreeMem

Exec/AllocSignal

Exec/AllocSignal

NAME AllocSignal -- allocate a signal bit

SYNOPSIS

signalNum = AllocSignal(signalNum)
D0

FUNCTION

Allocate a signal bit from the current tasks pool. Either a particular bit, or the next free bit may be allocated. The signal associated with the newly allocated bit will be properly initialized (cleared).

If the signal is already in use (or no free signals are available) a -1 is returned.

This function can only be used by the currently running task.

WARNING

Signals may not be allocated or freed from exception handling code.

INPUTS

signalNum - the desired signal number (of 0..31) or -1 for no preference.

RESULTS

signalNum - the signal bit number allocated (0..31).

If no signals are available, this function returns -1.

SEE ALSO

FreeSignal

Exec/AllocTrap

Exec/AllocTrap

NAME AllocTrap -- allocate a processor trap vector

SYNOPSIS
trapNum = AllocTrap (trapNum)
D0

FUNCTION
Allocate a trap number from the current task's pool. These trap numbers are those associated with the 68000 TRAP type instructions. Either a particular number, or the next free number may be allocated.

If the trap is already in use (or no free traps are available) a -1 is returned.

This function can only be used by the currently running task.

WARNING
Signals may not be allocated or freed from exception handling code.

INPUTS
trapNum - the desired trap number (of 0..15) or -1 for no preference.

RESULTS
trapNum - the trap number allocated (of 0..15). If no traps are available, this function returns -1.

SEE ALSO
FreeTrap

Exec/AvailMem

Exec/AvailMem

NAME AvailMem -- memory available given certain requirements

SYNOPSIS
size = AvailMem (requirements)
D0 DI

FUNCTION
This function returns the size of memory given certain requirements.

INPUTS
requirements - a requirements mask as specified in AllocMem

RESULT
size - total free space remaining

Exec/Cause

Exec/Cause

NAME Cause -- cause a software interrupt

SYNOPSIS Cause (interrupt)
AI

FUNCTION

This function causes a software interrupt to occur. If it is called from user mode (and processor level 0), the software interrupt will pre-empt the current task.

Currently only 5 software interrupt priorities are implemented: -32, -16, 0, +16, and +32. Priorities in between these values are truncated. Priorities outside the -32/+32 range are not allowed.

INPUTS

interrupt - pointer to a properly initialized interrupt node

Exec/CheckIO

Exec/CheckIO

NAME CheckIO -- get the IO request status

SYNOPSIS result = CheckIO(iORequest)
D0 AI

FUNCTION

This function determines the current state of an I/O request and returns FALSE if the I/O has not yet completed. This function effectively hides the internals of the I/O completion mechanism.

If the I/O has completed, CheckIO will not remove the returned IORequest from the reply port. This should be performed with Remove.

This function SHOULD NOT be used to busy loop, waiting for an IO to complete.

INPUTS

iORequest - pointer to an I/O request block

RESULTS

result - null if I/O is still in progress. Otherwise D0 points to the IORequest block.

Exec/CloseDevice

Exec/CloseDevice

NAME CloseDevice -- conclude access to a device

SYNOPSIS
CloseDevice(iORequest)
AI

FUNCTION

This function informs the system that access to a device/unit previously opened has been concluded. The device may perform certain house-cleaning operations. The I/O request structure is now free to be recycled.

INPUTS
iORequest - pointer to an I/O request structure

SEE ALSO
OpenDevice

Exec/CloseLibrary

Exec/CloseLibrary

NAME CloseLibrary -- conclude access to a library

SYNOPSIS
CloseLibrary(library)
AI

FUNCTION

This function informs the system that access to the given library has been concluded. The user should not reference the library or any routine in the library after this close.

INPUTS
library - pointer to a library node

SEE ALSO
OpenLibrary

Exec/ColdReset

Exec/ColdReset

NAME ColdReset -- cause a system coldstart to occur

SYNOPSIS ColdReset ()

FUNCTION

This function causes a coldstart system reset sequence identical to that which occurs at power-on. All current system activities will be stopped, and the entire software system will be re-initialized. Nothing will be preserved. This function will assert processor RESET to reset all hardware devices.

EXCEPTION

This function operates in supervisor mode only. Any attempt to perform this function from user mode will result in a privileges violation trap.

Exec/Deallocate

Exec/Deallocate

NAME Deallocate -- deallocate a block of memory

SYNOPSIS Deallocate(freeList, memoryBlock, byteSize)
A0 A1 D0

FUNCTION

This function deallocates memory by returning it to the appropriate free memory pool. This function can be used to free an entire block allocated with the above function, or it can be used to free a sub-block of a previously allocated block.

If memoryBlock is not on a block boundary (MEM_BLOCKSIZE) then it will be rounded down. Note that this will work correctly with all the memory allocation routines, but may cause surprises if one is freeing only part of a region. If byteSize is null, nothing happens. Also, the size of the block will be rounded up, so the freed block will fill an entire memory block.

INPUTS

freeList - points to the free list
memoryBlock - memory block to return
byteSize - the size of the desired block in bytes

SEE ALSO

Allocate

Exec/DoIO

Exec/DoIO

NAME DoIO -- perform an I/O command and wait for completion

SYNOPSIS
error = DoIO(iorequest)
D0 AI

FUNCTION
This function requests a device driver to perform the I/O command specified in the I/O request. This function will always block until the I/O request is completed.

INPUTS
iorequest - pointer to a properly initialized I/O request

RESULTS
error - see WaitIO

SEE ALSO
SendIO, WaitIO

Exec/Enqueue

Exec/Enqueue

NAME Enqueue -- Insert or append node to a system queue

SYNOPSIS
Enqueue(list, node)
A0 AI

FUNCTION
Insert or append a node into a system queue. The insert is performed based on the node priority -- it will keep the list properly sorted. New nodes will be inserted in front of the first node with a lower priority. Hence a FIFO queue for nodes of equal priority

INPUTS
list - a pointer to the system queue header
node - the node to enqueue

Exec/FindName

Exec/FindName

NAME FindName -- find a system list node with a given name

SYNOPSIS
node = FindName(start, name)
D0 A0 A1

FUNCTION
Traverse a system list until a node with the given name is found. To find multiple occurrences of a string, this function may be called with a node starting point.

INPUTS
start - a list header or a list node to start the search (if node, this one is skipped)
name - a pointer to a name string terminated with null

RESULTS
node - a pointer to the node with the same name else zero to indicate that the string was not found.

Exec/FindPort

Exec/FindPort

NAME FindPort -- find a given system message port

SYNOPSIS
port = FindPort(name)
D0 A1

FUNCTION
This function will search the system message port list for a port with the given name. The first port matching this name will be returned.

INPUT
name - name of the port to find
RETURN
port - a pointer to the message port, or zero if not found.

Exec/FindTask

Exec/FindTask

NAME FindTask -- find a task with the given name or find oneself

SYNOPSIS
task = FindTask(name)
D0 A1

FUNCTION
This function will check all task queues for a task with the given name, and return a pointer to its task control block. If a null name pointer is given a pointer to the current task will be returned.

INPUT
name - pointer to a name string

RESULT
task - pointer to the task

Exec/FreeEntry

Exec/FreeEntry

NAME FreeEntry -- free many regions of memory

SYNOPSIS
FreeEntry(memList)
A0

FUNCTION
This routine takes a memList structure (as returned by AllocEntry) and frees all the entries.

INPUTS
memList -- pointer to structure filled in with memEntry structures

Exec/FreeMem

Exec/FreeMem

NAME FreeMem -- deallocate with knowledge

SYNOPSIS FreeMem(memoryBlock, byteSize)
AI DO

FUNCTION Free a region of memory, returning it to the pool from which it came.

INPUTS memoryBlock - memory block to free
If the memoryBlock previously returned by an allocation routine.
byteSize - the size of the block in bytes

SEE ALSO AllocMem, AllocAbs

Exec/FreeSignal

Exec/FreeSignal

NAME FreeSignal -- free a signal bit

SYNOPSIS FreeSignal(signalNum)
DO

FUNCTION This function frees a previously allocated signal bit for reuse. This call must be performed while running in the same task in which the signal was allocated.

WARNING Signals may not be allocated or freed from exception handling code.

INPUTS signalNum - the signal number to free {0..31}

Exec/FreeTrap

Exec/FreeTrap

NAME FreeTrap -- free a processor trap

SYNOPSIS
FreeTrap (trapNum)
D0

FUNCTION

This function frees a previously allocated trap number for reuse. This call must be performed while running in the same task in which the trap was allocated.

WARNING

Traps may not be allocated or freed from exception handling code.

INPUTS

trapNum - the trap number to free (of 0..15)

Exec/CatCC

Exec/CatCC

NAME CatCC -- get condition codes in a 68010 compatible way.

SYNOPSIS
conditions = CatCC()
D0

FUNCTION

This function provides a means of obtaining the CPU condition codes in a manner that will make 68010 upgrades transparent.

INPUTS

RESULTS

conditions - the 68000/68010 condition codes

Exec/GetMsg

Exec/GetMsg

NAME GetMsg -- get next message from a message port

SYNOPSIS
message = GetMsg(port)
D0 A0

FUNCTION
This function receives a message from a given message port. It provides a fast, non-copying message receiving mechanism.

The received message is removed from the message port.

This function will not wait. If a message is not present this function will return zero. If a program must wait for a message, it can wait on the signal specified for the port or use the WaitPort function. There can only be one task waiting for any given port.

Getting the message does not imply that the message is now free to be reused. When the receiver is finished with the message, it may ReplyMsg it.

INPUT
port - a pointer to the receiver message port

RESULT
message - a pointer to the first message available. If there are no messages, return zero.

SEE ALSO
PutMsg, ReplyMsg, WaitPort

Exec/InitStruct

Exec/InitStruct

NAME InitStruct - initialize memory from a table

SYNOPSIS
InitStruct(InitTable, memory, size);
A1 A2 D0-0:16

FUNCTION
Clear a memory area except those words whose data and offset values are provided in the initialization table. This initialization table has byte commands to

```
|a ||byte| |given||byte| |once|
load |count||word| into |next ||rptr| offset, |repeatedly|.
```

Not all combinations are supported. The offset, when specified, is relative to the memory pointer provided (Memory), and is initially zero. The initialization data (InitTable) contains byte commands whose 8 bits are interpreted as follows:

```
ddssnnnn
dd the destination type (and size):
00 next destination, nnnn is count
01 next destination, nnnn is repeat
10 destination offset is next byte, nnnn is count
11 destination offset is next rptr, nnnn is count
ss the size and location of the source:
00 long, from the next two aligned words
01 word, from the next aligned word
10 byte, from the next byte
11 ERROR - will cause an ALERT (see below)
nnnn the count or repeat:
count the (number+1) of source items to copy
repeat the source is copied (number+1) times.
```

InitStruct commands are always read from the next even byte. Given destination offsets are always relative to memory (A2). The command 00000000 ends the InitTable stream: use 00010001 if you really want to copy one longword.

24 bit APTR not supported for 68020 compatibility -- use long.

INPUTS

InitTable - the beginning of the commands and data to init Memory with. Must be on an even boundary unless only byte initialization is done.
memory - the beginning of the memory to initialize. Must be on an even boundary if size is specified.
size - the size of memory, which is used to clear it before initializing it via the InitTable. If Size is zero,

memory is not cleared before initializing. Size is rounded down to the nearest even number before use.

IMPLEMENTATION

D0 clear size, command, count and repeat
D1 destination offset, command type
A0 current Memory pointer
A1 current InitTable pointer
D0,D1,A0,A1 destroyed

Exec/Insert

Exec/Insert

NAME Insert -- insert a node into a list

SYNOPSIS

Insert(list, node, listNode)
A0 A1 A2

FUNCTION

Insert a node into a doubly linked list AFTER a given node position. Insertion at the head of a list is performed by passing a zero value for listNode.

INPUTS

list - a pointer to the target list header
node - the node to insert
listNode - the node after which to insert

Exec/MakeLibLibrary

Exec/MakeLibLibrary

NAME MakeLibLibrary -- construct a library

SYNOPSIS library = MakeLibLibrary(vectors, structure, init, dataSize, segList)
D0 A0 A1 A2 D0 D1

FUNCTION This function is used for constructing a library vector and data area. Space for the library is allocated from the system's free memory pool. The size fields of the library are filled. The data portion of the library is initialized. A library specific entrypoint is called (init) if present.

INPUTS vectors - pointer to an array of function pointers or function displacements. If the first word of the array is -1, then the array contains relative word displacements (based off of vectors); otherwise, the array contains absolute function pointers.

structure - points to an "InitStruct" data region. If null, then it will not be called.

init - an entry point that will be called before adding the library to the system. If null, it will not be called. When it is called, it will be called with the libAddr in D0, and its result will be the result of this function.

dSize - the size of the library data area, including the standard library node data.

segList - pointer to a memory segment list (used by DOS) This is passed to a library's init code.

RESULT library - the reference address of the library. This is the address used in references to the library, not the beginning of the memory area allocated.

EXCEPTION If the library vector table require more system memory than is available, this function will cause a system panic.

SEE ALSO InitStruct

Exec/OpenDevice

Exec/OpenDevice

NAME OpenDevice -- gain access to a device

SYNOPSIS error = OpenDevice(devName, unitNumber, iORequest, flags)
D0 A0 D0 A1 D1

FUNCTION This function opens the named device/unit and initializes the given I/O request block.

INPUTS devName - requested device name

unitNumber - the unit number to open on that device. The format of the unit number is device specific.

iORequest - the I/O request block to be returned with appropriate fields initialized.

flags - additional driver specific information. This is sometimes used to request opening a device with exclusive access.

RESULTS error - zero if successful, else an error is returned

SEE ALSO CloseDevice

Exec/OpenLibrary

Exec/OpenLibrary

NAME OpenLibrary -- gain access to a library

SYNOPSIS
library = OpenLibrary(libName, version)
D0 AI D0

FUNCTION
This function returns a pointer to a library that was previously installed into the system. If the requested library is exists, and if the library version is greater than or equal to the requested version, then the open will succeed.

INPUTS
libName - the name of the library to open
version - the version of the library required.

RESULTS
library - a library pointer for a successful open, else zero
SEE ALSO
CloseLibrary

Exec/OpenResource

Exec/OpenResource

NAME OpenResource -- gain access to a resource

SYNOPSIS
resource = OpenResource(resName)
D0 AI

FUNCTION
This function returns a pointer to a resource that was previously installed into the system.

INPUTS
resName - the name of the resource requested.
RESULTS
resource - if successful, a resource pointer, else null
SEE ALSO
CloseResource

Exec/PutMsg

Exec/PutMsg

NAME PutMsg -- put a message to a message port

SYNOPSIS PutMsg(port, message)
A0 A1

FUNCTION This function attaches a message to a given message port. It provides a fast, non-copying message sending mechanism. Messages can be attached to only one port at a time. The message body can be of any size or form. Because messages are not copied, cooperating tasks share the same message memory. The sender task should not recycle the message until it has been replied by the receiver. Of course this depends on the message handling conventions setup by the involved tasks. If the ReplyPort field is non-zero, when the message is replied by the receiver, it will be sent back to that port.

Any one of the following actions can be set to occur when a message is put:

1. no special action
2. signal a given task
3. cause a software interrupt

3. cause a software interrupt
The action is selected depending on the value set in PB_ACTION of MP_FLAGS.

INPUT port - pointer to a message port
message - pointer to a message

SEE ALSO GetMsg, ReplyMsg

Exec/RamDevice

Exec/RamDevice

NAME RamDevice -- remove a device from the system

SYNOPSIS error = RamDevice(device)
D0 A1

FUNCTION This function removes an existing device from the system. This function deletes the device from the device name list, so no new opens can occur.

INPUTS device - pointer to a device node

RESULTS error - zero if successful, else an error is returned

SEE ALSO AddDevice

Exec/RemHead

Exec/RemHead

NAME RemHead -- remove the head node from a list

SYNOPSIS
node = RemHead(list)
D0 A0

FUNCTION Get a pointer to the head node and remove it from the list.

INPUTS list - a pointer to the target list header

RESULT node - the node removed or zero when empty list

Exec/RemIntServer

Exec/RemIntServer

NAME RemIntServer -- remove an interrupt server

SYNOPSIS
RemIntServer(intNum, interrupt)
D0-0:4 A1

FUNCTION This function removes an interrupt server node from the given server chain.

If this server was the last one in the chain interrupts will be disabled for intNum.

INPUTS intNum - the Portia interrupt bit (0..14)
interrupt - pointer to an interrupt server node

SEE ALSO
AddIntServer

Exec/RemLibrary

Exec/RemLibrary

NAME RemLibrary -- remove a library from the system

SYNOPSIS
error = RemLibrary(library)
D0 Al

FUNCTION
This function removes an existing library from the system.
It will delete it from the system library name list, so no
new opens may be performed.

INPUTS
library - pointer to a library node structure

RESULTS
error - zero if successful, else an error number

SEE ALSO
AddLibrary

Exec/Remove

Exec/Remove

NAME Remove -- remove a node from a list

SYNOPSIS
Remove (node)
Al

FUNCTION
Remove a node from a list.

INPUTS
node - the node to remove

Exec/RemPort

Exec/RemPort

NAME RemPort -- remove a message port from the system

SYNOPSIS
RemPort (port)
AI

FUNCTION
This function removes a message port structure from the system's message port list. Subsequent attempts to rendezvous by name with this port will fail.

INPUTS
port - pointer to a message port

SEE ALSO
AddPort, FindPort

Exec/RemResource

Exec/RemResource

NAME RemResource -- remove a resource from the system

SYNOPSIS
RemResource (resource)
AI

FUNCTION
This function removes an existing resource from the system.

INPUTS
resource - pointer to a resource node

SEE ALSO
AddResource

Exec/RemFail

Exec/RemFail

NAME RemFail -- remove the tail node from a list

SYNOPSIS
node = RemFail(list)
D0 A0

FUNCTION Get a pointer to the tail node and remove it from the list.

INPUTS list - a pointer to the target list header

RESULT node - the node removed or zero when empty list

Exec/RemTask

Exec/RemTask

NAME RemTask -- remove a task from the system

SYNOPSIS
RemTask(task)
A1

FUNCTION This function removes a task from the system. Deallocation of resources should have been performed prior to calling this function.

INPUTS task - pointer to the task node representing the task to be removed. A zero value indicates self removal, and will cause the next ready task to begin execution.

SEE ALSO AddTask

Exec/ReplyMsg

Exec/ReplyMsg

NAME ReplyMsg -- put a message to its reply port

SYNOPSIS ReplyMsg(message)
A1

FUNCTION This function sends a message to its reply port. This is usually done when the receiver of a message has finished and wants to return it to the sender (so that it can be re-used or deallocated, whatever).

INPUT message - a pointer to the message

SEE ALSO ReplyMsg

Exec/SandIO

Exec/SandIO

NAME SandIO -- initiate an I/O command

SYNOPSIS SandIO(IORrequest)
A1

FUNCTION This function requests the device driver to initiate the command specified in the given I/O request. The device will return regardless of whether the I/O has completed.

INPUTS IORrequest - pointer to an I/O request

SEE ALSO DoIO, WaitIO

Exec/SetExcept Exec/SetExcept

NAME SetExcept -- define certain signals to cause exceptions

SYNOPSIS
oldSignals = SetExcept(newSignals, signalMask)
D0 D1

FUNCTION
This function defines which of the task's signals will cause an exception. When any of the signals occurs the task's exception handler will be dispatched. If the signal occurred prior to calling SetExcept, the exception will happen immediately.

INPUTS
newSignals - the new values for the signals specified in signalMask.
signalMask - the set of signals to be effected

RESULTS
oldSignals - the prior exception signals

EXAMPLE
Get the current state of all exception signals:
SetExcept(0,0)
Change a few exception signals:
SetExcept(\$1374,\$1074)

SEE ALSO
Signal, SetSignal

Exec/SetFunction

Exec/SetFunction

NAME SetFunction -- change a function vector in a library

SYNOPSIS
oldFunc = SetFunction(library, funcOffset, funcEntry)
D0 A1 A0.W D0

FUNCTION
SetFunction is a functional way of changing those parts of a library that are checksummed. They are changed in such a way that the summing process will never falsely declare a library to be invalid.

INPUTS
library - a pointer to the library to be changed
funcOffset - the offset that FuncEntry should be put at.
funcEntry - pointer to new function

Exec/SetIntVector

Exec/SetIntVector

NAME SetIntVector -- set a system interrupt vector

SYNOPSIS
oldInterrupt = SetIntVector (intNumber, interrupt)
D0 D0-0:4 A1

FUNCTION This function provides a mechanism for setting the system interrupt vectors. Both the code and data pointers of the vector are set to the new values. A pointer to the old interrupt structure is returned. When the system calls the specified interrupt code the registers are setup as follows:

- D0 - scratch
 - D1 - scratch (on entry: active portia interrupts)
 - A0 - scratch (on entry: pointer to chipbase)
 - A1 - scratch (on entry: interrupt's data segment)
 - A5 - jump vector register (scratch on call)
 - A6 - library base pointer (scratch on call)
- all other registers - must be preserved

INPUTS
intNum - the Portia interrupt bit number (0..14)
interrupt - a pointer to a node structure containing the handler's entry point and data segment pointer.
It is a good idea to give the node a name so that other users may identify who currently has control of the interrupt.

RESULT
A pointer to the prior interrupt node which had control of this interrupt.

Exec/SetSignal

Exec/SetSignal

NAME SetSignal -- define the state of this task's signals

SYNOPSIS
oldSignals = SetSignal (newSignals, signalMask)
D0 D0 D1

FUNCTION This function defines the states of the task's signals. This function is considered dangerous.

INPUTS
newSignals - the new values for the signals specified in signalSet.
signalMask - the set of signals to be effected

RESULTS
oldSignals - the prior values for all signals

EXAMPLE
Get the current state of all signals:
SetSignal(0,0)
Clear all signals:
SetSignal(0,FFFFFFFF)

SEE ALSO
Signal, Wait

Exec/SetSR

Exec/SetSR

NAME SetSR -- get and/or set processor status register

SYNOPSIS
oldSR = SetSR (newSR, mask)
D0 D0 D1

FUNCTION This function provides a means of modifying the CPU status register in a "safe" way (well, how safe can a function like this be anyway?). This function will only effect the status register bits specified in the mask parameter. The prior content of the entire status register is returned.

INPUTS
newSR - new values for bits specified in the mask.
All other bits are not effected.
mask - bits to be changed

RESULTS
oldSR - the entire status register before new bits

EXAMPLES
To get the current SR:
currentSR = SetSR(0, 0);
To change the processor interrupt level to 3:
oldSR = SetSR(\$0300,\$0700);
Set processor interrupts back to prior level:
SetSR(oldSR,\$0700);

Exec/SetTaskPri

Exec/SetTaskPri

NAME SetTaskPri -- get and set the priority of a task

SYNOPSIS
oldPriority = SetTaskPri(task, priority)
D0-0:8 A1 D0-0:8

FUNCTION This function changes the priority of a task regardless of its state. The old priority of the task is returned. A reschedule is performed, and a context switch may result.

INPUTS
task - task to be affected
priority - the new priority for the task

RESULT
oldPriority - the tasks previous priority

Exec/Signal

Exec/Signal

NAME Signal -- signal a task

SYNOPSIS Signal(task, signals)
AI D0

FUNCTION This function signals a task with the given signals. If the task is currently waiting for one or more of these signals, it will be made ready and a reschedule will occur. If the task is not waiting for any of these signals, the signals will be posted to the task for possible later use. A signal may be sent to a task regardless of whether it's running, ready, or waiting.

This function is considered "low level". Its main purpose is to support multiple higher level functions like PutMsg. Generally a user need not perform Signals directly.

INPUT task - the task to be signalled
signals - the signals to be sent

SEE ALSO Wait, SetSignal

Exec/SumLibrary

Exec/SumLibrary

NAME SumLibrary -- compute and check the checksum on a library

SYNOPSIS SumLibrary(library)
AI

FUNCTION SumLibrary computes a new checksum on a library. It can also be used to check an old checksum. If an old checksum does not match and the library has not been marked as changed then the system will alert the user.

INPUTS library - a pointer to the library to be changed

EXCEPTIONS An alert will occur if the checksum fails.

Exec/SuperState

Exec/SuperState

NAME SuperState -- enter supervisor state with user stack

SYNOPSIS
oldSysStack = SuperState()
D0

FUNCTION

Enter supervisor mode while running on the user's stack. The user still has access to user stack variables. Be careful though, the user stack must be large enough to accommodate space for all interrupt data -- this includes all possible nesting of interrupts. This function is a nop when called from supervisor state.

RESULTS

oldSysStack - system stack pointer
Save this. It will come in useful when you return to user state. If the system is already in supervisor mode, oldSysStack is zero.

SEE ALSO

UserState

Exec/UserState

Exec/UserState

NAME UserState -- return to user state with user stack

SYNOPSIS
UserState(sysStack)
D0

FUNCTION

Return to user state with user stack, from supervisor state with user stack. This function is normally used in conjunction with the SuperState function above.

This function must not be called from the user state.

INPUT

sysStack - supervisor stack pointer

SEE ALSO

SuperState

Exec/Wait

Exec/Wait

NAME Wait -- wait for one or more signals

SYNOPSIS
signals = Wait(signalSet)
D0

FUNCTION
This function will cause the current task to suspend waiting for one or more signals. When any of the specified signals occurs, the task will return to the ready state. If a signal occurred prior to calling Wait, the wait condition will be immediately satisfied, and the task will continue to run.

This function cannot be called while in supervisor mode!

INPUT
signalSet - the set of signals for which to wait.
Each bit represents a particular signal.

RESULTS

Exec/WaitIO

Exec/WaitIO

NAME WaitIO -- wait for completion of an I/O request

SYNOPSIS
error = WaitIO(iOrequest)
D0 A1

FUNCTION
This function waits for the specified I/O request to complete. If the I/O has already completed, this function will return immediately.

This function should be used with care, as it does not return until the I/O request completes; if the I/O never completes, this function will never return, and your task will hang. If this situation is a possibility, it is safer to use the Wait function, which will return when any particular signal is received. This is how I/O timeouts can be properly handled.

INPUTS
iOrequest - pointer to an I/O request block

RESULTS
error - zero if successful, else an error is returned

SEE ALSO
SendIO

Exec/WaitPort

Exec/WaitPort

NAME WaitPort -- wait for a given port to be non-empty

SYNOPSIS
message = WaitPort (port)
D0 A0

FUNCTION
This function waits for the given port to become non-empty. If necessary, the Wait function will be called to wait for the port signal. If a message is already present at the port, this function will return immediately. The return value is always a pointer to the first message queued (but it is not removed from the queue).

INPUT
port - a pointer to the message port

RETURN
message - a pointer to the first available message

SEE ALSO
GetMsg

TABLE OF CONTENTS

graphics.library/AddAnimOb
graphics.library/AddBob
graphics.library/AddFont
graphics.library/AddVSprite
graphics.library/AllcRaster
graphics.library/AndRectRegion
graphics.library/Animate
graphics.library/AreaDraw
graphics.library/AreaEnd
graphics.library/AreaMove
graphics.library/AskFont
graphics.library/AskSoftStyle
graphics.library/BitBitMap
graphics.library/BitClear
graphics.library/BitPattern
graphics.library/BitTemplate
graphics.library/CEND
graphics.library/ChangeSprite
graphics.library/CINIT
graphics.library/CLEAR
graphics.library/ClearRegion
graphics.library/ClearScreen
graphics.library/CloseFont
graphics.library/CMOVE
graphics.library/CopySBitMap
graphics.library/CWAIT
graphics.library/DisownBlitter
graphics.library/DisposeRegion
graphics.library/DoCollision
graphics.library/Draw
graphics.library/DrawGList
graphics.library/Flood
graphics.library/FreeColorMap
graphics.library/FreeCplList
graphics.library/FreeCprList
graphics.library/FreeCbuffers
graphics.library/FreeRaster
graphics.library/FreeSprite
graphics.library/FreePortCplLists
graphics.library/GetColorMap
graphics.library/GetCbuffers
graphics.library/GetCGBA
graphics.library/GetSprite
graphics.library/InitArea
graphics.library/InitBitMap
graphics.library/InitCells
graphics.library/InitMasks
graphics.library/InitMaskPort
graphics.library/InitNpMas
graphics.library/InitView
graphics.library/InitVPort

graphics.library/LoadRCBA
graphics.library/LoadView
graphics.library/LockLayerRom
graphics.library/MakeVPort
graphics.library/Move
graphics.library/MovSprite
graphics.library/MrgCop
graphics.library/NewRegion
graphics.library/OpenFont
graphics.library/OvRectRegion
graphics.library/OwnBlitter
graphics.library/PolyDraw
graphics.library/QBilit
graphics.library/QBSBlit
graphics.library/ReadPixel
graphics.library/RectFill
graphics.library/RemFont
graphics.library/RemIBob
graphics.library/RemVSprite
graphics.library/ScrollRaster
graphics.library/ScrollVPort
graphics.library/SetAPen
graphics.library/SetPen
graphics.library/SetCollision
graphics.library/SetDrMd
graphics.library/SetFont
graphics.library/SetRast
graphics.library/SetRCBA
graphics.library/SetSoftStyle
graphics.library/SortGList
graphics.library/SyncSBitMap
graphics.library/Text
graphics.library/TextLength
graphics.library/UnlockLayerRom
graphics.library/VBeamPos
graphics.library/WaitBlit
graphics.library/WaitBOWP
graphics.library/WaitTOF
graphics.library/WritePixel
graphics.library/XorRectRegion

graphics.library/AddAnimOb graphics.library/AddAnimOb

NAME AddAnimOb -- add an AnimOb to the linked list of AnimObs

SYNOPSIS AddAnimOb(anOb, anKey, RPort)
a0 a1 a2

FUNCTION Links this AnimOb into the current list pointed to by animKey. Initializes all the Timers of the AnimOb's components. Calls AddBob with each component's Bob. Note that the RPort must be correctly initialized before you call here, including a valid GelsInfo.

INPUTS anOb = pointer to the AnimOb structure to be added to the list
anKey = address of a ptr to the first AnimOb in the list (NULL if none)
RPort = pointer to a valid RastPort

RESULT Nothing

BUGS None known

SEE ALSO Nothing

graphics.library/AddBob graphics.library/AddBob

NAME AddBob -- adds a Bob to current gel list

SYNOPSIS AddBob(Bob, RPort)
a0 a1

FUNCTION Sets up the system Bob flags, then links this gel into the list via AddVSprite

INPUTS Bob = pointer to the Bob structure to be added to the gel list
RPort = pointer to a RastPort structure

RESULT Nothing

BUGS None known

SEE ALSO AddVSprite

graphics.library/AddFont

graphics.library/AddFont

NAME
AddFont -- add a font to the system list

SYNOPSIS
AddFont(textFont), GraphicsLib
A1 A6

FUNCTION
This function adds the text font to the system, making it available for use by any application. The font added must be in public memory, and remain until successfully removed.

INPUTS
textFont - a TextFont structure in public ram.

graphics.library/AddVSprite

graphics.library/AddVSprite

NAME
AddVSprite -- add a VSprite to the current gel list

SYNOPSIS
AddVSprite(VS, RPort) as called by C
a0 a1

FUNCTION
Sets up the system VSprite flags
Links this VSprite into the current gel list using its Y,X
INPUTS
VS = pointer to the VSprite structure to be added to the gel list
RPort = pointer to a RastPort structure

RESULT
Nothing

BUGS
None known

SEE ALSO
Nothing

graphics.library/AllocRaster graphics.library/AllocRaster

NAME AllocRaster -- allocate space for a Bit Plane

SYNOPSIS AllocRaster(width, height)
 d0 d1

FUNCTION This function calls the memory allocation routines to allocate memory space for a bitplane width bits wide and height bits high.
Returns a pointer to the first word if successful.
Return 0 if unable to allocate that amount of space.

INPUTS x,y are maximum dimensions of the array in bits.
SEE ALSO FreeRaster

graphics.library/AndRectRegion graphics.library/AndRectRegion

NAME AndRectRegion -- Perform 2d AND operation of rectangle with region, leaving result in region

SYNOPSIS AndRectRegion(region, rectangle)
 a0 a1

Function Clip away any portion of the region that exists outside of the rectangle. Leave the result in region.

INPUTS region = pointer to Region structure
rectangle = pointer to Rectangle structure

BUGS

graphics.library/Animate

graphics.library/Animate

NAME

Animate -- processes every AnimOb in the current animation list

SYNOPSIS

Animate (key, RPort)
a0 a1

FUNCTION

For every AnimOb in the list

- update its location and velocities
- call the AnimOb's special routine if one is supplied
- for each component of the AnimOb
 - if this sequence times out, switch to the new one
 - call this component's special routine if one is supplied
 - set the sequence's sprite's y,x coordinates based on all this

INPUTS

key = address of the variable that points to the head AnimOb
RPort = pointer to the RastPort structure

RESULT

Nothing

BUGS

None known

SEE ALSO

Nothing

graphics.library/AreaDraw

graphics.library/AreaDraw

NAME

AreaDraw -- add a point to a list of end points for arefill.

SYNOPSIS

error = (lint) AreaDraw(rp, x, y)
A1 D0 D1

FUNCTION

Add point to the vector buffer.

INPUTS

x,y are coordinates of a point in the raster
rp points to a RastPort structure

RETURNS

0 if no error
-1 if no space left in vector list

SEE ALSO

AreaMove, InitArea, AreaEnd

graphics.library/areadEnd

graphics.library/AreaEnd

NAME

AreaEnd -- Process table of vectors and produce areafill

SYNOPSIS

AreaEnd(rp)
A1

FUNCTION

Trigger the filling operation.

Process the vector buffer and generate required fill into the raster planes. After the fill is complete reinitialize for the next AreaMove. Use the raster set up by InitRaster when generating an areafill mask.

INPUTS

rp points to a RastPort structure

SEE ALSO

InitArea, AreaMove, AreaDraw

graphics.library/AreaMove

graphics.library/AreaMove

NAME

AreaMove -- defines a new starting point for a new shape in the vector list.

SYNOPSIS

error = AreaMove(rp, x, y)
A1 D0 D1

FUNCTION

Close the last polygon and start another polygon at (x,y). Enter necessary points in vector buffer.

Closing a polygon may result in the generation of another AreaDraw() to close previous polygon.

INPUTS

x,y are positions in the raster
rp points to a RastPort structure

RETURNS

0 if no error
-1 if no space left in vector list

SEE ALSO

InitArea, AreaDraw, AreaEnd

graphics.library/AskFont

graphics.library/AskFont

NAME

AskFont - get the text attributes of the current font

SYNOPSIS

AskFont (rastPort, textAttr), graphicalLib
A1 A0 A6

FUNCTION

This function fills the text attributes structure with the attributes of the current font in the rastPort.

INPUTS

rastPort - the RastPort from which the text attributes are extracted.

textAttr - the TextAttr structure to be filled.

graphics.library/AskSoftStyle

graphics.library/AskSoftStyle

NAME

AskSoftStyle - get the soft style bits of the current font

SYNOPSIS

enable = AskSoftStyle (rastPort), graphicalLib
A1 A6

FUNCTION

This function returns those style bits of the current font that are not intrinsic in the font itself, but algorithmically generated. These are the bits that are valid to set in the enable mask for SetSoftStyle

INPUTS

rastPort - the RastPort from which the font and style are extracted.

RESULTS

enable - those bits in the style algorithmically generated. Style bits that are not defined are also set.

graphics.library/BitBitmap

graphics.library/BitBitmap

NAME

BitBitmap -- move a rectangle in a raster

SYNOPSIS

```

Planes = BitBitmap(SrcBitmap, SrcX, SrcY, DestBitmap,
D0          D0          D1          A1
DestX, DestY, SizeX, SizeY, Minterm, Mask, TempA);
D2          D3 D4          D5          D6          D7          A2

```

FUNCTION

perform non-destructive blits to move a rectangle from one area in a raster to another area, which can be on a different raster.

INPUTS

SrcBitmap, DestBitmap - the BitMap(s) containing the rectangles

- the planes copied from the source to the destination are only those whose plane numbers are identical and less than the minimum plane count and whose write mask is non-zero.

SrcX, SrcY - the x and y coordinates of the upper left corner of the source rectangle. Valid range is positive signed integer such that the raster word's offset 0..(32767-Size)

DestX, DestY - the x and y coordinates of the upper left corner of the destination for the rectangle. Valid range is as for Src.

SizeX, SizeY - the size of the rectangle to be moved. Valid range is (X: 1..976; Y: 1..1023 such that final raster word's offset is 0..32767)

Minterm - the logic function to apply to the rectangle when A is non-zero (i.e. within the rectangle). B is the source rectangle and C, D is the destination for the rectangle.

- \$0C0 is a vanilla copy

- \$030 inverts the source before the copy

- \$050 ignores the source and inverts the destination

Mask - see the hardware reference manual for other combinations indicate the write mask to apply to this operation. Bits set the minimum plane count) are to participate in the operation. Typically this is set to 0xff.

TempA - If the copy overlaps exactly to the left or right (i.e. the scan line addresses overlap), and TempA is non-zero, it points to enough chip accessible memory to hold a line of A source for the blit.

RESULTS

Planes - the number of planes actually involved in the blit.

EXCEPTIONS

This blt is assumed to be friendly: no errors conditions (e.g. a rectangle outside the BitMap bounds) are tested or reported. A plane count that is less than expected can be attributed to a failure to allocate a TempA when it was needed and null

graphics.library/BitClear graphics.library/BitClear

NAME

BitClear - Clear a block of memory words to zero.

SYNOPSIS

BitClear(memBlock, bytecount, flags)
 a1 d0 d1

FUNCTION

For memory that is local and blitter accessible. The most efficient way to clear a range of memory locations is to use the system's most efficient data mover, the blitter. This command accepts the starting location and count and clears that block to zeros.

INPUTS

memBlock pointer to local memory to be cleared
 memBlock must be even
flags set bit 0 to force function to wait until blit is done.
bytecount set bit1 to use row/bytesperrow
 if (flags & 2) == 0 then
 even number of bytes to clear.
 else
 low 16 bits is taken as number of bytes
 per row and upper 16 bits taken as
 number of rows.

This function is somewhat hardware dependant. In the rows/bytesperrow mode, rows must be <=1024 and bytesperrow must be <=128
In standard bytecount mode multiple runs of the blitter may be used to clear all the memory.

RESULT

The block of memory is set to zeros.

BUGS

None known.

SEE ALSO

graphics.library/BitPattern

graphics.library/BitPattern

NAME

BitPattern -- Using standard drawing rules for areafill, blit through a mask

SYNOPSIS

BitPattern(RastPort *, char *, x1, y1, maxx, maxy, bytecnt)
 a1, a0 d0 d1 d2 d3 d4

FUNCTION

Blit using drawmode,areafill pattern,outline, mask pointed to by a0, at position rectangle (x1,y1) (maxx,maxy). The image is not shifted but must be word aligned.

INPUTS

a1 points to RastPort
a0 points to 2 dimensional mask if needed
x1,y1 upper left of rectangular region in RastPort
maxx,maxy points to lower right of rectangular region in RastPort
bytecnt number of BytesPerRow for char * a0

RETURNS

SEE ALSO

graphics.library/BitTemplate

graphics.library/BitTemplate

NAME

BitTemplate -- cookie cut a shape in a rectangle to the RastPort

SYNOPSIS

BitTemplate(source, srcX, srcMod, destRastPort,
A0 D0 D1 A1
destX, destY, sizeX, sizeY), graphicsLib
D2 D3 D4 D5 A6

FUNCTION

This function draws the image in the template into the RastPort in the current color and drawing mode at the specified position. The template is assumed not to overlap the destination.

EXCEPTIONS

If the template falls outside the RastPort boundary, it is truncated to that boundary.

graphics.library/CEND

graphics.library/CEND

NAME

CEND -- terminate user copper list.

SYNOPSIS

CEND(c)

FUNCTION

add instruction to terminate user copper list.

INPUTS

c = pointer to UCopList structure

RESULTS

this is actually a macro that calls CMait(c) to wait for the end of the user copper list and then calls CBump(c) to bump the local pointer to the next instruction.

BUGS

None Known

SEE ALSO

CINIT();
CMOVE();
CMAIT();

Dec 3 17:05 1985 graphics.doc Page 21

graphics.library/ChangeSprite graphics.library/ChangeSprite

NAME

ChangeSprite -- change the sprite image pointer.

SYNOPSIS

```
ChangeSprite( vp, s, newdata)
              a0 a1 a2
```

FUNCTION

The sprite image is changed to use the data starting at newdata

INPUTS

```
vp = pointer to ViewPort structure that this sprite is
    relative to.
s = pointer to SimpleSprite structure
newdata = pointer to data structure of the following form.
          struct spriteImage
          {
            UNWORD posctl[2]; /* used by simple sprite machine */
            UNWORD data[height][2]; /* actual sprite image */
            UNWORD reserved[2]; /* initialized to *
                                /* 0xEFFF, 0xEFFF */
          }

```

programmer must initialize reserved[2]. spriteImage must be in CHIP memory. the height subfield of the SimpleSprite structure must be set to reflect the height of the new spriteImage BEFORE calling ChangeSprite. The programmer may allocate two sprites to handle a single attached sprite. After GetSprite, ChangeSprite, the programmer can set the SPRITE_ATTACHED bit in posctl[1] of the odd numbered sprite.

RESULTS

BUGS

SEE ALSO
sprite.h FreeSprite ChangeSprite MoveSprite

Dec 3 17:05 1985 graphics.doc Page 22

graphics.library/CINIT

graphics.library/CINIT

NAME

CINIT -- initialize user copperlist to accept intermediate user copper instructions

SYNOPSIS

```
struct CopperList *CINIT( c, n )
```

FUNCTION

allocates/initializes copperlist datastructures/buffers

INPUTS

```
c = pointer to UCopList structure
n = number of instructions buffer must hold
```

RESULTS

this is actually a macro that calls UCopperListInit(c,n) if (c== 0) allocate CopperList structure and a buffer to hold n copper instructions. If (c != 0) then just reinitialize the list to accept copper instructions and ignore n.

BUGS

graphics.library/ClearEOL

graphics.library/ClearEOL

NAME

ClearEOL - clear from current position to end of line

SYNOPSIS

ClearEOL(rastPort), graphicsLib
A1 A6

FUNCTION

Clear a rectangular swath from the current position to the right edge of the rastPort. The height of the swath is taken from that of the current text font, and the vertical positioning of the swath is adjusted by the text baseline, such that text output at this position would lie wholly on this newly cleared area.

Clearing consists of setting the color of the swath to zero, or, if the DrawMode is 2, to the BgPen.

graphics.library/ClearRegion

graphics.library/ClearRegion

NAME

ClearRegion -- set this region to size 0

SYNOPSIS

ClearRegion(region)
a0

Function

Clip away all rectangles in the region leaving nothing.

INPUTS

region = pointer to Region structure

BUCKS

graphics.library/ClearScreen

graphics.library/ClearScreen

NAME

ClearScreen - clear from current position to end of RastPort

SYNOPSIS

ClearScreen(rastPort), graphicsLib
A1 A6

FUNCTION

Clear a rectangular swath from the current position to the right edge of the rastPort with ClearEOL, then clear the rest of the screen from just beneath the swath to the bottom of the rastPort.

Clearing consists of setting the color of the swath to zero, or, if the DrawMode is 2, to the BgPen.

graphics.library/CloseFont

graphics.library/CloseFont

NAME

CloseFont - release a pointer to a system font.

SYNOPSIS

CloseFont(font), GraphicsLib
A1 A6

FUNCTION

This function indicates that the font specified is no longer in use. It is used to close a font opened by OpenFont, so that fonts that are no longer in use do not consume system resources.

INPUTS

font -
A font, as returned by OpenFont.

graphics.library/CMOVE

graphics.library/CMOVE

NAME

CMOVE -- append copper move instruction to user copper list.

SYNOPSIS

CMOVE (c , a , v)

FUNCTION

add instruction to move value v to hardware register a.

INPUTS

c = pointer to UCopList structure
a = hardware register
v = 16 bit value to be written

RESULTS

this is actually a macro that calls Cmove(c, &a, v)
and then calls CBump(c) to bump the local pointer
to the next instruction.

BUGS

graphics.library/CopySBitMap

graphics.library/CopySBitMap

NAME

CopySBitMap -- Synchronize Layer window with contents of
Super BitMap

SYNOPSIS

CopySBitMap(layer *)
a0

FUNCTION

This is the inverse of SyncSBitMap
Copy all bits from SuperBitMap to Layer bounds.
This is used for those functions that do not
want to deal with the ClipRect structures but do want
to be able to work with a SuperBitMap Layer.

INPUTS

layer * is a pointer to a Layer that has a SuperBitMap
The Layer should already be locked by the caller.

SEE ALSO

SyncSBitMap

Dec 3 17:05 1985 graphics.doc Page 29

graphics.library/CWAIT graphics.library/CWAIT

NAME CWAIT -- append copper wait instruction to user copper list.

SYNOPSIS CWAIT(c , v , h)

FUNCTION add instruction to wait for vertical beam position v and horizontal position h

INPUTS

c = pointer to UCopList structure
v = vertical beam position (relative to top of viewport)
h = horizontal beam position

RESULTS

this is actually a macro that calls CWait(c,v,h) and then calls CBump(c) to bump the local pointer to the next instruction.

BUGS

Dec 3 17:05 1985 graphics.doc Page 30

graphics.library/DisownBlitter

graphics.library/DisownBlitter

NAME DisownBlitter - return blitter to free state.

SYNOPSIS DisownBlitter ()

FUNCTION Free blitter up for use by other blitter users.

INPUTS

RETURNS

SEE ALSO OwnBlitter

graphics.library/DisposeRegion graphics.library/DisposeRegion

NAME DisposeRegion -- return all space for this region to free
memory pool

SYNOPSIS DisposeRegion(region)
a0

Function Free all RegionRectangles for this Region then
free the Region itself

INPUTS region = pointer to Region structure

BUGS

graphics.library/DoCollision graphics.library/DoCollision

NAME DoCollision -- tests every gel in gel list for collisions

SYNOPSIS DoCollision(RPort)
a1

FUNCTION

Tests each gel in gel list for boundary and gel-to-gel collisions
On detecting one of these collisions, the appropriate collision-handling
routine is called. See the documentation for a thorough description of
which collision routine is called.
This routine expects to find the gel list correctly sorted in Y,X order.
The system routine SortGList performs this function for the user.

INPUTS

RPort = pointer to a struct RastPort

RESULT

Nothing

BUGS

Doesn't handle gel-to-gel collisions completely correctly

SEE ALSO

SortGList

graphics.library/Draw

graphics.library/Draw

NAME

Draw -- draw a line between the current pen position and the new x,y position

SYNOPSIS

Draw(rp, x, y
A1 D0 D1

FUNCTION

Draw a line from the current pen position to (x,y).

INPUTS

rp pointer to a RastPort
x,y point in the RastPort to end the line.

graphics.library/DrawCList

graphics.library/DrawCList

NAME

DrawCList -- process the gel list, queuing VSprites, drawing Bobs

SYNOPSIS

DrawCList(RPort, VPort)
a1 a0 as called by C

FUNCTION

Performs one pass of the current gel list
- if nextLine and lastColor are defined, these are initialized
- for each gel
- if it's a VSprite build it into the copper list
- if it's a Bob, draw it into the current raster
- copy the save values into the "old" variables, double-buffering if required

INPUTS

a1 = pointer to the RastPort where Bobs will be drawn
a5 = pointer to GfxBase

RESULT

Nothing

BUGS

MUSTDRAW isn't implemented yet. Probably won't be for this release either. We are sad.

SEE ALSO

Nothing

Dec 3 17:05 1985 graphics.doc Page 35

graphics.library/Flood

graphics.library/Flood

NAME

Flood -- flood rastport like areafill

SYNOPSIS

Flood(rp, mode, x, y)
a1 d2 d0 d1

FUNCTION

Search the BitMap starting at (x,y). Fill all adjacent pixels if they are:

a: arenot the same as ACPen Mode 0
a: same as the one at (x,y) Mode 1

When actually doing the fill use the modes that apply to standard areafill routines such as drawmodes and patterns.

INPUTS

rp pointer to RastPort
(x,y) coordinate in BitMap
mode 0 fill all adjacent pixels searching for border
1 fill all adjacent pixels that have same pen number as (x,y)

SEE ALSO

BUGS

None known

Dec 3 17:05 1985 graphics.doc Page 36

graphics.library/FreeColorMap

graphics.library/FreeColorMap

NAME

FreeColorMap -- free the ColorMap structure and return memory to free memory pool

SYNOPSIS

FreeColorMap(colormap)
a0

INPUTS

colormap pointer to ColorMap allocated with GetColorMap

RESULT

The space is made available for others to use.

BUGS

SEE ALSO

SetRGB4 GetColorMap

graphics.library/FreesCoplList

graphics.library/FreesCoplList

NAME

FreesCoplList -- deallocate intermediate copper list

SYNOPSIS

FreesCoplList(coplList)

FUNCTION

deallocate all memory associated with this copper list

INPUTS

coplList pointer to structure CoplList

RESULTS

memory returned to memory manager

BUGS

none known

SEE ALSO

graphics.library/FreesCprList

graphics.library/FreesCprList

NAME

FreesCprList -- deallocate hardware copper list

SYNOPSIS

FreesCprList(cprList)

FUNCTION

return cprList to free memory pool

INPUTS

cprList pointer to cprList structure

RESULTS

none known

BUGS

SEE ALSO

graphics.library/FreeCBuffers graphics.library/FreeCBuffers

NAME FreeCBuffers -- deallocate memory gotten by GetCBuffers

SYNOPSIS FreeCBuffers(anOb, RPort, db) as called by C
a0 a1 d0

FUNCTION For each sequence of each component of the AnimOb, deallocate memory for:
SaveBuffer
BorderLine
CollMask and ImageShadow (point to same buffer)
if db is set (user wants double-buffering) deallocate:
DBufPacket
BufBuffer

INPUTS a1 = pointer to the AnimOb structure
a2 = pointer to the current RastPort
d0 = double-buffer indicator (set TRUE for double-buffering)

RESULT

BUCS None known

SEE ALSO Nothing

graphics.library/FreeRaster

graphics.library/FreeRaster

NAME FreeRaster -- release an allocated area to the system free memory pool.

SYNOPSIS FreeRaster (p, width, height)
a0 d0 d1

INPUTS p = a pointer to a memory space returned as a result of a call to AllocRaster.

width the width in bits of the bitplane.
height the height in bits of the bitplane.
the same values of width and height with which you called AllocRaster in the first place, when the pointer p returned. This defines the size of the memory space which is to be returned to the free memory pool.

FUNCTION

To return to the free memory pool the memory space which had been allocated by a call to AllocRast.

NOTE: Always use the same values that were used with AllocRaster

Dec 3 17:05 1985 graphics.doc Page 41

graphics.library/FreeSprite graphics.library/FreeSprite

NAME FreeSprite -- return sprite for use by others and virtual
sprite machine

SYNOPSIS
FreeSprite(pick)
do

FUNCTION mark sprite as available for others to use.

INPUTS pick = 0-7

RESULTS sprite made available for subsequent callers of GetSprite
as well as use by Virtual Sprite Machine

BUGS These sprite routines are provided to ease sharing of sprite
hardware and to handle simple cases of sprite usage and
movement. It is assumed the programs that use these routines
do want to be good citizens in their hearts. ie: they will
not FreeSprite unless they actually own the sprite.
Virtual Sprite machine may ignore simple sprite machines.

SEE ALSO sprite.h GetSprite ChangeSprite MoveSprite

Dec 3 17:05 1985 graphics.doc Page 42

graphics.library/FreePortCopLists graphics.library/FreePortCopLists

NAME FreePortCopLists -- deallocate all intermediate copper lists and
their headers from a viewport

SYNOPSIS
FreePortCopLists (viewport)

FUNCTION recursively search display, color, sprite, and user copper
lists and call FreeMem() to deallocate them from memory

INPUTS viewport pointer to ViewPort structure

RESULTS vp->DspIns == NULL; vp->SprIns == NULL; vp->ClrIns == NULL;
vp->UCopIns == NULL;

BUGS none known

SEE ALSO

Dec 3 17:05 1985 graphics.doc Page 43

graphics.library/GetColorMap graphics.library/GetColorMap

NAME GetColorMap -- allocate and initialize ColorMap

SYNOPSIS GetColorMap(entries)
 d0

INPUTS entries number of entries for this colormap

RESULT allocate and initial the required structures to be attached to the ViewPort to save color values.
returns 0 if cannot allocate memory for structures

BUGS

SEE ALSO SetRCB4 FreeColorMap

Dec 3 17:05 1985 graphics.doc Page 44

graphics.library/GetCBuffers graphics.library/GetCBuffers

NAME GetCBuffers -- attempts to allocate ALL the buffers of an entire AnimOb

SYNOPSIS GetCBuffers(anOb, RPort, db) as called by C
 a0 a1 d0

FUNCTION For each sequence of each component of the AnimOb, allocate memory for:
SaveBuffer
BorderLine

CollMask and ImageShadow (point to same buffer)
if db is set (user wants double-buffering) allocate:
DBufPacket
BufBuffer

INPUTS

a1 = pointer to the AnimOb structure
a2 = pointer to the current RastPort
d0 = double-buffer indicator (set TRUE for double-buffering)

RESULT

TRUE if the memory allocations were all successful, else FALSE

BUGS

None known

SEE ALSO

Nothing

graphics.library/GetRCBA

graphics.library/GetRCBA

NAME

GetRCBA -- Inquire value of entry in ColorMap

SYNOPSIS

GetRCBA(colormap, entry)
a0 d0

INPUTS

colormap pointer to ColorMap structure
entry index into colormap

RESULT

returns -1 if no valid entry
return UMOR0 RGB value 4 bits per gun right justified

BUGS

SEE ALSO

SetRCBA LoadRCBA GetColorMap FreeColorMap

graphics.library/GetSprite

graphics.library/GetSprite

NAME

GetSprite -- attempt to get a sprite for the simple sprite manager.

SYNOPSIS

Sprite_Number = GetSprite(sprite, pick)
d0 d0

FUNCTION

attempt to allocate one of the eight sprites for private use with the simple sprite manager. This must be done before using further calls to simple sprite machine.

INPUTS

sprite = ptr to programmers SimpleSprite structure.
pick = 0-7
-1 if programmer just wants the next one.

RESULTS

if pick is 0-7 attempt to allocate the sprite. If the sprite is already allocated then return -1
if pick -1 allocate the next sprite. If no sprites are available return -1.

If the sprite is available for allocation, mark it allocated and fill in the 'num' entry of the SimpleSprite structure. If successful return the sprite number.

BUGS

SEE ALSO

sprite.h FreeSprite ChangeSprite MoveSprite GetSprite

graphics.library/InitArea

graphics.library/InitArea

NAME

InitArea - Initialize vector collection matrix

SYNOPSIS

```
InitArea( AreaInfo *, buffer *, max vectors )
          a0          a1          d0
```

FUNCTION

This function provides initialization for the vector collection matrix such that it has a size of (max vectors). The size of the region pointed to by buffer (short pointer) should be five (5) times as large as (max vectors). This size is in bytes. Areafills done by using AreaMove, AreaDraw and AreaEnd must have enough space allocated in this table to store all the points of the largest fill. If not enough space the routines will return -1

INPUTS

```
AreaInfo = pointer to AreaInfo structure
buffer = pointer to chunk of memory to collect vertices
max vectors = max number of vectors this buffer can hold
```

RESULT

Pointers are set up to begin storage of vectors done by AreaMove and AreaDraw.

NOTE

The underlying graphics routines actually split the table into two parts to save coordinates and flags

BUGS

None known.

SEE ALSO

graph.h AreaEnd AreaMove AreaDraw

graphics.library/InitBitmap

graphics.library/InitBitmap

NAME

InitBitmap - Initialize bit map structure with input values

SYNOPSIS

```
InitBitmap( bm, depth, width, height )
            a0 d0          d1          d2
```

FUNCTION

Initialize various elements in the Bitmap structure to correctly reflect input depth, width, and height. Must be used before use of Bitmap in other graphics calls. The Planes[8] are not initialized and need to be set up by the caller. The Planes table was put at the end of the structure so that it may be truncated if needed, as well as extended.

INPUTS

```
bm = pointer to a Bitmap structure (gfx.h)
depth = number of bitplanes that this bitmap will have
width = number of bits (columns) wide for this Bitmap
height = number of bits (rows) tall for this Bitmap
```

BUGS

None known.

SEE ALSO

The gfx.h

graphics.library/InitCells

graphics.library/InitCells

NAME InitCells -- initialize a gel list; must be called before using gels

SYNOPSIS InitCells(head, tail, CInfo)
a0 a1 a2

FUNCTION Assigns the VSprites as the head and tail of the gel list in GfxBase
Links these two gels together as the keystones of the list
If the collHandler vector points to some memory array, sets the
BORDERHIT vector to NULL

INPUTS head = pointer to the VSprite structure to be used as the gel list head
tail = pointer to the VSprite structure to be used as the gel list tail
CInfo = pointer to the CelsInfo structure to be initialized

RESULT Nothing

BUCS None known

SEE ALSO Nothing

graphics.library/InitCMasks

graphics.library/InitCMasks

NAME InitCMasks -- initialize all the masks of an AnimOb

SYNOPSIS InitCMasks(anOb)
a0 as called by C

FUNCTION For every sequence of every component call InitMasks

INPUTS a1 = pointer to the AnimOb

RESULT Nothing

BUCS None known

SEE ALSO Nothing

graphics.library/InitMasks

graphics.library/InitMasks

NAME

InitMasks -- initializes the BorderLine and CollMask masks of a VSprite

SYNOPSIS

InitMasks (VS) as called by C
a0

FUNCTION

Creates the appropriate BorderLine and CollMask masks of the VSprite. Correctly detects if the VSprite is actually a Bob definition, handles the image data accordingly

INPUTS

VS = pointer to the VSprite structure

RESULT

Nothing

BUGS

SEE ALSO
Nothing

graphics.library/InitRastPort

graphics.library/InitRastPort

NAME

InitRastPort - Initialize raster port structure

SYNOPSIS

InitRastPort (rp)
a1

FUNCTION

Initialize a RastPort structure to standard values.

The struct Rastport describes a control structure for a write-able raster. The RastPort structure describes how a complete single playfield display will be written into. A RastPort structure is referenced whenever any drawing or filling operations are to be performed on a section of memory.

The section of memory which is being used in this way may or may not be presently a part of the current actual onscreen display memory. The name of the actual memory section which is linked to the RastPort is referred to here as a "raster" or as a bitmap.

NOTE: Calling the routine InitRastPort only establishes various defaults. It does NOT establish where, in memory, the rasters are located. To do graphics with this RastPort the user must set up the BitMap pointer in the RastPort.

INPUTS

rp = pointer to a RastPort structure.

RESULT

all entries in RastPort get zeroed out.

exceptions:

The following get -1:

Mask.FgPen,AOLPen,LinePtrn

DrawMode = JAM2

The font is set to the standard system font

BUGS

None known.

SEE ALSO

The rastport.h

graphics.library/InitTmpRas

graphics.library/InitTmpRas

NAME

InitTmpRas -- Initialize area of local memory for usage by
areafill, floodfill, text

SYNOPSIS

InitTmpRas(tmpRas *,buffer *, size)
a0 a1 d0

FUNCTION

The area of memory pointed to by buffer is set up to be used
by RastPort routines that may need to get some memory for
intermediate operations in preparation to putting the graphics
into the final BitMap.
tmpRas is used to control the usage of buffer.

INPUTS

tmpRas pointer to a TmpRas structure to be linked into
a RastPort
buffer pointer to a contiguous piece of chip memory.
size size in bytes of buffer

RESULT

makes buffer available for users of RastPort

BUGS

none known.
Would be nice if RastPorts could share one TmpRas.

graphics.library/InitView

graphics.library/InitView

NAME

InitView - Initialize View structure

SYNOPSIS

InitView(view)
 a1

FUNCTION

Initialize View structure to default values.

INPUTS

view = pointer to a View structure

RESULT

First, View structure set to all 0's.
Then values are put in DxOffset, DyOffset to properly position
default display about .5 inches from top and left on monitor.
InitView pays no attention to previous contents of view.

BUGS

None known.

SEE ALSO

view.h

graphics.library/InitVPort

graphics.library/InitVPort

NAME

InitVPort - Initialize ViewPort structure

SYNOPSIS

InitVPort(vp)
 a0

FUNCTION

Initialize ViewPort structure to default values.

INPUTS

vp = pointer to a ViewPort structure

RESULT

ViewPort structure set to all 0's.

BUGS

None known.

SEE ALSO

view.h

graphics.library/LoadRCB4

graphics.library/LoadRCB4

NAME

LoadRCB4 -- load RCB color values from table

SYNOPSIS

LoadRCB4(vp, colormap, count)
a0 a1 d0

FUNCTION

load the count words of the colormap from table

INPUTS

vp = pointer to ViewPort, whose colors you want to change
colormap= pointer to table of RCB values set up like an array
of USHORTS

background-- 0x0RCB
color1 -- 0x0RCB
color2 -- 0x0RCB
etc. UNORD per value.

The colors are interpreted as 15 = maximum intensity.
0 = minimum intensity.

count = number of UNORDs in the table to load into the
colormap starting at color 0 (background) and proceeding
to the next higher color number

RESULTS

store the colors in the ViewPorts colormap. This is a
table of gotten from GetColorMap(number of entries).

This colormap will be initialized from the Default colormap.

BUGS

None known

SEE ALSO

view.h

graphics.library/LoadView

graphics.library/LoadView

NAME

LoadView -- Use a (possibly freshly created) coprocessor instruction
list to create the current display.

SYNOPSIS

LoadView(View)
A1

FUNCTION

See NAME field. Coprocessor instruction list has been created by
InitVPort, MakeView, and MrgCop.

INPUTS

View - a pointer to the View structure which contains the
pointer to the constructed coprocessor instructions list.

RESULT

The new View is displayed, according to your instructions.
The vertical blank routine will pick this pointer up and
direct copper to start displaying this View.

BUGS

...

SEE ALSO

InitVPort, MakeView, MrgCop
Intuition's RethinkDisplay()

graphics.library/LockLayerRom graphics.library/LockLayerRom

NAME LockLayerRom -- Lock Layer structure by rom(gfx lib) code

SYNOPSIS LockLayerRom(layer)
a5

FUNCTION Return when the layer is locked and no other may alter the ClipRect structure in the Layer structure.

INPUTS layer = pointer to Layer structure

NOTE This call does not destroy any registers. This call nests so that callers in this chain will not lock themselves out.

This lock does not prevent another task from calling LockLayerRom() and not blocking. This is potentially dangerous in the case of ScrollRaster which will resort the list of ClipRects although it does not add any new ClipRects or remove any ClipRects.

SEE ALSO layers.h

graphics.library/MakeVPort graphics.library/MakeVPort

NAME MakeVPort -- generate display copper list

SYNOPSIS MakeVPort(view, viewport)
a0 a1

FUNCTION Using information in the View, ViewPort construct intermediate copper list for this ViewPort.

INPUTS view = pointer to View structure
viewport = pointer to ViewPort structure
The viewport must have valid ptr to RasInfo

RESULTS constructs intermediate copper list and puts pointers in viewport.DspIns
If the ColorMap ptr in ViewPort is nil then it uses colors from the default color table.
If DUALPF in Modes then there must be a second RasInfo pointed to by the first RasInfo

BUCS

SEE ALSO MrGCop() view.h
Intuition's MakeScreen(), RemakeDisplay(), and RethinkDisplay()

graphics.library/Move

graphics.library/Move

NAME

Move -- Move graphics pen position

SYNOPSIS

Move (rp, x, y)
a1 d0 d1

FUNCTION

Move graphics pen position to (x,y) relative to upper left (0,0) of RastPort.

Note: Text uses the same position.

INPUTS

rp = pointer to a RastPort structure
x,y= point in the RastPort

graphics.library/MoveSprite

graphics.library/MoveSprite

NAME

MoveSprite -- Move sprite to a point relative to top of viewport

SYNOPSIS

MoveSprite(vp, sprite, x, y)
a0 a1 d0 d1

FUNCTION

Move sprite image to new place on display.

INPUTS

vp = pointer to ViewPort structure
= 0, if sprite positioned relative to View
sprite = pointer to SimpleSprite structure
x,y = new position relative to top of viewport

RESULTS

BUGS

SEE ALSO

sprite.h FreeSprite ChangeSprite GetSprite

graphics.library/MrgCop

graphics.library/MrgCop

NAME

MrgCop -- Merge together coprocessor instructions.

SYNOPSIS

MrgCop(View)
AI

FUNCTION

Merge together the display, color, sprite and user coprocessor instructions into a single coprocessor instruction stream. This essentially creates a per-display-frame program for the coprocessor. This function MrgCop is used, for example, by the graphics animation routines which effectively add information into an essentially static background display. This changes some of the user or sprite instructions, but not those which have formed the basic display in the first place. When all forms of coprocessor instructions are merged together, you will have a complete per-frame instruction list for the coprocessor.

Restrictions: Each of the coprocessor instruction lists MUST be internally sorted in min to max Y-X order. The merge routines depend on this!
Each list must be terminated using CEND(copperlist)

INPUTS

View - a pointer to the view structure whose coprocessor instructions are to be merged.

RESULT

The view structure will now contain a complete, sorted/merged list of instructions for the coprocessor, ready to be used by the display processor. The display processor is told to use this new instruction stream through the instruction LoadView().

BUGS

...

SEE ALSO

InitVPort, MrgCop, LoadView
Intuition's RethinkDisplay()

graphics.library/NewRegion

graphics.library/NewRegion

NAME

NewRegion -- get a region of size 0

SYNOPSIS

rgn = (struct Region *)NewRegion()
do

Function

Create a Region structure, initialize it to empty and return a pointer to it.

INPUTS

none

BUGS

graphics.library/OpenFont

graphics.library/OpenFont

NAME

OpenFont - get a pointer to a system font.

SYNOPSIS

font = OpenFont(textAttr), graphicsLib

D0 A0 A6

FUNCTION

This function searches the system font space for the graphics text font that best matches the attributes specified. The pointer to the font returned can be used in subsequent SetFont and CloseFont calls. It is important to match this call with a corresponding CloseFont call for effective management of ram fonts.

INPUTS

textAttr - a TextAttr structure that describes the text font attributes desired

EXCEPTIONS

D0 is zero if the desired font cannot be found. If the named font is found, but the size and style specified are not available, a font with the nearest attributes is returned.

graphics.library/OrRectRegion

graphics.library/OrRectRegion

NAME

OrRectRegion -- Perform 2d OR operation of rectangle with region, leaving result in region

SYNOPSIS

OrRectRegion(region, rectangle)

a0 a1

Function

If any portion of rectangle is not in the region then add that portion to the region

INPUTS

region = pointer to Region structure
rectangle = pointer to Rectangle structure

BUGS

graphics.library/OwnBlitter

graphics.library/OwnBlitter

NAME OwnBlitter - get the blitter for private usage

SYNOPSIS OwnBlitter ()

FUNCTION return when the blitter has been locked from others using it and can now be used by this task. Before actually using the new owner should call WaitBlit, which waits until any previous blit that the blitter may have been doing is actually done.

INPUTS

RETURNS

SEE ALSO DisownBlitter

graphics.library/PolyDraw

graphics.library/PolyDraw

NAME PolyDraw -- Draw lines from table of (x,y) values.

SYNOPSIS PolyDraw(rp, count, array)
 a1 d0 a1

FUNCTION starting with the first pair draw connected lines to it and every succeeding pair.

INPUTS rp = pointer to RastPort structure
count = number of points in array (x,y) pairs
array = pointer to first (x,y) pair

BUGS none known

SEE ALSO Draw()

graphics.library/QBlit

graphics.library/QBlit

NAME

QBlit -- Queue up a request for blitter usage

SYNOPSIS

QBlit(bp)
Al

FUNCTION

Link a request for the use of the blitter to the end of the current blitter queue. The pointer bp points to a blit structure containing, among other things, the link information, and the address of your routine which is to be called when the blitter queue finally gets around to this specific request. When your routine is called, you are in control of the blitter ... it is not busy with anyone else's requests. This means that you can directly specify the register contents and start the blitter. See the description of the blit structure and the uses of QBlit in the section titled Graphics Support in the OS Kernel Manual. The header of a blitter structure is shown in hardware/blit.h

INPUTS

bp = pointer to a blit structure

RESULT

Your routine is called when the blitter is ready for you.

NOTE

In general requests for blitter usage through this channel are put in front of those who use the blitter via OwnBlitter and DisownBlitter. However for small blits there is more overhead using the queuer than Own/Disown Blitter.

BUGS

None known

SEE ALSO

QBSBlit blit.h

graphics.library/QBSBlit

graphics.library/QBSBlit

NAME

QBSBlit -- Synchronize the blitter request with the video beam.

SYNOPSIS

QBSBlit(bsp)
al

FUNCTION

Call a user routine for use of the blitter, enqueued separately from the QBlit queue. Calls the user routine contained in the blit structure when the video beam is located at a specified position onscreen. Useful when you are trying to blit into a visible part of the screen and wish to perform the data move while the beam is not trying to display that same area. (prevents showing part of an old display and part of a new display simultaneously). Blitter requests on the QBSBlit queue take precedence over those on the regular blitter queue. The beamposition is specified the blitnode.

INPUTS

bsp = pointer to a blit structure. See description in the Graphics Support section of the manual for more info.

RESULT

User routine is called when the QBSBlit queue reaches this request AND the video beam is in the specified position.

BUGS

...

SEE ALSO

QBlit

graphics.library/ReadPixel

graphics.library/ReadPixel

NAME

ReadPixel -- read the pen number value of the pixel at a specified x,y location within a certain RastPort.

SYNOPSIS

penno = (int)ReadPixel(rp, x, y)
 D0 al D0 D1

FUNCTION

Combine the bits from each of the bit-planes used to describe a particular RastPort into the pen number selector which that bit combination normally forms for the system hardware selection of pixel color.

INPUTS

x is the X coordinate within the range of the RastPort size.
y is the Y coordinate within the range of the RastPort size.
rp is a pointer to a RastPort structure
rp is a pointer to a RastPort structure

RESULT

Pen (0..255) number at that position is returned.
-1 is returned if cannot read that pixel

BUGS

...

SEE ALSO

WritePixel

graphics.library/RectFill

graphics.library/RectFill

NAME

RectFill -- Fill a defined rectangular area with the current drawing pen color, outline color, secondary color, and pattern.

SYNOPSIS

RectFill(rp, xmin, ymin, xmax, ymax)
 A1 D0 D1 D2 D3

FUNCTION

Fill the rectangular region specified by the parameters with the chosen pen colors, areafill pattern, and drawing mode.

INPUTS

(xmin,ymin) (xmax,ymax) are the coordinates of the upper left corner and the lower right corner, respectively, of the rectangle.
(xmax >= xmin) and (ymax >= ymin)

rp points to the RastPort which receives the filled rectangle.

SEE ALSO

graphics.library/RemFont

graphics.library/RemFont

NAME

RemFont - remove a font from the system list

SYNOPSIS

error = RemFont(textFont), GraphicsLib
D0 A1 A6

FUNCTION

This function removes a font from the system, ensuring that access to it is restricted to those applications that currently have an active pointer to it: i.e. no new GetFont requests to this font are satisfied.

INPUTS

textFont - the Textfont structure to remove.

graphics.library/RemIBob

graphics.library/RemIBob

NAME

RemIBob -- removes immediately a Bob from the gel list and the RastPort

SYNOPSIS

RemIBob(Bob, RPort, VPort)
a0 a1 a2

FUNCTION

Removes a Bob immediately by uncoupling it from the gel list and erasing it from the RastPort

INPUTS

Bob = pointer to the Bob to be removed
RPort = pointer to the RastPort if the Bob is to be erased
VPort = pointer to the ViewPort for beam-synchronizing

RESULT

Nothing

BUGS

None known

SEE ALSO

RemVSprite

graphics.library/RemvSprite

graphics.library/RemvSprite

NAME

RemvSprite -- removes a VSprite to the current gel list

SYNOPSIS

RemvSprite(VS)
a0

FUNCTION

Unlinks the VSprite from the current gel list

INPUTS

VS = pointer to the VSprite structure to be removed from the gel list

RESULT

Nothing

BUGS

None known

SEE ALSO

Nothing

graphics.library/ScrollRaster

graphics.library/ScrollRaster

NAME

ScrollRaster -- push bits in rectangle in raster around by dx,dy towards 0,0 inside rectangle

SYNOPSIS

ScrollRaster(rp, dx, dy, xmin, ymin, xmax, ymax)
a1 d0 d1 d2 d3 d4 d5

FUNCTION

move the bits in the raster by (dx,dy) towards (0,0) The space vacated is RectFilled with RFPen. Limit the scroll operation to the rectangle defined by (xmin,ymin)(xmax,ymax). Bits outside will not be affected.

INPUTS

rp must a valid pointer to a RastPort
dx,dy are integers that may be positive, zero, or negative

EXAMPLE

ScrollRaster(rp, 0, 1) /* shift raster up by one row */
ScrollRaster(rp, -1, -1) /* shift raster down and to the right by 1 pixel

BUGS

graphics.library/ScrollVPort graphics.library/ScrollVPort

NAME

ScrollVPort -- push bits in rectangle in vport around by dx,dy towards 0,0 inside rectangle

SYNOPSIS

ScrollVPort(vp)
a0

FUNCTION

After the programmer has adjusted the Offset values in the RasInfo structures of ViewPort, change the the copper lists to reflect the the Scroll positions.

INPUTS

vp must a valid pointer to a ViewPort that is currently on display

RESULTS

modifies hardware and intermediate copperlists to reflect new RasInfo

NOTE

Changing the BitMap ptr in RasInfo and not changing the the Offsets will effect a double buffering affect.

BUGS

pokes not fast enough to avoid some visible hashing of display

graphics.library/SetAPen graphics.library/SetAPen

NAME

SetAPen -- Set primary pen

SYNOPSIS

SetAPen(rp, pen)
a1 d0

FUNCTION

Set the primary drawing pen for lines, fills, and text.

INPUTS

rp = pointer to RastPort structure.
pen = 0-255

RESULT

Changes the minterns in the RastPort to reflect new primary pen.
Set line drawer to restart pattern.

BUGS

SEE ALSO

SetBPen

graphics.library/SetBPen graphics.library/SetBPen

NAME SetBPen -- Set secondary pen

SYNOPSIS SetBPen(rp, pen)
 a1 d0

FUNCTION Set the secondary drawing pen for lines, fills, and text.

INPUTS rp = pointer to RastPort structure.
 pen = 0-255

RESULT Changes the minterms in the RastPort to reflect new secondary pen.
Set line drawer to restart pattern.

BUGS
SEE ALSO SetBPen

graphics.library/SetCollision

graphics.library/SetCollision

NAME SetCollision -- sets a pointer to a user collision routine

SYNOPSIS SetCollision(num, routine, GInfo)
 d0 a0 a1

FUNCTION Sets entry h in the user's collision vectors table equal to the pointer p

INPUTS num = collision vector number
 routine = pointer to the user's collision routine
 GInfo = pointer to a Galsinfo structure

RESULT Nothing

BUGS None known

SEE ALSO Nothing

graphics.library/SetDrMd

graphics.library/SetDrMd

NAME

SetDrMd -- Set drawing mode

SYNOPSIS

SetDrMd(rp, mode)
 a1 d0

FUNCTION

Set the drawing mode for lines, fills and text.

INPUTS

rp = pointer to RastPort structure.
mode = 0-255
#define JAM1 0 /* jam 1 color into raster */
#define JAM2 1 /* jam 2 colors into raster */
#define COMPLEMENT 2 /* XOR bits into raster */
#define INVERSEVID 4 /* inverse video for drawing modes */
some combinations may not make much sense.

RESULT

The mode set is dependant on the bits selected.
Change minterms to reflect new drawing mode.
Set line drawer to restart pattern.

BUGS

SEE ALSO SetAPen

graphics.library/SetFont

graphics.library/SetFont

NAME

SetFont -- set the text font and attributes in a RastPort

SYNOPSIS

error = SetFont(rastPort, font), graphicsLib
D0 A1 A0 A6

FUNCTION

This function sets the font in the RastPort to that described by font, and updates the text attributes to reflect that change. If TextAttr is zero, this call leaves the RastPort with no font. This function clears the effect of any previous soft styles.

INPUTS

RastPort - the RastPort in which the text attributes are changed.
font - an open font.

graphics.library/SetRast

graphics.library/SetRast

NAME

SetRast - Set an entire drawing area to a specified color.

SYNOPSIS

SetRast (RastPort, pen)
 A1 D0

FUNCTION

Set the entire contents of the specified RastPort to the specified pen.

INPUTS

RastPort is a pointer to the rastPort you wish to use.
pen is the pen value which you wish to fill into that port. (0-255)

RESULT

The drawing area becomes the selected pen number.

BUGS

...

SEE ALSO

...

graphics.library/SetRGB4

graphics.library/SetRGB4

NAME

SetRGB4 -- set one color register for this viewport

SYNOPSIS

SetRGB4 (vp, n, r, g, b)
 a0 D0 D1 D2 D3

INPUTS

vp= viewport to affect
n = the color number (range from 0 to 31)

r = red level
g = green level
b = blue level

RESULT

If there is a ColorMap for this viewport store the value in the structure ColorMap.
The selected color register is changed to match your specs.
If the color value is unused then nothing will happen.

BUGS

If the color value is unused it may affect the color values in the next viewports.

...

SEE ALSO

LoadRGB4

graphics.library/SetSoftStyle

graphics.library/SetSoftStyle

NAME

SetSoftStyle - set the soft style of the current font

SYNOPSIS

newStyle = SetSoftStyle(rastPort, style, enable), graphicsLib
A1 D0 D1 A6

FUNCTION

This function alters the soft style of the current font. Only those bits that are also set in enable are affected. The resulting style is returned, since some style request changes will not be honored when the implicit style of the font precludes changing them.

INPUTS

rastPort - the RastPort from which the font and style are extracted.

style - the new font style to set, subject to enable.

enable - those bits in style to be changed. Any set bits here that would not be set as a result of AskSoftStyle will be ignored, and the newStyle result will not be as expected.

RESULTS

style - the resulting style, both as a result of previous soft style selection, the effect of this function, and the style inherent in the set font.

graphics.library/SortGList

graphics.library/SortGList

NAME

SortGList -- sort the current gel list according to the y,x coordinates

SYNOPSIS

SortGList(RPort) as called by C
a1

FUNCTION

Sorts the current gel list according to the gels' y,x coordinates. This sorting is essential before calls to DrawGList or DoCollision.

INPUTS

RPort = pointer to the RastPort structure containing the GelsInfo

RESULT

Nothing

BUGS

None known

SEE ALSO

DoCollision
DrawGList

graphics.library/SyncSBitMap

graphics.library/SyncSBitMap

NAME

SyncSBitMap -- Synchronize Super BitMap with whatever is in the standard Layer bounds

SYNOPSIS

SyncSBitMap(layer *)
a0

FUNCTION

Copy all bits from ClipRects in Layer into Super BitMap BitMap. This is used for those functions that do not want to deal with the ClipRect structures but do want to be able to work with a SuperBitMap Layer.

INPUTS

layer * is a pointer to a Layer that has a SuperBitMap
The Layer should already be locked by the caller.

SEE ALSO

CopySBitMap

graphics.library/Text

graphics.library/Text

NAME

Text - write text characters (no formatting)

SYNOPSIS

error = Text(RastPort, string, count), gfxLib
D0 A1 A0 D0-0:16 A6

FUNCTION

This graphics function writes printable text characters to the specified RastPort at the current position. No control meaning is applied to any of the characters, and only text on the current line is output.

INPUTS

RastPort - a pointer to the RastPort which describes where the text is to be output
count - the string length. If zero, there are no characters to be output.
string - the address of string to output

EXCEPTIONS

BOUNDS -

If the characters displayed run past the RastPort boundary, the current position is truncated to the boundary, and thus does not represent the true position.

graphics.library/TextLength

graphics.library/TextLength

NAME

TextLength - determine raster length of text data

SYNOPSIS

length = TextLength(rastPort, string, count)
D0 A1 A0 D0-0:16

FUNCTION

This graphics function determines the length that text data would occupy if output to the specified RastPort with the current attributes. The length is specified as the number of raster dots: to determine what the current position would be after a Write using this string, add the length to cp_x (cp_y is unchanged by Write).

INPUTS

RastPort - a pointer to the RastPort which describes where the text attributes reside.
string - the address of string to determine the length of
count - the string length. If zero, there are no characters in the string.

RESULTS

length - the number of pixels in x this text would occupy, not including any negative kerning that may take place at the beginning of the text string, nor taking into account the effects of any clipping that may take place.

BUGS

A length that would overflow single word arithmetic is not calculated correctly.

graphics.library/UnlockLayerRom

graphics.library/UnlockLayerRom

NAME

UnlockLayerRom -- Unlock Layer structure by rom(gfx lib) code

SYNOPSIS

UnlockLayerRom(layer)
a5

FUNCTION

Decrement Lock count and Unlock Layer if the result is 0. Once the Layer is really unlocked the layerlib may then modify this Layer.

INPUTS

layer = pointer to Layer structure

NOTE

There should be an UnlockLayer for every LockLayer. This call does destroy scratch registers.

SEE ALSO

layers.h LockLayer ()

graphics.library/VBeamPos graphics.library/VBeamPos

NAME

VBeamPos -- get vertical beam position at this instant

SYNOPSIS

pos = VBeamPos()
do

FUNCTION

get the vertical beam position from the hardware.

INPUTS

none

RESULT

interrogates hardware for beam position and returns value.
valid results in the range of 0-255

BUGS

Because of hardware constraints if the vertical beam is
between 256 and 262 then 0 through 6 may be returned.

NOTE

Because of multitasking, the actual value returned may have
no use.

graphics.library/WaitBlit graphics.library/WaitBlit

NAME

WaitBlit -- Wait for the blitter to be finished before proceeding
with anything else.

SYNOPSIS

WaitBlit()

FUNCTION

WaitBlit returns when the blitter is idle. This function should
normally only be used when dealing with the blitter in a
synchronous manner, such as when using OwnBlitter and DisownBlitter.
WaitBlit does not wait for all blits queued up using QBlit or
QBSBlit.

INPUTS

none

RESULT

Your program waits until the blitter is finished.

BUGS

Because of a bug in agnus. This code may return too soon when
the blitter has infact not started the blit yet, even though
BltSize has been written. This most often occurs in a heavily
loaded system with extended memory, HIRRES, and 4 bitplanes.

SEE ALSO

OwnBlitter, DisownBlitter

Dec 3 17:05 1985 graphics.doc Page 93

graphics.library/MaitBOVP graphics.library/MaitBOVP

NAME MaitBOVP -- Wait till vertical beam reached bottom of
this viewport.

SYNOPSIS
MaitBOVP (ViewPort)
a0

FUNCTION Returns when vertical beam reaches bottom of this viewport

INPUTS ViewPort = pointer to ViewPort structure

Dec 3 17:05 1985 graphics.doc Page 94

graphics.library/MaitTOE graphics.library/MaitTOE

NAME MaitTOE -- Wait for the top of the next video frame

SYNOPSIS
MaitTOE ()

FUNCTION Wait for vertical blank to occur and all vertical blank
service routines to complete before returning to caller.

BUCS
INPUTS none

Dec 3 17:05 1985 graphics.doc Page 95

graphics.library/WritePixel

graphics.library/WritePixel

NAME

WritePixel -- change the pen num of one specific pixel in a specified RasterPort.

SYNOPSIS

WritePixel(rp, x, y)
 a1 D0 D1

FUNCTION

Changes the pen number of the selected pixel in the specified RasterPort to that currently specified by PenA, the primary drawing pen. Obey DrawModes and minterms in RasterPort.

INPUTS

x - the X coordinate within the RasterPort at which the selected pixel is located.
y - the Y coordinate.
rp - a pointer to the RasterPort to use.

RESULT

The pixel is changed.

BUGS

...

SEE ALSO

ReadPixel

Dec 3 17:05 1985 graphics.doc Page 96

graphics.library/XorRectRegion

graphics.library/XorRectRegion

NAME

XorRectRegion -- Perform 2d XOR operation of rectangle with region, leaving result in region

SYNOPSIS

XorRectRegion(region, rectangle)
 a0 a1

Function

Clip away any portion of the region that exists outside of the rectangle. Leave the result in region.

INPUTS

region = pointer to Region structure
rectangle = pointer to Rectangle structure

BUGS

this one does not work yet

Dec 3 17:05 1985 graphics.doc Page 97

TABLE OF CONTENTS

icon.library/AddFreeList
 icon.library/AllocMBOBject
 icon.library/BumpRevision
 icon.library/FindToolType
 icon.library/FreeDiskObject
 icon.library/FreeFreeList
 icon.library/FreeMBOBject
 icon.library/GetDiskObject
 icon.library/GetIcon
 icon.library/GetMBOBject
 icon.library/MatchToolValue
 icon.library/PutDiskObject
 icon.library/PutIcon
 icon.library/PutMBOBject
 Local

icon.library/AddFreeList
 icon.library/AddFreeList

NAME
 AddFreeList - add memory to the free list

SYNOPSIS
 status = AddFreeList(free, mem, len)
 D0 A0 A1 A2

FUNCTION
 This routine adds the specified memory to the free list.
 The free list will be extended (if required). If there
 is not enough memory to complete the call, a null is returned.
 Note that AddFreeList does NOT allocate the requested memory.
 It only records the memory in the free list.

INPUTS
 free -- a pointer to a FreeList structure
 mem -- the base of the memory to be recorded
 len -- the length of the memory to be recorded

RESULTS
 status -- nonzero if the call succeeded.

EXCEPTIONS
 SEE ALSO
 AllocEntry, FreeEntry, FreeFreeList

BUGS

Icon.library/Alloc#BObject

Icon.library/Alloc#BObject

NAME Alloc#BObject - allocate a Workbench object

SYNOPSIS
object = Alloc#BObject()
D0

FUNCTION

This routine allocates a Workbench object, and initializes its free list. A subsequent call to Free#BObject will free all of its memory.

If memory cannot be obtained, a NULL is returned.

This routine is intended only for internal users that can track changes to the Workbench.

INPUTS

RESULTS

object - the BObject (if memory is available)

EXCEPTIONS

SEE ALSO

AllocEntry, FreeEntry, Free#BObject

BUGS

Icon.library/BumpRevision

Icon.library/BumpRevision

NAME BumpRevision - reformat a name for a second copy

SYNOPSIS
result = BumpRevision(newbuf, oldname)
D0 A0 A1

FUNCTION

BumpRevision takes a name and turns it into a "copy of name". It knows how to deal with copies of copies. The routine will truncate the new name to the maximum dos name size (currently 30 characters).

INPUTS

newbuf - the new buffer that will receive the name (it must be at least 31 characters long).
oldname - the original name

RESULTS

result - a pointer to newbuf

EXCEPTIONS

EXAMPLE

| | |
|----------------------------------|----------------------------------|
| oldname | newbuf |
| ----- | ----- |
| "foo" | "copy of foo" |
| "copy of foo" | "copy 2 of foo" |
| "copy 2 of foo" | "copy 3 of foo" |
| "copy 199 of foo" | "copy 200 of foo" |
| "copy foo" | "copy of copy foo" |
| "copy 0 of foo" | "copy 1 of foo" |
| "012345678901234567890123456789" | "copy of 0123456789012345678901" |

SEE ALSO

BUGS

icon.library/FindToolType

icon.library/FindToolType

NAME FindToolType - find the value of a ToolType variable

SYNOPSIS
value = FindToolType(toolTypeArray, typeNames)
D0 A0 A1

FUNCTION
This function searches a tool type array for a given entry, and returns a pointer to that entry. This is useful for finding standard tool type variables. The returned value is not a new copy of the string but is only a pointer to the part of the string after typeName.

INPUTS
toolTypeArray - an array of strings
typeNames - the name of the tooltype entry

RESULTS
value - a pointer to a string that is the value bound to typeName, or NULL if typeNames is not in the toolTypeArray.

EXCEPTIONS

EXAMPLE
Assume the tool type array has two strings in it:
"FILETYPE=text"
"TEMPDIR=:t"

FindToolType(toolTypeArray, "FILENAME") returns "text"
FindToolType(toolTypeArray, "TEMPDIR") returns ":t"
FindToolType(toolTypeArray, "MAXSIZE") returns NULL

SEE ALSO
MatchToolValue

BUGS

icon.library/FreeDiskObject

icon.library/FreeDiskObject

NAME FreeDiskObject - free all memory in a Workbench disk object

SYNOPSIS
FreeDiskObject(diskobj)
A0

FUNCTION
This routine frees all memory in a Workbench disk object, and the object itself. It is implemented via FreeFreeList().

CatDiskObject() takes care of all the initialization required to set up the objects free list. This procedure may ONLY be called on DiskObject allocated via GetDiskObject().

INPUTS
diskobj -- a pointer to a DiskObject structure

RESULTS

EXCEPTIONS

SEE ALSO
GetDiskObject, FreeFreeList

BUGS

icon.library/FreeFreeList

icon.library/FreeFreeList

NAME FreeFreeList - free all memory in a free list

SYNOPSIS
FreeFreeList(free)
AO

FUNCTION

This routine frees all memory in a free list, and the free list itself. It is useful for easily getting rid of all memory in a series of structures. There is a free list in a Woribench object, and this contains all the memory associated with that object.

A FreeList is a list of MemList structures. See the MemList and MemEntry documentation for more information.

If the FreeList itself is in the free list, it must be in the first MemList in the FreeList.

INPUTS

free -- a pointer to a FreeList structure

RESULTS

EXCEPTIONS

SEE ALSO

AllocEntry, FreeEntry, AddFreeList

BUGS

icon.library/FreeBObject

icon.library/FreeBObject

NAME FreeBObject - free all memory in a Woribench object

SYNOPSIS
FreeBObject(obj)
AO

FUNCTION

This routine frees all memory in a Woribench object, and the object itself. It is implemented via FreeFreeList().

AllocBObject() takes care of all the initialization required to set up the objects free list.

This routine is intended only for internal users that can track changes to the Woribench.

INPUTS

free -- a pointer to a FreeList structure

RESULTS

EXCEPTIONS

SEE ALSO

AllocEntry, FreeEntry, AllocBObject, FreeFreeList

BUGS

icon.library/GetDiskObject

icon.library/GetDiskObject

NAME GetDiskObject - read in a Workbench disk object

SYNOPSIS
diskobj = GetDiskObject(name)
D0 A0

FUNCTION
This routine reads in a Workbench disk object in from disk. The name parameter will have a ".info" postpended to it, and the info file of that name will be read. If the call fails, it will return zero. The reason for the failure may be obtained via IoErr().

This routine is very similar to CetIcon, but shields the programmer from the worst of the grunginess associated with CetIcon. A FreeList structure is allocated just after the DiskObject structure; FreeDiskObject makes use of this to get rid of the memory that was allocated.

INPUTS
name -- name of the object

RESULTS
diskobj -- the Workbench disk object in question

EXCEPTIONS

SEE ALSO
CetIcon, FreeDiskObject

BUGS

icon.library/GetIcon

icon.library/GetIcon

NAME GetIcon - read in a DiskObject structure from disk

SYNOPSIS
status = GetIcon(name, icon, free)
D0 A0 A1 A2

FUNCTION
This routine reads in a DiskObject structure, and its associated information. All memory will be automatically allocated, and stored in the specified FreeList. The file name of the info file will be the name parameter with a ".info" postpended to it. If the call fails, a zero will be returned. The reason for the failure may be obtained via IoErr().

Users are encouraged to use GetDiskObject instead of this routine

INPUTS
name -- name of the object
icon -- a pointer to a DiskObject
free -- a pointer to a FreeList

RESULTS
status -- non-zero if the call succeeded.

EXCEPTIONS

SEE ALSO

BUGS

icon.library/CeathBObject icon.library/CeathBObject

NAME CeathBObject - read in a Workbench object

SYNOPSIS
object = CeathBObject(name)
D0 A0

FUNCTION
This routine reads in a Workbench object in from disk. The name parameter will have a ".info" postpended to it, and the info file of that name will be read. If the call fails, it will return zero. The reason for the failure may be obtained via IoErr().

This routine is intended only for internal users that can track changes to the Workbench.

INPUTS
name -- name of the object

RESULTS
object -- the Workbench object in question

EXCEPTIONS

SEE ALSO

BUGS

icon.library/MatchToolValue icon.library/MatchToolValue

NAME MatchToolValue - check a tool type variable for a particular value

SYNOPSIS
result = MatchToolValue(typeString, value)
D0 A0 A1

FUNCTION
MatchToolValue is useful for parsing a tool type value for a known value. It knows how to parse the syntax for a tool type value (in particular, it knows that '|' separates alternate values).

INPUTS
typeString - a ToolType value (as returned by FindToolType)
value - you are interested if value appears in typeString

RESULTS
result - a one if the value was in typeString

EXCEPTIONS

EXAMPLE
Assume there are two type strings:
type1 = "text"
type2 = "a|b|c"

MatchToolValue(type1, "text") returns 1
MatchToolValue(type1, "data") returns 0
MatchToolValue(type2, "a") returns 1
MatchToolValue(type2, "b") returns 1
MatchToolValue(type2, "d") returns 0
MatchToolValue(type2, "a|b") returns 0

SEE ALSO
FindToolType

BUGS

icon.library/PutDiskObject

icon.library/PutDiskObject

NAME PutDiskObject - write out a DiskObject to disk

SYNOPSIS status = PutDiskObject(name, diskobj)
D0 A0 A1

FUNCTION This routine writes out a DiskObject structure, and its associated information. The file name of the info file will be the name parameter with a ".info" postpended to it. If the call fails, a zero will be returned. The reason for the failure may be obtained via IoErr().

PutDiskObject and PutIcon are functionally identical. They are both provided so there is a Put/Get/Free triple for disk objects.

INPUTS

name -- name of the object
diskobj -- a pointer to a DiskObject

RESULTS

status -- non-zero if the call succeeded

EXCEPTIONS

SEE ALSO

GetDiskObject, FreeDiskObject, PutIcon

BUGS

icon.library/PutIcon

icon.library/PutIcon

NAME PutIcon - write out a DiskObject to disk

SYNOPSIS status = PutIcon(name, icon)
D0 A0 A1

FUNCTION This routine writes out a DiskObject structure, and its associated information. The file name of the info file will be the name parameter with a ".info" postpended to it. If the call fails, a zero will be returned. The reason for the failure may be obtained via IoErr().

PutDiskObject and PutIcon are functionally identical. They are both provided so there is a Put/Get/Free triple for disk objects.

Users are encouraged to use PutDiskObject instead of this routine

INPUTS

name -- name of the object
icon -- a pointer to a DiskObject

RESULTS

status -- non-zero if the call succeeded

EXCEPTIONS

SEE ALSO

BUGS

icon.library/PutWObject

icon.library/PutWObject

NAME PutWObject - write out a Workbench object

SYNOPSIS
status = PutWObject(name, object)
DO A0 A1

FUNCTION

This routine writes a Workbench object out to disk. The name parameter will have a ".info" postpended to it, and that file name will have the disk-resident information written into it. If the call fails, it will return a zero. The reason for the failure may be obtained via IoErr().

This routine is intended only for internal users that can track changes to the Workbench.

INPUTS

name -- name of the object
object -- the Workbench object to be written out

RESULTS

status -- non-zero if the call succeeded.

EXCEPTIONS

SEE ALSO

BUGS

TABLE OF CONTENTS

AddCadget
AllocRemember
AutoRequest
BeginRefresh
BuildSysRequest
ClearDRRequest
ClearMenuStrip
ClearPointer
CloseScreen
CloseWindow
CloseWorkBench
CurrentTime
DisplayAlert
DisplayBeep
DoubleClick
DrawBorder
DrawImage
EndRefresh
EndRequest
FreeRemember
FreeSysRequest
GetDefPrefs
GetPrefs
InitRequester
IntuiTextLength
ItemAddress
MakeScreen
ModifyIDCMP
ModifyProp
MoveScreen
MoveWindow
OffCadget
OffMenu
OnCadget
OnMenu
OpenScreen
OpenWindow
OpenWorkBench
PrintText
RefreshCgadgets
RemakeDisplay
RemoveCadget
ReportMouse
Request
RethinkDisplay
ScreenToBack
ScreenToFront
SetDRRequest
SetMenuStrip
SetPointer
SetWindowTitle
ShowTitle

SizeWindow
ViewAddress
ViewPortAddress
WBenchToFront
WindowLimits
WindowToBack
WindowToFront

AddGadget AddGadget

NAME AddGadget -- add a Gadget to the Gadget list of the Window or Screen

SYNOPSIS SHORT AddGadget(Pointer, Gadget, Position);

FUNCTION

Adds the specified Gadget to the Gadget list of the given Window or Screen, linked in at the position in the list specified by the Position argument (that is, if Pos = 0, the Gadget will be inserted at the head of the list, and if Position = 1 then the Gadget will be inserted after the first Gadget and before the second). If the Position you specify is greater than the number of Gadgets in the list, your Gadget will be added to the end of the list. The SCRGADGET flag of the Gadget specifies whether the Pointer argument points to a Window (SCRGADGET not set) or a Screen (SCRGADGET is set). This procedure returns the position at which your Gadget was added.

NOTE: A relatively safe way to add the Gadget to the end of the list is to specify a Position of -1. That way, only the 65536th (and multiples of it) will be inserted at the wrong position. The return value of the procedure will tell you where it was actually inserted.

NOTE: The System Window and Screen Gadgets are initially added to the front of the Gadget List. The reason for this is: if you position your own Gadgets in some way that interferes with the graphical representation of the system Gadgets, the system's ones will be "hit" first by User. If you then start adding Gadgets to the front of the list, you will disturb this plan, so beware. On the other hand, if you don't violate the design rule of never overlapping your Gadgets, there's no problem.

INPUTS

Pointer = pointer to the Window or Screen to get your Gadget
Gadget = pointer to the new Gadget
Position = integer position in the list for the new Gadget (starting from zero as the first position in the list)

RESULT

Returns the position of where the Gadget was actually added.

BUCKS None

SEE ALSO RemoveGadget()

AllocRemember AllocRemember

AllocRemember

NAME AllocRemember -- AllocMem and create a link node to make FreeMem easy

SYNOPSIS

AllocRemember (RememberKey, Size, Flags);

FUNCTION

This routine calls the EXEC AllocMem() function for you, but also links the parameters of the allocation into a master list, so that you can simply call the Intuition routine FreeRemember() at a later time to deallocate all allocated memory without being required to remember the details of the memory you've allocated.

This routine will have two primary uses:

- Let's say that you're doing a long series of allocations in a procedure (such as the Intuition OpenWindow() procedure). If any one of the allocations fails for lack of memory, you need to abort the procedure. Abandoning ship correctly involves freeing up what memory you've already allocated. This procedure allows you to free up that memory easily, without being required to keep track of how many allocations you already done, what the sizes of the allocations were, where the memory was allocated.

- Also, in the more general case, you may do all of the allocations in you entire program using this routine. Then, when your program is exiting, you can free it all up at once with a simple call to FreeRemember().

You create the "anchor" for the allocation master list by creating a variable that's a pointer to struct Remember, and initializing that pointer to NULL. This is called the RememberKey. Whenever you call AllocRemember(), the routine actually does two memory allocations, one for the memory you want and the other for a copy of a Remember structure. The Remember structure is filled in with data describing your memory allocation, and it's linked into the master list pointed to by your RememberKey. Then, to free up any memory that's been allocated, all you have to do is call FreeRemember() with your RememberKey.

Please read the FreeRemember() header too. As you will see, you can select to either free just the link nodes and keep all the allocated memory for yourself, or you can elect to free both the nodes and your memory buffers.

See the "Amiga ROM Kernel Manual" for a description of the AllocMem() call and the values you should use for the Size and Flags variables.

INPUTS

RememberKey = the address of a pointer to struct Remember. Before the very first call to AllocRemember, initialize this pointer to NULL. For instance:
struct Remember *RememberKey;
RememberKey = NULL;

AllocRemember (dRememberKey, BUFSIZE, MEME_CHIP);
FreeRemember (dRememberKey, TRUE);

Size = the size in bytes of the memory allocation. Please refer to the EXEC AllocMem() function in the "Amiga ROM Kernel Manual" for details.

Flags = the specifications for the memory allocation. Please refer to the EXEC AllocMem() function in the "Amiga ROM Kernel Manual" for details.

RESULT

If the memory allocation is successful, this routine returns the byte address of your requested memory block. Also, the node to your block will be linked into the list pointed to by your RememberKey variable. If the allocation fails, this routine returns NULL and the list pointed to by RememberKey, if any, will be undisturbed.

BUCS

None

SEE ALSO

FreeRemember ()
The EXEC AllocMem() function

AutoRequest

NAME

AutoRequest -- Automatically build and get response from a Requester

SYNOPSIS

AutoRequest(Window, BodyText, PositiveText, NegativeText,
PositiveFlags, NegativeFlags, Width, Height);

FUNCTION

This procedure automatically builds a Requester for you and then waits for a response from the user or the system to satisfy your request. If the response is Positive, this procedure returns TRUE. If the response is negative, this procedure returns FALSE.

This procedure first preserves the state of the IDCMP values of the Window argument. Then it creates an IDCMPFlag specification by merging together your PositiveFlags, NegativeFlags, and the IDCMP class GADGETUP. You may choose to specify no flags for either the PositiveFlags or NegativeFlags arguments.

The IntuiText arguments, and the Width and Height values, are passed directly to the BuildSysRequest() procedure along with your Window pointer and the IDCMP flags. Please refer to BuildSysRequest() for a description of the IntuiText that you are expected to supply when calling this routine. It's an important but long-winded description that need not be duplicated here.

If the BuildSysRequest() procedure does not return a pointer to a Window, it will return TRUE or FALSE (not valid structure pointers) instead, and these BOOL values will be returned to you immediately.

On the other hand, if a valid Window pointer is returned, that Window will have had its IDCMP Ports and flags initialized according to your specifications. AutoRequest() then waits for an IDCMP message on the UserPort, which message will satisfy one of three requirements:

- either the message is of a class that matches one of your PositiveFlags arguments (if you've supplied any), in which case this routine returns TRUE. Or
- the message class matches one of your NegativeFlags arguments (if you've supplied any), in which case this routine returns FALSE. Or
- the only other possibility is that the IDCMP message is of class GADGETUP, which means that one of the two Gadgets, as specified by the PositiveText and NegativeText arguments, was selected by the user. If the TRUE Gadget was selected, TRUE is returned. If the FALSE Gadget was selected, FALSE is returned.

When the dust has settled, this routine calls FreeSysRequest() if necessary to clean up the Requester and any other allocated memory.

INPUTS

Window = pointer to a Window structure

BodyText = pointer to an IntuiText structure
 PositiveText = pointer to an IntuiText structure
 NegativeText = pointer to an IntuiText structure
 PositiveFlags = flags for the IDCMP
 NegativeFlags = flags for the IDCMP
 Width, Height = the sizes required for the rendering of the Requester

RESULT

The return value is either TRUE or FALSE. See the text above for a complete description of the chain of events that might lead to either of these values being returned.

BUCS

None

SEE ALSO

BuildSysRequest()

BeginRefresh

BeginRefresh

NAME

BeginRefresh -- Sets up a Window for optimized refreshing

SYNOPSIS

BeginRefresh(Window);

FUNCTION

This routine sets up your Window for optimized refreshing. It sets Intuition internal states and then sets up the layer underlying your Window for a call to the layer library. There, the "clip rectangles" of the layer are reorganized in a fashion where any reordering performed in your Window (until you call to EndRefresh()) will occur only in the regions which need to be refreshed. The phrase "clip rectangles" refers to the division of your Window into visible and concealed rectangles.

For more information about clipping rectangles and the layer library, refer to the "Amiga ROM Kernel Manual".

For instance, if you have a SIMPLE_REFRESH Window which is partially concealed and the user brings it to the front, you will receive a message asking you to refresh your display. If you call BeginRefresh() before doing any of the rendering, then the layer that underlies your Window will be arranged such that the only rendering that will actually take place will be that which goes to the newly-revealed areas. This is very performance-efficient.

After you have performed your refresh of the display, you should call EndRefresh() to reset the state of the layer and the Window. Then you may proceed with rendering to the Window as usual.

You learn that your Window needs refreshing by receiving either a message of class REFRESHWINDOW through the IDCMP, or an input event of class IECLASS_REFRESHWINDOW through the Console Device. Whenever you are told that your Window needs refreshing, you should call BeginRefresh() and EndRefresh() to clear the refresh-needed state, even if you don't plan on doing any rendering.

INPUTS

Window = pointer to the Window structure which needs refreshing

RESULT

None

BUCS

None

SEE ALSO

EndRefresh()

The "Windows" chapter in this book

BuildSysaRequest

BuildSysaRequest

NAME BuildSysaRequest -- Build and display a system Requester

SYNOPSIS

```
BuildSysaRequest(Window, BodyText, PositiveText, NegativeText,
  IDCMPFlags, Width, Height);
```

FUNCTION

This procedure builds a Requester based on the supplied information. If all goes well and the Requester is constructed, this procedure returns a pointer to the Window in which the Requester appears. That Window will have the IDCMP UserPort and WindowPort initialized to reflect the flags found in the IDCMPFlags argument. You may then Wait() on those ports to detect the user's response to your Requester, which response may include either selecting one of the Gadgets or causing some other event to be noticed by Intuition (like DISKINSERTED, for instance). After the Requester is satisfied, you should call the FreeSysaRequest() procedure to remove the Requester and free up any allocated memory.

If it isn't possible to construct the Requester for any reason, this procedure will instead use the text arguments to construct a text string for a call to the DisplayAlert() procedure, and then will return either a TRUE or FALSE depending on whether DisplayAlert() returned a FALSE or TRUE respectively.

If the Window argument you supply is equal to NULL, a new Window will be created for you in the Workbench Screen. If you want the Requester created by this routine to be bound to a particular Window, you should not supply a Window argument of NULL.

The text arguments are used to construct the display. They are pointers to instances of the struct IntuiText.

The BodyText argument should be used to describe the nature of the Requester. As usual with IntuiText data, you may link several lines of text together, and the text may be placed in various locations in the Requester. This IntuiText pointer will be stored in the ReqText variable of the new Requester.

The PositiveText argument describes the text that you want associated with the user choice of "Yes. TRUE. Retry. Good." If the Requester is successfully opened, this text will be rendered in a Gadget in the lower-left of the Requester, which Gadget will have the GadgetID field set to TRUE. If the Requester cannot be opened and the DisplayAlert() mechanism is used, this text will be rendered in the lower-left corner of the Alert display with additional text specifying that the left mouse button will select this choice. This pointer can be set to NULL, which specifies that there is no TRUE choice that can be made.

The NegativeText argument describes the text that you want associated with the user choice of "No. FALSE. Cancel. Bad." If the Requester

is successfully opened, this text will be rendered in a Gadget in the lower-right of the Requester, which Gadget will have the GadgetID field set to FALSE. If the Requester cannot be opened and the DisplayAlert() mechanism is used, this text will be rendered in the lower-right corner of the Alert display with additional text specifying that the right mouse button will select this choice. This pointer cannot be set to NULL. There must always be a way for the user to cancel this Requester.

The Positive and Negative Gadgets created by this routine have the following features:

- BOOLGADGET
- RELVERIFY
- REQGADGET
- TOGGLESELECT

When defining the text for your Gadgets, you may find it convenient to use the special defines used by Intuition for the construction of the Gadgets. These include defines like AUTORANMODE, AUTOLEFTEDGE, AUTOTOPEDGE and AUTOFONTPEN. You can find these in your local intuition.h (or intuition.i) file.

The Width and Height values describe the size of the Requester. All of your BodyText must fit within the Width and Height of your Requester. The Gadgets will be created to conform to your sizes.

VERY IMPORTANT NOTE: for the preliminary release of this procedure, a new Window is opened in the same Screen as the one containing your Window. However, with a forthcoming update of Intuition, this will change such that the Requester will be opened in the Window supplied as an argument to this routine, if possible. The primary implication of this will be that the IDCMPFlags and Ports will be disturbed by a call to this routine. To assure upward-compatibility, it's your responsibility to make sure that the Ports and IDCMPFlags of the Window passed to the routine are protected before the call.

INPUTS

Window = pointer to a Window structure
 BodyText = pointer to an IntuiText structure
 PositiveText = pointer to an IntuiText structure
 NegativeText = pointer to an IntuiText structure
 IDCMPFlags = the IDCMP flags you want used for the initialization of the Window
 Width, Height = the size required to render your Requester

RESULT

If the Requester was successfully rendered in a Window, the value returned by this procedure is a pointer to the Window in which the Requester was rendered. If, however, the Requester cannot be rendered in the Window, this routine will have called DisplayAlert() before returning and will pass back TRUE if the user pressed the left mouse button and FALSE if the user pressed the right mouse button.

BUGS

This procedure currently opens a Window as wide as the Screen in

which it was rendered, and then opens the Requester within that Window. Also, if DisplayAlert() is called, the PositiveText and NegativeText are not rendered in the lower corners of the Alert.

SEE ALSO

FreeSysRequest()
DisplayAlert()
ModifyIDCMP()
The Executive's Wait() instruction
AutoRequest()

ClearDMRequest

ClearDMRequest

NAME

ClearDMRequest -- clears the DMRequest of the Window

SYNOPSIS

ClearDMRequest(Window);

FUNCTION

Attempts to clear the DMRequester from the specified window. The DMRequester is the special Requester that you attach to the double-click of the menu button which the user can then bring up on demand. This routine WILL, NOT clear the DMRequester if it's active (in use by the user). After having called SetDMRequest(), if you want to change the DMRequester, the correct way to start is by calling ClearDMRequest() until it returns a value of TRUE; then you can call SetDMRequest() with the new DMRequester.

INPUTS

Window = pointer to the window from which the DMRequest is to be cleared

RESULT

If the DMRequest was not currently in use, zeroes out the DMRequest pointer in the Window and returns TRUE.

If the DMRequest was currently in use, doesn't change the pointer and returns FALSE

BUGS

None

SEE ALSO

SetDMRequest()
Request()

ClearMenuStrip

ClearMenuStrip

NAME ClearMenuStrip -- Clears the Menu strip from the Window

SYNOPSIS ClearMenuStrip(Window);

FUNCTION Clears the menu strip from the Window.

INPUTS Window = pointer to a Window structure

RESULT None

BUGS None

SEE ALSO SetMenuStrip()

ClearPointer

ClearPointer

NAME ClearPointer -- clears the Pointer definition from a Window

SYNOPSIS ClearPointer(Window);

FUNCTION Clears the Window of its own definition of the Intuition pointer. After calling ClearPointer(), every time this Window is the active one the default Intuition pointer will be the pointer displayed to the user. If your Window is the active one when this routine is called, the change will take place immediately.

INPUTS Window = pointer to the Window to be cleared of its Pointer definition

RESULT None

BUGS None

SEE ALSO SetPointer()

CloseScreen CloseScreen

NAME CloseScreen -- Closes an Intuition Screen

SYNOPSIS CloseScreen(Screen);

FUNCTION

Unlinks the Screen, unlinks the ViewPort, deallocates everything. Doesn't care whether or not there are still any Windows attached to the Screen. Doesn't try to close any attached Windows; in fact, ignores them altogether. If this is the last Screen to go, attempts to reopen Workbench.

INPUTS Screen = pointer to the Screen to be deallocated

RESULT

None

BUGS

Don't think so

SEE ALSO

OpenScreen

CloseWindow CloseWindow

NAME CloseWindow -- Closes an Intuition Window

SYNOPSIS CloseWindow(Window);

FUNCTION

Closes an Intuition Window. Unlinks it from the system, unallocates its memory, and if its Screen is a system one that would be empty without the Window, closes the system Screen too

A Grim, Foreboding Note: if you are ever rude enough to CloseWindow() on a Window that has an IDCMP without first having Reply()'d to all of my messages to the IDCMP port, Intuition in turn will so rudely as to reclaim and deallocate its messages without waiting for your permission.

Another grim note: if you have added a Menu strip to this Window (via a call to SetMenuStrip()) you must be sure to remove that Menu strip (via a call to ClearMenuStrip()) before closing your Window. CloseWindow() doesn't check whether or not the menus of your Window are currently being used when the Window is closed. If this in fact happens to be the case, then as soon as the user releases the Menu button the system will crash with pyrotechnics that are usually quite lovely.

INPUTS

Window = a pointer to a Window structure

RESULT

None

BUGS

Don't think so. What do you think?

SEE ALSO

OpenWindow(), CloseScreen()

CloseWorkBench CloseWorkBench

NAME CloseWorkBench -- Closes the WorkBench Screen

SYNOPSIS
BOOL CloseWorkBench();

FUNCTION
This routine attempts to close the WorkBench. The actions taken are:
- Test whether or not any applications have opened Windows on the WorkBench, and return FALSE if so. Otherwise ...
- Clean up all special buffers
- Close the WorkBench Screen
- Make the WorkBench program mostly inactive (it will still monitor disk activity)
- Return TRUE

INPUTS
None

RESULT
TRUE if the WorkBench Screen closed successfully
FALSE if anything went wrong and the WorkBench Screen is still out there

BUGS
None

SEE ALSO
None

CurrentTime CurrentTime

NAME CurrentTime -- Get the current time values

SYNOPSIS
ULONG Seconds, Micros;
CurrentTime(&Seconds, &Micros);

FUNCTION
Puts copies of the current time into the supplied argument pointers.
This time value is not extremely accurate, nor is it of a very fine resolution. This time will be updated no more than sixty times a second, and will typically be updated far fewer times a second.

INPUTS
Seconds = pointer to a LONG variable to receive the current seconds value
Micros = pointer to a LONG variable for the current microseconds value

RESULT
Puts the time values into the memory locations specified by the arguments

BUGS
None

SEE ALSO
None

DisplayAlert

NAME DisplayAlert -- Create a display of an Alert message

SYNOPSIS

DisplayAlert(AlertNumber, String, Height);

FUNCTION

Creates an Alert display with the specified message.

If the system can recover from this Alert, its a RECOVERY_ALERT and this routine waits until the user presses one of the mouse buttons, after which the display is restored to its original state and a BOOL value is returned by this routine to specify whether or not the User pressed the LEFT mouse button.

If the system cannot recover from this Alert, it's a DEADEND_ALERT and this routine returns immediately upon creating the Alert display. The return value is FALSE.

The AlertNumber is a LONG value, related to the value sent to the Alert() routine. But the only bits that are pertinent to this routine are the ALERT_TYPE bits. These bits must be set to either RECOVERY_ALERT for Alerts from which the system may safely recover, or DEADEND_ALERT for those fatal Alerts. These states are described in the paragraph above. There is a third type of Alert, the DAISY_ALERT, which is used only by the Executive.

The String argument points to an AlertMessage string. The AlertMessage string is comprised of one or more substrings, each of which is comprised of the following components:

- first, a 16-bit x-coordinate and an 8-bit y-coordinate, describing where on the Alert display you want this string to appear. The y-coordinate describes the offset to the baseline of the text.
- then, the bytes of the string itself, which must be null-terminated (end with a byte of zero)
- lastly, the continuation byte, which specifies whether or not there's another substring following this one. If the continuation byte is non-zero, there IS another substring to be processed in this Alert Message. If the continuation byte is zero, this is the last substring in the message.

The last argument, Height, describes how many video lines tall you want the Alert display to be.

INPUTS

AlertNumber = the number of this Alert Message. The only pertinent bits of this number are the ALERT_TYPE bits. The rest of the number is ignored by this routine
String = pointer to the Alert message string, as described above
Height = minimum display lines required for your message

RESULT

A BOOL value of TRUE or FALSE. If this is a DEADEND_ALERT, FALSE is always the return value. If this is a RECOVERY_ALERT, the return value will be TRUE if the User presses the left mouse button in response to your message, and FALSE if the User presses the right hand button in response to your text.

BUGS

If the system is worse off than you think, the level of your Alert may become DEADEND_ALERT without you ever knowing about it.

SEE ALSO

None

DisplayBeep DisplayBeep

NAME DisplayBeep -- "beeps" the video display

SYNOPSIS DisplayBeep (Screen);

FUNCTION "Beeps" the video display by flashing the background color of the specified Screen. If the Screen argument is NULL, every Screen in the display will be beeped. Flashing everyone's Screen is not a polite thing to do, so this should be reserved for dire circumstances.

The reason such a routine is supported is because the Amiga has no internal bell or speaker. When the user needs to know of an event that is not serious enough to require the use of a Requester, the DisplayBeep() function should be called.

INPUTS Screen = pointer to a Screen. If NULL, every Screen in the display will be flashed

RESULT None

BUGS None

SEE ALSO None

DoubleClick DoubleClick

DoubleClick

NAME DoubleClick -- Test two time values for double-click timing

SYNOPSIS DoubleClick(StartSeconds, StartMicros, CurrentSeconds, CurrentMicros);

FUNCTION Compares the difference in the time values with the double-click timeout range that the user (using the "Preferences" tool) or some other source has configured into the system. If the difference between the specified time values is within the current double-click time range, this function returns TRUE, else it returns FALSE.

These time values can be found in InputEvents and IDCMP Messages. The time values are not perfect; however, they are precise enough for nearly all applications.

INPUTS StartSeconds, StartMicros = the timestamp value describing the start of the double-click time period you are considering
CurrentSeconds, CurrentMicros = the timestamp value describing the end of the double-click time period you are considering

RESULT If the difference between the supplied timestamp values is within the double-click time range in the current set of Preferences, this function returns TRUE, else it returns FALSE

BUGS None
SEE ALSO CurrentTime();

DrawBorder DrawBorder

NAME DrawBorder -- draws the specified border into the RastPort

SYNOPSIS DrawBorder(RastPort, Border, LeftOffset, TopOffset);

FUNCTION

First, sets up the DrawMode and Pens in the RastPort according to the arguments of the Border structure. Then, draws the vectors of the Border argument into the RastPort, offset by the Left and Top Offsets. This routine does Intuition window clipping as appropriate -- if you draw a line outside of your Window, your imagery will be clipped at the Window's edge.

If the NextBorder field of the Border argument is non-zero, the next Border is rendered as well (return to the top of this FUNCTION section for details).

INPUTS

RastPort = pointer to the RastPort to receive the border crossing
Border = pointer to a Border structure

LeftOffset = the offset which will be added to each vector's x coordinate

TopOffset = the offset which will be added to each vector's y coordinate

RESULT

None

BUGS

None

SEE ALSO

None

DrawImage DrawImage

NAME DrawImage -- draws the specified Image into the RastPort

SYNOPSIS DrawImage(RastPort, Image, LeftOffset, TopOffset);

FUNCTION

First, sets up the DrawMode and Pens in the RastPort according to the arguments of the Image structure. Then, moves the Image data of the Image argument into the RastPort, offset by the Left and Top Offsets. This routine does Intuition window clipping as appropriate -- if you draw an image outside of your Window, your imagery will be clipped at the Window's edge.

If the NextImage field of the Image argument is non-zero, the next Image is rendered as well (return to the top of this FUNCTION section for details).

INPUTS

RastPort = pointer to the RastPort to receive the border crossing
Image = pointer to an Image structure

LeftOffset = the offset which will be added to the Image's x coordinate

TopOffset = the offset which will be added to the Image's y coordinate

RESULT

None

BUGS

None

SEE ALSO

None

EndRefresh EndRefresh

NAME EndRefresh -- Ends the optimized refresh state of the Window

SYNOPSIS EndRefresh(Window, Complete);

FUNCTION

This function gets you out of the special refresh state of your Window. It is called following a call to BeginRefresh(), which routine puts you into the special refresh state. While your Window is in the refresh state, the only rendering that will be wrought in your Window will be to those areas which were recently revealed and need to be refreshed.

After you've done all the refreshing you want to do for this Window, you should call this routine to restore the Window to its non-refreshing state. Then all rendering will go to the entire Window, as usual.

The Complete argument is a boolean TRUE or FALSE value used to describe whether or not the refreshing you've done was all the refreshing that needs to be done at this time. Most often, this argument will be TRUE. But if, for instance, you have multiple tasks or multiple procedure calls which must run to completely refresh the Window, then each can call its own Begin/EndRefresh() pair with a Complete argument of FALSE, and only the last call with a Complete argument of TRUE.

INPUTS

Window = pointer to the Window currently in optimized-refresh mode
Complete = Boolean TRUE or FALSE describing whether or not this Window is completely refreshed

RESULT

None

BUGS

None

SEE ALSO

BeginRefresh()
The "Screens" chapter in this book

EndRequest EndRequest

NAME EndRequest -- Ends the Request and resets the Window

SYNOPSIS EndRequest(Requester, Window);

FUNCTION

Ends the Request by erasing the Requester and resetting the Window. Note that this doesn't necessarily clear all Requesters from the Window, only the specified one. If the Window labors under other Requesters, they will remain in the Window.

INPUTS

Requester = pointer to the Requester to be removed
Window = pointer to the Window structure with which this Requester is associated

RESULT

None

BUGS

None

SEE ALSO

None

FreeRemember FreeRemember

NAME FreeRemember -- Free memory allocated by calls to AllocRemember ()

SYNOPSIS FreeRemember (RememberKey, ReallyForget);

FUNCTION

This function frees up memory allocated by the AllocRemember () function. It will either free up just the Remember structures, which supply the link nodes that tie your allocations together, or it will deallocate both the link nodes AND your memory buffers too.

If you want to deallocate just the Remember structure link nodes, you should set the ReallyForget argument to FALSE. However, if you want FreeRemember to really forget about all the memory, including both the Remember structure link nodes and the buffers you requested via earlier calls to AllocRemember () then you should set the ReallyForget argument to TRUE.

INPUTS

RememberKey = the address of a pointer to struct Remember. This pointer should either be NULL or set to some value (possibly NULL) by a call to AllocRemember (). For example:

```
struct Remember *RememberKey;
```

```
RememberKey = NULL;
```

```
AllocRemember (&RememberKey, BUFSIZE, MEME_CHIP);
```

```
FreeRemember (&RememberKey, TRUE);
```

ReallyForget = a BOOL FALSE or TRUE describing, respectively,

whether you want to free up only the Remember nodes or

if you want this procedure to really forget about all of the memory, including both the nodes and the memory buffers pointed to by the nodes.

RESULT

None

BUGS

None

SEE ALSO

AllocRemember ()

FreeSysRequest

FreeSysRequest

NAME FreeSysRequest -- Frees up memory used by a call to BuildSysRequest ()

SYNOPSIS FreeSysRequest (Window);

FUNCTION

This routine frees up all memory allocated by a successful call to the BuildSysRequest () procedure. If BuildSysRequest () returned a pointer to a Window, then you are able to Wait () on the message port of that Window to detect an event which satisfies the Requester. If when you want to remove the Requester, you call this procedure. It ends the Requester and deallocates any memory used in the creation of the Requester.

NOTE: If BuildSysRequest () did not return a pointer to a Window, you should not call FreeSysRequest () !

INPUTS

Window = a copy of the Window pointer returned by a successful call to the BuildSysRequest () procedure

RESULT

None

BUGS

None

SEE ALSO

BuildSysRequest ()
The Executive's Wait () instruction
AutoRequest ()

GetDefPrefs GetDefPrefs

NAME
GetDefPrefs -- Get a copy of the the Intuition default Preferences

SYNOPSIS
GetDefPrefs(PrefBuffer, Size);

FUNCTION

Gets a copy of the Intuition default preferences data. Writes the data into the buffer you specify. The number of bytes you want copied is specified by the Size argument.

The default Preferences are those that Intuition uses when it is first opened. If no preferences file is found, these are the preferences that are used. These would also be the startup Preferences in an AmigaDOS-less environment.

It is legal to take a partial copy of the Preferences structure. The more pertinent Preferences variables have been grouped near the top of the structure to facilitate the memory conservation that can be had by taking a copy of only some of the Preferences structure.

INPUTS

PrefBuffer = pointer to the memory buffer to receive your copy of the Intuition Preferences
Size = the number of bytes in your PrefBuffer, the number of bytes you want copied from the system's internal Preference settings

RESULT

Returns your Preferences pointer

BUGS

None

SEE ALSO

GetPrefs()

GetPrefs

GetPrefs

NAME
GetPrefs -- Get the current setting of the Intuition Preferences

SYNOPSIS
GetPrefs(PrefBuffer, Size);

FUNCTION

Gets a copy of the current Intuition Preferences data. Writes the data into the buffer you specify. The number of bytes you want copied is specified by the Size argument.

It is legal to take a partial copy of the Preferences structure. The more pertinent Preferences variables have been grouped near the top of the structure to facilitate the memory conservation that can be had by taking a copy of only some of the Preferences structure.

INPUTS

PrefBuffer = pointer to the memory buffer to receive your copy of the Intuition Preferences
Size = the number of bytes in your PrefBuffer, the number of bytes you want copied from the system's internal Preference settings

RESULT

Returns a copy of your Preferences pointer

BUGS

None

SEE ALSO

GetDefPrefs()

InitRequester InitRequester

NAME InitRequester -- initializes a Requester structure

SYNOPSIS InitRequester (Requester);

FUNCTION

Initializes a requester for general use. After calling InitRequester, you need fill in only those Requester values that fit your needs. The other values are set to states that Intuition regards as NULL

INPUTS

Requester = a pointer to a Requester

RESULT

None

BUGS

None

SEE ALSO

None

IntuiTextLength

IntuiTextLength

NAME IntuiTextLength -- Return the length (pixel-width) of an IntuiText

SYNOPSIS IntuiTextLength(IText);

FUNCTION

This routine accepts a pointer to an instance of an IntuiText structure, and returns the length (the pixel-width) of the string that that instance of the structure represents.

All of the usual IntuiText rules apply. Most notably, if the Font pointer of the structure is set to NULL, you'll get the pixel-width of your text in terms of the current default font.

INPUTS

IText = pointer to an instance of an IntuiText structure

RESULT

Returns the pixel-width of the text specified by the IntuiText data

BUGS

None

SEE ALSO

None

ItemAddress ItemAddress

NAME ItemAddress -- Returns the address of the specified MenuItem

SYNOPSIS ItemAddress(MenuItem, MenuItemNumber):

FUNCTION This routine feels through the specified MenuItem and returns the address of the Item specified by the MenuItemNumber. Typically, you will use this routine to get the address of a MenuItem from a MenuItemNumber sent to you by Intuition after User has played with your Menus. This routine requires that the arguments are well-defined. MenuItemNumber may be equal to MENU_NULL, in which case this routine returns MENU_NULL. If MenuItemNumber doesn't equal MENU_NULL, it's presumed to be a valid MenuItem number selector for your MenuItem, which includes:
- a valid MenuItem number
- a valid Item Number
- if the Item specified by the above two components has a SubItem, the MenuItemNumber may have a SubItem component too

Note that there must be BOTH a MenuItem number and an Item number. Because a SubItem specifier is optional, the address returned by this routine may point to either an Item or a SubItem.

INPUTS MenuItem = a pointer to the first MenuItem in your MenuItemStrip
MenuItemNumber = the value which contains the packed data that selects the MenuItem and Item (and SubItem)

RESULT If MenuItemNumber == MENU_NULL, this routine returns MENU_NULL
else this routine returns the address of the MenuItem specified by MenuItemNumber

BUGS None

SEE ALSO The "Menus" chapter in this book for more information about MenuItemNumbers

MakeScreen MakeScreen

NAME MakeScreen -- Do an Intuition-integrated MakeVPort() of a custom screen

SYNOPSIS MakeScreen(Screen):

FUNCTION This procedure allows you to do a MakeVPort() for the ViewPort of your Custom Screen in an Intuition-integrated way. This allows you to do your own Screen manipulations without worrying about interference with Intuition's usage of the same ViewPort. After calling this routine, you can call RethinkDisplay() to incorporate the new ViewPort of your custom screen into the Intuition display.

INPUTS Screen = address of the Custom Screen structure

RESULT None

BUGS None

SEE ALSO RethinkDisplay()
RemakeDisplay()
The graphics library's MakeVPort()

ModifyIDCMP

ModifyIDCMP

NAME

ModifyIDCMP -- Modify the state of the Window's IDCMP

SYNOPSIS

ModifyIDCMP (Window, IDCMPFlags);

FUNCTION

This routine modifies the state of your Window's IDCMP (Intuition Direct Communication Message Port). The state is modified to reflect your desires as described by the flag bits in the value IDCMPFlags. When you call ModifyIDCMP(), if the IDCMPFlags equals NULL, you are asking that if the Port is currently opened, you want it closed. If you set any of the IDCMPFlags, this means that you want the message ports to be open; if not currently opened, the Ports will be opened now.

The four actions that might be taken are:

- if there is currently no IDCMP in the given Window, and IDCMPFlags is NULL, nothing happens
- if there is currently no IDCMP in the given Window, and any of the IDCMPFlags is selected (set), then the IDCMP of the Window is created, including allocating and initializing the message ports and allocating a Signal bit for your Port. See the "Input and Output Methods" chapter of this book for full details
- if the IDCMP for the given Window is opened, and the IDCMPFlags argument is NULL, this says that you want Intuition to close the Ports, free the buffers and free your Signal bit. You MUST be the same Task that was active when this Signal bit was allocated
- if the IDCMP for the given Window is opened, and the IDCMPFlags argument is not NULL, this means that you want to change the state of which events will be broadcast to you through the IDCMP

NOTE: You can set up the Window->UserPort to any Port of your own before you call ModifyIDCMP(). If IDCMPFlags is non-null but your UserPort is already initialized, Intuition will assume that it's a valid Port with Task and Signal data preset and Intuition won't disturb your set-up at all, Intuition will just allocate the Intuition Message Port half of it. The converse is true as well: if UserPort is NULL when you call here with IDCMPFlags == NULL, I'll deallocate only the Intuition Port. This allows you to use a Port that you already have allocated:

- OpenWindow() with IDCMPFlags equal to NULL (open no ports)
- set the UserPort variable of your Window to any valid Port of your own choosing
- call ModifyIDCMP with IDCMPFlags set to what you want
- then, to clean up later, set UserPort equal to NULL before calling CloseWindow() (leave IDCMPFlags alone)

A Grim, Foreboding Note: if you are ever rude enough to close an IDCMP without first having Reply() 'd to all of the messages sent to the IDCMP port, Intuition in turn will so rude as to reclaim and deallocate its messages without waiting for your permission.

INPUTS

Window = pointer to the Window structure containing the IDCMP Ports
IDCMPFlags = the flag bits describing the new desired state of the IDCMP

RESULT

None

BUGS

None

SEE ALSO

OpenWindow

ModifyProp ModifyPropop

NAME ModifyPropop -- Modify the current parameters of a Proportional Cadget

SYNOPSIS ModifyPropop(Cadget, Pointer, Requester, Flags, HorizPot, VertPot, HorizBody, VertBody);

FUNCTION Modifies the parameters of the specified Proportional Cadget. The Cadget's internal state is then recalculated and the Imagery is redisplayed, wherever it is that the Pointer argument points. The Pointer argument can point to either a Window or a Screen structure, which it actually points to is decided by examining the SCRCADGET flag of the Cadget; if the flag is set, Pointer points to a Screen structure, otherwise it points to a Window structure.

The Requester variable can point to a Requester structure. If the Cadget has the REQCADGET flag set, the Cadget is in a Requester and the Pointer must necessarily point to a Window. If this is not the Cadget of a Requester, the Requester argument may be NULL.

INPUTS PropCadget = pointer to a Proportional Cadget
Pointer = pointer to the "owning" display element of the Cadget, be it a Window or a Screen
Requester = pointer to a Requester (may be NULL if this isn't a Requester Cadget)
Flags = value to be stored in the Flags variable of the PropInfo
HorizPot = value to be stored in the HorizPot variable of the PropInfo
VertPot = value to be stored in the VertPot variable of the PropInfo
VertBody = value to be stored in the HorizBody variable of the PropInfo

RESULT None
BUCS None
SEE ALSO None

MoveScreen MoveScreen

NAME MoveScreen -- attempts to move the Screen by the delta amounts

SYNOPSIS MoveScreen(Screen, DeltaX, DeltaY);

FUNCTION Attempts to move the specified Screen. This movement must follow certain constraints (only for the current release of the software):
- the bottom of the Screen must not move higher than the bottom of the video display
- horizontal movements are ignored
If the DeltaX and DeltaY variables you specify would move the Screen in a way that violates the above restrictions, the Screen will be moved as far as possible

INPUTS Screen = pointer to a Screen structure
DeltaX = amount to move the screen on the x-axis
DeltaY = amount to move the screen on the y-axis

RESULT None
BUCS None
SEE ALSO None

MoveWindow MoveWindow

NAME MoveWindow -- Ask Intuition to move a Window

SYNOPSIS MoveWindow(Window, DeltaX, DeltaY);

FUNCTION This routine sends a request to Intuition asking to move the Window the specified distance. The delta arguments describe how far to move the Window along the respective axes.

Note that the Window will not be moved immediately, but rather will be moved the next time Intuition receives an input event, which happens currently at a minimum rate of ten times per second, and a maximum of sixty times a second.

This routine does no error-checking. If your delta values specify some far corner of the Universe, Intuition will attempt to move your Window to the far corners of the Universe. Because of the distortions in the space-time continuum that can result from this, as predicted by special relativity, the result is generally not a pretty sight.

INPUTS

Window = pointer to the structure of the Window to be moved
DeltaX = signed value describing how far to move the Window on the x-axis
DeltaY = signed value describing how far to move the Window on the y-axis

RESULT

None

BUCS

None

SEE ALSO

SizeWindow(), WindowToFront(), WindowToBack()

OffGadget OffGadget

NAME OffGadget -- disables the specified Gadget

SYNOPSIS OffGadget(Gadget, Pointer, Requester);

FUNCTION This command disables the specified Gadget. When a Gadget is disabled, these things happen:
- its Imagery is displayed ghosted
- the GADGDISABLED flag is set
- the Gadget cannot be selected by User

The Pointer argument can point to either a Window or a Screen structure. Which it actually points to is decided by examining the SCRGADGET flag of the Gadget; if the flag is set, Pointer points to a Screen structure, else it points to a Window structure. The Requester variable can point to a Requester structure. If the Gadget has the REQGADGET flag set, the Gadget is in a Requester and the Pointer must necessarily point to a Window. If this is not the Gadget of a Requester, the Requester argument may be NULL.

NOTE: It's never safe to tinker with the Gadget list yourself. Don't supply some Gadget that Intuition hasn't already processed in the usual way.

NOTE: If you have specified that this is the Gadget list of a Requester, that Requester must be currently displayed

INPUTS

Gadget = pointer to the Gadget that you want disabled
Pointer = pointer to either a Screen or Window structure (defined by the SCRGADGET flag of the Gadget)
Requester = pointer to a Requester (may be NULL if this isn't a Requester Gadget list)

RESULT

None

BUCS

None

SEE ALSO

None

OffMenu

NAME OffMenu -- disables the given menu or menu item

SYNOPSIS OffMenu(Window, MenuNumber);

FUNCTION

This command disables a sub-item, an item, or a whole menu. If the base of the menu number matches the menu currently revealed, the menustrip is redisplayed.

INPUTS

Window = pointer to the window
MenuNumber = the menu piece to be enabled

RESULT

None

BUGS

None

SEE ALSO

None

OnGadget

NAME OnGadget -- enables the specified Gadget

SYNOPSIS OnGadget(Gadget, Pointer, Requester);

FUNCTION

This command enables the specified Gadget. When a Gadget is enabled, these things happen:
- its Imagery is displayed normally (not ghosted)
- the CACHED flag is cleared
- the Gadget can thereafter be selected by the user

The Pointer argument can point to either a Window or a Screen structure: which it actually points to is decided by examining the SCRGADGET flag of the Gadget: if the flag is set, Pointer points to a Screen struct, else it points to a Window.
The Requester variable can point to a Requester structure. If the Gadget has the REQGADGET flag set, the Gadget is in a Requester and the Pointer must necessarily point to a Window. If this is not the Gadget of a Requester, the Requester argument may be NULL.

NOTE: It's never safe to tinker with the Gadget list yourself. Don't supply some Gadget that Intuition hasn't already processed in the usual way.

NOTE: If you have specified that this is the Gadget list of a Requester, that Requester must be currently displayed.

INPUTS

Gadget = pointer to the Gadget that you want enabled
Pointer = pointer to either a Screen or Window structure (defined by the SCRGADGET flag of the Gadget)
Requester = pointer to a Requester (may be NULL if this isn't a Requester Gadget list)

RESULT

None

BUGS

None

SEE ALSO

None

OnGadget

OnMenu OnMenu

NAME OnMenu -- enables the given menu or menu item

SYNOPSIS OnMenu(Window, MenuNumber);

FUNCTION This command enables a sub-item, an item, or a whole menu. If the base of the menu number matches the menu currently revealed, the manustrip is redisplayed.

INPUTS Window = pointer to the window
MenuNumber = the menu piece to be enabled

RESULT None

BUGS None

SEE ALSO None

OpenScreen

NAME OpenScreen -- open an Intuition Screen

SYNOPSIS OpenScreen(NewScreen);

where NewScreen is a structure is initialized with:
Left, Top, Width, Height, Depth, DetailPen, BlockPen,
ViewModes, Type, Font, DefaultTitle, Gadgets

FUNCTION

Opens an Intuition Screen according to the specified parameters. Does all the allocations, sets up the Screen structure and all substructures completely, and links this Screen's ViewPort into Intuition's View of the world.

Before you call OpenScreen(), you must initialize an instance of a NewScreen structure. NewScreen is a structure that contains all of the arguments needed to open a Screen. The NewScreen structure may be discarded immediately after it is used to open the Screen.

The TextAttr pointer that you supply as an argument will be used as the default font for all Intuition-managed text that appears in the Screen and its Windows. This includes, but is not limited to, the text on the title bars of both Screen and Windows.

The SHOWTITLE flag is set to TRUE by default when a Screen is opened. To change this, you must call the routine ShowTitle().

INPUTS

NewScreen = pointer to an instance of a NewScreen structure.
That structure is initialized with the following information:

Left = initial x-position of your Screen (should be zero for now)
Top = initial y-position of the opening Screen
Width = the width for this Screen's RastPort
Height = the height for his Screen's RastPort
Depth = number of BitPlanes
DetailPen = pen number for details (like gadgets or text in title bar)
BlockPen = pen number for block fills (like title bar)
Type = Screen type (if you are not Intuition, this should be equal to CUSTOMSCREEN). Types currently supported include:
CUSTOMSCREEN -- this is your own Screen

You may also set the Type flags CUSTOMBITMAP and then supply your own BitMap for Intuition to use rather than allocating the display memory for you.
ViewModes = the appropriate argument for the data type ViewPort.Modes. these might include:
HIRES for this screen to be HIRES width
INTERLACE for the display to switch to interlace
SPRITES for this Screen to use sprites
DUALPF for dual-playfield mode (not supported yet)

Font = pointer to the default TextAttr structure for this Screen and

all Windows that open in this Screen
 DefaultTitle = pointer to a line of text that will be displayed along the Screen's Title Bar. Null terminated, or just a NULL pointer to get no text
 Gadgets = first in a linked list of the Gadgets you want for this Screen
 CustomBitMap = if you're not supplying a custom BitMap, this value is ignored. However, if you have your own display memory that you want used for this Screen, the CustomBitMap argument should point to the BitMap that describes your display memory. See the "Screens" chapter and the "Amiga ROM Kernel Manual" for more information about BitMaps.

RESULT

If all is well, returns the pointer to your new Screen
 If anything goes wrong, returns NULL.

BUGS

No way

SEE ALSO

OpenWindow

OpenWindow

OpenWindow

NAME

OpenWindow -- Opens an Intuition Window

SYNOPSIS

OpenWindow(NewWindow);
 where the NewWindow structure is initialized with:
 Left, Top, Width, Height, DetailPen, BlockPen, Flags,
 IDCMPFlags, Gadgets, CheckMark, Text, Type, Screen, BitMap,
 MinWidth, MinHeight, MaxWidth, MaxHeight

FUNCTION

Opens an Intuition window of the given height, width and depth, including the specified system Gadgets as well as any of your own. Allocates everything you need to get going.

Before you call OpenWindow(), you must initialize an instance of a NewWindow structure. NewWindow is a structure that contains all of the arguments needed to open a Window. The NewWindow structure may be discarded immediately after it is used to open the Window.

If Type == CUSTOMSCREEN, you must have opened your own Screen already via a call to OpenScreen(). Then Intuition uses your screen argument for the pertinent information needed to get your Window going. On the other hand, if type == one of the Intuition's standard Screens, your screen argument is ignored. Instead, Intuition will check to see whether or not that Screen already exists: if it doesn't, it will be opened first before Intuition opens your window in the Standard Screen. If the flag SUPER_BITMAP is set, the bitmap variable must point to your own BitMap.

The DetailPen and the BlockPen are used for system rendering; for instance, the Title bar is first filled using the BlockPen, and then the Gadgets and text are rendered using DetailPen. You can either choose to supply special pens for your Window, or, by setting either of these arguments to -1, the Screen's Pens will be used instead.

INPUTS

NewWindow = pointer to an instance of a NewWindow structure. That structure is initialized with the following data:

- Left = the initial x-position for your window
 Top = the initial y-position for your window
 Width = the initial width of this window
 Height = the initial height of this window
 DetailPen = pen number (or -1) for the rendering of Window details (like gadgets or text in title bar)
 BlockPen = pen number (or -1) for Window block fills (like Title Bar)
 Flags = specifiers for your requirements of this window. Including:
 - which system Gadgets you want attached to your window;
 - WINDOWDRAG allows this Window to be dragged
 - WINDOWDEPTH lets the user depth-arrange this Window
 - WINDOWCLOSE attaches the standard Close Gadget

- **WINDOSIZING** allows this Window to be sized. If you ask the **WINDOSIZING** Gadget, you must specify one or both of the flags **SIZERIGHT** and **SIZEBOTTOM** below; if you don't, the default is **SIZERIGHT**. See the following items **SIZERIGHT** and **SIZEBOTTOM** for extra information.
- **SIZERIGHT** is a special system Gadget flag that you set to specify whether or not you want the **RIGHT** Border adjusted to account for the physical size of the Sizing Gadget. The Sizing Gadget must, after all, take up room in either the right or bottom border (or both, if you like) of the Window. Setting either this or the **SIZEBOTTOM** flag selects which edge will take up the slack. This will be particularly useful to applications that want to use the extra space for other Gadgets (like a Proportional Gadget and two Booleans done up to look like scroll bars) or, for instance, applications that want every possible horizontal bit and are willing to lose lines vertically.
- NOTE:** If you select **WINDOSIZING**, you must select either **SIZERIGHT** or **SIZEBOTTOM** or both. If you select neither, the default is **SIZERIGHT**.
- **SIZEBOTTOM** is a special system Gadget flag that you set to specify whether or not you want the **BOTTOM** Border adjusted to account for the physical size of the Sizing Gadget. For details, refer to **SIZERIGHT** above.
- NOTE:** If you select **WINDOSIZING**, you must select either **SIZERIGHT** or **SIZEBOTTOM** or both. If you select neither, the default is **SIZERIGHT**.
- **GIMMEZERO** for easy but expensive output
- what type of window raster you want, either:
 - **SIMPLE_REFRESH**
 - **SMART_REFRESH**
 - **SUPER_BITMAP**
- **BACKDROP** for whether or not you want this window to be one of Intuition's special backdrop windows. See **BORDERLESS** as well.
- **REPORTMOUSE** for whether or not you want to "listen" to mouse movement events whenever your Window is the active one. After you've opened your Window, if you want to change you can later change the status of this via a call to **ReportMouse()**. Whether or not your Window is listening to Mouse is affected by Gadgets too, since they can cause you to start getting reports too if you like.
- The mouse move reports (either **InputEvents** or messages on the **IDCMP**) that you get will have the x/y coordinates of the current mouse position, relative to the upper-left corner of your Window (**GIMMEZERO** notwithstanding).
- This flag can work in conjunction with the **IDCMP** Flag called **MOUSEMOVE**, which allows you to listen via the **IDCMP**.
- **BORDERLESS** should be set if you want a Window with no Border padding. Your Window may have the Border variables set anyway, depending on what Gadgetry you've requested for the Window, but you won't get the standard border lines and

- spacing that comes with typical Windows. This is a good way to take over the entire Screen, since you can have a Window cover the entire width of the Screen using this flag. This will work particularly well in conjunction with the **BACKDROP** flag (see above), since it allows you to open a Window that fills the **ENTIRE** Screen.
- NOTE:** This is not a flag that you want to set casually, since it may cause visual confusion on the Screen. The Window borders are the only dependable visual division between various Windows and the background Screen. Taking away that Border takes away that visual cue, so make sure that your design doesn't need it at all before you proceed.
- **ACTIVATE** is the flag you set if you want this Window to automatically become the active Window. The active Window is the one that receives input from the keyboard and mouse. It's usually a good idea to have the Window you open when your application first starts up be an **ACTIVATED** one, but all others opened later not be **ACTIVATED** (if the user is off doing something with another Screen, for instance, your new Window will change where the input is going, which would have the effect of yanking the input rug from under the user). Please use this flag thoughtfully and carefully.
 - **MBIRAP**, when set, causes the right mouse button events to be trapped and broadcast as events. You can receive these events through either the **IDCMP** or the Console.
- IDCMPFlags** = **IDCMP** is the acronym for Intuition Direct Communications Message Port. It's Intuition's sole acronym, given in honor of all hack-heads who love to mangle our brains with maniacal names, and fashioned especially cryptic and unpronounceable to make them squirm with sardonic delight. Here's to you, my chums. Meanwhile, I still opt (and argue) for simplicity and elegance.
- If any of the **IDCMP** Flags is selected, Intuition will create a pair of messageports and use them for direct communications with the Task opening this Window (as compared with broadcasting information via the Console Device). See the "Input and Output Methods" chapter of this book for complete details.
- You request an **IDCMP** by setting any of these flags. Except for the special **VERIFY** flags, every other flag you set tells me that if a given event occurs which your program wants to know about, I'm to broadcast the details of that event through the **IDCMP** rather than via the Console device. This allows a program to interface with Intuition directly, rather than going through the Console device.
- Remember, if you are going to open both an **IDCMP** and a Console, it will be far better to get most of the event messages via the Console. Reserve your usage of the **IDCMP** for special performance cases; that is, when you aren't going to open a Console for your Window and you do want to learn about a certain set of events (for instance, **CLOSEWINDOW**); another example would be **SIZEVERIFY**, which is a function that you get ONLY through the use of the **IDCMP** (because the Console doesn't give you any way to talk to Intuition directly).
- On the other hand, if the **IDCMPFlags** argument is equal to

zero, no IDCMP is created and the only way you can learn about any Window event for this Window is via a Console opened for this Window. And you have no way to SIZEVERIFY.

If you want to change the state of the IDCMP some time after you've opened the Window (including opening or closing the IDCMP) you call the routine ModifyIDCMP().

The flags you can set are:

- **RECOVERIFY** is the flag which, like **SIZEVERIFY** and (see **MENUVERIFY** (see immediately below), specifies that you want to make sure that your graphical state is quiescent before something extraordinary happens. In this case, the extraordinary event is that a rectangle of graphical data is about to be blasted into your Window. If you're drawing into that Window, you probably will wish to make sure that you've ceased drawing before the user is allowed to bring up the DMRequest you've set up, and the same for when system has a request for the user. Set this flag to ask for that verification step.
- **REQCLEAR** is the flag you set to hear about it when the last Requester is cleared from your Window and it's safe for you to start output again (presuming you're using **REQVERIFY**)
- **REQSET** is a flag that you set to receive a broadcast when the first Requester is opened in your Window. Compare this with **REQCLEAR** above. This function is distinct from **REQVERIFY**. This function merely tells you that a Requester has opened, whereas **REQVERIFY** requires you to respond before the Requester is opened.

MENUVERIFY is the flag you set to have Intuition stop and wait for you to finish all graphical output to your Window before rendering the menus. Menus are currently rendered in the most memory-efficient way, which involves interrupting output to all Windows in the Screen before the Menus are drawn. If you need to finish your graphical output before this happens, you can set this flag to make sure that you do.

- **SIZEVERIFY** means that you will be doing output to your Window which depends on a knowledge of the current size of the Window. If the user wants to resize the Window, you may want to make sure that any queued output completes before the sizing takes place (critical text, for instance). If this is the case, set this flag. Then, when the user wants to size, Intuition will send you the **SIZEVERIFY** message and **Wait()** until you reply that it's OK to proceed with the sizing. **NOTE:** when I say that Intuition will **Wait()** until you reply, what I'm really saying is that User will **WAIT** until you reply, which suffers the great negative potential of User-Unfriendliness. So remember: use this flag sparingly, and, as always with any IDCMP Message you receive, Reply to it promptly! Then, after User has sized the Window, you can find out about it using **NEWSIZE**:

- **NEWSIZE** is the flag that tells Intuition to send an IDCMP Message to you after the user has resized your Window. At this point, you could examine the size variables

in your Window structure to discover the new size of the Window

- **REFRESHWINDOW** when set will cause a Message to be sent whenever your Window needs refreshing. This flag makes sense only with **SIMPLE_REFRESH** and **SMART_REFRESH** Windows.
 - **MOUSEBUTTONS** will get reports about Mouse-button Up/Down events broadcast to you (Note: only the ones that don't mean something to Intuition. If the user clicks the Select button over a Gadget, Intuition deals with it and you don't find out about it through here).
 - **MOUSEMOVE** will work only if you've set the flag **REPORTMOUSE** above, or if one of your Gadgets has the flag **FOLLOWMOUSE** set. Then all mouse movements will be reported here.
 - **GADGETDOWN** means that when the User "selects" a Gadget you've created with the **GADGETIMMEDIATE** flag set, the fact will be broadcast through the IDCMP.
 - **GADGETUP** means that when the User "releases" a Gadget that you've created with the **RELVERIFY** flag set, the fact will be broadcast through the IDCMP.
 - **MENUPICK** selects that MenuNumber data will come this way the IDCMP rather than the Console.
 - **CLOSEWINDOW** means broadcast the **CLOSEWINDOW** event through the IDCMP. Note that all **RAWKEY** events are transmitted via the IDCMP. Note that these are absolutely RAW keycodes, which you will have to massage before using. Setting this and the **MOUSE** flags effectively eliminates the need to open a Console Device to get input from the keyboard and mouse. Of course, in exchange you lose all of the Console features, most notably the "cooking" of input data and the systematic output of text to your Window.
- Gadgets = the pointer to the first of a linked list of the your own Gadgets which you want attached to this Window. Can be NULL
- If you have no Gadgets of your own
- CheckdMark** = a pointer to an instance of the struct Image where can be found the imagery you want used when any of your MenuItems is to be checked. If you don't want to supply your own imagery and you want to just use Intuition's own checkmark, set this argument to NULL
- Text** = a null-terminated line of text to appear on the title bar of your window (may be null if you want no text)
- Type** = the Screen type for this window. If this equal **CUSTOMSCREEN**, you must have already opened a **CUSTOMSCREEN** (see text above). Types available include:
- **MBENCHSCREEN**
 - **CUSTOMSCREEN**

Screen = if your type is one of Intuition's Standard Screens, then this argument is ignored. However, if Type == **CUSTOMSCREEN**, this must point to the structure of your own Screen

BitMap = if you have specified **SUPER_BITMAP** as the type of raster you want for this Window, then this value points to a instance of the struct **BitMap**. However, if the raster type is NOT **SUPER_BITMAP**, this pointer is ignored

MinWidth, **MinHeight**, **MaxWidth**, **MaxHeight** = the size limits for this Window. These must be reasonable values, which is to say

that the minimums cannot be greater than the current size, nor can the maximums be smaller than the current size. If they are, they're ignored. Any one of these can be initialized to zero, which means that that limit will be set to the current dimension of that axis. The limits can be changed after the Window is opened by calling the WindowLimits() routine. If you haven't requested the WINDOWIZING option, these variables are ignored so you don't have to initialize them.

RESULT

If all is well, returns the pointer to your new Window
If anything goes wrong, returns NULL

BUGS

None

SEE ALSO

OpenScreen()
ModifyIDCMP()
WindowTitles()

OpenWorkBench

OpenWorkBench

NAME

OpenWorkBench -- Opens the WorkBench Screen

SYNOPSIS

BOOL OpenWorkBench();

FUNCTION

This routine attempts to reopen the WorkBench. The actions taken are:
- general good stuff and nice things, and then return TRUE
- find that something has gone wrong, and return FALSE

Even though this routine does return a BOOL value, you can ignore the return value if you want

INPUTS

None

RESULT

TRUE if the WorkBench Screen opened successfully, or was already opened
FALSE if anything went wrong and the WorkBench Screen isn't out there

BUGS

None

SEE ALSO

None

PrintIntText PrintIntText

NAME PrintIntText -- prints the text according to the IntuiText argument

SYNOPSIS PrintIntText(RastPort, IText, LeftEdge, TopEdge)

FUNCTION

Prints the IntuiText into the specified RastPort. Sets up the RastPort as specified by the IntuiText values, then prints the text into the RastPort at the IntuiText x/y coordinates offset by the left/top arguments.

This routine does Intuition window clipping as appropriate -- If you print text outside of your Window, your characters will be clipped at the Window's edge.

If the RastPort field of the IntuiText argument is non-zero, the next IntuiText is rendered as well (return to the top of this FUNCTION section for details).

INPUTS

RastPort = the RastPort destination of the text
IText = pointer to an instance of the structure IntuiText
LeftEdge = left offset of the IntuiText into the RastPort
TopEdge = top offset of the IntuiText into the RastPort

RESULT

None

BUGS

None

SEE ALSO

None

RefreshGadgets

RefreshGadgets

NAME RefreshGadgets -- Refresh (redraw) the Gadget display

SYNOPSIS RefreshGadgets(Gadgets, Pointer, Requester);

FUNCTION

Refreshes (redraws) all of the Gadgets in the Gadget List starting from the specified Gadget.

The Pointer argument can point to either a Window or a Screen structure. Which it actually points to is decided by examining the SCRGADGET flag in the first Gadget of the list; if the flag is set, then Pointer points to a Screen structure, else it points to a Window structure.

The Requester variable can point to a Requester structure. If the first Gadget in the list has the REQCADGET flag set, the Gadget list refers to Gadgets in a Requester and the Pointer must necessarily point to a Window. If these are not the Gadgets of a Requester, the Requester argument may be NULL.

The two main reasons why you might want to use this routine are: first, that you've modified the imagery of the Gadgets in your display and you want the new imagery to be displayed; secondly, if you think that some graphic operation you just performed trashed the Gadgets of your display, this routine will refresh the imagery for you.

The Gadgets argument can be a copy of the FirstGadget variable in either the Screen or Window structure that you want refreshed; the effect of this will be that all Gadgets will be redrawn. However, you can selectively refresh just some of the Gadgets by starting the refresh part-way into the list: for instance, redrawing your Window non-GIMMEZERO Gadgets only, which you've conveniently grouped at the end of your Gadget list.

NOTE: It's never safe to tinker with the Gadget list yourself. Don't supply some Gadget list that Intuition hasn't already processed in the usual way.

NOTE: If you have specified that this is the Gadget list of a Requester, that Requester must be currently displayed

INPUTS

Gadgets = pointer to the first in the list of Gadgets wanting refreshment
Pointer = pointer to either a Screen or Window structure (defined by the SCRGADGET flag of the first Gadget (see next))
Requester = pointer to a Requester (may be NULL if this isn't a Requester Gadget list)

RESULT

None

BUGS
None
SEE ALSO
None

RemakeDisplay

RemakeDisplay

NAME
RemakeDisplay -- Remake the entire Intuition display

SYNOPSIS
RemakeDisplay();

FUNCTION
This is the big one.

This procedure remakes the entire Intuition display. It calls MakeScreen() for every Screen in the system, and then it calls RethinkDisplay() which rethinks the relationships of the Screens to one another and then rethinks the display copper lists.

WARNING: This routine can take several milliseconds to run, so do not use it lightly. RethinkDisplay() (called by this routine) does a Forbid() on entry and a Permit() on exit, which can seriously degrade the performance of the multi-tasking Executive.

INPUTS

None

RESULT

None

BUGS

None

SEE ALSO

RethinkDisplay()
The graphics library's MakeScreen()

RemoveGadget RemoveGadget

NAME RemoveGadget -- removes a Gadget from a Window or a Screen

SYNOPSIS
USHORT RemoveGadget(Pointer, Gadget);

FUNCTION
Removes the given Gadget from the Gadget list of the specified Window or Screen. Returns the ordinal position of the removed Gadget. If the Gadget's SCRGADGET flag is set, the Pointer variable is regarded as a pointer to a Screen; else, it's regarded as a pointer to a Window. If the Gadget pointer points to a Gadget that isn't in the appropriate list, -1 is returned. If there aren't any Gadgets in the list, -1 is returned. If you remove the 65535th Gadget from the list -1 is returned.

INPUTS
Pointer = pointer to the Window or Screen from which the Gadget is to be removed. the Gadget's SCRGADGET flag describes whether this is a pointer to a Window or a Screen
Gadget = pointer to the Gadget to be removed. The Gadget itself describes whether this is a Gadget that should be removed from the Window or the Screen

RESULT
Returns the ordinal position of the removed Gadget. If the Gadget wasn't found in the appropriate list, or if there are no Gadgets in the list, returns -1

BUCS
None
SEE ALSO
AddGadget

ReportMouse ReportMouse

NAME ReportMouse -- tells Intuition whether or not to report mouse movement

SYNOPSIS
ReportMouse(Window, Boolean);

FUNCTION
Tells Intuition whether or not to broadcast mouse-movement events to this Window when it's the active one. The Boolean value specifies whether to start or stop broadcasting position information of mouse-movement. If the Window is the active one, mouse-movement reports start coming immediately afterwards. This same routine will change the current state of the FOLLOWMOUSE function of a currently-selected Gadget too. Note that calling ReportMouse() when a Gadget is selected will only temporarily change whether or not mouse movements are reported while the Gadget is selected; the next time the Gadget is selected, its FOLLOWMOUSE flag is examined anew. Note also that calling ReportMouse() when no Gadget is currently selected will change the state of the Window's REPORTMOUSE flag, but will have no effect on any Gadget that may be subsequently selected.

The ReportMouse() function is first performed when OpenWindow() is first called; if the flag REPORTMOUSE is included among the options, then all mouse-movement events are reported to the opening task and will continue to be reported until ReportMouse() is called with a Boolean value of FALSE. If REPORTMOUSE is not set, then no mouse-movement reports will be broadcast until ReportMouse() is called with a Boolean of TRUE.

INPUTS
Window = pointer to a Window structure associated with this request
Boolean = TRUE or FALSE value specifying whether to turn this function on or off

RESULT
None
BUCS
None
SEE ALSO
None

Request Request

NAME Request -- Activates a Requester

SYNOPSIS Request(Requester, Window);

FUNCTION Links in and displays a Requester into the specified Window.
This routine ignores the Window's **REQVERIFY** flag.

INPUTS Requester = pointer to the Requester to be displayed
Window = pointer to the Window into which this Requester goes

RESULT If the Requester is successfully opened, **TRUE** is returned.
If the Requester could not be opened, **FALSE** is returned.

BUGS None
SEE ALSO None

RethinkDisplay RethinkDisplay

NAME RethinkDisplay -- the grand manipulator of the entire Intuition display

SYNOPSIS RethinkDisplay();

FUNCTION This function performs the Intuition global display reconstruction. This includes: massaging internal state data, rethinking about all of the ViewPorts and their relationship to one another, and, finally, reconstructing the entire display based on the results of all this rethinking.

The reconstruction of the display includes calls to the graphics library to perform **MgCop()** and **LoadView()** for all of Intuition's Screens.

You may perform a **MakeScreen()** on your Custom Screen before calling this routine. The results will be incorporated in the new display.

WARNING: This routine can take several milliseconds to run, so do not use it lightly. **RethinkDisplay()** does a **Forbid()** on entry and a **Permit()** on exit, which can seriously degrade the performance of the multi-tasking Executive.

INPUTS None

RESULT None

BUGS None

SEE ALSO **RemakeDisplay()**
The graphics library's **MakeVPort()**, **MgCop()**, and **LoadView()**

ScreenToBack ScreenToBack
NAME ScreenToBack -- send the specified Screen to the back of the display
SYNOPSIS ScreenToBack(Screen);
FUNCTION Sends the specified Screen to the back of the display
INPUTS Screen = pointer to a Screen structure
RESULT None
BUGS None
SEE ALSO None

ScreenToFront ScreenToFront
NAME ScreenToFront -- brings the specified Screen to the front of the display
SYNOPSIS ScreenToFront(Screen);
FUNCTION Brings the specified Screen to the front of the display
INPUTS Screen = a pointer to a Screen structure
RESULT None
BUGS None
SEE ALSO None

SetDMRequest

SetDMRequest

NAME

SetDMRequest -- sets the DMRequest of the Window

SYNOPSIS

SetDMRequest(Window, DMRequester);

FUNCTION

Attempts to set the DMRequester into the specified window. The DMRequester is the special Requester that you attach to the double-click of the menu button which the user can then bring up on demand. This routine WILL NOT set the DMRequester if it's already set and is currently active (in use by the user). After having called SetDMRequest(), if you want to change the DMRequester, the correct way to start is by calling ClearDMRequest() until it returns a value of TRUE; then you can call SetDMRequest() with the new DMRequester.

INPUTS

Window = pointer to the window from which the DMRequest is to be set
DMRequester = a pointer to a Requester

RESULT

If the current DMRequest was not in use, sets the DMRequest pointer into the Window and returns TRUE.
If the DMRequest was currently in use, doesn't change the pointer and returns FALSE

BUGS

None

SEE ALSO

ClearDMRequest()
Request()

SetMenuStrip

SetMenuStrip

NAME

SetMenuStrip -- Attaches the Menu strip to the Window

SYNOPSIS

SetMenuStrip(Window, Menu);

FUNCTION

Attaches the Menu strip to the Window. After calling this routine, if the user presses the menu button, this specified menu strip will be displayed and accessible.

NOTE: You should always design your Menu strip changes to be a two-way operation, where for every Menu strip you add to your Window you should always plan to clear that strip sometimes. Even in the simplest case, where you will have just one Menu strip for the lifetime of your Window, you should always clear the Menu strip before closing the Window. If you already have a Menu strip attached to this Window, the correct procedure for changing to a new Menu strip involves calling ClearMenuStrip() to clear the old first. The sequence of events should be:

- OpenWindow()
- zero or more iterations of:
 - SetMenuStrip()
 - ClearMenuStrip()
- CloseWindow()

INPUTS

Window = pointer to a Window structure
Menu = pointer to the first Menu in the Menu strip

RESULT

None

BUGS

None

SEE ALSO

ClearMenuStrip()

SetPointer SetPointer

NAME SetPointer -- sets a Window with its own Pointer

SYNOPSIS SetPointer (Window, Pointer, Height, Width, Xoffset, Yoffset);

FUNCTION Sets up the Window with the sprite definition for the Pointer. Then whenever the Window is the active one, the Pointer image will change to its version of the Pointer. If the Window is the active one when this routine is called, the change takes place immediately.

The Xoffset and Yoffset are used to offset the top-left corner of the hardware sprite imagery from what Intuition regards as the current position of the Pointer. Another way of describing it is as the offset from the "hot spot" of the Pointer to the top-left corner of the sprite. For instance, if you specify offsets of zero, zero, then the top-left corner of your sprite image will be placed at the Pointer position. On the other hand, if you specify an Xoffset of -7 (remember, sprites are 16 pixels wide) then your sprite will be centered over the Pointer position. If you specify an Xoffset of -15, the right-edge of the sprite will be over the Pointer position.

INPUTS Window = pointer to the Window to receive this Pointer definition Pointer = pointer to the data definition of a Sprite Height = the height of the Pointer Width = the Width of the sprite (must be less than or equal to sixteen) Xoffset = the offset for your sprite from the Pointer position Yoffset = the offset for your sprite from the Pointer position

RESULT None
BUGS None
SEE ALSO ClearPointer ()

SetWindowTitles SetWindowTitles

NAME SetWindowTitles -- Sets the Window's titles for both Window and Screen

SYNOPSIS SetWindowTitles(Window, WindowTitle, ScreenTitle);

FUNCTION Allows you to set the text which appears in the Window and/or Screen title bars.

The Window Title appears at all times along the Window Title Bar. The Window's Screen Title appears at the Screen Title Bar whenever this Screen is the active one.

When this routine is called, your Window Title will be changed immediately. If your Window is the active one when this routine is called, the Screen Title will be changed immediately. You can specify a value of -1 (negative one) for either of the title pointers. This designates that you want to Intuition to leave the current setting of that particular title alone, and modify only the other one. Of course, you could set both to -1.

Furthermore, you can set a value of 0 (zero) for either of the title pointers. Doing so specifies that you want no title to appear (the title bar will be blank).

INPUTS Window = pointer to your Window structure WindowTitle = pointer to a null-terminated text string, or set to either the value of -1 (negative one) or 0 (zero) ScreenTitle = pointer to a null-terminated text string, or set to either the value of -1 (negative one) or 0 (zero)

RESULT None
BUGS None
SEE ALSO OpenWindow ()

ShowTitle ShowTitle

NAME ShowTitle -- Set the Screen title bar display mode

SYNOPSIS ShowTitle(Screen, ShowIt);

FUNCTION

This routine sets the SHOWTITLE flag of the specified Screen, and then coordinates the redisplay of the Screen and its Windows.

The Screen title bar can appear either in front of or behind BACKDROP Windows. This is contrasted with the fact that non-BACKDROP Windows always appear in front of the Screen Title Bar. You specify whether you want the Screen Title Bar to be in front of or behind the Screen's BACKDROP Windows by calling this routine.

The ShowIt argument should be set to either TRUE or FALSE. If TRUE, the Screen's Title Bar will be shown in front of BACKDROP Windows. If FALSE, the Title Bar will be rendered behind all Windows.

When a Screen is first opened, the default setting of the SHOWTITLE flag is TRUE.

INPUTS

Screen = pointer to a Screen structure
ShowIt = Boolean TRUE or FALSE describing whether to show or hide the Screen Title Bar

RESULT

None

BUGS

None

SEE ALSO

None

SizeWindow

SizeWindow

NAME SizeWindow -- Ask Intuition to size a Window

SYNOPSIS SizeWindow(Window, DeltaX, DeltaY);

FUNCTION

This routine sends a request to Intuition asking to size the Window the specified amounts. The delta arguments describes how much to size the Window along the respective axes.

Note that the Window will not be sized immediately, but rather will be sized the next time Intuition receives an input event, which happens currently at a minimum rate of ten times per second, and a maximum of sixty times a second. You can discover when you Window has finally been sized by setting the NEWSIZE flag of the IDCMP of your Window. See the "Input and Output Methods" chapter of this book for description of the IDCMP.

This routine does no error-checking. If your delta values specify some far corner of the Universe, Intuition will attempt to size your Window to the far corners of the Universe. Because of the distortions in the space-time continuum that can result from this, as predicted by special relativity, the result is generally not a pretty sight.

INPUTS

Window = pointer to the structure of the Window to be sized
DeltaX = signed value describing how much to size the Window on the x-axis
DeltaY = signed value describing how much to size the Window on the y-axis

RESULT

None

BUGS

None

SEE ALSO

MoveWindow(), WindowToFront(), WindowToBack()

ViewAddress ViewAddress

NAME ViewAddress -- returns the address of the Intuition View structure

SYNOPSIS ViewAddress();

FUNCTION

Returns the address of the Intuition View structure. If you want to use any of the graphics, text, or animation primitives in your Window and that primitive requires a pointer to a View, this routine will return the address of the View for you.

INPUTS

None

RESULT

Returns the address of the Intuition View structure

BUGS

Would be hard for this routine to have a bug

SEE ALSO

All of the graphics, text, and animation primitives

ViewPortAddress

ViewPortAddress

NAME ViewPortAddress -- returns the address of a Window's ViewPort structure

SYNOPSIS ViewPortAddress(Window);

FUNCTION

Returns the address of the ViewPort associated with the specified Window. The ViewPort is actually the ViewPort of the Screen within which the Window is displayed. If you want to use any of the graphics, text, or animation primitives in your Window and that primitive requires a pointer to a ViewPort, you can use this call.

INPUTS

Window = pointer to the Window for which you want the ViewPort address

RESULT

Returns the address of the Intuition View structure

BUGS

Would be hard for this routine to have a bug

SEE ALSO

All of the graphics, text, and animation primitives

WBenchToBack WBenchToBack

NAME
WBenchToBack -- Sends the WorkBench Screen in back of all Screens

SYNOPSIS
WBenchToBack ();

FUNCTION

Causes the WorkBench Screen, if it's currently opened, to go to the background. This does not 'move' the Screen up or down, instead only affects the depth-arrangement of the Screen.

If the WorkBench Screen was opened, this function returns TRUE, otherwise it returns FALSE.

INPUTS
None

RESULT

If the WorkBench Screen was opened, this function returns TRUE, otherwise it returns FALSE.

BUGS
None

SEE ALSO
WBenchToFront ()

WBenchToFront

WBenchToFront

NAME
WBenchToFront -- Brings the WorkBench Screen in front of all Screens

SYNOPSIS
WBenchToFront ();

FUNCTION

Causes the WorkBench Screen, if it's currently opened, to come to the foreground. This does not 'move' the Screen up or down, instead only affects the depth-arrangement of the Screen.

If the WorkBench Screen was opened, this function returns TRUE, otherwise it returns FALSE.

INPUTS
None

RESULT

If the WorkBench Screen was opened, this function returns TRUE, otherwise it returns FALSE.

BUGS
None

SEE ALSO
WBenchToBack ()

WindowLimits WindowLimits

NAME WindowLimits -- Set the minimum and maximum limits of the Window

SYNOPSIS WindowLimits (Window, MinWidth, MinHeight, MaxWidth, MaxHeight);

FUNCTION

Sets the minimum and maximum limits of the Window's size. Until this routine is called, the Window's size limits are equal to the Window's initial size, which means that the user won't be able to size it at all. After the call to this routine, the Window will be able to be sized to any dimensions within the specified limits.

If you don't want to change any one of the dimensions, set the limit argument for that dimension to zero. If any of the limit arguments is equal to zero, that argument is ignored and the initial setting of that parameter remains undisturbed.

If any of the arguments is out of range (minimums greater than the current size, maximums less than the current size), that limit will be ignored, though the others will still take effect if they are in range. If any are out of range, the return value from this procedure will be FALSE. If all arguments are valid, the return value will be TRUE.

If the user is currently sizing this Window, the new limits will not take effect until after the sizing is completed.

INPUTS

Window = pointer to a Window structure
MinWidth, MinHeight, MaxWidth, MaxHeight = the new limits for the size of this Window. If any of these is set to zero, it will be ignored and that setting will be unchanged.

RESULT

Returns TRUE if everything was in order. If any of the parameters was out of range (minimums greater than current size, maximums less than current size), FALSE is returned and the errant limit request is not fulfilled (though the valid ones will be).

BUGS

None

SEE ALSO

None

WindowToBack WindowToBack

NAME WindowToBack -- Ask Intuition to send this Window to the back

SYNOPSIS WindowToBack (Window);

FUNCTION

This routine sends a request to Intuition asking to send the Window in back of all other Windows in the Screen.

Note that the Window will not be depth-arranged immediately, but rather will be arranged the next time Intuition receives an input event, which happens currently at a minimum rate of ten times per second, and a maximum of sixty times a second.

Remember that BACKDROP Windows cannot be depth-arranged.

INPUTS

Window = pointer to the structure of the Window to be sent to the back

RESULT

None

BUGS

None

SEE ALSO

MoveWindow(), SizeWindow(), WindowToFront()

WindowToFront WindowToFront

NAME WindowToFront -- Ask Intuition to bring this Window to the front

SYNOPSIS WindowToFront(Window);

FUNCTION

This routine sends a request to Intuition asking to bring the Window in front of all other Windows in the Screen.

Note that the Window will not be depth-arranged immediately, but rather will be arranged the next time Intuition receives an input event, which happens currently at a minimum rate of ten times per second, and a maximum of sixty times a second.

Remember that BACKDROP Windows cannot be depth-arranged.

INPUTS

Window = pointer to the structure of the Window to be brought to front

RESULT

None

BUGS

None

SEE ALSO

MoveWindow(), SizeWindow(), WindowToFront()

TABLE OF CONTENTS

layers.library/BeginUpdate
 layers.library/BehindLayer
 layers.library/CreateBehindLayer
 layers.library/CreateUpfrontLayer
 layers.library/DeleteLayer
 layers.library/DisposeLayerInfo
 layers.library/EndUpdate
 layers.library/FattenLayerInfo
 layers.library/InitLayers
 layers.library/LockLayer
 layers.library/LockLayerInfo
 layers.library/MoveLayer
 layers.library/MoveLayerToFrontOf
 layers.library/NewLayerInfo
 layers.library/ScrollLayer
 layers.library/SizeLayer
 layers.library/SwapBitsLastPortClipRect
 layers.library/ThinLayerInfo
 layers.library/UnlockLayer
 layers.library/UnlockLayerInfo
 layers.library/UpfrontLayer
 layers.library/WhichLayer

layers.library/BeginUpdate layers.library/BeginUpdate

NAME BeginUpdate -- prepare to repair damaged layer

SYNOPSIS BeginUpdate(1)
 a0

INPUTS 1 = pointer to a layer

FUNCTION convert damage list to ClipRect list and swap in for programmer to redraw through. This routine simulates the rom library environment. The layer is locked against changes made by the layer library.

SEE ALSO layers.h EndUpdate()

layers.library/BehindLayer layers.library/BehindLayer

NAME BehindLayer -- Put layer behind other layers.

SYNOPSIS BehindLayer (l1, l)
 a0 a1

INPUTS l1 = pointer to LayerInfo structure
 l = pointer to a layer

FUNCTION Move this layer to the most behind position swapping bits in and out of the display with other layers.
If other layers are REFRESH then collect their damage lists and set bit in Flags of those layers that may be revealed.
If this layer is a backdrop layer then put this layer behind all other backdrop layers.
If this layer is NOT a backdrop layer then put in front of the top backdrop layer and behind all other layers.

RETURNS TRUE if operation successful
 FALSE if operation unsuccessful (probably out of memory)

BUGS
SEE ALSO layers.h

layers.library/CreateBehindLayer layers.library/CreateBehindLayer

NAME CreateBehindLayer -- Create a new layer behind all existing layers.

SYNOPSIS CreateBehindLayer (l1,bm,x0,y0,x1,y1,flags [,bm2])
 a0 a1 d0 d1 d2 d3 d4 [a2]

INPUTS l1 = pointer to LayerInfo structure
 bm = pointer to common BitMap used by all Layers
 bm2 = pointer to optional Super BitMap
 flags= various types of layers supported as bit sets.
 x0,y0= upper left hand corner of layer
 x1,y1= lower right hand corner of layer

FUNCTION Create a new Layer of position and size (x0,y0)->(x1,y1) Make this layer of type found in flags if SuperBitMap, use bm2 as pointer to real SuperBitMap. and copy contents of Superbitmap into display layer. If this layer is a backdrop layer then place it behind all other layers including other backdrop layers. If this is not a backdrop layer then place it behind all nonbackdrop layers.

SEE ALSO layers.h

Dec 3 17:50 1985 layers.Doc Page 5

layers.library/CreteupfrontLayer layers.library/CreteupfrontLayer

NAME CreteupfrontLayer -- Create a new layer on top of existing layers.

SYNOPSIS

CreteupfrontLayer (l1,bm,x0,y0,x1,y1,flags [,bm2])
a0 a1 d0 d1 d2 d3 d4 [a2]

INPUTS

l1 = pointer to LayerInfo structure
bm = pointer to common BitMap used by all Layers
bm2 = pointer to optional Super BitMap
flags= various types of layers supported as bit sets.
x0,y0= upper left hand corner of layer
x1,y1= lower right hand corner of layer

FUNCTION

Create a new Layer of position and size (x0,y0)->(x1,y1) and place it on top of all other layers.
Make this layer of type found in flags
if SuperBitMap, use bm2 as pointer to real SuperBitMap.
and copy contents of Superbitmap into display layer.

SEE ALSO layers.h

Dec 3 17:50 1985 layers.Doc Page 6

layers.library/Deletelayer layers.library/Deletelayer

NAME Deletelayer -- delete layer from layer list.

SYNOPSIS

Deletelayer (l1, l)
a0 a1

INPUTS

l1 = pointer to LayerInfo structure
l = pointer to a layer

FUNCTION

Remove this layer from the list of layers. Release memory associated with it. Restore other layers that may have been obscured by it. Trigger refresh in those that may need it. If this is a superbitmap make sure SuperBitMap is current. The SuperBitMap is not removed from the system but is available for program sans rest of layer stuff

SEE ALSO layers.h

Dec 3 17:50 1985 layers.Doc Page 7

layers.library/DisposeLayerInfo layers.library/DisposeLayerInfo

NAME

DisposeLayerInfo -- Return all memory for LayerInfo to mem pool

SYNOPSIS

DisposeLayerInfo(l1)
a0

INPUTS

l1 = pointer to LayerInfo structure

FUNCTION

return LayerInfo and any other memory attached to this LayerInfo to memory allocator

SEE ALSO

layers.h

Dec 3 17:50 1985 layers.Doc Page 8

layers.library/EndUpdate

layers.library/EndUpdate

NAME

EndUpdate -- remove damage list and restore state of layer to normal.

SYNOPSIS

EndUpdate(l, flag)
a0 d0

INPUTS

l = pointer to a layer
flag= TRUE if update was successful. The damage list is cleared.

FUNCTION

After the programmer has redrawn his picture he calls this routine to restore the ClipRects to point to his standard layer tiling.

Use flag=0 if you are only making a partial update. You may use the other region functions to clip adjust the DamageList to reflect a partial update.

SEE ALSO

layers.h BeginUpdate()

Dec 3 17:50 1985 layers.Doc Page 9

layers.library/FattenLayerInfo layers.library/FattenLayerInfo

NAME FattenLayerInfo -- convert 1.0 LayerInfo to 1.1 LayerInfo

SYNOPSIS FattenLayerInfo(1i)
a0

INPUTS 1i = pointer to LayerInfo structure

FUNCTION

1.1 software on need to have more info in the Layer_Info structure. To do this in a 1.0 supportable manner requires allocation and deallocation of the memory whenever most layer library functions are called. To prevent unnecessary allocation/deallocation FattenLayerInfo will preallocate the necessary data structures and fake out the layer library into thinking it has a LayerInfo gotten from NewLayerInfo. NewLayerInfo is the approved method for getting this structure. When a program needs to give up the LayerInfo structure it must call ThinLayerInfo before freeing the memory. ThinLayerInfo is not necessary if New/DisposeLayerInfo are used however.

SEE ALSO NewLayerInfo ThinLayerInfo DisposeLayerInfo layers.h

Dec 3 17:50 1985 layers.Doc Page 10

layers.library/InitLayers layers.library/InitLayers

NAME InitLayers -- Initialize Layer_Info structure

SYNOPSIS InitLayers(1i)
a0

INPUTS 1i = pointer to LayerInfo structure

FUNCTION

Initialize Layer_Info structure in preparation to use other layer operations on this list of layers. Make the Layers unlocked (open).

SEE ALSO layers.h

Dec 3 17:50 1985 layers.Doc Page 11

layers.library/LockLayer

layers.library/LockLayer

NAME

LockLayer -- lock layer to make changes to ClipRects.

SYNOPSIS

LockLayer (l1, l1)
 a0 a1

INPUTS

l1 = pointer to LayerInfo structure
l1 = pointer to a layer

FUNCTION

Makes this layer unavailable for other tasks to use.
If another task is already using this layer then wait for
it to complete and then take the layer.

SEE ALSO

layers.h

Dec 3 17:50 1985 layers.Doc Page 12

layers.library/LockLayerInfo

layers.library/LockLayerInfo

NAME

LockLayerInfo -- lock the LayerInfo structure.

SYNOPSIS

LockLayerInfo (l1)
 a0

INPUTS

l1 = pointer to LayerInfo structure

FUNCTION

After the operation is complete that required a LockLayerInfo,
unlock the LayerInfo structure so that other tasks may
affect the layers.

SEE ALSO

layers.h LockLayerInfo()

Dec 3 17:50 1985 layers.Doc Page 13

layers.library/LockLayers layers.library/LockLayers

NAME LockLayers -- lock all layers from graphics output

SYNOPSIS
LockLayers (ll)
 a0

INPUTS ll = pointer to LayerInfo structure

FUNCTION first calls LockLayerInfo
make all layers in this layer list locked.

SEE ALSO layers.h LockLayer() LockLayerInfo()

Dec 3 17:50 1985 layers.Doc Page 14

layers.library/MoveLayer

layers.library/MoveLayer

NAME MoveLayer -- Move nonbackdrop layer to new position in BitMap

SYNOPSIS
MoveLayer (ll, l, dx, dy)
 a0 a1 d0 d1

INPUTS ll = pointer to LayerInfo structure
l = pointer to a nonbackdrop layer
dx = delta to add to current x position
dy = delta to add to current y position

FUNCTION Move this layer to new position in shared BitMap.
If any refresh layers become revealed, collect damage and
set REFRESH bit in layer Flags.

SEE ALSO
layers.h

Dec 3 17:50 1985 layers.Doc Page 15

layers.library/MoveLayerInFrontOf layers.library/MoveLayerInFrontOf

NAME MoveLayerInFrontOf-- Put layer in front of another layer

SYNOPSIS

BOOLEAN MoveLayerInFrontOf(layertomove, target)
a0 a1

INPUTS

layertomove : layer to moved
target : move layertomove infront of target

FUNCTION

Move this layer to in front of target swapping bits
in and out of the display with other layers.
If this is a refresh layer then collect damage list and
set bit in Flags if redraw required.
By clearing the BACKDROP bit in the layers Flags you may
bring a Backdrop layer up to the front of all other layers.

RETURNS

TRUE if operation successful
FALSE if operation unsuccessful (probably out of memory)

SEE ALSO

layers.h

Dec 3 17:50 1985 layers.Doc Page 16

layers.library/NewLayerInfo layers.library/NewLayerInfo

NAME NewLayerInfo -- Allocate and Initialize full Layer_Info structure

SYNOPSIS

NewLayerInfo()

INPUTS

None

FUNCTION

Allocate memory required for full Layer_Info structure.
Initialize Layer_Info structure in preparation to use
other layer operations on this list of layers.
Make the Layers unlocked (open).

RETURNS

pointer to Layer_Info structure if successful
NULL if not enough memory

SEE ALSO

layers.h

Dec 3 17:50 1985 layers.Doc Page 17

layers.library/ScrollLayer layers.library/ScrollLayer

NAME ScrollLayer -- scroll around in a superbitmap

SYNOPSIS ScrollLayer (ll, l, dx, dy)
 a0 a1 d0 d1

INPUTS ll = pointer to LayerInfo structure
 l = pointer to a nonbackdrop layer
 dx = delta to add to current x scroll value
 dy = delta to add to current y scroll value

FUNCTION Copy bits between layer and superbitmap to reposition layer
 over different portion of superbitmap.

SEE ALSO layers.h

Dec 3 17:50 1985 layers.Doc Page 18

layers.library/SizeLayer layers.library/SizeLayer

NAME SizeLayer -- Change the size of this nonbackdrop layer.

SYNOPSIS SizeLayer (ll, l, dx, dy)
 a0 a1 d0 d1

INPUTS ll = pointer to LayerInfo structure
 l = pointer to a nonbackdrop layer
 dx = delta to add to current x size
 dy = delta to add to current y size

FUNCTION Change the size of this layer by (dx,dy). The lower right hand
 corner is extended to make room for the larger layer.
 If there is SuperBitmap for this layer then copy pixels into
 or out of the layer depending on whether the layer increases or
 decreases in size.
 Collect damage list for those layers that may need to be
 refreshed if damage occurred.

NOTE The current implementation forces layer to front. This is not to
 be depended upon and may change in future releases of layer lib.

SEE ALSO layers.h

Dec 3 17:50 1985 layers.Doc Page 19

layers.library/SwapBitsRastPortClipRect

NAME

SwapBitsRastPortClipRect -- Swap bits between common bitmap
and obscured ClipRect

SYNOPSIS

SwapBitsRastPortClipRect(rp, cr)
a0 a1

INPUTS

rp = pointer to rastport
cr = pointer to cliprect to swap bits with

FUNCTION

Support routine useful for those that need to do some
operations not done by the layer library. Allows programmer
to swap the contents of a small BitMap with a subsection of
the display. This is accomplished without using extra memory.
The bits in the display RastPort are exchanged with the
bits in the ClipRect's BitMap.

SEE ALSO

Dec 3 17:50 1985 layers.Doc Page 20

layers.library/ThinLayerInfo

NAME

ThinLayerInfo -- convert 1.1 LayerInfo to 1.0 LayerInfo

SYNOPSIS

ThinLayerInfo(l1)
a0

INPUTS

l1 = pointer to LayerInfo structure

FUNCTION

return the extra memory needed that was allocated with
FattenLayerInfo. This is must be done prior to freeing
the Layer_info structure itself. V1.1 software should be
using DisposeLayerInfo

SEE ALSO

layers.h
DisposeLayerInfo FattenLayerInfo

layers.library/ThinLayerInfo

Dec 3 17:50 1985 layers.Doc Page 21

layers.library/UnlockLayer

layers.library/UnlockLayer

NAME

UnlockLayer -- unlock layer and allow graphics routines to use it.

SYNOPSIS

UnlockLayer (1)
a0

INPUTS

1 = pointer to a layer

FUNCTION

When finished changing the ClipRects or whatever you were doing with this layer you must UnlockLayer it to allow the other task to proceed with it's graphic output.

SEE ALSO

layers.h

Dec 3 17:50 1985 layers.Doc Page 22

layers.library/UnlockLayerInfo

layers.library/UnlockLayerInfo

NAME

UnlockLayerInfo -- unlock the LayerInfo structure.

SYNOPSIS

UnlockLayerInfo (11)
a0

INPUTS

11 = pointer to LayerInfo structure

FUNCTION

Before doing an operation that requires the LayerInfo structure make sure that no other task is also using the LayerInfo structure. This procedure returns when the LayerInfo belongs to this task. There should be an UnlockLayerInfo for every LockLayerInfo.

All layer routines presently LockLayerInfo when they startup and UnlockLayerInfo as they exit. Programmers will need to use these Lock/Unlock routines if they wish to do something with the LayerStructure that is not supported by the layer library.

SEE ALSO

layers.h UnlockLayerInfo ()

Dec 3 17:50 1985 layers.Doc Page 23

layers.library/UnlockLayers layers.library/UnlockLayers

NAME

UnlockLayers -- unlock all layers from graphics output
Restart graphics output to layers that
have been waiting

SYNOPSIS

UnlockLayers(ll)
 a0

INPUTS

ll = pointer to LayerInfo structure

FUNCTION

make all layers in this layer list unlocked.
Then call UnlockLayerInfo

SEE ALSO

layers.h UnlockLayer()

Dec 3 17:50 1985 layers.Doc Page 24

layers.library/UpfrontLayer

layers.library/UpfrontLayer

NAME

UpfrontLayer -- Put layer in front of all other layers.

SYNOPSIS

BOOLEAN UpfrontLayer(ll, l)
 a0 a1

INPUTS

ll = pointer to LayerInfo structure
l = pointer to a nonbackdrop layer

FUNCTION

Move this layer to the most upfront position swapping bits
in and out of the display with other layers.
If this is a refresh layer then collect damage list and
set bit in Flags if redraw required.
By clearing the BACKDROP bit in the layers Flags you may
bring a Backdrop layer up to the front of all other layers.

RETURNS

TRUE if operation successful
FALSE if operation unsuccessful (probably out of memory)

SEE ALSO

layers.h

layers.library/WhichLayer layers.library/WhichLayer

NAME WhichLayer -- Which Layer is this point in?

SYNOPSIS layer = (struct Layer *)WhichLayer(li, x, y)
a0 d0 dl

INPUTS li = pointer to LayerInfo structure
(x,y) = coordinate in the BitMap

FUNCTION Starting at the topmost layer check to see if this point (x,y) occurs in this layer. If it does return the pointer to this layer. Return 0 if there is no layer at this point.

SEE ALSO layers.h

Appendix B

Device Summaries

This appendix contains Unix-like summaries for the commands that may be applied to ROM (or kickstart) resident devices, as well as summaries of routines in disk-loadable devices.

These documentation files are organized on a device-by-device basis. The tutorial sections of this manual give you information that shows how these device commands relate to each other and the prerequisites for calling them.

To use any of the device commands, you must first open the device. The correct calling sequence for opening each device is shown in the device tutorial chapter itself. This introduction lists the names of the current set of devices that are included with the system.

If the device is disk-resident, it gets loaded and initialized. The **OpenDevice()** fills in the **io_Device** and **io_Unit** fields of your I/O request block, thereby tying that request block to a specific device. When you say **DoIO(IORrequest)**, the **DoIO()** routine, among others, looks in the **IORrequest** to find out which device is to be used. This prevents having to have a complete (duplicate) set of I/O transmit and control functions for each device.

The following is a list of the names of the devices currently a part of the Amiga software. All of these are to be treated as null-terminated strings, given to the **OpenDevice()** function. For example:

```
error = OpenDevice("keyboard.device",0,IORrequest,0);
```

See the **exec.doc** summary for **OpenDevice()** for the meaning of the various fields of this command.

Device Names

audio.device
console.device
gameport.device
keyboard.device
parallel.device
printer.device
serial.device
timer.device
trackdisk.device

When you have finished using a device, at the end of your program, you should close it, using the **CloseDevice()** function as follows:

```
CloseDevice(IOResult);
```

You must also free whatever memory you may have dedicated to the device communication before your program ends. Note that you must assure that the device has responded to all of your I/O requests by returning your IOResult blocks before you attempt to close the device or deallocate the memory.

If the system is running out of memory and needs to free up space, it can check the **accessors** field for various devices. If you have closed the device, it decrements its **accessors** count. For those devices whose **accessors** value is zero, the system can retrieve the memory that the device had used.

Certain devices have routines associated with them, and can almost be treated as libraries. The devices that have specific interface routines are the timer device and the console device. To access these routines, you must, as with a library, provide a value to a specific **Base** variable name:

| Device | Base Address Name |
|---------|-------------------|
| timer | TimerBase |
| console | ConsoleDevice |

To get this base address, you must open the device, then copy the **io_Device** field from your **IOResult** block as the base address for this "library" routine. Note that unlike libraries, you need not issue a **CloseLibrary()** command after using the device routines. The **CloseDevice()** function call is sufficient.

An example showing how to obtain the base address for the timer device is shown in the "Timer Device" chapter in this manual.

| | |
|-----------------|------------------|
| AbortIO | audio device |
| AbortIO | serial device |
| AbortIO | narrator device |
| AbortIO | parallel device |
| AddHandler | input device |
| AddResetHandler | keyboard device |
| AddTime | timer device |
| ALLOCATE | audio device |
| AskCType | gameport device |
| AskTrigger | gameport device |
| background | timer device |
| BeginIO | audio device |
| BeginIO | serial device |
| BeginIO | parallel device |
| BeginIO | clipboard device |
| Break | serial device |
| CDAskKeyMap | console device |
| CDAskKeyMap | console device |
| CDInputHandler | console device |
| CDInputHandler | console device |
| CDSetKeyMap | console device |
| CDSetKeyMap | console device |
| CLEAR | audio device |
| Clear | input device |
| Clear | serial device |
| Clear | console device |
| Clear | console device |
| Clear | gameport device |
| Clear | keyboard device |
| Clear | parallel device |
| Close | serial device |
| Close | narrator device |
| Close | parallel device |
| Close | clipboard device |
| CloseDevice | audio device |
| CmpTime | timer device |
| CurrentReadID | clipboard device |
| CurrentWriteID | clipboard device |
| DumpRPort | printer device |
| Expunge | audio device |
| Expunge | clipboard device |
| FINISH | audio device |
| FLUSH | audio device |
| Flush | serial device |
| Flush | printer device |
| Flush | narrator device |
| Flush | parallel device |
| FREE | audio device |
| Invalid | printer device |
| LOCK | audio device |
| Open | input device |
| Open | serial device |
| Open | gameport device |
| Open | narrator device |
| Open | parallel device |
| Open | clipboard device |

| | |
|------------------|------------------|
| OpenDevice | audio device |
| OpenDevice | console device |
| OpenDevice | console device |
| PERVOL | audio device |
| Post | clipboard device |
| PrtCommand | printer device |
| Query | serial device |
| Query | parallel device |
| RawKeyConvert | console device |
| RawKeyConvert | console device |
| RawWrite | printer device |
| READ | audio device |
| Read | serial device |
| Read | console device |
| Read | console device |
| Read | narrator device |
| Read | parallel device |
| Read | clipboard device |
| ReadEvent | gameport device |
| ReadEvent | keyboard device |
| ReadMatrix | keyboard device |
| RemHandler | input device |
| RemResetHandler | keyboard device |
| RESET | audio device |
| Reset | input device |
| Reset | serial device |
| Reset | printer device |
| Reset | keyboard device |
| Reset | narrator device |
| Reset | parallel device |
| Reset | clipboard device |
| ResetHandlerDone | keyboard device |
| SetCType | gameport device |
| SetMPort | input device |
| SetMTrig | input device |
| SetMType | input device |
| SetParams | serial device |
| SetParams | parallel device |
| SetPeriod | input device |
| SETPREC | audio device |
| SetThresh | input device |
| SetTrigger | gameport device |
| START | audio device |
| Start | input device |
| Start | serial device |
| Start | printer device |
| Start | narrator device |
| Start | parallel device |
| STOP | audio device |
| Stop | serial device |
| Stop | printer device |
| Stop | parallel device |
| SubTime | timer device |
| TR_ADDREQUEST | timer device |
| TR_GETSYSTIME | timer device |
| TR_SETSYSTIME | timer device |

UPDATE
Update
WAITCYCLE
WRITE
Write
Write
Write
Write
Write
Write
Write
Write
WriteEvent

audio device
clipboard device
audio device
audio device
serial device
console device
console device
printer device
narrator device
parallel device
clipboard device
input device

TABLE OF CONTENTS

audio.device/AbortIO
audio.device/BeginIO
audio.device/BeginIO/ADCMD_ALLOCATE
audio.device/BeginIO/ADCMD_FINISH
audio.device/BeginIO/ADCMD_FREE
audio.device/BeginIO/ADCMD_LOCK
audio.device/BeginIO/ADCMD_PERVOL
audio.device/BeginIO/ADCMD_SETPREC
audio.device/BeginIO/ADCMD_WAITCYCLE
audio.device/BeginIO/CMD_CLEAR
audio.device/BeginIO/CMD_FLUSH
audio.device/BeginIO/CMD_READ
audio.device/BeginIO/CMD_RESET
audio.device/BeginIO/CMD_START
audio.device/BeginIO/CMD_STOP
audio.device/BeginIO/CMD_UPDATE
audio.device/BeginIO/CMD_WRITE
audio.device/CloseDevice
audio.device/Expunge
audio.device/OpenDevice

audio.device/AbortIO audio.device/AbortIO

NAME AbortIO - abort a device command

SYNOPSIS
AbortIO(iORequest);
AI

FUNCTION
AbortIO tries to abort a device command. It is allowed to be unsuccessful. If the Abort is successful, the io_Error field of the iORequest contains an indication that IO was aborted.

INPUTS
iORequest -- pointer to the I/O Request for the command to abort

audio.device/BeginIO

audio.device/BeginIO

NAME

BeginIO - dispatch a device command

SYNOPSIS

BeginIO(iORrequest);
 AI

FUNCTION

BeginIO has the responsibility of dispatching all device commands. Immediate commands are always called directly, and all other commands are queued to make them single threaded.

INPUTS

iORrequest -- pointer to the I/O Request for this command

audio.device/BeginIO/ADCMD_ALLOCATE

NAME

ADCMD_ALLOCATE -- allocate a set of audio channels

FUNCTION

ADCMD_ALLOCATE is a command that allocates multiple audio channels. ADCMD_ALLOCATE takes an array of possible channel combinations (ioaData) and an allocation precedence (inPri) and tries to allocate one of the combinations of channels.

If the channel combination array is zero length (ioaLength), the allocation succeeds; otherwise, ADCMD_ALLOCATE checks each combination, one at a time, in the specified order, to find one combination that does not require ADCMD_ALLOCATE to steal allocated channels.

If it must steal allocated channels, it uses the channel combination that steals the lowest precedence channels.

ADCMD_ALLOCATE cannot steal a channel of equal or greater precedence than the allocation precedence (inPri).

If it fails to allocate any channel combination and the no-wait flag (ADIOE_NOWAIT) is set ADCMD_ALLOCATE returns a zero in the unit field of the I/O request (ioUnit) and an error (IOERR_ALLOCFAILED). If the no-wait flag is clear, it places the I/O request in a list that tries to allocate again whenever ADCMD_FREE frees channels or ADCMD_SETPREC lowers the channels' precedences.

If the allocation is successful, ADCMD_ALLOCATE checks if any channels are locked (ADCMD_LOCK) and if so, replies (ReplyMsg) the lock I/O request with an error (ADIOERR_CHANNELSTOLEN). Then it places the allocation I/O request in a list waiting for the locked channels to be freed. When all the allocated channels are un-locked, ADCMD_ALLOCATE:

- . resets (CMD_RESET) the allocated channels,
- . generates a new allocation key (ioaAllocKey), if it is zero,
- . copies the allocation key into each of the allocated channels
- . copies the allocation precedence into each of the allocated channels, and
- . copies the channel bit map into the unit field of the I/O request.

If channels are allocated with a non-zero allocation key, ADCMD_ALLOCATE allocates with that same key; otherwise, it generates a new and unique key.

ADCMD_ALLOCATE is synchronous:

- . if the allocation succeeds and there are no locked channels to be stolen, or
 - . if the allocation fails and the no-wait flag is set.
- In either case, ADCMD_ALLOCATE only replies (mReplyPort) if the quick flag (IOF_QUICK) is clear; otherwise, the allocation is asynchronous, so it clears the quick flag and replies the I/O request after the allocation is finished. If channels are stolen, all audio

device commands return an error (IOERR_NOALLOCATION) when the former user tries to use them again. Do not use ADCMD_ALLOCATE in Interrupt code.

If you decide to store directly to the audio hardware registers, you must either lock the channels you've allocated, or set the precedence to maximum (ADALLOC_MAXPREC) to prevent the channels from being stolen.

Under all circumstances, unless channels are stolen, you must free (ADCMD_FREE) all allocated channels when you are finished using them.

INPUTS

in_Pri - allocation precedence (-128 thru 127)
mn_ReplyPort- pointer to message port that receives I/O request after the allocation completes is asynchronous or quick flag (ADIOF_QUICK) is set
io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
io_Command - command number for ADCMD_ALLOCATE
io_Flags - flags, must be cleared if not used:
 IOF_QUICK - (CLEAR) reply I/O request only if (SET) only reply I/O request (see above text) asynchronous (see above text) if allocation fails, wait till it succeeds
 ADIOF_NOWAIT- (CLEAR) if allocation fails, return error (SET) if allocation fails, return error (ADIOERR_ALLOCFAILED)
ioa_AllocKey- allocation key, zero to generate new key; otherwise, it must be set by (or copied from I/O block set by) OpenDevice function or previous ADCMD_ALLOCATE command
ioa_Data - pointer to channel combination options (byte array, bits 0 thru 3 correspond to channels 0 thru 3)
ioa_Length - length of the channel combination option array (0 thru 16, 0 always succeeds)

OUTPUTS

io_Unit - bit map of successfully allocated channels (bits 0 thru 3 correspond to channels 0 thru 3)
io_Flags - IOF_QUICK flag cleared if asynchronous (see above text)
io_Error - error number:
 0 - no error
 ADIOERR_ALLOCFAILED - allocation failed

audio.device/BeginIO/ADCMD_FINISH

audio.device/BeginIO/ADCMD_FINISH

NAME

ADCMD_FINISH -- abort writes in progress to audio channels

FUNCTION

ADCMD_FINISH is a command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct and there is a write (CMD_WRITE) in progress, ADCMD_FINISH aborts the current write immediately or at the end of the current cycle depending on the sync flag (ADIOF_SYNC_CYCLE). If the allocation key is incorrect ADCMD_FINISH returns an error (ADIOERR_NOALLOCATION). ADCMD_FINISH is synchronous and only replies (mn_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do not use ADCMD_FINISH in interrupt code at interrupt level 5 or higher.

INPUTS

mn_ReplyPort- pointer to message port that receives I/O request if the quick flag (IOF_QUICK) is clear
io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
io_Unit - bit map of channels to finish (bits 0 thru 3 correspond to channels 0 thru 3)
io_Command - command number for ADCMD_FINISH
io_Flags - flags, must be cleared if not used:
 IOF_QUICK - (CLEAR) reply I/O request
 ADIOF_SYNC_CYCLE- (CLEAR) finish immediately (SET) finish at the end of current cycle

ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels successfully finished (bits 0 thru 3 correspond to channels 0 thru 3)
io_Error - error number:
 0 - no error
 ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/ADCMD_FREE

audio.device/BeginIO/ADCMD_FREE

NAME ADCMD_FREE -- free audio channels for allocation

FUNCTION

ADCMD_FREE is a command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, ADCMD_FREE does the following:
- restores the channel to a known state (CMD_RESET),
- changes the channels allocation key, and
- makes the channel available for re-allocation.
If the channel is locked (ADCMD_LOCK) ADCMD_FREE unlocks it and clears the bit for the channel (io_Unit) in the lock I/O request. If the lock I/O request has no channel bits set ADCMD_FREE replies the lock I/O request, and checks if there are allocation requests (ADCMD_ALLOCATE) waiting for the channel.

Otherwise, ADCMD_FREE returns an error (ADIOERR_NOALLOCATION). ADCMD_FREE is synchronous and only replies (m_ReplyPort) if the quick flag (IOE_QUICK) is clear. Do not use ADCMD_FREE in interrupt code.

INPUTS

m_ReplyPort- pointer to message port that receives I/O request
if the quick flag (IOE_QUICK) is clear
io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
io_Unit - bit map of channels to free (bits 0 thru 3 correspond to channels 0 thru 3)
io_Command - command number for ADCMD_FREE
io_Flags - flags, must be cleared if not used:
IOE_QUICK - (CLEAR) reply I/O request
ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels successfully freed (bits 0 thru 3 correspond to channels 0 thru 3)
io_Error - error number:
0 - no error
ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/ADCMD_LOCK

audio.device/BeginIO/ADCMD_LOCK

NAME ADCMD_LOCK -- prevent audio channels from being stolen

FUNCTION

ADCMD_LOCK is a command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, ADCMD_LOCK locks the channel, preventing subsequent allocations (ADCMD_ALLOCATE or OpenDevice) from stealing the channel. Otherwise, ADCMD_LOCK returns an error (ADIOERR_NOALLOCATION) and will not lock any channels.

Unlike setting the precedence (ADCMD_SETPREC, ADCMD_ALLOCATE or OpenDevice) to maximum (ADALLOC_MAXPREC) which would cause all subsequent allocations to fail, ADCMD_LOCK causes all higher precedence allocations, even no-wait (ADIOE_NOWAIT) allocations, to wait until the channels are un-locked.

Locked channels can only be unlocked by freeing them (ADCMD_FREE), which clears the channel select bits (io_Unit). ADCMD_LOCK does not reply the I/O request (m_ReplyPort) until all the channels it locks are freed, unless a higher precedence allocation attempts to steal one of the locked channels. If a steal occurs, ADCMD_LOCK replies and returns an error (ADIOERR_CHANNELSTOLEN). If the lock is replied (m_ReplyPort) with this error, the channels should be freed as soon as possible. To avoid a possible deadlock, never make the freeing of stolen channels dependent on another allocations completion.

ADCMD_LOCK is only asynchronous if the allocation key is correct, in which case it clears the quick flag (IOE_QUICK); otherwise, it is synchronous and only replies if the quick flag (IOE_QUICK) is clear. Do not use ADCMD_LOCK in interrupt code.

INPUTS

m_ReplyPort- pointer to message port that receives I/O request
if the quick flag (IOE_QUICK) is clear
io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
io_Unit - bit map of channels to lock (bits 0 thru 3 correspond to channels 0 thru 3)
io_Command - command number for ADCMD_LOCK
io_Flags - flags, must be cleared
ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of successfully locked channels (bits 0 thru 3 correspond to channels 0 thru 3) not freed (ADCMD_FREE)
io_Flags - IOE_QUICK flag cleared if the allocation key is correct (noADIOERR_NOALLOCATION error)
io_Error - error number:
0 - no error
ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

ADIOERR_CHANNELSTOLEN- allocation attempting to steal locked channel

audio.device/BeginIO/ADCMD_PERVOL audio.device/BeginIO/ADCMD_PERVOL

NAME ADCMD_PERVOL -- change the period and volume for writes in progress to audio channels

FUNCTION

ADCMD_PERVOL is a command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct and there is a write (CMD_WRITE) in progress, ADCMD_PERVOL loads a new volume and period immediately or at the end of the current cycle depending on the sync flag (ADIOF_SYNC_CYCLE). If the allocation key is incorrect, ADCMD_PERVOL returns an error (ADIOERR_NOALLOCATION). ADCMD_PERVOL is synchronous and only replies (m_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do not use ADCMD_PERVOL in interrupt code at interrupt level 5 or higher.

INPUTS

- m_ReplyPort- pointer to message port that receives I/O request if the quick flag (IOF_QUICK) is clear
- io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
- io_Unit - bit map of channels to load period and volume (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Command - command number for ADCMD_PERVOL
- io_Flags - flags, must be cleared if not used:
 - IOF_QUICK - (CLEAR) reply I/O request
 - ADIOF_SYNC_CYCLE- (CLEAR) finish immediately
 - (SET) finish at the end of current cycle
- ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command
- ioa_Period - new sample period in 279.365 ns increments (127 thru 65536, anti-aliasing filter works below 300 to 500 depending on waveform)
- ioa_Volume - new volume (0 thru 64, linear)

OUTPUTS

- io_Unit - bit map of channels that successfully loaded period and volume (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Error - error number:
 - 0 - no error
 - ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/ADCMD_SETPREC audio.device/BeginIO/ADCMD_SETPREC

NAME ADCMD_SETPREC -- set the allocation precedence for audio channels

FUNCTION

ADCMD_SETPREC is a command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, ADCMD_SETPREC sets the allocation precedence to a new value (In_Pri) and checks if there are allocation requests (ADCMD_ALLOCATE) waiting for the channel which now have higher precedence; otherwise, ADCMD_SETPREC returns an error (ADIOERR_NOALLOCATION). ADCMD_SETPREC is synchronous and only replies (mm_ReplyPort) if the quick flag (IOE_QUICK) is clear. Do not use ADCMD_SETPREC in interrupt code.

INPUTS

In_Pri - new allocation precedence (-128 thru 127)
mm_ReplyPort - pointer to message port that receives I/O request
if the quick flag (IOE_QUICK) is clear
io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
io_Unit - bit map of channels to set precedence (bits 0 thru 3 correspond to channels 0 thru 3)
io_Command - command number for ADCMD_SETPREC
io_Flags - flags, must be cleared if not used:
IOE_QUICK - (CLEAR) reply I/O request
ioa_AllocKey - allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels that successfully set precedence (bits 0 thru 3 correspond to channels 0 thru 3)
io_Error - error number:
0 - no error
ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/ADCMD_WAITCYCLE

NAME ADCMD_WAITCYCLE -- wait for an audio channel to complete the current cycle of a write

FUNCTION

ADCMD_WAITCYCLE is a command for a single audio channel (io_Unit). If the allocation key (ioa_AllocKey) is correct and there is a write (CMD_WRITE) in progress on selected channel, ADCMD_WAITCYCLE does not reply (mm_ReplyPort) until the end of the current cycle. If there is no write in progress, ADCMD_WAITCYCLE replies immediately. If the allocation key is incorrect, ADCMD_WAITCYCLE returns an error (ADIOERR_NOALLOCATION). ADCMD_WAITCYCLE returns an error (IOERR_ABORTED) if it is canceled (AbortIO) or the channel is stolen (ADCMD_ALLOCATE). ADCMD_WAITCYCLE is only asynchronous if it is waiting for a cycle to complete, in which case it clears the quick flag (IOE_QUICK); otherwise, it is synchronous and only replies if the quick flag (IOE_QUICK) is clear. Do not use ADCMD_WAITCYCLE in interrupt code at interrupt level 5 or higher.

INPUTS

mm_ReplyPort - pointer to message port that receives I/O request, if the quick flag (IOE_QUICK) is clear, or if a write is in progress on the selected channel and a cycle has completed
io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
io_Unit - bit map of channel to wait for cycle (bits 0 thru 3 correspond to channels 0 thru 3), if more than one bit is set lowest bit number channel is used
io_Command - command number for CMD_WAITCYCLE
io_Flags - flags, must be cleared if not used:
IOE_QUICK - (CLEAR) reply I/O request
(SEI) only reply I/O request if a write is in progress on the selected channel and a cycle has completed
ioa_AllocKey - allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channel that successfully waited for cycle (bits 0 thru 3 correspond to channels 0 thru 3)
io_Flags - IOE_QUICK flag cleared if a write is in progress on the selected channel
io_Error - error number:
0 - no error
IOERR_ABORTED - canceled (AbortIO) or channel stolen
ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/CMD_CLEAR audio.device/BeginIO/CMD_CLEAR

NAME CMD_CLEAR -- throw away internal caches

FUNCTION

CMD_CLEAR is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, CMD_CLEAR does nothing; otherwise, CMD_CLEAR returns an error (ADIOERR_NOALLOCATION). CMD_CLEAR is synchronous and only replies (m_ReplyPort) if the quick flag (IOF_QUICK) is clear.

INPUTS

- m_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOF_QUICK) is clear
- io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
- io_Unit - bit map of channels to clear (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Command - command number for CMD_CLEAR
- io_Flags - flags, must be cleared if not used: IOF_QUICK - (CLEAR) reply I/O request
- ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

- io_Unit - bit map of channels successfully cleared (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Error - error number:
 - 0 - no error
- ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/CMD_FLUSH audio.device/BeginIO/CMD_FLUSH

NAME CMD_FLUSH -- cancel all pending I/O

FUNCTION

CMD_FLUSH is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, CMD_FLUSH aborts all writes (CMD_WRITE) in progress or queued and any I/O requests waiting to synchronize with the end of the cycle (ADCMD_WAITCYCLE); otherwise, CMD_FLUSH returns an error (ADIOERR_NOALLOCATION). CMD_FLUSH is synchronous and only replies (m_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do not use CMD_FLUSH in interrupt code at interrupt level 5 or higher.

INPUTS

- m_ReplyPort- pointer to message port that receives I/O request if the quick flag (IOF_QUICK) is clear
- io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
- io_Unit - bit map of channels to flush (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Command - command number for CMD_FLUSH
- io_Flags - flags, must be cleared if not used: IOF_QUICK - (CLEAR) reply I/O request
- ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

- io_Unit - bit map of channels successfully flushed (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Error - error number:
 - 0 - no error
- ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/CMD_READ

audio.device/BeginIO/CMD_READ

NAME CMD_READ -- normal I/O entry point

FUNCTION

CMD_READ is a standard command for a single audio channel (io_Unit). If the allocation key (io_AllocKey) is correct, CMD_READ returns a pointer (io_Data) to the I/O block currently writing (CMD_WRITE) on the selected channel; otherwise, CMD_READ returns an error (ADIOERR_NOALLOCATION). If there is no write in progress, CMD_READ returns zero. CMD_READ is synchronous and only replies (m_ReplyPort) if the quick bit (IOF_QUICK) is clear.

INPUTS

m_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOF_QUICK) is clear
 io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
 io_Unit - bit map of channel to read (bit 0 thru 3 corresponds to channel 0 thru 3), if more than one bit is set lowest bit number channel read
 io_Command - command number for CMD_READ
 io_Flags - flags, must be cleared if not used:
 IOF_QUICK - (CLEAR) reply I/O request
 io_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channel successfully read (bit 0 thru 3 corresponds to channel 0 thru 3)
 io_Error - error number:
 0 - no error
 ADIOERR_NOALLOCATION - allocation key (io_AllocKey) does not match key for channel
 io_Data - pointer to I/O block for current write, zero if none is progress

audio.device/BeginIO/CMD_RESET

audio.device/BeginIO/CMD_RESET

NAME CMD_RESET -- restore device to a known state

FUNCTION

CMD_RESET is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (io_AllocKey) is correct, CMD_RESET:
 . clears the hardware audio registers and attach bits,
 . sets the audio interrupt vector,
 . cancels all pending I/O (CMD_FLUSH), and
 . un-stops the channel if it is stopped (CMD_STOP).

Otherwise, CMD_RESET returns an error (ADIOERR_NOALLOCATION).

CMD_RESET is synchronous and only replies (m_ReplyPort) if the quick flag (IOF_QUICK) is clear. Do not use CMD_RESET in interrupt code at interrupt level 5 or higher.

INPUTS

m_ReplyPort- pointer to message port that receives I/O request if the quick flag (IOF_QUICK) is clear
 io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
 io_Unit - bit map of channels to reset (bits 0 thru 3 correspond to channels 0 thru 3)
 io_Command - command number for CMD_RESET
 io_Flags - flags, must be cleared if not used:
 IOF_QUICK - (CLEAR) reply I/O request
 io_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels to successfully reset (bits 0 thru 3 correspond to channels 0 thru 3)
 io_Error - error number:
 0 - no error
 ADIOERR_NOALLOCATION - allocation key (io_AllocKey) does not match key for channel

audio.device/BeginIO/CMD_START audio.device/BeginIO/CMD_START

NAME CMD_START -- start device processing (like 'Q')

FUNCTION

CMD_START is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct and the channel was previously stopped (CMD_STOP), CMD_START immediately starts all writes (CMD_WRITE) to the channel. If the allocation key is incorrect, CMD_START returns an error (ADIOERR_NOALLOCATION). CMD_START starts multiple channels simultaneously to minimize distortion if the channels are playing the same waveform and their outputs are mixed. CMD_START is synchronous and only replies (m_ReplyPort) if the quick flag (IOE_QUICK) is clear. Do not use CMD_START in interrupt code at interrupt level 5 or higher.

INPUTS

- m_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOE_QUICK) is clear
- io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
- io_Unit - bit map of channels to start (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Command - command number for CMD_START
- io_Flags - flags, must be cleared if not used: IOE_QUICK - (CLEAR) reply I/O request
- ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

- io_Unit - bit map of channels successfully started (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Error - error number:
 - 0 - no error
 - ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/CMD_STOP audio.device/BeginIO/CMD_STOP

NAME CMD_STOP -- stop device processing (like 'S')

FUNCTION

CMD_STOP is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, CMD_STOP immediately stops any writes (CMD_WRITE) in progress; otherwise, CMD_STOP returns an error (ADIOERR_NOALLOCATION). CMD_WRITE queues up writes to a stopped channel until CMD_START starts the channel or CMD_RESET resets the channel. CMD_STOP is synchronous and only replies (m_ReplyPort) if the quick flag (IOE_QUICK) is clear. Do not use CMD_STOP in interrupt code at interrupt level 5 or higher.

INPUTS

- m_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOE_QUICK) is clear
- io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
- io_Unit - bit map of channels to stop (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Command - command number for CMD_STOP
- io_Flags - flags, must be cleared if not used: IOE_QUICK - (CLEAR) reply I/O request
- ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

- io_Unit - bit map of channels successfully stopped (bits 0 thru 3 correspond to channels 0 thru 3)
- io_Error - error number:
 - 0 - no error
 - ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/CMD_UPDATE audio.device/BeginIO/CMD_UPDATE

NAME CMD_UPDATE -- force dirty buffers out

FUNCTION

CMD_UPDATE is a standard command for multiple audio channels. For each selected channel (io_Unit), if the allocation key (ioa_AllocKey) is correct, CMD_UPDATE does nothing; otherwise, CMD_UPDATE returns an error (ADIOERR_NOALLOCATION). CMD_UPDATE is synchronous and only replies (m_ReplyPort) if the quick flag (IOF_QUICK) is clear.

INPUTS

m_ReplyPort- pointer to message port that receives I/O request after if the quick flag (IOF_QUICK) is clear
 io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
 io_Unit - bit map of channels to update (bits 0 thru 3 correspond to channels 0 thru 3)
 io_Command - command number for CMD_UPDATE
 io_Flags - flags, must be cleared if not used:
 IOF_QUICK - (CLEAR) reply I/O request
 ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command

OUTPUTS

io_Unit - bit map of channels successfully updated (bits 0 thru 3 correspond to channels 0 thru 3)
 io_Error - error number:
 0 - no error
 ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey) does not match key for channel

audio.device/BeginIO/CMD_WRITE audio.device/BeginIO/CMD_WRITE

NAME CMD_WRITE -- normal I/O entry point

FUNCTION

CMD_WRITE is a standard command for a single audio channel (io_Unit). If the allocation key (ioa_AllocKey) is correct, CMD_WRITE plays a sound using the selected channel; otherwise, it returns an error (ADIOERR_NOALLOCATION). CMD_WRITE queues up requests if there is another write in progress or if the channel is stopped (CMD_STOP). When the write actually starts; if the ADIOF_PERVOL flag is set, CMD_WRITE loads volume (ioa_Volume) and period (ioa_Period), and if the ADIOF_WRITEMESSAGE flag is set, CMD_WRITE replies the write message (ioa_WriteMsg). CMD_WRITE returns an error (IOERR_ABORTED) if it is canceled (AbortIO) or the channel is stolen (ADCMD_ALLOCATE). CMD_WRITE is only asynchronous if there is no error, in which case it clears the quick flag (IOF_QUICK) and replies the I/O request (m_ReplyPort) after it finishes writing; otherwise, it is synchronous and only replies if the quick flag (IOF_QUICK) is clear. Do not use CMD_WRITE in interrupt code at interrupt level 5 or higher.

INPUTS

m_ReplyPort- pointer to message port that receives I/O request after the write completes
 io_Device - pointer to device node, must be set by (or copied from I/O block set by) OpenDevice function
 io_Unit - bit map of channel to write (bit 0 thru 3 corresponds to channel 0 thru 3), if more than one bit is set lowest bit number channel is written
 io_Command - command number for CMD_WRITE
 io_Flags - flags, must be cleared if not used:
 ADIOF_PERVOL - (SET) load volume and period
 ADIOF_WRITEMESSAGE - (SET) reply message at write start
 ioa_AllocKey- allocation key, must be set by (or copied from I/O block set by) OpenDevice function or ADCMD_ALLOCATE command
 ioa_Data - pointer to waveform array (signed bytes (-128 thru 127) in custom chip addressable ram and word aligned)
 ioa_Length - length of the wave array in bytes (2 thru 131072, must be even number)
 ioa_Period - sample period in 279.365 ns increments (127 thru 65536, anti-aliasing filter works below 300 to 500 depending on waveform), if enabled by ADIOF_PERVOL
 ioa_Volume - volume (0 thru 64, linear), if enabled by ADIOF_PERVOL
 ioa_Cycles - number of times to repeat array (0 thru 65535, 0 for infinite)
 ioa_WriteMsg- message replied at start of write, if enabled by ADIOF_WRITEMESSAGE

OUTPUTS

io_Unit - bit map of channel successfully written (bit 0 thru 3 corresponds to channel 0 thru 3)
 io_Flags - IOF_QUICK flag cleared if there is no error
 io_Error - error number:

0
 IOERR_ABORTED - no error
 - canceled (AbortIO) or channel
 stolen
 ADIOERR_NOALLOCATION - allocation key (ioa_AllocKey)
 does not match key for channel

BUSCS

If OMD_WRITE starts the write immediately after stopping a previous write, you must set the ADIOF_PERVOL flag or else the new data pointer (ioa_Data) and length (ioa_Length) may not be loaded.

audio.device/CloseDevice audio.device/CloseDevice

NAME
 CloseDevice - terminate access to the audio device

SYNOPSIS
 CloseDevice(iORequest);
 AI

FUNCTION

The CloseDevice routine notifies the audio device that it will no longer be used. It takes an I/O audio request block (IOAudio) and clears the device pointer (io.Device). If there are any channels allocated with the same allocation key (ioa_AllocKey), CloseDevice frees (ADCMD_FREE) them. CloseDevice decrements the open count, and if it falls to zero and an expunge (Expunge) is pending, the device is expunged.

INPUTS

iORequest - pointer to audio request block (struct IOAudio)
 io.Device - pointer to device node, must be set by (or copied from I/O block set by) open (OpenDevice)
 io.Unit - bit map of channels to free (ADCMD_FREE) (bits 0 thru 3)
 ioa_AllocKey - allocation key, used to free channels

OUTPUTS

iORequest - pointer to audio request block (struct IOAudio)
 io.Device - set to -1
 io.Unit - set to zero

audio.device/Expunge

audio.device/Expunge

NAME

EXPUNGE - indicate a desire to remove the Audio device

FUNCTION

The Expunge routine is called when a user issues a RemDevice call. By the time it is called, the device has already been removed from the device list, so no new opens will succeed. The existence of any other users of the device, as determined by the device open count being non-zero, will cause the Expunge to be deferred. When the device is not in use, or no longer in use, the Expunge is actually performed.

audio.device/OpenDevice

audio.device/OpenDevice

NAME

OpenDevice - open the audio device

SYNOPSIS

error = OpenDevice("audio.device", unitNumber, IORequest, flags);

FUNCTION

The OpenDevice routine grants access to the audio device. It takes an I/O audio request block (IORequest) and if it can successfully open the audio device, it loads the device pointer (io_Device) and the allocation key (io_AllocKey); otherwise, it returns an error (IOERR_OPENFAIL). OpenDevice increments the open count keeping the device from being expunged (Expunge). If the length (ioa_Length) is non-zero, OpenDevice tries to allocate (ADCMD_ALLOCATE) audio channels from a array of channel combination options (ioa_Data). If the allocation succeeds, the allocated channel combination is loaded into the unit field (ioa_Unit); otherwise, OpenDevice returns an error (ADIOERR_ALLOCFAILED). OpenDevice does not wait for allocation to succeed and closes (CloseDevice) the audio device if it fails. To allocate channels, OpenDevice also requires a properly initialized reply port (m_ReplyPort) with an allocated signal bit.

INPUTS

unitNumber - not used
 IORequest - pointer to audio request block (struct IOAudio)
 in_Pri - allocation precedence (-128 thru 127), only necessary for allocation (non-zero length)
 m_ReplyPort - pointer to message port for allocation, only necessary for allocation (non-zero length)
 ioa_Data - pointer to channel combination options (byte array, bits 0 thru 3 correspond to channels 0 thru 3), only necessary for allocation (non-zero length)
 ioa_Length - length of the channel combination option array (0 thru 16), zero for no allocation
 flags - not used

OUTPUTS

IORequest - pointer to audio request block (struct IOAudio)
 io_Device - pointer to device node if OpenDevice succeeds, otherwise -1
 io_Unit - bit map of successfully allocated channels (bits 0 thru 3 correspond to channels 0 thru 3) if allocation, otherwise 0
 io_Error - error number:
 0 - no error
 IOERR_OPENFAIL - open failed
 ADIOERR_ALLOCFAILED - allocation failed, no open
 ioa_AllocKey - unique allocation key, if OpenDevice succeeds
 error - copy of io_Error

Dec 3 17:04 1985 audio.doc Page 25

TABLE OF CONTENTS

clipboard.device/BeginIO
clipboard.device/Close
clipboard.device/CurrentReadID
clipboard.device/CurrentWriteID
clipboard.device/Expunge
clipboard.device/Open
clipboard.device/Post
clipboard.device/Read
clipboard.device/Reset
clipboard.device/Update
clipboard.device/Write

clipboard.device/BeginIO

clipboard.device/BeginIO

NAME

BeginIO - initiate clipboard device IO

SYNOPSIS

SendIO(iOrequest), sysBase
DoIO(iOrequest), sysBase

FUNCTION

The BeginIO is the workhorse device function used to initiate device commands. It can be called directly or via the exec library functions SendIO and DoIO.

clipboard.device/Close

clipboard.device/Close

NAME

Close - terminate access to the clipboard device

SYNOPSIS

CloseDevice(iOrequest), sysBase

FUNCTION

The close routine notifies the clipboard device that the iOrequest will no longer be used.

clipboard.device/CurrentReadID

clipboard.device/CurrentReadID

NAME

CurrentReadID - determines the current read identifier.

FUNCTION

CurrentReadID fills the io_ClipID with a clip identifier that can be compared with that of a post command: if greater than the post identifier then the post data held privately by an application is not valid for its own pasting.

IO REQUEST

io_Message mm_ReplyPort set up
io_Device preset by OpenDevice
io_Unit preset by OpenDevice
io_Command CMD_CLIPREADID

io_ClipID

the ClipID of the current write is set

clipboard.device/CurrentWriteID

clipboard.device/CurrentWriteID

NAME CurrentWriteID - determine the current write identifier.

FUNCTION CurrentWriteID fills the io.ClipID with a clip identifier that can be compared with that of a post command: if greater than the post identifier then the post is obsolete and need never be satisfied.

IO REQUEST io_Message mn_ReplyPort set up
io_Device preset by OpenDevice
io_Unit preset by OpenDevice
io_Command CMD_CLIPWRITEID

io.ClipID the ClipID of the current write is set

clipboard.device/Expunge

clipboard.device/Expunge

NAME Expunge - indicate a desire to remove the clipboard device

SYNOPSIS <Expunge is not generally called by application programs>

FUNCTION The Expunge routine is called when the system needs the memory used by the clipboard device, and the clipboard device has no open units. The clipboard device is removed from memory until next needed (i.e. until the next OpenDevice("clipboard.device", ...));

clipboard.device/Open

clipboard.device/Open

NAME

Open - a request to open the clipboard device

SYNOPSIS

OpenDevice("clipboard.device", unit, iOrequest, 0), sysBase

FUNCTION

The open routine grants access to a device. There are two fields in the iOrequest block that will be filled in: io_Device and io_Unit.

A successful OpenDevice call must be matched by a CloseDevice call when access to the device is no longer needed.

RESULTS

If the open was unsuccessful, OpenDevice returns a non-zero result and the iOrequest is not valid.

clipboard.device/Post

clipboard.device/Post

NAME

Post - post clip to clipboard

FUNCTION

IO REQUEST

| | |
|------------|---------------------------------|
| io_Message | mm_ReplyPort set up |
| io_Device | presert by OpenDevice |
| io_Unit | presert by OpenDevice |
| io_Command | CB0_F0ST |
| io_Data | pointer to satisfy message port |
| io_ClipID | zero |

clipboard.device/Read

clipboard.device/Read

NAME Read - read clip from clipboard

FUNCTION

IO REQUEST
 io_Message
 io_Device
 io_Unit
 io_Command
 io_Length
 io_Data
 io_Offset
 io_ClipID

m_ReplyPort set up
 preset by OpenDevice
 preset by OpenDevice
 CMD_READ
 number of bytes in data buffer
 pointer to buffer of data to fill
 zero if this is the initial read, Offset of
 the SeekType if successful SeekType performed
 zero if this is the initial read

clipboard.device/Reset

clipboard.device/Reset

NAME Reset - reset the clipboard

FUNCTION

Reset resets the clipboard device without destroying handles
 to the open device.

IO REQUEST
 io_Message
 io_Device
 io_Command
 io_Flags

m_ReplyPort set up
 preset by OpenDevice
 CMD_RESET
 IOB_QUICK set if quick I/O is possible

clipboard.device/Update

clipboard.device/Update

NAME Update - terminate the writing of a cut to the clipboard

FUNCTION

Indicate to the clipboard that the previous write commands are complete and can be used for any pending pastes (reads). This command cannot be issued while any of the write commands are pending.

IO REQUEST

io_Message m_ReplyPort set up
io_Device preset by OpenDevice
io_Unit preset by OpenDevice
io_Command CMD_UPDATE
io_ClipID the ClipID of the write

clipboard.device/Write

clipboard.device/Write

NAME Write - write clip to clipboard

FUNCTION

IO REQUEST

io_Message m_ReplyPort set up
io_Device preset by OpenDevice
io_Unit preset by OpenDevice
io_Command CMD_WRITE
io_Length number of bytes to process
io_Data pointer to block of data to process
io_Offset zero if this is the initial write
io_ClipID zero if this is the initial write, ClipID of the Post if this is to satisfy a post

Dec 3 17:04 1985 clipboard.doc Page 13

TABLE OF CONTENTS

console.device/CDAskKeyMap
console.device/CDInputHandler
console.device/CDSetKeyMap
console.device/Clear
console.device/OpenDevice
console.device/RawKeyConvert
console.device/Read
console.device/Write

console.device/CDAskKeyMap console.device/CDAskKeyMap

NAME

AskKeyMap - get the current key map structure for this console

FUNCTION

Fill the IO_DATA buffer with the current KeyMap structure in use by this console unit.

IO REQUEST

io_Message m_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Unit preset by the call to OpenDevice
io_Command CD_ASKEYMAP
io_Flags IOF_QUICK if quick I/O possible, else zero
io_Length sizeof(*keyMap)
io_Data struct KeyMap *keyMap
 eight longwords to describe the raw keycodes
 to byte stream conversion.

RESULTS

This function sets the error field in the ioRequest, and fills the structure at IO_DATA with the current key map.

console.device/CDInputHandler

console.device/CDInputHandler

NAME

CDInputHandler - handle an input event for the console device

SYNOPSIS

CDInputHandler (events, consoleDev)
A0 AI

FUNCTION

Accept input events from the producer, which is usually the
rom input.task.

NOTES

This function is different from standard device commands in
that it is a function in the console device library vectors.
The "OpenLibrary" call for the console device is to
OpenDevice("console.device", -1, iORquest, 0), and then grab
the io_Device field out of the iORquest as the library
vector.

console.device/CDSetKeyMap

console.device/CDSetKeyMap

NAME

SetKeyMap - set the current key map structure for this console

FUNCTION

Set the current KeyMap structure used by this console unit to
the structure pointed to by IO_DATA

IO REQUEST

io_Message mn_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Unit CD_SETKEYMAP
io_Command IOF_QUICK if quick I/O possible, else zero
io_Flags sizeof(*keyMap)
io_Length struct KeyMap *keyMap
io_Data eight longwords that describe the raw keycode
to byte stream conversion.

RESULTS

This function sets the error field in the iORquest, and fills
the current key map from IO_DATA.

console.device/Clear

console.device/Clear

NAME Clear - clear console input buffer

FUNCTION

Remove from the input buffer any reports waiting to satisfy read requests.

IO REQUEST

- io_Message
- io_Device
- io_Unit
- io_Command
- io_Flags

mn_ReplyPort set if quick I/O is not possible
 preset by the call to OpenDevice
 CMD_CLEAR preset by the call to OpenDevice
 IOB_QUICK set if quick I/O is possible

console.device/OpenDevice

console.device/OpenDevice

NAME OpenDevice - a request to open a Console device

SYNOPSIS

OpenDevice("console.device", unit, ioRequest, 0)

FUNCTION

The open routine grants access to a device. There are two fields in the ioRequest block that will be filled in: the IO_DEVICE field and possibly the IO_UNIT field.

This open command differs from most other device open commands in that it requires some information to be supplied in the IO_DATA field of the ioRequest block. This initialization information supplies the window that is used by the console device for output.

The unit number that is a standard parameter for an open call is used specially by this device. A unit of -1 indicates that no actual console is to be opened, and is used to get a pointer to the device library vector. A unit of zero binds the supplied window to a unique console. Sharing a console must be done at a level higher than the device. There are no other valid unit numbers.

IO REQUEST

io_Data

struct Window *window

This is the window that will be used for this console. It must be supplied if the unit in the OpenDevice call is 0 (see above). The RPort of this window is potentially in use by the console whenever there is an outstanding write command.

console.device/RawKeyConvert

console.device/RawKeyConvert

NAME RawKeyConvert - decodes raw input classes

SYNOPSIS

actual = RawKeyConvert(event, buffer, length, keyMap),
consoleDev
D0 A6 A0 A1 D1 A2

FUNCTION

This console function converts input events of type IECLASS_RAWKEY to ANSI bytes, based on the keyMap, and places the result into the buffer.

INPUTS

event - an InputEvent structure pointer.
buffer - a byte buffer large enough to hold all anticipated characters generated by this conversion.
length - maximum anticipation, i.e. the buffer size in bytes.
keyMap - a KeyMap structure pointer, or null if the default console device key map is to be used.
consoleDev - the io_Device of the console device.

RESULTS

actual - the number of characters in the buffer, or -1 if a buffer overflow was about to occur.

ERRORS

if actual is -1, a buffer overflow condition was detected.
Not all of the characters in the buffer are valid.

NOTES

This function is different from standard device commands in that it is a function in the console device library vectors. The "OpenLibrary" call for the console device is to OpenDevice("console.device", -1, IORequest, 0), and then grab the io_Device field out of the IORequest as the library vector.

console.device/Read

console.device/Read

NAME

Read - return the next input from the keyboard

FUNCTION

Read the next input, generally from the keyboard. The form of this input is as an ANSI byte stream: i.e. either ASCII text or control sequences. Raw input events received by the console device can be selectively filtered via the SRE and RRE control sequences (see the write command). Keys are converted via the keymap associated with the unit, which is modified with AskKeyMap and SetKeyMap

If, for example, raw keycodes had been enabled by writing <CSI>is to the console (where <CSI> is \$9B or Esc[]), keys would return raw keycode reports with the information from the input event itself, in the form: <CSI>1;0;<keycode>;<modifiers>;0;0;<seconds>;<microseconds>q

If there is no pending input, this command will not be satisfied, but if there is some input, but not as much as can fill IO_LENGTH, the request will be satisfied with the input currently available.

IO REQUEST

io_Message m_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Unit preset by the call to OpenDevice
io_Command CMD_READ
io_Flags IOF_QUICK if quick I/O possible, else zero
io_Length sizeof(*buffer)
io_Data char buffer[]
The destination for the characters to read from the keyboard.

RESULTS

This function sets the error field in the IORequest, and fills the IORequest IO_DATA area with the next input, and IO_ACTUAL with the number of bytes read.

console.device/write console.device/write

NAME Write - write text to the display

FUNCTION Write a text record to the display. Note that the RPort of the console window is in use while this write command is pending.

IO REQUEST
 io_Message mn_replyPort set if quick I/O is not possible
 io_Device preset by the call to OpenDevice
 io_Unit preset by the call to OpenDevice
 io_Command CMD_WRITE
 io_Flags IOF_QUICK if quick I/O possible, else zero
 io_Length sizeof(buffer)
 io_Data char buffer[]
 a buffer containing the ANSI text to write to the console device.

ANSI CODES SUPPORTED

| Code | Name | Definition |
|-------|------|-----------------|
| 00/ 8 | BS | BACKSPACE |
| 00/10 | LF | LINE FEED |
| 00/11 | VT | VERTICAL TAB |
| 00/12 | FF | FORM FEED |
| 00/13 | CR | CARRIAGE RETURN |
| 00/14 | SO | SHIFT OUT |
| 00/15 | SI | SHIFT IN |
| 01/11 | ESC | ESCAPE |

| Code or Esc | Name | Definition |
|-------------|------|---|
| 08/ 4 | D | IND INDEX: move the active position down one line |
| 08/ 5 | E | NEL NEXT LINE. |
| 08/13 | M | RI REVERSE INDEX: |
| 09/11 | I | CSI CONTROL SEQUENCE INTRODUCER: see next list |

ISO Compatible Escape Sequences (introduced by Esc) --

| Esc | Name | Definition |
|-----|------|---|
| a | INT | INTERRUPT (will not be supported later) |
| c | RIS | RESET TO INITIAL STATE |

Control Sequences (introduced by CSI, i.e. \$9B or Esc[]) with parameters: "1" is an optional numeric parameter. "2" are two numeric parameters - e.g. '14;94', and "3" is any number of numeric parameters, separated by semicolons --

Esc[Name Definition
 1@ ICH INSERT CHARACTER

| | | |
|-----|-------|--|
| 1A | CUU | CURSOR UP |
| 1B | CUD | CURSOR DOWN |
| 1C | CUF | CURSOR FORWARD |
| 1D | CUB | CURSOR BACKWARD |
| 1E | CNL | CURSOR NEXT LINE |
| 1F | CPL | CURSOR PRECEDING LINE |
| 2H | CUP | CURSOR POSITION |
| 1J | ED | ERASE IN DISPLAY (only to end of display) |
| 1K | EL | ERASE IN LINE (only to end of line) |
| 1L | IL | INSERT LINE |
| 1M | DL | DELETE LINE |
| 1P | DCH | DELETE CHARACTER |
| 2R | CPR | CURSOR POSITION REPORT (in Read stream only) |
| 1S | SU | SCROLL UP |
| 1T | SD | SCROLL DOWN |
| 3h | SM | SET MODE |
| 3l | RM | RESET MODE |
| 3m | SCR | SELECT GRAPHIC RENDITION |
| 1n | DSR | DEVICE STATUS REPORT |
| 1t | asLPP | SET PAGE LENGTH (private Amiga sequence) |
| 1u | asLL | SET LINE LENGTH (private Amiga sequence) |
| 1x | asLO | SET LEFT OFFSET (private Amiga sequence) |
| 1y | asTO | SET TOP OFFSET (private Amiga sequence) |
| 3{ | asRE | SET RAW EVENTS (private Amiga sequence) |
| 6 | aIER | INPUT EVENT REPORT (private Amiga Read sequence) |
| 3} | aARR | RESET RAW EVENTS (private Amiga sequence) |
| 1~ | aSCR | SET CURSOR RENDITION (private Amiga sequence) |
| 1 p | aMSR | SPECIAL KEY REPORT (private Amiga sequence) |
| 0 q | aMSR | WINDOW STATUS REQUEST (private Amiga sequence) |
| 4 r | aMSR | WINDOW BOUNDS REPORT (private Amiga Read sequence) |

Dec 3 17:04 1985 console.doc Page 11

TABLE OF CONTENTS

narrator.device/AbortIO
narrator.device/Close
narrator.device/Flush
narrator.device/Open
narrator.device/Read
narrator.device/Reset
narrator.device/Start
narrator.device/Write

narrator.device/AbortIO narrator.device/AbortIO

NAME AbortIO - Abort an IO request

SYNOPSIS AbortIO(IOrequest)

FUNCTION Aborts a speech IO request. The request may be in the queue or currently active.

INPUTS IOORB of request to abort.

RESULTS io_Error field of IOORB set to IOERR_ABORTED

SEE ALSO

narrator.device/Close

narrator.device/Close

NAME

CloseDevice - terminates access to the narrator device

SYNOPSIS

CloseDevice(IORequest)

FUNCTION

Close invalidates the IO_UNIT and IO_DEVICE fields in the IOCB, preventing subsequent IO until another OpenDevice. CloseDevice also reduces the open count. If the count goes to 0 and the expurge bit is set, the device is expurged. If the open count goes to zero and the delayed expurge bit is not set, CloseDevice sets the expurge bit.

INPUTS

IORequest block

RESULTS

IORequest block with unit and device pointers invalidated.

SEE ALSO

narrator.device/Flush

narrator.device/Flush

NAME

Flush - Aborts all inprogress and queued requests

SYNOPSIS

Standard device command. See DoIO/SendIO

FUNCTION

Aborts all inprogress and queued speech requests.

INPUTS

io_Command - CMD_FLUSH

RESULTS

SEE ALSO

narrator.device/Open

narrator.device/Open

NAME

OpenDevice - open the narrator device.

SYNOPSIS

error = OpenDevice("narrator.device", 0, IORrequest, 0);

FUNCTION

The OpenDevice routine grants access to the narrator device. OpenDevice checks the unit number, and if non-zero, returns an error (ND_UnitErr). If this is the first time the driver has been opened, OpenDevice will attempt to open the audio device and allocate the driver's static buffers. If either of these operations fail, an error is returned (see the .hi files for possible error return codes). Next, OpenDevice (done for all opens, not just the first one) initializes the user's IORrequest block (IORB). Default values for sex, rate, pitch, pitch mode, sampling frequency, and mouths are set in the appropriate fields of the IORB. Note that if users wish to use non-default values for these parms, the values must be set after the open is done. OpenDevice then assigns a pseudo-unit number to the IORB for use in synchronizing read and write requests. See the .read command for more details. Finally, OpenDevice stores the device mode pointer in the IORB and clears the delayed expunge bit.

INPUTS

deviceName - must be "narrator.device"
unitNumber - must be 0
IORrequest - the user's IORB (need not be initialized)
flags - not used

RESULTS

IORB fields set:
rate - 150 words/minute
pitch - 110 Hz
mode - Natural
sex - Male
mouths - Off
sampfreq - 22200
volume - 64 (max)

error - same as io_Error field of IORB

SEE ALSO

Write command.

narrator.device/Read

narrator.device/Read

NAME

Read - Return the next different mouth shape from an associated write

SYNOPSIS

Standard device command. See DoIO/SendIO.

FUNCTION

The read command of the narrator device returns mouth shapes to the user. The shape returned is guaranteed to be different from the previously returned shape (allowing updating to be done only when something has changed). Each read request is associated with a write request by the pseudo-unit number assigned by the OpenDevice call. Since the first structure in the read-mouth IORB is a narrator (write) IORB, this association is easily made by copying the narrator IORB into the narrate_rb field of the read IORB. See the .hi files. If there is no write in progress or in the device input queue with the same pseudo-unit number as the read request, the read will be returned to the user with an error. This is also how the user knows that the write request has finished and that s/he should not issue any more reads. Note that in this case the mouth shapes may not be different from previously returned values.

INPUTS

IORB with the narrator_rb structure copied from the associated write request except for:

io_Message - message port for read request
io_Command - CMD_READ
io_Error - 0
width - 0
height - 0

RESULTS

IORB fields set:
width - mouth width in millimeters/3.67
(division done for scaling)
height - mouth height in millimeters
shape - compressed form of mouth shapes
(internal use only)

SEE ALSO

Write command.

narrator.device/Reset

narrator.device/Reset

NAME

Reset - Reset the device to a known state

SYNOPSIS

Standard device command. See DoIO/SendIO.

FUNCTION

Resets the device as though it has just be initialized.
Aborts all read/write requests whether active of enqueued.
Restarts device if it has been stopped.

INPUTS

io_Command = CMD_RESET

RESULTS

SEE ALSO

narrator.device/Start

narrator.device/Start

NAME

Stop - Stops the device.
Start - Restarts the device after Stop

SYNOPSIS

Standard device commands. See DoIO/SendIO

FUNCTION

StartIO halts the currently active speech (if any) and prevents any queued requests from starting.

StartIO restarts the currently active speech (if any) and allows queued request to start.

INPUTS

io_Command = CMD_STOP or CMD_START

RESULTS

SEE ALSO

narrator.device/Write narrator.device/Write

NAME Write - Send speech request to the narrator device

SYNOPSIS Standard device command. See DoIO/SendIO.

FUNCTION Performs the speech request. If there is an associated read request on the device input queue, write will remove it and return an initial mouth shape to the user. Note that if you are going to be doing reads, the mouths parameter must be set to 1.

INPUTS

- Narrator IORB
- ch_masks - array of audio channel selection masks (see audio device documentation for description of this field)
- rm_masks - number of audio channel selection masks
- mouths - 0 if no mouths are desired
1 if mouths are to be read
- rate - speaking rate
- pitch - pitch
- mode - pitch mode
0 if natural mode
1 if robotic mode
- sex - 0 if male
1 if female
- io_Message - message port
- io_Command - CMD_WRITE
- io_Data - input string
- io_Length - length of input string

RESULTS

The function sets the io_Error field of the IORB. The io_Actual field is set to the length of the input string that was actually processed. If the return code indicates a phoneme error (ND_PhonErr), io_Actual is the position in the input string where the error occurred.

SEE ALSO

- Read command.
- Audio device documentation.

TABLE OF CONTENTS

parallel.device/AbortIO
parallel.device/BeginIO
parallel.device/Clear
parallel.device/Close
parallel.device/Flush
parallel.device/Open
parallel.device/Query
parallel.device/Read
parallel.device/Reset
parallel.device/SetParams
parallel.device/Start
parallel.device/Stop
parallel.device/Write

parallel.device/AbortIO

parallel.device/AbortIO

NAME

AbortIO -- abort the specified I/O request

FUNCTION

This function aborts the specified read or write request. If the request is active, it is stopped immediately. If the request is queued, it is painlessly removed.

INPUTS

ioRequest -- pointer to the IOreqst Block that is to be aborted.

RESULTS

Error -- if the Abort succeeded, then Error will be #IOERR_ABORTED (-2) and the request will be flagged as aborted (bit 5 of io_Flags set). If the Abort failed, then the Error will be zero.

Parallel.device/BeginIO

parallel.device/BeginIO

NAME BeginIO -- start up an I/O process

FUNCTION

This function initiates a I/O request made to the parallel device. Other than read or write, the functions are performed synchronously, and do not depend on any interrupt handling logic (or it's associated discontinuities), and hence, if so selected, can be performed as IO_QUICK. Reads and writes are merely initiated by BeginIO, and thusly return to the caller as begun, not completed. Completion is signalled via the standard ReplyMsg routine. A valid read or write request is performed asynchronously, and never as IO_QUICK. Multiple requests are handled via FIFO queuing.

INPUTS

ioRequest -- pointer to an I/O Request Block of size ioExtParSize (see parallel.i for size/definition), containing a valid function in io.Command to process, as well as the function's other required parameters. deviceNode -- pointer to the "parallel.device" node built at init, and put into io.Device at Open.

RESULTS

Error -- if the BeginIO succeeded, then Error will be null. If the BeginIO failed, then the Error will be non-zero. Most I/O errors won't be reported until the ReplyMsg.

parallel.device/Clear

parallel.device/Clear

NAME Clear -- clear the parallel port buffer

FUNCTION

This function just RTS's (no buffer to clear)

IO REQUEST

io_Message mn_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command CMD_CLEAR

RESULTS

Error -- none

parallel.device/Close parallel.device/Close

NAME

Close -- close the parallel port

SYNOPSIS

CloseDevice(deviceNode)

FUNCTION

This function closes software access to the parallel device.

INPUTS

deviceNode - pointer the device node, set by Open

SEE ALSO

parallel.device/Open

parallel.device/Flush

parallel.device/Flush

NAME

Flush -- clear all queued I/O requests for the parallel port

FUNCTION

This function purges the read and write request queues for the parallel device.

I/O REQUEST

io_Message mm_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command CMD_FLUSH

RESULTS

Error -- if the Flush succeeded, then Error will be null.
 If the Flush failed, then the Error will be non-zero.

parallel.device/Open parallel.device/Open

NAME Open -- a request to open the parallel port
SYNOPSIS OpenDevice(parname, unit, ioRequest, flags)
FUNCTION

This function allows the requestor software access to the parallel device. Unless the shared-access bit (bit 5 of ioParFlags) is set, exclusive use is granted and no other access is allowed until the owner closes the device. OpenDevice initializes the io_Device and io_Unit fields to 0, since there is only one parallel device/unit.

INPUTS

parname - pointer to literal string "parallel.device"
unit - ignored
ioRequest - pointer to an ioRequest block of size io_ExtParSize (see parallel.i for size/definition) to be initialized by the Open routine.
NOTE use of io_ParFlags (see FUNCTION above)
flags - ignored

RESULTS

D0 -- pointer to the device node
Error -- if the Open succeeded, then Error will be null. If the Open failed, then the Error will be non-zero.

parallel.device/Query

parallel.device/Query

NAME Query -- query parallel port/line status
FUNCTION

This function return the status of the parallel port lines and registers.

IO REQUEST
io_Message mm_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command PDCMD_QUERY (0A)

RESULTS

io_Status BIT ACTIVE FUNCTION
0 low printer selected
1 low paper out
2 low printer in busy toggle
3 read=0, write=1
4-7 reserved

parallel.device/Read

parallel.device/Read

NAME Read -- read input from parallel port

FUNCTION

This function causes a stream of characters to be read from the parallel I/O register. The number of characters is specified in io_Length, unless -1 is used, in which case input is read until an EOF is read (currently 0x00). If no read request has been made, pending input (i.e. handshake request) is not acknowledged.

IO REQUEST

- io_Message mn_ReplyPort initialized
- io_Device set by OpenDevice
- io_Unit set by OpenDevice
- io_Command CMD_READ
- io_Flags IOF_QUICK if quick I/O possible and desired
- io_Length number of characters to receive, or if set to -1 receives until EOF read in
- io_Data pointer where to put the data.

RESULTS

Error -- if the Read succeeded, then Error will be null.
If the Read failed, then the Error will be non-zero.

SEE ALSO

parallel.device/BeginIO, parallel.device/SetParams

parallel.device/Reset

parallel.device/Reset

NAME Reset -- reinitializes the parallel port

FUNCTION

This function resets the parallel port to its freshly initialized condition. It aborts all I/O requests both queued and current and sets the port's flags and parameters to their boot-up time default values.

IO REQUEST

- io_Message mn_ReplyPort initialized
- io_Device set by OpenDevice
- io_Unit set by OpenDevice
- io_Command CMD_RESET

RESULTS

Error -- if the Reset succeeded, then Error will be null.
If the Reset failed, then the Error will be non-zero.

parallel.device/SetParams

parallel.device/SetParams

NAME

SetParams -- change parameters for the parallel port

FUNCTION

This function allows the caller to change parameters for the parallel port. It will disallow changes if any reads or writes are active or queued. The EofMode bit of io_SerFlags can be set/reset without a call to SetParams. The Shared bit of io_SerFlags pertains to OpenDevice calls only. ALL OTHER PARAMETERS CAN ONLY BE CHANGED BY THE SETPARAMS FUNCTION. (!!!!)

IO REQUEST

io_Message mn_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command PFORM_SETPARAMS (09)

NOTE

- that the following fields are filled by Open to reflect the parallel device's current configuration.

io_PExtFlags not used in Vi.1 (MUST be set to zero)
io_ParFlags see definition in parallel.i or parallel.h

NOTE

- that x00 yields exclusive access, termarray inactive.

io_PTermArray ASCII descending-ordered 8-byte array of termination characters. If less than 8 chars used, fill out array w/lowest valid value.

Terminators are used only if EOFMODE bit of io_ParFlags is set. (e.g. x512F0403030303) This field is filled on OpenDevice only if the EOFMODE bit is set.

RESULTS

Error -- If the SetParams succeeded, then Error will be null.
If the SetParams failed, then the Error will be non-zero.

parallel.device/Start

parallel.device/Start

NAME

Start -- restart paused I/O over the parallel port

FUNCTION

This function restarts the current I/O activity on the parallel port by reactivating the handshaking sequence.

IO REQUEST

io_Message mn_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command CMD_START

RESULTS

Error -- If the Start succeeded, then Error will be null.
If the Start failed, then the Error will be non-zero.

SEE ALSO

parallel.device/Stop

parallel.device/Stop

parallel.device/Stop

NAME

Stop -- pause current activity on the parallel port

FUNCTION

This function halts the current I/O activity on the parallel device by discontinuing the handshaking sequence.

IO REQUEST

io_Message mm_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command CMD_STOP

RESULTS

Error -- If the Stop succeeded, then Error will be null.
If the Stop failed, then the Error will be non-zero.

SEE ALSO

parallel.device/Start

parallel.device/Write

parallel.device/Write

NAME

Write -- send output to parallel port

FUNCTION

This function causes a stream of characters to be written to the parallel output register. The number of characters is specified in io_Length, unless -1 is used, in which case output is sent until an EOF is encountered (currently 0x00).

IO REQUEST

io_Message mm_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command CMD_WRITE
io_Flags IOF_QUICK if quick I/O possible and desired
io_Length number of characters to transmit, or if set to -1 send until EOF encountered
io_Data pointer to block of data to transmit

RESULTS

Error -- If the Write succeeded, then Error will be null.
If the Write failed, then the Error will be non-zero.

SEE ALSO

parallel.device/BeginIO, parallel.device/SetParams

Dec 3 17:04 1985 parallel.doc Page 15

TABLE OF CONTENTS

pr-inter.device/DumpRPort
 pr-inter.device/Flush
 pr-inter.device/Invalid
 pr-inter.device/PrintCommand
 pr-inter.device/RawWrite
 pr-inter.device/Reset
 pr-inter.device/Start
 pr-inter.device/Stop
 pr-inter.device/Write

pr-inter.device/DumpRPort

pr-inter.device/DumpRPort

NAME

DumpRPort - dump the specified RastPort to a graphics printer.

FUNCTION

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

IO REQUEST

io_Message mm_ReplyPort set if quick I/O is not possible
 io_Command PRD_DUMP_RPORT
 io_Flags IOB_QUICK set if quick I/O is possible
 io_RastPort ptr to a RastPort.
 io_ColorMap ptr to a ColorMap.
 io_Modes the 'modes' flag as from a ViewPort structure
 the upper word is reserved and should be zero
 io_SrcX the x offset into the RastPort
 io_SrcY the y offset into the RastPort
 io_SrcWidth the x size in the RastPort to be printed
 io_SrcHeight the y size in the RastPort to be printed
 io_DestCols
 io_DestRows these two parameters describe the size of the area to print to on the printer, as described below.

io_Special

Interpretation of Dest parameters:
 If SPECIAL_MIL is set, then the associated parameter is specified in thousandths of an inch on the printer.
 If SPECIAL_FULL is set, then the dimension is set to the maximum possible (as determined by the printer limits or the configuration limits, whichever is less).
 If SPECIAL_FRAC is set, the parameter is taken to be a longword binary fraction of the maximum for that dimension.
 If ASPECT is set, one of the dimensions may be reduced to preserve the aspect ratio of the print.
 If all bits for a dimension are clear, the parameter is specified in printer pixels.

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

HINTS

The printer selected in preferences must have graphics capability to use this command.
 Color printers may not be able to print black and white or greyscale pictures -- specifically, the Okimate 20 cannot print these with a color ribbon: use a black one.

If the printer has an input buffer option, use it.
If the printer can be uni- or bi-directional, select uni-directional.

printer.device/Flush

printer.device/Flush

NAME

Flush - abort all I/O requests (immediate)

FUNCTION

Flush aborts all stopped I/O at the unit.

IO REQUEST

io_Message
io_Device
io_Command
io_Flags

no_ReplyPort set if quick I/O is not possible
preset by the call to OpenDevice
CMD_FLUSH
IOB_QUICK set if quick I/O is possible

printer.device/Invalid printer.device/Invalid

NAME Invalid - invalid command

FUNCTION

Invalid is always an invalid command, and sets the device error appropriately.

IO REQUEST

io_Message mn_ReplyPort set if quick I/O is not possible
io_Command CMD_INVALID
io_Flags IOB_QUICK set if quick I/O is possible

printer.device/PrtCommand printer.device/PrtCommand

NAME PCPrCommand -- send a command to the printer

FUNCTION

This function sends a command to either the parallel or serial device. The printer device maps this command to the control code set of the current printer. The commands supported can be found with the printer.device/write command. All printers may not support all functions.

IO REQUEST IOPrCommand
io_Message mn_ReplyPort set
io_Device preset by OpenDevice
io_Unit preset by OpenDevice
io_Command PRD_PRtCOMMAND
io_PrtCommand the actual command number
io_Param0 parameter for the command
io_Param1 parameter for the command
io_Param2 parameter for the command
io_Param3 parameter for the command

RESULTS

Errors: if the PCPrCommand succeeded, then Error will be zero. Otherwise the Error will be non-zero.

SEE ALSO

printer.device/write printer.h, parallel.device, Preferences

printer.device/RawWrite

printer.device/RawWrite

NAME RawWrite - transparent write command

FUNCTION

This is a non standard write command that performs no processing on the data passed to it.

IO REQUEST

io_Message m_ReplyPort set if quick I/O is not possible
io_Command PRD_RAWWRITE
io_Flags IOB_QUICK set if quick I/O is possible
io_Length the number of bytes in io.Data
io_Data the raw bytes to write to the printer

printer.device/Reset

printer.device/Reset

NAME Reset - reset the printer

FUNCTION

Reset resets the printer device without destroying handles to the open device.

IO REQUEST

io_Message m_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Command CMD_RESET
io_Flags IOB_QUICK set if quick I/O is possible

printer.device/Start printer.device/Start

NAME Start - restart after stop (immediate)

FUNCTION

Start restarts the unit after a stop command.

IO REQUEST

io_Message mn_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Command CMD_START
io_Flags IOB_QUICK set if quick I/O is possible

printer.device/Stop

printer.device/Stop

NAME Stop - pause current and queued I/O requests (immediate)

FUNCTION

Stop pauses all queued requests for the unit, and tries to pause the current I/O request. The only commands that will be subsequently allowed to be performed are immediate I/O requests, which include those to start, flush, and finish the I/O after the stop command.

IO REQUEST

io_Message mn_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Command CMD_STOP
io_Flags IOB_QUICK set if quick I/O is possible

printer.device/Write printer.device/Write

NAME POWrite -- send output to the printer

FUNCTION

This function causes a buffer of characters to be written to the either the parallel or serial device. The number of characters is specified in io_Length, unless -1 is used, in which case output is send until a 0x00 is encountered. The Printer device, like the Console device, maps ANSI X3.64 style 7-bit printer control codes to the control code set of the current printer. The ANSI codes supported can be found below. All printers may not support all functions.

IO REQUEST

- io_Message mn_ReplyPort set
- io_Device preset by OpenDevice
- io_Unit preset by OpenDevice
- io_Command CMD_WRITE
- io_Length number of characters to process, or if -1, process until EOF encountered
- io_Data pointer to block of data to process

RESULTS

Errors: If the POWrite succeeded, then Error will be zero. Otherwise the Error will be non-zero.

SEE ALSO

printer.h, parallel.device, serial.device, Preferences

ANSI X3.64 style COMMANDS

| | |
|---------|----------------------|
| aRIS | reset |
| aRIN | initialize |
| aIND | if |
| aNEL | return,if |
| aRI | reverse if |
| aSCR0 | normal char set |
| aSCR3 | italics on |
| aSCR23 | italics off |
| aSCR4 | underline on |
| aSCR24 | underline off |
| aSCR1 | boldface on |
| aSCR22 | boldface off |
| aSFC | set foreground color |
| aSBC | set background color |
| aSHORP0 | normal pitch |
| aSHORP2 | elite on |
| aSHORP1 | elite off |
| aSHORP4 | condensed line on |
| aSHORP3 | condensed off |

| | | |
|---------|--------------|-------------------------|
| aSHORP6 | ESC[6w | enlarged on |
| aSHORP5 | ESC[5w | enlarged off |
| aDEN6 | ESC[6"z | shadow print on |
| aDEN5 | ESC[5"z | shadow print off |
| aDEN4 | ESC[4"z | doublestrike on |
| aDEN3 | ESC[3"z | doublestrike off |
| aDEN2 | ESC[2"z | NLQ on |
| aDEN1 | ESC[1"z | NLQ off |
| aSUS2 | ESC[2v | superscript on |
| aSUS1 | ESC[1v | superscript off |
| aSUS4 | ESC[4v | subscript on |
| aSUS3 | ESC[3v | subscript off |
| aSUS0 | ESC[0v | normalize the line |
| aPLU | ESCL | partial line up |
| aPLD | ESCK | partial line down |
| aFNT0 | ESC(B | US char set |
| aFNT1 | ESC(R | French char set |
| aFNT2 | ESC(K | German char set |
| aFNT3 | ESC(A | UK char set |
| aFNT4 | ESC(E | Danish I char set |
| aFNT5 | ESC(H | Sweden char set |
| aFNT6 | ESC(Y | Italian char set |
| aFNT7 | ESC(Z | Spanish char set |
| aFNT8 | ESC(J | Japanese char set |
| aFNT9 | ESC(6 | Norwegian char set |
| aFNT10 | ESC(C | Danish II char set |
| aPROP2 | ESC[2p | proportional on |
| aPROP1 | ESC[1p | proportional off |
| aPROF0 | ESC[0p | proportional clear |
| aTSS | ESC[n E | set proportional offset |
| aJFY5 | ESC[5 F | auto left justify |
| aJFY7 | ESC[7 F | auto right justify |
| aJFY6 | ESC[6 F | auto full justify |
| aJFY0 | ESC[0 F | auto justify off |
| aJFY3 | ESC[3 F | letter space (justify) |
| aJFY1 | ESC[1 F | word fill(auto center) |
| aVERP0 | ESC[0z | 1/8" line spacing |
| aVERP1 | ESC[1z | 1/6" line spacing |
| aSLPP | ESC[nt | set form length n |
| aPERF | ESC[nq | perf skip n (n>0) |
| aPERF0 | ESC[0q | perf skip off |
| aLMS | ESC#9 | Left margin set |
| aRMS | ESC#0 | Right margin set |
| aTMS | ESC#8 | Top margin set |
| aBMS | ESC#2 | Bottom marg set |
| aSBM | ESC[Pn1;Pn2r | T&B margins |
| aSLRM | ESC[Pn1;Pn2s | L&R margins |
| aCAM | ESC#3 | Clear margins |
| aHTS | ESCH | Set horiz tab |

| | | |
|---------|----------|--------------------|
| aVTS | ESCJ | Set vertical tabs |
| aTBC0 | ESC[0g | Clr horiz tab |
| aTBC3 | ESC[3g | Clear all h tab |
| aTBC1 | ESC[1g | Clr vertical tabs |
| aTBC4 | ESC[4g | Clr all v tabs |
| aTBCALL | ESC#4 | Clr all h & v tabs |
| aTBSALL | ESC#5 | Set default tabs |
| aXTEND | ESC[Pn"x | extended commands |

TABLE OF CONTENTS

console.device/CDAskKeyMap
 console.device/CDInputHandler
 console.device/CDSetKeyMap
 console.device/Clear
 console.device/OpenDevice
 console.device/RawKeyConvert
 console.device/Read
 console.device/Write
 gameport.device/AskCType
 gameport.device/AskTrigger
 gameport.device/Clear
 gameport.device/Open
 gameport.device/ReadEvent
 gameport.device/SetCType
 gameport.device/SetTrigger
 input.device/AddHandler
 input.device/Clear
 input.device/Open
 input.device/RemHandler
 input.device/Reset
 input.device/SetMPort
 input.device/SetMPort19
 input.device/SetMType
 input.device/SetPeriod
 input.device/SetThresh
 input.device/Start
 input.device/WriteEvent
 keyboard.device/AddrResetHandler
 keyboard.device/Clear
 keyboard.device/ReadEvent
 keyboard.device/ReadMatrix
 keyboard.device/RemResetHandler
 keyboard.device/Reset
 keyboard.device/ResetHandlerDone

console.device/CDAskKeyMap console.device/CDAskKeyMap

NAME

AskKeyMap - get the current key map structure for this console

FUNCTION

Fill the IO_DATA buffer with the current KeyMap structure in use by this console unit.

IO REQUEST

io_Message mn_ReplyPort set if quick I/O is not possible
 io_Device preset by the call to OpenDevice
 io_Unit preset by the call to OpenDevice
 io_Command CD_ASKKEYMAP
 io_Flags IOF_QUICK if quick I/O possible, else zero
 io_Length sizeof(*keyMap)
 io_Data struct KeyMap *keyMap
 eight longwords to describe the raw keycode
 to byte stream conversion.

RESULTS

This function sets the error field in the ioRequest, and fills the structure at IO_DATA with the current key map.

console.device/CDInputHandler

console.device/CDInputHandler

NAME

CDInputHandler - handle an input event for the console device

SYNOPSIS

CDInputHandler(events, consoleDev)
A0 A1

FUNCTION

Accept input events from the producer, which is usually the
raw input.task.

NOTES

This function is different from standard device commands in
that it is a function in the console device library vectors.
The "OpenLibrary" call for the console device is to
OpenDevice("console.device", -1, iORequest, 0), and then grab
the io_Device field out of the iORequest as the library
vector.

console.device/CDSetKeyMap

console.device/CDSetKeyMap

NAME

SetKeyMap - set the current key map structure for this console

FUNCTION

Set the current KeyMap structure used by this console unit to
the structure pointed to by IO_DATA

IO REQUEST

io_Message mn_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Unit preset by the call to OpenDevice
io_Command CD_SETKEYMAP
io_Flags IOF_QUICK if quick I/O possible, else zero
io_Length sizeof(*keyMap)
io_Data struct KeyMap *keyMap
eight longwords that describe the raw keycode
to byte stream conversion.

RESULTS

This function sets the error field in the iORequest, and fills
the current key map from IO_DATA.

console.device/Clear

console.device/Clear

NAME

Clear - clear console input buffer

FUNCTION

Remove from the input buffer any reports waiting to satisfy read requests.

IO REQUEST

io_Message
io_Device
io_Unit
io_Command
io_Flags
no_ReplyPort set if quick I/O is not possible
preset by the call to OpenDevice
CMD_CLEAR
IOB_QUICK set if quick I/O is possible

console.device/OpenDevice

console.device/OpenDevice

NAME

OpenDevice - a request to open a Console device

SYNOPSIS

OpenDevice("console.device", unit, iORequest, 0)

FUNCTION

The open routine grants access to a device. There are two fields in the iORequest block that will be filled in: the IO_DEVICE field and possibly the IO_UNIT field.

This open command differs from most other device open commands in that it requires some information to be supplied in the IO_DATA field of the iORequest block. This initialization information supplies the window that is used by the console device for output.

The unit number that is a standard parameter for an open call is used specially by this device. A unit of -1 indicates that no actual console is to be opened, and is used to get a pointer to the device library vector. A unit of zero binds the supplied window to a unique console. Sharing a console must be done at a level higher than the device. There are no other valid unit numbers.

IO REQUEST

io_Data

struct Window *window

This is the window that will be used for this console. It must be supplied if the unit in the OpenDevice call is 0 (see above). The RPort of this window is potentially in use by the console whenever there is an outstanding write command.

console.device/RawKeyConvert

console.device/RawKeyConvert

NAME

RawKeyConvert - decode raw input classes

SYNOPSIS

```
actual = RawKeyConvert(event, buffer, length, keyMap),
consoleDev
D0      A0      A1      D1      A2
A6
```

FUNCTION

This console function converts input events of type IECLASS_RAWKEY to ANSI bytes, based on the keyMap, and places the result into the buffer.

INPUTS

event - an InputEvent structure pointer.
 buffer - a byte buffer large enough to hold all anticipated characters generated by this conversion.
 length - maximum anticipation, i.e. the buffer size in bytes.
 keyMap - a KeyMap structure pointer, or null if the default console device key map is to be used.
 consoleDev - the io_Device of the console device.

RESULTS

actual - the number of characters in the buffer, or -1 if a buffer overflow was about to occur.

ERRORS

if actual is -1, a buffer overflow condition was detected. Not all of the characters in the buffer are valid.

NOTES

This function is different from standard device commands in that it is a function in the console device library vectors. The "OpenLibrary" call for the console device is to OpenDevice("console.device", -1, iORequest, 0), and then grab the io_Device field out of the iORequest as the library vector.

console.device/Read

console.device/Read

NAME

Read - return the next input from the keyboard

FUNCTION

Read the next input, generally from the keyboard. The form of this input is as an ANSI byte stream: i.e. either ASCII text or control sequences. Raw input events received by the console device can be selectively filtered via the SRE and RRE control sequences (see the write command). Keys are converted via the keymap associated with the unit, which is modified with AskKeyMap and SetKeyMap

If, for example, raw keycodes had been enabled by writing <CSI>is to the console (where <CSI> is \$9B or Esc()), keys would return raw keycode reports with the information from the input event itself, in the form:
 <CSI>I;0;<keycode>;<modifiers>;0;<seconds>;<microseconds>q

If there is no pending input, this command will not be satisfied, but if there is some input, but not as much as can fill IO_LENGTH, the request will be satisfied with the input currently available.

IO REQUEST

```
io_Message      mm_ReplyPort set if quick I/O is not possible
io_Device       preset by the call to OpenDevice
io_Unit         CMD_READ
io_Command      IOE_QUICK if quick I/O possible, else zero
io_Flags        sizeof(*buffer)
io_Length       char buffer[]
io_Data         The destination for the characters to read
                from the keyboard.
```

RESULTS

This function sets the error field in the iORequest, and fills the iORequest IO_DATA area with the next input, and IO_ACTUAL with the number of bytes read.

console.device/Write console.device/Write

NAME Write - write text to the display

FUNCTION

Write a text record to the display. Note that the RPort of the console window is in use while this write command is pending.

IO REQUEST

io_Message m_ReplyPort set if quick I/O is not possible
 io_Device preset by the call to OpenDevice
 io_Unit preset by the call to OpenDevice
 io_Command CMD_WRITE
 io_Flags IOF_QUICK if quick I/O possible, else zero
 io_Length sizeof(*buffer), or -1 if null terminated
 io_Data char buffer[]
 a buffer containing the ANSI text to write to the console device.

ANSI CODES SUPPORTED

| Code | Name | Definition |
|-------|------|-----------------|
| 00/ 8 | BS | BACKSPACE |
| 00/10 | LF | LINE FEED |
| 00/11 | VT | VERTICAL TAB |
| 00/12 | FF | FORM FEED |
| 00/13 | CR | CARRIAGE RETURN |
| 00/14 | SO | SHIFT OUT |
| 00/15 | SI | SHIFT IN |
| 01/11 | ESC | ESCAPE |

Code or Esc Name Definition

| | | |
|-------|---|---|
| 08/ 4 | D | IND INDEX: move the active position down one line |
| 08/ 5 | E | NEL NEXT LINE: |
| 08/ 8 | H | HTS HORIZONTAL TABULATION SET |
| 08/13 | M | RI REVERSE INDEX: |
| 09/11 | [| CSI CONTROL SEQUENCE INTRODUCER: see next list |

ISO Compatible Escape Sequences (introduced by Esc) --

| Esc | Name | Definition |
|-----|------|------------------------|
| c | RIS | RESET TO INITIAL STATE |

Control Sequences (introduced by CSI, i.e. \$9B or Esc[]) with numeric parameters - e.g. '14;94', and '3' is any number of numeric parameters, separated by semicolons --

| Esc[] | Name | Definition |
|-------|------|------------------|
| 1@ | ICH | INSERT CHARACTER |

| | | |
|-----|-------|--|
| 1A | CUU | CURSOR UP |
| 1B | CUD | CURSOR DOWN |
| 1C | CUF | CURSOR FORWARD |
| 1D | CUB | CURSOR BACKWARD |
| 1E | CNL | CURSOR NEXT LINE |
| 1F | CPL | CURSOR PRECEDING LINE |
| 2H | CUP | CURSOR POSITION |
| 1I | CHT | CURSOR HORIZONTAL TABULATION |
| 1J | ED | ERASE IN DISPLAY (only to end of display) |
| 1K | EL | ERASE IN LINE (only to end of line) |
| 1L | IL | INSERT LINE |
| 1M | DL | DELETE LINE |
| 1P | DCH | DELETE CHARACTER |
| 2R | CFR | CURSOR POSITION REPORT (in Read stream only) |
| 1S | SU | SCROLL UP |
| 1T | SD | SCROLL DOWN |
| 3M | CTC | CURSOR TABULATION CONTROL |
| 1Z | CBT | CURSOR BACKWARD TABULATION |
| 2f | HVP | HORIZONTAL AND VERTICAL POSITION |
| 1g | TBC | TABULATION CLEAR |
| 3h | SM | SET MODE |
| 3l | RM | RESET MODE |
| 3m | SCR | SELECT GRAPHIC RENDITION |
| 1n | DSR | DEVICE STATUS REPORT |
| 1t | aSLPP | SET PAGE LENGTH (private Amiga sequence) |
| 1u | aSLL | SET LINE LENGTH (private Amiga sequence) |
| 1x | aSLO | SET LEFT OFFSET (private Amiga sequence) |
| 1y | aSRE | SET RAW EVENTS (private Amiga sequence) |
| 3{ | aIER | INPUT EVENT REPORT (private Amiga Read sequence) |
| 8 | aARRE | RESET RAW EVENTS (private Amiga sequence) |
| 3} | aSKR | SPECIAL KEY REPORT (private Amiga Read sequence) |
| 1^ | aSCR | SET CURSOR RENDITION (private Amiga sequence) |
| 1 p | aMSR | WINDOW STATUS REQUEST (private Amiga sequence) |
| 0 q | aWBR | WINDOW BOUNDS REPORT (private Amiga Read sequence) |
| 4 r | | |

gameport.device/AskCType

gameport.device/AskCType

NAME

AskCType - inquire the current game port controller type

FUNCTION

This command identifies the type of controller at the game port, so that the signals at the port may be properly interpreted. The controller type has been set by a previous SetCType.

This command always executes immediately.

IO REQUEST

io_Message mm_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Unit preset by the call to OpenDevice
io_Command GPD_ASKCTYPE
io_Flags IOB_QUICK set if quick I/O is possible
io_Length at least 1
io_Data the address of the byte variable for the result

gameport.device/AskTigger

gameport.device/AskTigger

NAME

AskTigger - Inquire the conditions for a game port report

FUNCTION

This command inquires what conditions must be met by a game port unit before a pending Read request will be satisfied. These conditions, called triggers, are independent -- that any one occurs is sufficient to queue a game port report to the Read queue. These conditions are set by SetTigger.

This command always executes immediately.

IO REQUEST

io_Message mm_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Unit preset by the call to OpenDevice
io_Command GPD_ASKTRIGGER
io_Flags IOB_QUICK set if quick I/O is possible
io_Length sizeof(gameportTigger)
io_Data a structure of type GameportTigger, which has the following elements
gpt_Keys - GPTB_DOWNKEYS set if button down transitions trigger a report, and GPTB_UPKEYS set if button up transitions trigger a report
gpt_Timeout - a time which, if exceeded, triggers a report;
measured in vertical blank units (60/sec)
gpt_XDelta - a distance in x which, if exceeded, triggers a report
gpt_YDelta - a distance in y which, if exceeded, triggers a report

gameport.device/Clear

gameport.device/Clear

NAME

Clear - clear gameport input buffer

FUNCTION

Remove from the input buffer any gameport reports waiting to satisfy read requests.

IO REQUEST

- io_Message
- io_Device
- io_Unit
- io_Command
- io_Flags

- m_ReplyPort set if quick I/O is not possible
- presert by the call to OpenDevice
- presert by the call to OpenDevice
- CMD_CLEAR
- IOB_QUICK set if quick I/O is possible

gameport.device/Open

gameport.device/Open

NAME

Open - a request to open the GamePort device

SYNOPSIS

OpenDevice("gameport.device", unit, iOrequest, 0)

FUNCTION

The open routine grants access to a device. There are two fields in the iOrequest block that will be filled in: the IO_DEVICE field and the IO_UNIT field.

The device open count will be incremented. The device cannot be expunged unless this open is matched by a Close device.

INPUTS

- unit - 0 unit associated with left game port controller
- 1 unit associated with right game port controller

RESULTS

If the open was unsuccessful, IO_ERROR will be set, IO_UNIT and IO_DEVICE will not be valid.

gameport.device/ReadEvent

gameport.device/ReadEvent

NAME ReadEvent - return the next game port event.

FUNCTION

Read game port events from the game port and put them in the data area of the IORequest. If there are no pending game port events, this command will not be satisfied, but if there are some events, but not as many as can fill IO_LENGTH, the request will be satisfied with those currently available.

IO REQUEST

io_Message mn_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Unit preset by the call to OpenDevice
io_Command GPD_READEVENT
io_Flags IOB_QUICK set if quick I/O is possible
io_Length the size of the io_Data area in bytes; there are sizeof(inputEvent) bytes per input event.
io_Data a buffer area to fill with input events. The fields of the input event are:

- ie_NextEvent links the events returned
- ie_Class is IECLASS_MOUSE
- ie_SubClass is 0 for the left, 1 for the right game port
- ie_Code contains any gameport button reports. No report is indicated by the value 0xff.
- ie_Qualifier only the relative and button bits are set
- ie_X, ie_Y the x and y values for this report, in either relative or absolute device dependent units.
- ie_TimeStamp the delta time since the last report, given not as a standard timestamp, but as the frame count in the IV_SECS field.

RESULTS

This function sets the error field in the IORequest, and fills the IORequest with the next game port events (but not partial events).

SEE ALSO

gameport.device/SetCType, gameport.device/SetTrigger

gameport.device/SetCType

gameport.device/SetCType

NAME SetCType - set the current game port controller type

FUNCTION

This command sets the type of device at the game port, so that the signals at the port may be properly interpreted. The port can also be turned off, so that no reports are generated.

This command always executes immediately.

IO REQUEST

io_Message mn_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Unit preset by the call to OpenDevice
io_Command GPD_SETCTYPE
io_Flags IOB_QUICK set if quick I/O is possible
io_Length 1
io_Data the address of the byte variable describing the controller type, as per the equates in the gameport include file

gameport.device/SetTrigger gameport.device/SetTrigger

NAME

SetTrigger - set the conditions for a game port report

FUNCTION

This command sets what conditions must be met by a game port unit before a pending Read request will be satisfied. These conditions, called triggers, are independent -- that any one occurs is sufficient to queue a game port report to the Read queue. These conditions are inquired with AskTrigger.

This command always executes immediately.

IO REQUEST

io_Message m_ReplyPort set if quick I/O is not possible
 io_Device preset by the call to OpenDevice
 io_Unit preset by the call to OpenDevice
 io_Command CPD SETTRIGGER
 io_Flags IOB_QUICK set if quick I/O is possible
 io_Length sizeof(gameportTrigger)
 io_Data a structure of type GameportTrigger, which has the following elements

get_Keys -
 CPTB_DOWNKEYS set if button down transitions trigger a report, and CPTB_UPKEYS set if button up transitions trigger a report

get_Timeout -
 a time which, if exceeded, triggers a report;
 measured in vertical blank units (60/sec)

get_XDelta -
 a distance in x which, if exceeded, triggers a report

get_YDelta -
 a distance in y which, if exceeded, triggers a report

input.device/AddHandler input.device/AddHandler

NAME

AddHandler - add an input handler to the device

FUNCTION

Add a function to the list of functions called to handle input events generated by this device. The function is called as
 newInputEvents = Handler(inputEvents, handlerData);
 D0 A0
 A1

IO REQUEST

io_Message m_ReplyPort set
 io_Device preset by OpenDevice
 io_Unit preset by OpenDevice
 io_Command IHD_ADDHANDLER
 io_Data a pointer to an interrupt structure.
 io_IsData the handlerData pointer described above
 io_IsCode the Handler function address

NOTES

The interrupt structure is kept by the input device until a RemHandler command is satisfied for it.

input.device/Clear

input.device/Clear

NAME Clear - clear input buffer

FUNCTION Remove from input buffers any input reports waiting to satisfy read requests.

IO REQUEST

io_Message
io_Device
io_Unit
io_Command
io_Flags

mn_ReplyPort set if quick I/O is not possible
presert by the call to OpenDevice
presert by the call to OpenDevice
CMD_CLEAR
IOB_QUICK set if quick I/O is possible

input.device/Open

input.device/Open

NAME Open - a request to open the input device

SYNOPSIS OpenDevice("input.device", 0, IORequest, 0)

FUNCTION

The open routine grants access to a device. There are two fields in the IORequest block that will be filled in: the IO_DEVICE field and the IO_UNIT field.

The device open count will be incremented. The device cannot be expunged unless this open is matched by a Close device.

RESULTS

If the open was unsuccessful, IO_ERROR will be set, IO_UNIT and IO_DEVICE will not be valid.

input.device/RemHandler

input.device/RemHandler

NAME RemHandler - remove an input handler from the device

FUNCTION Remove a function previously added to the list of handler functions.

IO REQUEST io_Message mn_ReplyPort set
io_Device preset by OpenDevice
io_Unit preset by OpenDevice
io_Command IND_REPHANDLER
io_Data a pointer to the interrupt structure.

NOTES This command is not immediate

input.device/Reset

input.device/Reset

NAME Reset - reset the Input

FUNCTION Reset resets the keyboard device without destroying handles to the open device.

IO REQUEST io_Message mn_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Unit preset by the call to OpenDevice
io_Command CMD_RESET
io_Flags IOB_QUICK set if quick I/O is possible

input.device/SetMPort

NAME

SetMPort - set the current mouse port

input.device/SetMPort

FUNCTION

This command sets the gameport port at which the mouse is connected.

IO REQUEST

io_Message
 io_Device
 io_Unit
 io_Command
 io_Flags
 io_Length
 io_Data

mn_ReplyPort set if quick I/O is not possible
 preset by the call to OpenDevice
 preset by the call to OpenDevice
 IND_SETMPORT
 IOB_QUICK set if quick I/O is possible
 1
 a pointer to a byte that is either 0 or 1,
 indicating that mouse input should be obtained
 from either the left or right controller port,
 respectively.

input.device/SetMTrig

NAME

SetMTrig - set the conditions for a mouse port report

input.device/SetMTrig

FUNCTION

This command sets what conditions must be met by a mouse before a pending Read request will be satisfied. The trigger specification is that used by the gameport device.

IO REQUEST

io_Message
 io_Device
 io_Unit
 io_Command
 io_Flags
 io_Length
 io_Data

mn_ReplyPort set if quick I/O is not possible
 preset by the call to OpenDevice
 preset by the call to OpenDevice
 IND_SETMTRIGGER
 IOB_QUICK set if quick I/O is possible
 sizeof(gameportTrigger)
 a structure of type GameportTrigger, which
 has the following elements

gpt_Keys -
 GPTB_DOWNKEYS set if button down transitions
 trigger a report, and GPTB_UPKEYS set if button up
 transitions trigger a report

gpt_Timeout -
 a time which, if exceeded, triggers a report;
 measured in vertical blank units (60/sec)

gpt_XDelta -
 a distance in x which, if exceeded, triggers a
 report

gpt_YDelta -
 a distance in y which, if exceeded, triggers a
 report

Input.device/SetMType

Input.device/SetMType

NAME SetMType - set the current mouse port controller type

FUNCTION

This command sets the type of device at the mouse port, so the signals at the port may be properly interpreted.

IO REQUEST

io_Message m_nReplyPort set if quick I/O is not possible
 io_Device preset by the call to OpenDevice
 io_Unit preset by the call to OpenDevice
 io_Command IND_SETMTYPE
 io_Flags IOB_QUICK set if quick I/O is possible
 io_Length 1
 io_Data the address of the byte variable describing the controller type, as per the equates in the gameport include file

Input.device/SetPeriod

Input.device/SetPeriod

NAME SetPeriod - set the key repeat period

FUNCTION

This command sets the period at which a repeating key repeats. This command always executes immediately.

IO REQUEST - a timerrequest

io_Message m_nReplyPort set if quick I/O is not possible
 io_Device preset by the call to OpenDevice
 io_Unit preset by the call to OpenDevice
 io_Command IND_SETPERIOD
 io_Flags IOB_QUICK set if quick I/O is possible
 io_tv_Secs the repeat period seconds
 io_tv_Micro the repeat period microseconds

input.device/SetThresh

input.device/SetThresh

NAME

SetThresh - set the key repeat threshold

FUNCTION

This command sets the time that a key must be held down before it can repeat. The repeatability of a key may be restricted (as, for example, are the shift keys).

This command always executes immediately.

IO REQUEST - a timerrequest

- io_Message m_ReplyFort set if quick I/O is not possible
- io_Device preset by the call to OpenDevice
- io_Unit preset by the call to OpenDevice
- io_Command IND_SETTHRESH
- io_Flags IOB_QUICK set if quick I/O is possible
- io_tv_Secs the threshold seconds
- io_tv_Micro the threshold microseconds

input.device/Start

input.device/Start

NAME

Start - restart after stop

FUNCTION

Start restarts the unit after a stop command.

IO REQUEST

- io_Message m_ReplyFort set if quick I/O is not possible
- io_Device preset by the call to OpenDevice
- io_Unit preset by the call to OpenDevice
- io_Command CMD_START
- io_Flags IOB_QUICK set if quick I/O is possible

input.device/WriteEvent input.device/WriteEvent

NAME WriteEvent - propagate input event(s) to all handlers

FUNCTION

IO REQUEST io_Message mn_ReplyPort set if quick I/O is not possible
 io_Device preset by the call to OpenDevice
 io_Unit preset by the call to OpenDevice
 io_Command IND_WRITEEVENT
 io_Flags IOB_QUICK set if quick I/O is possible
 io_Length the size of the io_Data area in bytes; there
 io_Data are sizeof(inputEvent) bytes per input event.
 a buffer area with input events(s). The
 fields of the input event are:

 ie_NextEvent links the events together, the last event
 has a zero ie_NextEvent.

 ie_Class ie_SubClass
 ie_Code ie_Code
 ie_Qualifier ie_Qualifier
 ie_X, ie_Y ie_X, ie_Y
 ie_TimeStamp ie_TimeStamp as desired

NOTES

The contents of the input event(s) are destroyed.

keyboard.device/AddResetHandler keyboard.device/AddResetHandler

NAME AddResetHandler - add a reset handler to the device

FUNCTION

Add a function to the list of functions called to clean up
before a hard reset:
 Handler(handlerData):
 AI

IO REQUEST

 io_Message mn_ReplyPort set
 io_Device preset by OpenDevice
 io_Unit preset by OpenDevice
 io_Command KBD_ADDRESETHANDLER
 io_Data a pointer to an interrupt structure.
 the handlerData pointer described above
 is_Data is_Data
 is_Code is_Code the Handler function address

NOTES

The interrupt structure is kept by the keyboard device until a
RemResetHandler command is satisfied for it.

keyboard.device/Clear

keyboard.device/Clear

NAME Clear - clear keyboard input buffer

FUNCTION

Remove from the input buffer any keys transitions waiting to satisfy read requests.

IO REQUEST

io_Message m_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Command CMD_CLEAR
io_Flags IOB_QUICK set if quick I/O is possible

keyboard.device/ReadEvent

keyboard.device/ReadEvent

NAME ReadEvent - return the next keyboard event.

FUNCTION

Read raw keyboard events from the keyboard and put them in the data area of the IORequest. If there are no pending keyboard events, this command will not be satisfied, but if there are some events, but not as many as can fill IO_LENGTH, the request will be satisfied with those currently available.

IO REQUEST

io_Message m_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Command KBD_READEVENT
io_Flags IOB_QUICK set if quick I/O is possible
io_Length the size of the io_Data area in bytes: there are sizeof(InputEvent) bytes per input event.
io_Data a buffer area to fill with input events. The fields of the input event are:

ie_NextEvent

links the events returned

ie_Class

is IECLASS_RAMEY

ie_Code

contains the next key up/down reports

ie_Qualifier

only the shift and numeric pad bits are set
ie_SubClass, ie_X, ie_Y, ie_TimeStamp
are not used, and set to zero

RESULTS

This function sets the error field in the IORequest, and fills the IORequest with the next keyboard events (but not partial events).

keyboard.device/ReadMatrix

keyboard.device/ReadMatrix

NAME ReadMatrix - read the current keyboard key matrix

FUNCTION This function reads the up/down state of every key in the key matrix.

IO REQUEST io_Message mn_ReplyPort set if quick I/O is not possible
io_Device preset by the call to OpenDevice
io_Command KBD_READMATRIX
io_Flags IOB_QUICK set if quick I/O is possible
io_Length the size of the io_Data area in bytes: this must be big enough to hold the key matrix:
io_Data a buffer area to fill with the key matrix:
an array of bytes whose component bits reflect each keys state: the state of the key for keycode n is at bit (n MOD 8) in byte (n DIV 8) of this matrix.

RESULTS This function sets the error field in the IORequest, and sets matrix to the current key matrix.

keyboard.device/RemResetHandler

keyboard.device/RemResetHandler

NAME RemResetHandler - remove a reset handler from the device

FUNCTION Remove a function previously added to the list of handler functions.

IO REQUEST io_Message mn_ReplyPort set
io_Device preset by OpenDevice
io_Unit preset by OpenDevice
io_Command KBD_REMSETHANDLER
io_Data a pointer to the handler interrupt structure.

keyboard.device/Reset

keyboard.device/Reset

NAME Reset - reset the keyboard

FUNCTION

Reset resets the keyboard device without destroying handles to the open device.

IO REQUEST

io_Message
io_Device
io_Command
io_Flags

mn_ReplyPort set if quick I/O is not possible
presert by the call to OpenDevice
CMD_RESET
IOB_QUICK set if quick I/O is possible

keyboard.device/ResetHandlerDone

keyboard.device/ResetHandlerDone

NAME ResetHandlerDone - indicates that reset can occur

FUNCTION

Indicate that reset cleanup associated with the handler has completed.

IO REQUEST

io_Message
io_Device
io_Whit
io_Command
io_Data

mn_ReplyPort set
presert by OpenDevice
presert by OpenDevice
KBD_RESETHANDLERDONE
a pointer to the handler interrupt structure.

Dec 3 17:04 1985 ravisput.doc Page 37

TABLE OF CONTENTS

serial.device/AbortIO
 serial.device/BeginIO
 serial.device/Break
 serial.device/Clear
 serial.device/Close
 serial.device/Flush
 serial.device/Open
 serial.device/Query
 serial.device/Read
 serial.device/Reset
 serial.device/SetParams
 serial.device/Start
 serial.device/Stop
 serial.device/Write

serial.device/AbortIO

serial.device/AbortIO

NAME

AbortIO -- abort the specified I/O request

FUNCTION

This function aborts the specified read or write request. If the request is active, it is stopped immediately. If the request is queued, it is painlessly removed.

INPUTS

ioRequest -- pointer to the IOReqst Block that is to be aborted.

RESULTS

Error -- If the Abort succeeded, then Error will be #IOERR_ABORTED (-2) and the request will be flagged as aborted (set bit 5 of io_Flags). If the Abort failed, then the Error will be zero.

serial.device/BeginIO

serial.device/BeginIO

NAME

BeginIO -- start up an I/O process

FUNCTION

This function initiates a I/O request made to the serial device. Other than read or write, the functions are performed synchronously, and do not depend on any interrupt handling logic (or it's associated discontinuities), and hence, if so selected, can be performed as IO_QUICK.

With one exception, reads and writes are merely initiated by BeginIO, and thusly return to the caller as begun, not completed. Completion is signalled via the standard ReplyMsg routine. Multiple requests are handled via FIFO queuing.

The only exception to this non-QUICK handling of reads and writes is for READS when:

- IO_QUICK bit is set
- There are no pending read requests
- There is already enough data in the input buffer to satisfy this I/O Request immediately.

In this case, the IO_QUICK flag is not cleared, and the request is completed by the time it returns to the caller. There is no ReplyMsg or signal bit activity in this case.

INPUTS

ioRequest -- pointer to an I/O Request Block of size io_ExtSerSize (see serial.i for size/definition), containing a valid command in io_Command to process, as well as the command's other required parameters.

deviceNode -- pointer to the "serial.device" node built at init, and put into io_Device at Open.

RESULTS

Error -- if the BeginIO succeeded, then Error will be null. If the BeginIO failed, then the Error will be non-zero. Most I/O errors won't be reported until the ReplyMsg.

serial.device/Break

serial.device/Break

NAME

Break -- send a break signal over the serial line

FUNCTION

This function sends a break signal (serial line held low for an extended period) out the serial port. This is accomplished by setting the UARTBRK bit of reg ADKCON. After a duration (user specifiable via setparams, default 250000 microseconds) the bit is reset and the signal discontinued. If the QUEUEDEBRK bit of io_SerFlags is set in the io_Request block, the request is placed at the back of the write-request queue and executed in turn. If the QUEUEDEBRK bit is not set, the break is started immediately, control returns to the caller, and the timer discontinues the signal after the duration is completed. It is up to the caller to co-ordinate his/her intentions with the proper commands such as ABORT, FLUSH, STOP, START, etc...

IO REQUEST

io_Message mm_ReplyPort initialized
 io_Device set by OpenDevice
 io_Unit set by OpenDevice
 io_Command SDCMD_BREAK
 io_Flags set/reset IO_QUICK per above description

RESULTS

Error -- if the Break succeeded, then Error will be null. If the Break failed, then the Error will be non-zero.

serial.device/Clear

serial.device/Clear

NAME Clear -- clear the serial port buffers

FUNCTION This function resets the serial port's read buffer pointers.

IO REQUEST io_Message set by OpenDevice
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command CMD_CLEAR

RESULTS Error -- if the Clear succeeded, then Error will be null.
If the Clear failed, then the Error will be non-zero.

serial.device/Close

serial.device/Close

NAME Close -- close the serial port

SYNOPSIS CloseDevice(deviceNode)

FUNCTION This function closes software access to the serial device. Upon closing, the device's input buffer is freed.

INPUTS deviceNode - pointer the device node, set by Open

SEE ALSO serial.device/Open

serial.device/Flush

serial.device/Flush

NAME

Flush -- clear all queued I/O requests for the serial port

FUNCTION

This function purges the read and write request queues for the serial device. Flush will not affect active requests.

IO REQUEST

- io_Message mn_ReplyPort initialized
- io_Device set by OpenDevice
- io_Unit set by OpenDevice
- io_Command CMD_FLUSH

RESULTS

Error -- if the Flush succeeded, then Error will be null.
If the Flush failed, then the Error will be non-zero.

serial.device/Open

serial.device/Open

NAME

Open -- a request to open the serial port

SYNOPSIS

OpenDevice(sername, unit, ioRequest, flags)

FUNCTION

This function allows the requestor software access to the serial device. Unless the shared-access bit (bit 5 of io_SerFlags) is set, exclusive use is granted and no other access is allowed until the owner closes the device. All serial-specific fields are initialized to their most recent values (or default, if the first time open). OpenDevice initializes the io_Device and io_Unit fields to 0, since there is only one serial device/unit. If the user wants to support 7-wire handshaking (i.e. RS232-C CTS/RTS protocol), he should set the 7WIRE bit before opening.

INPUTS

- sername - pointer to literal string "serial.device"
- unit - ignored
- ioRequest - pointer to an ioRequest block of size io_ExtSerSize (see serial.i.h for size/definition) to be initialized by the OpenDevice routine.

NOTE use of io_SerFlags (see FUNCTION above)
#@@! IMPORTANT !!! ioRequest block MUST (!!) be of size io_ExtSerSize !!!
flags - ignored

RESULTS

D0 -- pointer to the device node
Error -- if the Open succeeded, then Error will be null.
If the Open failed, then the Error will be non-zero.

serial.device/Query

serial.device/Query

NAME Query -- query serial port/line status

FUNCTION

This function return the status of the serial port lines and registers. The number of unread bytes in the serial device's read buffer is shown in io_Actual.

IO REQUEST

io_Message m_ReplyPort initialized
 io_Device set by OpenDevice
 io_Unit set by OpenDevice
 io_Command SDOWD_QUERY (0A)

RESULTS

io_Status BIT ACTIVE FUNCTION

| | | | |
|-----|-------|------|----------------------------------|
| LSB | 0 | low | reserved |
| | 1 | low | reserved |
| | 2 | low | reserved |
| | 3 | low | Data Set Ready |
| | 4 | low | Clear To Send |
| | 5 | low | Carrier Detect |
| | 6 | low | Ready To Send |
| | 7 | low | Data Terminal Ready |
| | 8 | high | read buffer overflow |
| | 9 | high | break sent (most recent output) |
| | 10 | high | break received (as latest input) |
| | 11 | high | transmit x-OFFed |
| | 12 | high | receive x-OFFed |
| | 13-15 | | reserved |

io_Actual set to count of unread input characters
 Error -- if the Query succeeded, then Error will be null.
 If the Flush failed, then the Error will be non-zero.

serial.device/Read

serial.device/Read

NAME Read -- read input from serial port

FUNCTION

This function causes a stream of characters to be read in the serial port. The number of characters is specified in io_Length, unless -1 is used, in which case input is read until a null(0x00) is received. Input for which there is no request is stored in the input buffer until it can be dispatched to a requestor.

IO REQUEST

io_Message m_ReplyPort initialized
 io_Device set by OpenDevice
 io_Unit set by OpenDevice
 io_Command CMD_READ
 io_Flags IOF_QUICK if quick I/O possible and desired
 io_Length number of characters to receive, or if set to -1 receive until null(0x00) read in pointer to read buffer
 io_Data

RESULTS

Error -- if the Read succeeded, then Error will be null.
 If the Read failed, then the Error will be non-zero.

serial.device/Reset

serial.device/Reset

NAME
Reset -- reinitializes the serial port

FUNCTION

This function resets the serial port to its freshly initialized condition. It aborts all I/O requests both queued and current, relinquishes the current buffer, obtains a new default sized buffer, and sets the port's flags and parameters to their boot-up times default values. The functions places the reset parameter values in the ioRequest block.

IO REQUEST

io_Message mn_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command CMD_RESET

RESULTS

Error -- if the Reset succeeded, then Error will be null.
If the Reset failed, then the Error will be non-zero.

serial.device/SetParams

serial.device/SetParams

NAME
SetParams -- change parameters for the serial port

FUNCTION

This function allows the caller to change parameters for the serial device. Except for xON-xOFF enable/disable, it will reject a setparams call if any reads or writes are active or pending.
Note specifically:

1. Valid input for io_Baud is between 112 and 292000 baud inclusive; asynchronous i/o above 32KB (especially on a busy system) may be ambitious.
2. The ECFMODE and QUEUEDBRK bits of io_SerFlags can be set/reset in the io_Rqst block without a call to SetParams. The SHARED and 7WIRE bits of io_SerFlags are used in OpenDevice calls. ALL OTHER PARAMETERS CAN ONLY BE CHANGED BY THE SetParams COMMAND. (!!!!)
3. RBufLen must be at least 512.
4. io_ExtFlags is not used in V1.1, and MUST be set to zero to assure upward compatibility.
5. xON-xOFF is by default enabled. The XDISABLED bit is the only parameter that can be changed via a SetParams call while the device is active. Note that this will return the value SerErr_DevBusy in the io_Error field.
6. If trying to run MIDI, it is suggested to set the RAD_BOOGIE bit of io_SerFlags to bypass unneeded overhead. Specifically, this skips checks for parity, x-OFF handling, character lengths other than 8 bits, and testing for a break signal. Setting RAD_BOOGIE will also set the XDISABLED bit.
Note that writing data (that's already in MIDI format) at MIDI rates is easily accomplished. Using this driver alone for MIDI reads may, however, be inappropriate, due to MIDI timestamping requirements, and possibility of overruns in a busy multitasking and/or display intensive environment.

IO REQUEST

io_Message mn_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command SDCMD_SETPARAMS (0x0B)

NOTE

- that the following fields are filled in by Open a longword containing byte values for the xON,xOFF,INQ,ACK fields (respectively) (INQ/ACK not used at this time) length in bytes of input buffer (not used)

io_CtlChar

io_RBufLen
io_ExtFlags

NOTE

- that any change in buffer size causes the current buffer to be deallocated and a new, correctly sized one to be allocated. Thusly, the CONTENTS OF THE OLD BUFFER ARE LOST.
baud rate for reads AND writes. (See 1 above)

io_Baud

io_BrkTime duration of break signal in MICROseconds
 io_TermArray ASCII descending-ordered 8-byte array of termination characters. If less than 8 chars used, fill out array w/lowest valid value.
 Terminators are checked only if EOPMODE bit of io_SerFlags is set. (e.g. x512F040303030303)
 number of bits in read word (1-8) " "
 number of bits in write word (1-8) " "
 number of stop bits (1 normal, 2 can be specified for reads if ReadLen <= 7)
 io_StopBits see serial.1.h for bit equates. NOTE that x00 yields exclusive access, xON/OFF-enabled, no parity checking, 3-wire protocol and TermArray inactive.

RESULTS
 Error -- If the SetParams succeeded, then Error will be null.
 If the SetParams failed, then the Error will be non-zero.

serial.device/Start serial.device/Start

NAME Start -- restart paused I/O over the serial port

FUNCTION
 This function restarts all current I/O on the serial port by sending an xON to the "other side", and submitting a "logical xON" to "our side", if/when appropriate to current activity.

IO REQUEST
 io_Message m_ReplyPort initialized
 io_Device set by OpenDevice
 io_Unit set by OpenDevice
 io_Command CMD_START

RESULTS
 Error -- If the Start succeeded, then Error will be null.
 If the Start failed, then the Error will be non-zero.

SEE ALSO
 serial.device/Stop

serial.device/Stop

serial.device/Stop

NAME

Stop -- pause all current I/O over the serial port

FUNCTION

This function halts all current I/O on the serial port by sending an xOFF to the "other side", and submitting a "logical xOFF" to "our side", if/when appropriate to current activity.

IO REQUEST

io_Message mn_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command CMD_STOP

RESULTS

Error -- if the Stop succeeded, then Error will be null.
If the Stop failed, then the Error will be non-zero.

SEE ALSO

serial.device/Start

serial.device/Write

serial.device/Write

NAME

Write -- send output to serial port

FUNCTION

This function causes a stream of characters to be written out the serial port. The number of characters is specified in io_Length, unless -1 is used, in which case output is sent until a null(0x00) is encountered.

IO REQUEST

io_Message mn_ReplyPort initialized
io_Device set by OpenDevice
io_Unit set by OpenDevice
io_Command CMD_WRITE
io_Flags IOF_QUICK set if quick I/O possible and desired
io_Length number of characters to transmit, or if set to -1 transmit until null encountered in buffer
io_Data pointer to block of data to transmit

RESULTS

Error -- if the Write succeeded, then Error will be null.
If the Write failed, then the Error will be non-zero.

SEE ALSO

serial.device/BeginIO, serial.device/setParams

Dec 3 17:04 1985 serial.doc Page 17

TABLE OF CONTENTS

timer.device/AddTime
 timer.device/background
 timer.device/CmpTime
 timer.device/SubTime
 timer.device/TR_ADDREQUEST
 timer.device/TR_GETSYSTEME
 timer.device/TR_SETSYSTEME

timer.device/AddTime

timer.device/AddTime

NAME

AddTime - add one time request to another

SYNOPSIS

AddTime(Dest, Source), timer.device
A0 A1 A6

FUNCTION

This routine adds one timeval structure to another. The results are stored in the destination (Dest + Source -> Dest)

A0 and A1 will be left unchanged

INPUTS

Dest, Source -- pointers to timeval structures.

EXCEPTIONS

SEE ALSO

BUCS

timer.device/background

timer.device/background

TIMER REQUEST

A time request is a non standard IO Request. It has an IORequest followed by a timeval structure.

TIMEVAL

A timeval structure consists of two longwords. The first is the number of seconds, the latter is the fractional number of microseconds. The microseconds must always be "normalized" e.g. the longword must be between 0 and one million.

UNITS

The timer contains two units -- one that is precise but inaccurate, the other that has little system overhead, is very stable over time, but only has limited resolution.

UNIT_MICROHZ

This unit uses a programmable timer in the 8520 to keep track of its time. It has precision down to about 2 microseconds, but will drift as system load increases. The timer is typically accurate to within five percent.

UNIT_VBLANK

This unit is driven by the vertical blank interrupt. It is very stable over time, but only has a resolution of 16667 microseconds (or 20000 microseconds in PAL land). The timer is very cheap to use, and should be used by those who are waiting for long periods of time (typically 1/2 second or more).

LIBRARY

In addition to the normal device calls, the timer also supports three direct, library like calls. They are for manipulating timeval structures. Addition, subtraction, and comparison are supported.

timer.device/CmpTime

timer.device/CmpTime

NAME

CmpTime - Compare two timeval structures

SYNOPSIS

result = CmpTime(Dest, Source), timer.device
A0 A1 A6

FUNCTION

This routine compares two timeval structures.

A0 and A1 will be left unchanged

INPUTS

Dest, Source -- pointers to timeval structures.

RESULTS

result = 0 if Dest has the same time as Source
result = -1 if Dest has less time than Source
result = +1 if Dest has more time than Source

EXCEPTIONS

SEE ALSO

BUCS

timer.device/SubTime

timer.device/SubTime

NAME SubTime - subtract one time request from another

SYNOPSIS SubTime(Dest, Source), timer.device
A0 AI A6

FUNCTION This routine subtracts one timeval structure from another. The results are stored in the destination (Dest - Source -> Dest)
A0 and A1 will be left unchanged

INPUTS Dest, Source -- pointers to timeval structures.

EXCEPTIONS

SEE ALSO

BUGS

timer.device/TR_ADDREQUEST

timer.device/TR_ADDREQUEST

NAME TR_ADDREQUEST -- submit a request to time timer

FUNCTION Ask the timer to count off a specified amount of time. The timer will chain this request with its other requests, and will reply the message back to the user when the timer counts down to zero.

TIMER REQUEST
io_Message mReplyPort initialized
io_Device preset by timer in OpenDevice
io_Unit preset by timer in OpenDevice
io_Command TR_ADDREQUEST
io_Flags IOF_QUICK allowable
tr_time a timeval structure specify how long until the driver will reply

RESULTS tr_time will contain junk

timer.device/TR_GETSYSTEMTIME

timer.device/TR_GETSYSTEMTIME

NAME TR_GETSYSTEMTIME -- get the system time

FUNCTION

Ask the timer what time it is. The system time starts off at zero at power on, but may be initialized via the TR_SETSYSTEMTIME call.

System time is monotonically increasing, and guaranteed to be unique (except of someone sets the time backwards). The time is incremented every vertical blank by the vertical blanking interval; in addition it is changed every time someone asks what time it is. This way the return value of the system time is unique and unrepeating.

TIMER REQUEST

io_Message m_ReplyPort initialized
io_Device preset by timer in OpenDevice
io_Unit preset by timer in OpenDevice
io_Command TR_ADDRESSREQUEST
io_Flags IOF_QUICK allowable

RESULTS

tr_time the timeval structure will be filled in with the current system time

timer.device/TR_SETSYSTEMTIME

timer.device/TR_SETSYSTEMTIME

NAME TR_SETSYSTEMTIME -- set the system time

FUNCTION

Set the systems idea of what time it is. The system starts out at time "zero" so it is safe to set it forward to the "real" time. However care should be taken when setting the time backwards. System time is specced as being monotonically increasing.

TIMER REQUEST

io_Message m_ReplyPort initialized
io_Device preset by timer in OpenDevice
io_Unit preset by timer in OpenDevice
io_Command TR_ADDRESSREQUEST
io_Flags IOF_QUICK allowable
tr_time a timeval structure with the current system time

RESULTS

none

TABLE OF CONTENTS

translator.library/Close
translator.library/Open
translator.library/Translate

translator.library/Close

translator.library/Close

NAME

CloseLibrary - Closes the library

SYNOPSIS

CloseLibrary(libNode)

FUNCTION

The CloseLibrary routine is called when the user no longer needs the translator library. CloseLibrary reduces the open count, and if zero and the expunge bit is on, expunges the library.

INPUTS

libNode - The library node returned from OpenLibrary.

RESULTS

SEE ALSO

translator.library/Open translator.library/Open

NAME OpenLibrary - Grants access to the translator library

SYNOPSIS
libNode = OpenLibrary("translator.library", 0);

FUNCTION
The open routine grants access to the translator library and returns the library node. The name "TranslatorBase" must be used and it must be declared to be external and long.

INPUTS
library name - "translator.library"
version - 0

RESULTS
libNode - This must be called "TranslatorBase" and declared as extern long.

SEE ALSO

translator.library/Translate

translator.library/Translate

NAME Translate - Converts an English string into phonetics

SYNOPSIS
rtnCode = Translate(instring, inlen, outbuf, outlen)

FUNCTION
The translate function converts an English string into a string of phonetic codes suitable as input to the narrator device.

INPUTS
instring - pointer to English string
inlen - length of English string
outbuf - a char array which will hold the phonetic codes
outlen - the length of the output array

RESULTS
Translate will return a zero if no error has occurred. The only error that can occur is overflowing the output buffer. If Translate determines that an overflow will occur, it will stop the translation at a word boundary before the overflow happens. If this occurs, Translate will return a negative number whose absolute value indicates where in the INPUT string Translate stopped. The user can then use the offset -rtnCode from the beginning of the buffer in a subsequent Translate call to continue the translation where s/he left off.

SEE ALSO

Dec 4 09:14 1985 translator.doc Page 5

Appendix C

Resource Summaries

Resources are software entities in the Amiga Kernel software that enable cooperating tasks to gain exclusive access to certain parts of the Amiga hardware.

There are four resources in the Amiga system:

- disk allows access to one of four possible disk units.
- cia allows you to access specific bits in each of the Complex Interface Adaptors.
- potgo manages the bits of the POTGO register.
- misc manages the serial and parallel port register bits.

Each routine for resource management is outlined in the summary sections that follow.

NOTE: Resources need only be used if a user is attempting to use the associated hardware directly. The system software routines utilize these resources internally when they perform hardware operations. Tasks that also utilize these software resource controls will be compatible with Exec and the system software.

To utilize the routines listed for the resources, as with libraries, you must first open the resource and assign the value returned to a specific base pointer name. Here is a list of the resource names and their associated base pointer names. As with libraries, the name is a null-terminated string:

| Resource Name | Base Pointer Name | |
|----------------|-------------------|---|
| potgo.resource | PotgoBase | |
| disk.resource | ---- | none provided, for asm language programmers only |
| misc.resource | ---- | none provided, for asm language programmers only |
| ciaa.resource | <user-defined> | |
| ciab.resource | <user-defined> | |

Examples:

```

struct Library *PotgoBase;
PotgoBase = (struct Library *)OpenResource("potgo.resource");
/* then use the routines provided */

....

/* <user-defined> example */
struct Library *myCiaPointerA;

myCiaPointerA = (struct Library *)OpenResource("ciaa.resource");

/* then utilize myCiaPointerA as one of the explicit parameters
* for the C language calls to the resource routines. */

```

TABLE OF CONTENTS

cia.resource/AbleICR
cia.resource/AddICRVector
cia.resource/RemICRVector
cia.resource/SetICR

Note: There are two cia.resources: ciaa.resource, and ciab.resource. These correspond to the first and second 8520 in the system. See the software memory map for the definition of the bits that each cia controls.

cia.resource/AbleICR

cia.resource/AbleICR

NAME AbleICR -- enable/disable ICR interrupts

SYNOPSIS
oldMask = AbleICR(Resource, mask)
D0 A6 D0

FUNCTION
This function provides a means of enabling and disabling 8520 CIA interrupt control registers. In addition it returns the previous enable mask.

INPUTS
mask - a bit mask indicating which interrupts to be modified. If bit 7 is clear the mask indicates interrupts to be disabled. If bit 7 is set, the mask indicates interrupts to be enabled. Bit positions are identical to those in 8520 ICR.
resource - pointer to ciaa.resource or ciab.resource as obtained from the call to OpenResource

RESULTS
oldMask - the previous enable mask before the requested changes. To get the current mask without making changes, call the function with a null parameter.

EXAMPLES
Get the current mask:
mask = AbleICR(0)
Enable both timer interrupts:
AbleICR(0x83)
Disable serial port interrupt:
AbleICR(0x08)

EXCEPTIONS
Enabling the mask for a pending interrupt will cause an immediate processor interrupt (that is if everything else is enabled). You may want to clear the pending interrupts with SetICRx prior to enabling them.

SEE ALSO
SetICR

cia.resource/AddICRVector

cia.resource/AddICRVector

NAME

AddICRVector -- attach an interrupt handler to a CIA bit

SYNOPSIS

interrupt = AddICRVector(resource, ICRBit, interrupt)
D0 A6 D0 A1

FUNCTION

Assign interrupt processing code to a particular interrupt bit of the CIA ICR. If the interrupt bit has already been assigned, this function will fail, and return a pointer to the owner interrupt. If it succeeds, a null is returned.

This function will also enable the CIA interrupt for the given ICR bit.

INPUTS

ICRBit - bit number to set (0..4)
interrupt - pointer to interrupt structure
resource - pointer to ciaa.resource or ciab.resource as obtained from the call to OpenResource

RESULT

interrupt - zero if successful, otherwise returns a pointer to the current owner interrupt structure.

SEE ALSO

RemICRVector

cia.resource/RemICRVector

cia.resource/RemICRVector

NAME

RemICRVector -- detach an interrupt handler from a CIA bit

SYNOPSIS

RemICRVector(resource, ICRBit, interrupt)
A6 D0 A1

FUNCTION

Disconnect interrupt processing code for a particular interrupt bit of the CIA ICR.

This function will also disable the CIA interrupt for the given ICR bit.

INPUTS

ICRBit - bit number to set (0..4)
interrupt - pointer to interrupt structure
resource - pointer to ciaa.resource or ciab.resource as obtained from the call to OpenResource

RESULT

SEE ALSO

AddICRVector

cia.resource/SetICR

cia.resource/SetICR

NAME SetICR -- cause, clear, and sample ICR interrupts

SYNOPSIS
old@mask = SetICR(resource, mask)
D0 A6 D0

FUNCTION
This function provides a means of resetting, causing, and sampling 8520 CIA interrupt control registers.

INPUTS
mask - a bit mask indicating which interrupts to be effected. If bit 7 is clear the mask indicates interrupts to be reset. If bit 7 is set, the mask indicates interrupts to be caused.
Bit positions are identical to those in 8520 ICR.
resource - pointer to ciaa.resource or ciab.resource as obtained from the call to OpenResource

RESULTS
old@mask - the previous interrupt register status before making the requested changes. To sample current status without making changes, call the function with a null parameter.

EXAMPLES
Get the interrupt mask:
mask = SetICR(0)
Clear serial port interrupt:
SetICR(0x08)

EXCEPTIONS
Setting an interrupt bit for an enabled interrupt will cause an immediate interrupt.

SEE ALSO
AbleICR

TABLE OF CONTENTS

disk.resource/AllocdUnit
disk.resource/FreeUnit
disk.resource/GetUnit
disk.resource/GetUnitID
disk.resource/GiveUnit

disk.resource/AllocdUnit disk.resource/AllocdUnit

NAME AllocdUnit - allocate a unit of the disk

SYNOPSIS
Success = AllocdUnit(unitNum), DResource
 D0 D0 A6

FUNCTION
This routine allocates one of the units of the disk. It should
be called before trying to use the disk (via GetUnit).

INPUTS
unitNum -- a legal unit number (zero through three)

RESULTS
Success -- nonzero if successful. zero on failure.

EXCEPTIONS

SEE ALSO

BUGS

disk.resource/FreedUnit disk.resource/FreedUnit

NAME FreedUnit - deallocate the disk

SYNOPSIS FreedUnit(unitNum), DRResource
D0 A6

FUNCTION

This routine deallocates one of the units of the disk. It should be called when done with the disk. Do not call it if you did not successfully allocate the disk (there is no protection -- you will probably crash the disk system).

INPUTS

unitNum -- a legal unit number (zero through three)

RESULTS

EXCEPTIONS

SEE ALSO

BUGS

disk.resource/GetUnit

disk.resource/GetUnit

NAME GetUnit - allocate the disk for a driver

SYNOPSIS lastDriver = GetUnit(unitPointer), DRResource
D0 A1 A6

FUNCTION

This routine allocates the disk to a driver. It is either immediately available, or the request is saved until the disk is available. When it is available, your unitPointer is sent back to you (via ReplyMsg). You may then reattempt the GetUnit.

Allocating the disk allows you to use the disk's resources. Remember however that there are four units to the disk; you are only one of them. Please be polite to the other units (by never selecting them, and by not leaving interrupts enabled, etc.).

When you are done, please leave the disk in the following state:
dmacon dma bit ON
disklen dma bit OFF (write a #DISKMAOFF to dsklen)
adkcon disk bits -- any way you want
antenna:disk sync and disk block interrupts -- Both DISABLED
CIA resource index interrupt -- DISABLED
8520 outputs -- doesn't matter, because all bits will be set to inactive by the resource.
8520 data direction regs -- restore to original state.

INPUTS

unitPtr - a pointer you your disk resource unit structure.
Note that the message filled of the structure MUST be a valid message, ready to be replied to.

RESULTS

lastDriver - if the disk is not busy, then the last unit to use the disk is returned. This may be used to see if a driver needs to reset device registers. (If you were the last user, then no one has changed any of the registers. If someone else has used it, then any allowable changes may have been made). If the disk is busy, then a null is returned.

EXCEPTIONS

SEE ALSO

BUGS

disk.resource/GetUnitID disk.resource/GetUnitID

NAME GetUnitID - find out what type of disk is out there

SYNOPSIS
 idtype = GetUnitID(unitid), DRResource
 D0 A6

FUNCTION

INPUTS

RESULTS

 idtype -- the type of the disk drive. Standard types are
 defined in the resource include file.

EXCEPTIONS

SEE ALSO

BUGS

disk.resource/GiveUnit disk.resource/GiveUnit

NAME GiveUnit - Free the disk back up

SYNOPSIS
 GiveUnit(), DRResource
 A6

FUNCTION

 This routine frees the disk after a driver is done with it.
 If others are waiting, it will notify them.

INPUTS

RESULTS

EXCEPTIONS

SEE ALSO

BUGS

Dec 4 09:12 1985 disk.doc Page 7

TABLE OF CONTENTS

misc.resource/FreetMiscResource
misc.resource/CetMiscResource

misc.resource/FreetMiscResource misc.resource/FreetMiscResource

NAME

FreetMiscResource - make a resource available for reallocation

SYNOPSIS

FreetMiscResource(unitNum), DRResource
D0 A6

FUNCTION

This routine frees one of the resources allocated by AllocMiscResource. The resource is made available for reuse.

This routine may not be called from an interrupt routine

INPUTS

unitNum - the number of the miscellaneous resource to be freed.

RESULTS

EXCEPTIONS

SEE ALSO

BUGS

misc.resource/GetMiscResource misc.resource/GetMiscResource

NAME

GetMiscResource - allocate one of the misc resources

SYNOPSIS

CurrentUser = GetMiscResource(unitNum, name), DRResource
D0 A1 A6

FUNCTION

This routine allocates one of the miscellaneous resources.
If the resource is currently allocated, an error is returned.
If you do get it, your name is associated with the resource
(so a user can see who has it allocated).

This routine may not be called from an interrupt routine

INPUTS

unitNum - the number of the resource you want to allocate
name - a mnemonic name that will help the user figure out
what piece of software is hogging a resource.
(havoc breaks out if a name of null is passed in...)

RESULTS

CurrentUser - if the resource is busy, then the name of
the current user is returned. If the resource is
free, then null is returned.

EXCEPTIONS

SEE ALSO

BUGS

TABLE OF CONTENTS

potgo.resource/AllocPotBits
potgo.resource/FreePotBits
potgo.resource/WritePotgo

potgo.resource/AllocPotBits

potgo.resource/AllocPotBits

NAME

AllocPotBits - allocate bits in the potgo register

SYNOPSIS

allocated = AllocPotBits(bits), potgoResource
D0 A6

FUNCTION

The AllocPotBits routine allocates bits in the hardware potgo register that the application wishes to manipulate via WritePotgo. The request may be for more than one bit. A user trying to allocate bits may find that they are unavailable because they are already allocated, or because the start bit itself (bit 0) has been allocated, or if requesting the start bit, because input bits have been allocated. A user can block itself from allocation: i.e. it should FreePotgoBits the bits it has and re-AllocPotBits if it is trying to change an allocation involving the start bit.

INPUTS

bits - a description of the hardware bits that the application wishes to manipulate, loosely based on the register description itself:

START (bit 0) - set if you wish to use start (i.e. start thru proportional controller counters) with the input ports you allocate (below). You must allocate all the DATxx ports you want to apply START to in this same call, with the OUTxx bit clear.

DATLX (bit 8) - set if you wish to use the port associated with the left (0) controller, pin 5.

OUTLX (bit 9) - set if you promise to use the LX port in output mode only. The port is not set to output for you at this time -- this bit set indicates that you don't mind if STARTs are initiated at any time by others, since ports that are enabled for output are unaffected by START.

DATLY (bit 10) - as DATLX but for the left (0) controller, pin 9.

OUTLY (bit 11) - as OUTLX but for LY.

DATRX (bit 12) - the right (1) controller, pin 5.

OUTRX (bit 13) - OUT for RX.

DATRY (bit 14) - the right (1) controller, pin 9.

OUTRY (bit 15) - OUT for RY.

RESULTS

allocated - the START and DATxx bits of those requested that were granted. The OUTxx bits are don't cares.

potgo.resource/FreePotBits

potgo.resource/FreePotBits

NAME

FreePotBits - free allocated bits in the potgo register

SYNOPSIS

FreePotBits(allocated), potgoResource
D0

FUNCTION

The FreePotBits routine frees previously allocated bits in the hardware potgo register that the application had allocated via AllocPotBits and no longer wishes to use. It accepts the return value from AllocPotBits as its argument.

potgo.resource/WritePotgo

potgo.resource/WritePotgo

NAME

WritePotgo - write to the hardware potgo register

SYNOPSIS

WritePotgo(word, mask), potgoResource
D0 DI A6

FUNCTION

The WritePotgo routine sets and clears bits in the hardware potgo register. Only those bits specified by the mask are affected -- it is improper to set bits in the mask that you have not successfully allocated. The bits in the high byte are saved to be maintained when other users write to the potgo register. The START bit is not saved, it is written only explicitly as the result of a call to this routine with the START bit set: other users will not restart it.

INPUTS

word - the data to write to the hardware potgo register and save for further use, except the START bit, which is not saved.

mask - those bits in word that are to be written. Other bits may have been provided by previous calls to this routine, and default to zero.

Dec 3 17:04 1985 potgo.doc Page 5

Appendix D

C Include Files — “.h” Files

This appendix contains the C-language include files that define the system data structures used by the ROM (or kickstart) routines and the disk-loadable libraries.

As with the documentation files, these include-files are organized on a functional basis. In other words, things pertinent to the exec are listed under “exec/something.h”, things pertinent to graphics are listed under “graphics/graphicsitem.h” and so on.

This appendix is a hard-copy of the ”SYS:includes” directory on the Amiga C (Lattice C) disk.

Cross Reference Utility (C) 1984,1985 Commodore Amiga, Inc.

```

1:dictionary      2:adbbits.h      3:alerts.h      4:audio.h      5:blit.h
6:bootblock.h    7:cia.h          8:clip.h        9:clipboard.h  10:collide.h
11:console.h     12:comunit.h     13:copper.h     14:ctype.h     15:custom.h
16:dec.h         17:devices.h     18:diak.h       19:diakfont.h  20:display.h
21:dnabits.h     22:dos.h         23:dosextns.h  24:error.h     25:errors.h
26:exec.h        27:execbase.h   28:execname.h  29:fcntl.h     30:gameport.h
31:gels.h        32:gfx.h         33:gfxbase.h   34:gfxmacros.h 35:graphint.h
36:icon.h        37:input.h       38:inputevent.h 39:intbits.h   40:interrupts.h
41:intuition.h   42:intuitionbase.h 43:io.h         44:iosl.h     45:keyboard.h
46:keymap.h      47:layers.h     48:libraries.h 49:limits.h    50:lists.h
51:macros.h      52:math.h        53:mathffp.h   54:memory.h    55:misc.h
56:narrator.h    57:nodes.h       58:parallel.h  59:ports.h     60:potgo.h
61:printer.h     62:prtbase.h    63:rastport.h  64:regions.h  65:resident.h
66:serial.h      67:sprite.h     68:startup.h   69:stdio.h     70:tasks.h
71:text.h        72:timer.h      73:trackdiak.h 74:translator.h 75:types.h
76:view.h        77:workbench.h

```

```

A. 14-50, 14-51, 22-105
ACCESS_READ, 22-38
ACCESS_WRITE, 22-40
ADKB_FAST, 2-21
ADKB_MEMPREC, 2-17
ADKB_MSBSYNC, 2-20
ADKB_PRECOMP0, 2-16
ADKB_PRECOMP1, 2-15
ADKB_SETCLR, 2-14
ADKB_UARTERK, 2-18
ADKB_USEOP1, 2-25
ADKB_USEOV1, 2-29
ADKB_USE1P2, 2-24
ADKB_USE1V2, 2-26
ADKB_USE2P3, 2-23
ADKB_USE2V3, 2-27
ADKB_USE3PN, 2-22
ADKB_USE3VN, 2-26
ADKB_WORDSYNC, 2-19
ADKE_FAST, 2-38
ADKE_MEMPREC, 2-34
ADKE_MSBSYNC, 2-37
ADKE_PRE000NS, 2-48
ADKE_PRE140NS, 2-49
ADKE_PRE280NS, 2-50
ADKE_PRE560NS, 2-51
ADKE_PRECOMP0, 2-33, 2-49, 2-51
ADKE_PRECOMP1, 2-32, 2-50, 2-51
ADKE_SETCLR, 2-31
ADKE_UARTERK, 2-35
ADKE_USEOP1, 2-42
ADKE_USEOV1, 2-46
ADKE_USE1P2, 2-41
ADKE_USE1V2, 2-45
ADKE_USE2P3, 2-40
ADKE_USE2V3, 2-44

```

```

ADKE_USE3PN, 2-39
ADKE_USE3VN, 2-43
ADKE_WORDSYNC, 2-36
AFB_50HZ, 27-107
AFB_68010, 27-103
AFB_68020, 27-104
AFB_68081, 27-105
AFB_PAL, 27-106
AG_IOError, 3-50
AG_MakeLib, 3-46
AG_NoMemory, 3-45
AG_OpenDev, 3-48
AG_OpenLib, 3-47
AG_OpenRes, 3-49
ALERT_TYPE, 41-1228
ALPHA_F_101, 41-1153
ALTKEYMAP, 41-329
ALLLEFT, 41-1252
ALRIGHT, 41-1253
AMIGALEFT, 41-1254, 41-1256
AN_BaseChkSum, 3-84
AN_BltBitMap, 3-102
AN_CopDisplay, 3-93
AN_CopListOver, 3-96
AN_CopInstr, 3-94
AN_CopListHead, 3-97
AN_CopListOver, 3-95
AN_ExceptVect, 3-83
AN_ExecLib, 3-82
AN_FloodFill, 3-100
AN_GraphicsLib, 3-92
AN_InitAPtr, 3-89
AN_IntrMem, 3-88
AN_Intuition, 3-108
AN_LayersLib, 3-105
AN_LibChkSum, 3-85
AN_LibMem, 3-86
AN_LongFrame, 3-98
AN_MemCorrupt, 3-87
AN_ShortFrame, 3-99
AN_TextTapRas, 3-101
AO_AudioDev, 3-62
AO_BootStrap, 3-71
AO_CIARarc, 3-68
AO_CListLib, 3-68
AO_ConsoleDev, 3-63
AO_DOSLib, 3-69
AO_DiskRarc, 3-69
AO_ExecLib, 3-63
AO_GamePortDev, 3-64
AO_GraphicsLib, 3-64
AO_IconLib, 3-61
AO_Intuition, 3-66
AO_KeyboardDev, 3-65
AO_LayersLib, 3-65
AO_MathLib, 3-57

```

AO_MiscRsrc, 3-70
 AO_RAMLib, 3-60
 AO_TimerDev, 3-67
 AO_TrackDiskDev, 3-66
 AO_Workbench, 3-72
 AOlPen, 34-28, 63-63
 AREAOUTLINE, 34-28, 34-33, 63-101
 ASPECT_HORIZ, 41-1136
 ASPECT_VERT, 41-1137
 AT_DeadEnd, 3-41
 AT_Recovery, 3-42
 AUTOBACKPEN, 41-1239
 AUTODRAWMODE, 41-1240
 AUTOFRONTPEN, 41-1238
 AUTOITEXTFONT, 41-1243
 AUTOKNOB, 41-413
 AUTOLEFTEGE, 41-1241
 AUTONEXTTEXT, 41-1244
 AUTOTOPEDGE, 41-1242
 AddFreeList, 36-40
 AlertData, 27-48
 AllocWObject, 36-38
 AltKeyMap, 41-483
 AreaPtsz, 34-31, 63-65
 AreaPtrn, 34-31, 63-56
 AttrFlags, 27-68
 AttrResched, 27-69
 AudChannel, 15-86
 B, 22-105
 BADDR, 27-80
 BAUD_110, 41-1105
 BAUD_1200, 41-1107
 BAUD_19200, 41-1111
 BAUD_2400, 41-1108
 BAUD_300, 41-1106
 BAUD_4800, 41-1109
 BAUD_9600, 41-1110
 BAUD_MIDI, 41-1112
 BENAME_DOS, 6-30
 BENAME_KICK, 6-31
 BITSPERBYTE, 22-30
 BITSPERLONG, 22-32
 BLITMSG_FAULT, 33-63
 BNDRYOFF, 34-33
 BOOLGADGET, 41-352
 BOTTOMBORDER, 41-320
 BROTHER_15XL, 41-1154
 BYTEBITS, 75-33
 BYTEMASK, 75-54
 BYTESPERLONG, 22-31
 BackPen, 41-480, 41-509
 BitMap, 8-39, 8-64, 8-64, 32-33, 41-171, 41-884, 41-884, 41-941, 41-941, 41-1010, 63-55,
 63-55, 76-70, 76-70
 Buffer, 41-438
 BufferPos, 41-440
 C, 6-28, 6-31, 22-108

CBD_CURRENTREADID, 9-24
 CBD_CURRENTWRITEID, 9-25
 CBD_POST, 9-23
 CBERR_OBSOLETEID, 9-27
 CBM_MP51000, 41-1155
 CBump, 34-36, 34-37
 CEND, 34-38
 CHECKWIDTH, 41-1218
 Chgght, 41-405
 CIAANAME, 7-6
 CIABNAME, 7-7
 CINIT, 34-35
 CLOSE, 41-350
 CLeft, 41-448
 CMOVE, 34-36
 CMove, 34-36
 COMMWIDTH, 41-1219
 COUNT, 75-45
 CR_NEEDS_NO_CONCEALED_RASTERS, 8-74
 CTop, 41-448
 CUSTOM, 41-1149
 CUSTOM_NAME, 41-1152
 CWait, 34-37, 34-38
 CWidth, 41-404
 Carg, 35-19
 ChkBase, 27-39
 ChkSum, 27-51
 ClipRect, 8-28, 8-28, 8-40, 8-51, 8-54, 8-55, 8-59, 8-61, 8-62, 8-66
 ClipboardUnitPartial, 9-30
 Clock, 27-107, 31-212
 ClrIns, 76-36
 ColdCapture, 27-40
 ColorTable, 76-25
 CoolCapture, 27-41
 Custom, 15-20
 DEADEND_ALERT, 41-1230
 DEVICES_CLIPBOARD_H, 9-1, 9-2
 DEVICES_INPUT_H, 37-1, 37-2
 DEVICES_PRINTER_H, 61-1, 61-2
 DEVICES_TIMER_H, 38-13, 41-49, 62-37, 72-15, 72-16, 72-43
 Dhght, 76-38
 DIAB_630, 41-1156
 DIAB_ADV_D25, 41-1157
 DIAB_C_150, 41-1158
 DLT_DEVICE, 23-219
 DLT_DIRECTORY, 23-220
 DLT_VOLUME, 23-221
 DMAB_AUDIO, 21-38
 DMAB_AUD1, 21-39
 DMAB_AUD2, 21-40
 DMAB_AUD3, 21-41
 DMAB_BLITHOG, 21-48
 DMAB_BLITTER, 21-44
 DMAB_BLTDONE, 21-49
 DMAB_BLTZERO, 21-50

DMAB COPPER, 21-45
 DMAB_DISK, 21-42
 DMAB_MASTER, 21-47
 DMAB_RASTER, 21-46
 DMAB_STICLR, 21-37
 DMAB_SFRITE, 21-43
 DMAF_ALL, 21-30
 DMAF_BLITHO, 21-29
 DMAF_BLITTR, 21-25
 DMAF_BLDONE, 21-34
 DMAF_BLTZERO, 21-35
 DMAF COPPER, 21-26
 DMAF_DISK, 21-23
 DMAF_MASTER, 21-28
 DMAF_RASTER, 21-27, 34-20, 34-21
 DMAF_SFRITE, 21-24, 34-22, 34-23
 DOMAIN, 52-47
 DOSNAME, 22-13
 DOUBLE, 75-44
 DRAFT, 41-1124
 DUALPF, 76-57
 DWidth, 76-38
 DamageList, 8-49
 DateStamp, 22-42, 22-59, 23-165, 23-211
 DebugData, 27-47
 DebugEntry, 27-46
 Device, 9-39, 17-28, 43-24, 43-33, 61-125, 61-139
 DispCount, 27-61, 41-447
 DispPos, 41-442
 DrawMode, 41-481, 41-510, 63-64
 DspIns, 76-34
 DxOffset, 76-39, 76-50
 EIGHT_LPI, 41-1129
 ELITE, 41-1120
 ENDGADGET, 41-300
 EPSON, 41-1159
 EPSON_JX_80, 41-1160
 ERROR_NO_FREE_STORE, 22-111
 ETD_CLEAR, 73-101
 ETD_FORMAT, 73-99
 ETD_MOTOR, 73-97
 ETD_READ, 73-96
 ETD_SEEK, 73-98
 ETD_UPDATE, 73-100
 ETD_WRITE, 73-95
 EXCLUSIVE_LOCK, 22-39
 EXECNAME, 28-2
 EXEC_ALERTS_H, 3-1, 3-2, 3-193
 EXEC_EXECLIB_H, 27-1, 27-2
 EXEC_LISTS_H, 9-16, 18-25, 18-27, 19-16, 27-17, 27-19, 33-4, 40-21, 40-23, 47-12, 50-1, 50-2, 59-21, 59-23, 61-23, 62-18, 70-21, 70-23, 77-22, 77-24
 EXEC_MEMORY_H, 54-1, 54-2
 EXEC_NODES_H, 9-13, 19-13, 35-8, 40-17, 40-19, 48-17, 48-19, 50-17, 50-19, 54-17, 54-19, 57-1, 57-2, 59-17, 59-19, 61-19, 62-15, 65-17, 65-19, 70-17, 70-19, 77-18, 77-20
 EXEC_PORTS_H, 8-7, 9-19, 12-11, 17-21, 17-23, 18-29, 18-31, 23-16, 41-45, 43-17, 43-19,

EXEC_RESIDENT_H, 47-8, 59-1, 59-2, 61-27, 62-21, 68-13, 68-15, 71-12
 EXTRA_HALFBRITE, 65-1, 65-2
 Elapsed, 76-65
 ExecBase, 27-34, 27-99
 ExecBaseReserved, 27-96
 ExecMessage, 41-583
 FALSE, 75-51
 FANFOLD, 41-1115
 FIBB_DELETE, 22-69, 22-73
 FIBB_EXECUTE, 22-68, 22-72
 FIBB_READ, 22-66, 22-70
 FIBB_WRITE, 22-67, 22-71
 FIBF_DELETE, 22-73
 FIBF_EXECUTE, 22-72
 FIBF_READ, 22-70
 FIBF_WRITE, 22-71
 FINE, 41-1121
 FOLLOWMOUSE, 41-311
 FOREVER, 41-1210
 FPENAN, 52-62
 FPEOVE, 52-60
 FPEUND, 52-59
 FPEZDV, 52-61
 FPHALF, 53-20
 FPONE, 53-19
 FPTEN, 53-18
 FPZERO, 53-21
 FREEHORIZ, 41-414
 FREEVERT, 41-415
 FRST_DOT, 34-29, 63-94
 F_DUPED, 29-23
 F_GETED, 29-24
 FileInfoBlock, 22-50
 FileLock, 23-225
 Flags, 8-38, 8-69, 31-95, 31-148, 31-176, 31-254, 32-37, 33-47, 34-28, 34-29, 34-33, 41-66, 41-94, 41-156, 41-204, 41-370, 41-705, 41-856, 41-927, 47-32, 63-41, 63-68, 76-22
 FreeFreeList, 36-40
 FreeObject, 36-40
 FrontPan, 41-480, 41-509
 GADGETABLED, 41-282
 GADGET0002, 41-353
 GADGETTYPE, 41-337
 GADGBOX, 41-251
 GADGCOMP, 41-250
 GADGHIGHBITS, 41-249
 GADGHIMAGE, 41-252
 GADGNONE, 41-253
 GADGIMAGE, 41-258
 GADGIMMEDIATE, 41-295
 GENLOC, 33-60
 GENLOCK_AUDIO, 76-63
 GENLOCK_VIDEO, 76-64
 GLOBAL, 75-18
 GRAPHICS_CLIP_H, 8-1, 8-2, 41-25

GRAPHICS_GFXMACROS_H, 34-1, 34-2
 GRAPHICS_GFX_H, 8-4, 32-1, 32-2, 41-21, 63-4, 64-4, 76-4
 GRAPHICS_GRAPHINT_H, 35-1, 35-2
 GRAPHICS_RASPTOP_H, 34-16, 41-33, 63-1, 63-2
 GRAPHICS_REGIONS_H, 64-1, 64-2
 GRELBOTTOM, 41-265
 GRELHEIGHT, 41-270
 GRELRIGHT, 41-266
 GRELWIDTH, 41-268
 GZZGADGET, 41-340
 GetIcon, 36-39
 GetWObject, 36-38
 HAM, 76-60
 HARDWARE_ADKBITS_H, 2-11, 2-12, 2-53
 HARDWARE_CUSTOM_H, 15-11, 15-12, 15-114
 HIRES, 76-58
 HP_LASERJET, 41-1164
 HP_LASERJET_PLUS, 41-1165
 HPotRes, 41-406
 HUGE, 52-75
 HorizBody, 41-400
 HorizPot, 41-380
 ICONNAME, 36-30
 IDNestCnt, 27-65
 ID_DOS_DISK, 22-106
 ID_KICKSTART_DISK, 22-108
 ID_NOT_REALY_DOS, 22-107
 ID_NO_DISK_PRESENT, 22-104
 ID_UNREADABLE_DISK, 22-105
 ID_VALIDATED, 22-101
 ID_VALIDATING, 22-102
 ID_WRITE_PROTECTED, 22-99
 IECLASS_ACTIVEWINDOW, 38-53
 IECLASS_CLOSEWINDOW, 38-41
 IECLASS_DISKINSERTED, 38-51
 IECLASS_DISKREMOVED, 38-49
 IECLASS_EVENT, 38-27
 IECLASS_GADGETDOWN, 38-33
 IECLASS_GADGETUP, 38-35
 IECLASS_INACTIVEWINDOW, 38-55
 IECLASS_MENULIST, 38-39
 IECLASS_NEWPREFS, 38-47
 IECLASS_NULL, 38-21
 IECLASS_POINTERPOS, 38-29
 IECLASS_RAWKEY, 38-23
 IECLASS_RAWMOUSE, 38-25
 IECLASS_REFRESHWINDOW, 38-45
 IECLASS_REQUESTER, 38-37
 IECLASS_SIZEWINDOW, 38-43
 IECLASS_TIMER, 38-31
 IECODE_KEY_CODE_FIRST, 38-66
 IECODE_UP_PREFIX, 38-65, 41-1248, 41-1250
 IMAGE_NEGATIVE, 41-1133
 IMAGE_POSITIVE, 41-1132
 IMPORT, 75-19
 IND_ADDHANDLER, 37-16

IND_REMHANDLER, 37-17
 IND_SETIMPORT, 37-21
 IND_SETMIRG, 37-23
 IND_SETMTRIC, 37-22
 IND_SETMTRIC, 37-22
 IND_SETMTRIC, 37-20
 IND_SETMTRIC, 37-19
 IND_WRITEEVENT, 37-18
 INT_VERT, 34-25, 34-26, 39-46
 IOCLib, 9-37
 IOERR_ABORTED, 25-17
 IOERR_BADLENGTH, 25-19
 IOERR_NOCMD, 25-18
 IOERR_OPENFAIL, 25-16
 IOExtID, 73-109
 IOSERB_ABORT, 66-116
 IOSERB_ACTIVE, 66-118
 IOSERB_BUFREAD, 66-112
 IOSERB_QUEUED, 66-114
 IOSERF_ABORT, 66-117
 IOSERF_ACTIVE, 66-119
 IOSERF_BUFREAD, 66-113
 IOSERF_QUEUED, 66-115
 IOSTB_OVERRUN, 66-128
 IOSTB_READERBREAK, 66-124
 IOSTB_WRITERBREAK, 66-126
 IOSTB_XOFFREAD, 66-120
 IOSTB_XOFFWRITE, 66-122
 IOSTE_OVERRUN, 66-129
 IOSTE_READERBREAK, 66-125
 IOSTE_WRITERBREAK, 66-127
 IOSTE_XOFFREAD, 66-121
 IOSTE_XOFFWRITE, 66-123
 IOSer, 66-36
 IOTArray, 66-26, 66-67
 ISCRTR, 8-79
 ISCRTRY, 8-80
 ISLESSX, 8-77
 ISLESSY, 8-78
 ITEMNUM, 41-1195
 IText, 41-485
 ITextFont, 41-484
 I_PI, 52-72
 I_PID2, 52-73
 IdleCount, 27-60
 Image, 41-528, 41-570, 41-765, 41-869, 77-56, 77-57
 InfoData, 22-85
 IntVector, 27-55, 40-33
 IntVects, 27-55
 IntrList, 27-84
 IntuiMessage, 41-581, 41-618, 41-757
 Iptr, 35-16
 Isrvstr, 35-13, 35-16
 JAM2, 41-1240, 63-89
 KNOBHIT, 41-417
 KNOBMIN, 41-419
 KNOBVMIN, 41-420

LACE, 76-59
 LEFTBORDER, 41-318
 LETTER, 41-1125
 LIBRARIES_DOS_H, 22-1, 22-2, 22-159, 23-23, 68-17, 68-19
 LIBRARIES_ICON_H, 36-2, 36-3, 36-44
 LIBRARIES_MATHEFF_H, 53-1, 53-2, 53-46
 LIBRARIES_TRANSLATOR_H, 74-1, 74-2, 74-15
 LIBRARY_VERSION, 75-56
 LOECprList, 76-48
 LOG10, 53-16
 LOGHUGE, 52-77
 LOGTINY, 52-78
 LONGBITS, 75-27
 LONGINT, 41-327
 LOWCHECKWIDTH, 41-1220
 LOWCOMWIDTH, 41-1221
 LastAlert, 27-34
 Layer, 8-25, 8-27, 8-63, 41-161, 41-449, 41-790, 41-958, 47-24, 47-25, 47-26, 63-54, 63-54
 LayerInfo, 8-52, 41-942
 LayerLockCount, 8-35
 LayerLocker, 8-53
 LayerPtr, 41-449
 Layer_Info, 8-52, 41-942, 47-22
 LeftBorder, 41-407
 LibList, 27-85
 LibNode, 27-35, 33-26, 42-40
 LinePtrn, 34-29, 63-69
 Lock, 8-31, 47-29
 LockCount, 8-33
 LockMessage, 8-46
 LockPort, 8-45, 47-28
 LongInt, 41-456
 LowMemChkSum, 27-38
 MAXBODY, 41-421
 MAXINT, 22-33
 MAXPOT, 41-422
 MENUDOWN, 41-1251
 MENUNULL, 41-1206
 MENUNUM, 41-1194
 MENUUP, 41-1250
 MININT, 22-34
 MODE_NEWFILE, 22-20
 MODE_OLDFILE, 22-18
 Mask, 34-30, 63-60
 MatchToolValue, 36-39
 MaxChars, 41-441
 MaxLocMem, 27-45
 MemList, 27-81, 54-59
 Memory, 41-1182
 MsgPort, 8-45, 8-47, 12-33, 17-36, 23-33, 23-60, 23-61, 23-79, 23-209, 23-229, 41-756, 47-27, 47-28, 59-32, 59-53, 59-60, 62-62, 62-82, 68-23
 N, 22-107
 NEWCLIPRECTS_1_1, 8-68
 NOCROSSFILL, 63-102
 NOITEM, 41-1204

NOMENU, 41-1203
 NOSUB, 41-1205
 NOT, 41-1212
 NTSC, 27-106, 33-59
 N_TRACTOR, 41-1147
 Next, 8-61, 13-52, 13-59, 13-73, 64-14, 76-31, 76-69
 NextBorder, 41-514
 NextImage, 41-570
 NextRemember, 41-1180
 NextText, 41-486
 Node, 9-31, 19-51, 35-15, 40-27, 40-36, 48-35, 50-23, 50-24, 50-25, 54-33, 54-60, 57-18, 57-19, 57-20, 59-33, 59-52, 70-27, 77-94, 77-95, 77-96, 77-97
 NumChars, 41-446
 OFFSET_BEGINNING, 22-28
 OFFSET_BEGINNING, 22-24, 22-28
 OFFSET_CURRENT, 22-25
 OFFSET_END, 22-26
 OFF_DISPLAY, 34-21
 OFF_SPRITE, 34-23
 OFF_VBLANK, 34-26
 OKIMATE_20, 41-1161
 ON_DISPLAY, 34-20
 ON_SPRITE, 34-22
 ON_VBLANK, 34-25
 OVERFLOW, 52-49
 O_APPEND, 29-11
 O_CREAT, 29-12
 O_EXCL, 29-14
 O_NDELAY, 29-10
 O_RAW, 29-16
 O_RDONLY, 29-6
 O_RDWR, 29-8
 O_TRUNC, 29-13
 O_WRONLY, 29-7
 PAL, 27-106, 33-61
 PARALLEL_PRINTER, 41-1101
 PA_IGNORE, 59-46
 PFBA, 76-56
 PI2, 53-13
 PI4, 53-14
 PICA, 41-1119
 PID4, 52-71
 PLOSS, 52-52
 POTCONAME, 60-7
 PRD_DUMPPOINT, 61-33
 PRD_PRCOMMAND, 61-32
 PRD_RAWWRITE, 61-31
 PROPBOARDLESS, 41-416
 PROPGADGET, 41-354
 PaperLength, 41-1093
 PaperType, 41-1094
 PortList, 27-86
 Prev, 64-14
 PropInfo, 41-368, 77-58, 77-59
 PutIcon, 36-39
 PutWObject, 36-39

QUME_LP_20, 41-1162
 Quantum, 27-62
 RECOVERY_ALERT, 41-1229
 REGISTER, 75-21
 RELVERIFY, 41-289
 REQADGET, 41-341
 RESOURCES_POTGO_H, 60-1, 60-2
 RIGHTBORDER, 41-317
 RTC_MATCHWORD, 65-34
 RTE_AUTOINIT, 65-36
 RTE_COLDSTART, 65-37
 RTM_WHEN, 65-40
 RTW_COLDSTART, 65-42
 RTW_NEVER, 65-41
 RasInfo, 76-42, 76-42, 76-67, 76-69
 RastPort, 8-29, 41-718, 41-731, 41-940, 41-940, 61-144, 63-52
 Rate, 27-107
 Rectangle, 8-30, 8-65, 32-25, 64-15, 64-20
 Region, 8-49, 64-18
 RegionRectangle, 64-13, 64-14, 64-21, 64-21
 Remember, 41-1178, 41-1180
 RememberSize, 41-1181
 ReplyPort, 8-47
 ResModules, 27-70
 Resident, 65-21, 65-23
 ResourceList, 27-82
 RsvdExt, 27-49
 RxOffset, 76-71
 RyOffset, 76-71
 SCRCADGET, 41-339
 SDCMD_BREAK, 66-93
 SDCMD_QUERY, 66-92
 SDCMD_SETPARAMS, 66-94
 SDOWNBACK, 41-349
 SDRAGGING, 41-345
 SELECTDOWN, 41-1249
 SELECTED, 41-275
 SELECTUP, 41-1248
 SERB_7WIRE, 66-106
 SERB_EOFMODE, 66-98
 SERB_PARTY_ODD, 66-108
 SERB_PARTY_ON, 66-110
 SERB_QUEUEBRK, 66-104
 SERB_RAD_BOOGIE, 66-102
 SERB_SHARED, 66-100
 SERB_XDISABLED, 66-96
 SERF_7WIRE, 66-107
 SERF_EOFMODE, 66-99
 SERF_PARTY_ODD, 66-109
 SERF_PARTY_ON, 66-111
 SERF_QUEUEBRK, 66-105
 SERF_RAD_BOOGIE, 66-103
 SERF_SHARED, 66-101
 SERF_XDISABLED, 66-97
 SERIAL_PRINTER, 41-1102
 SE_ALERTWACK, 3-17

SHADE_BW, 41-1140
 SHADE_COLOR, 41-1142
 SHADE_GREYSCALE, 41-1141
 SHARED_LOCK, 22-37
 SHECPrList, 76-49
 SHIFITEM, 41-1199
 SHIFTMENU, 41-1198
 SHIFTSUB, 41-1200
 SIGN, 41-1211
 SING, 52-48
 SINGLE, 41-1116
 SIX_LPI, 41-1128
 SIZING, 41-343
 SPAbs, 53-31
 SPAGos, 53-38
 SPAdd, 53-33
 SPAsin, 53-38
 SPAtan, 53-38
 SPCap, 53-29
 SPCos, 53-29
 SPCosh, 53-40
 SPDiv, 53-36
 SPExp, 53-41
 SPFcos, 53-42
 SPFlx, 53-27
 SPFlt, 53-28
 SPLog, 53-41
 SPLog10, 53-41
 SPMod, 53-35
 SPNeg, 53-32
 SPPov, 53-41
 SPRITES, 76-61
 SPSin, 53-39
 SPSincos, 53-39
 SPSinh, 53-40
 SPSqrt, 53-42
 SPSub, 53-34
 SPTan, 53-39
 SPTanh, 53-40
 SPTat, 53-30
 STATIC, 75-20
 STRGADGET, 41-355
 STRINGCENTER, 41-324
 STRINGRIGHT, 41-325
 STRPTR, 9-46, 71-47, 75-34, 75-35
 SUBNUM, 41-1196
 SUPERONT, 41-347
 SYSBASESIZE, 27-99
 SYSGADGET, 41-338
 Scroll_X, 8-44
 Scroll_Y, 8-44
 Semaphore, 59-59
 SerErr_BaudMismatch, 66-132
 SerErr_BufErr, 66-134
 SerErr_BufOverflow, 66-142
 SerErr_DetectedBreak, 66-145

```

SerErr_DevBusy, 66-131
SerErr_InitErr, 66-140
SerErr_InvBaud, 66-133
SerErr_InvParam, 66-135
SerErr_LineErr, 66-136
SerErr_NoCTS, 66-144
SerErr_NoDSR, 66-143
SerErr_NotOpen, 66-137
SerErr_ParityErr, 66-139
SerErr_PortReset, 66-138
SerErr_TimerErr, 66-141
SetACpt, 34-31
SetDrPt, 34-29
SetOFen, 34-28
SetWrMsk, 34-30
SoftIntList, 27-90, 40-40
SoftInts, 27-90
SoftVer, 27-37
SprIns, 76-35
StringInfo, 41-435
SuperBitMap, 8-39
SuperClipRect, 8-40
SuperSaveClipRects, 8-54
SysFlags, 27-64
SysStkLower, 27-44
SysStkUpper, 27-43
TDERR_BadDriveType, 73-140
TDERR_BadHdrSum, 73-131
TDERR_BadSecHdr, 73-134
TDERR_BadSecID, 73-130
TDERR_BadSecPreamble, 73-129
TDERR_BadSecSum, 73-132
TDERR_BadUnitNum, 73-139
TDERR_DiskChanged, 73-136
TDERR_DriveInUse, 73-141
TDERR_NoMem, 73-138
TDERR_NoSecHdr, 73-128
TDERR_NotSpecified, 73-127
TDERR_SeekError, 73-137
TDERR_TooFewSecs, 73-133
TDERR_WriteProt, 73-135
TDE_EXTCOM, 73-75, 73-95, 73-96, 73-97, 73-98, 73-99, 73-100, 73-101
TDWestCnt, 27-66
TD_CHANGE_NUM, 73-82
TD_CHANGE_STATE, 73-83
TD_FORMAT, 73-80, 73-99
TD_LABELSIZE, 73-117
TD_LASTCOM, 73-86
TD_MOTOR, 73-78, 73-97
TD_NAME, 73-73
TD_PROTSTATUS, 73-84, 73-86
TD_REMOVE, 73-81
TD_SECSHIFT, 73-52
TD_SEEK, 73-79, 73-98
TEXT, 75-48
TICKS_PER_SECOND, 22-47

```

```

TINY, 52-76
TLOSS, 52-51
TOGGLESELECT, 41-322
TOPBORDER, 41-319
TR_MakeBad, 74-13
TR_NoMem, 74-12
TR_NotUsed, 74-11
TWO_PI, 53-12
Task, 8-53, 23-32, 27-59, 33-52, 47-33, 59-36, 62-83, 70-26
TaskExceptCode, 27-73
TaskExitCode, 27-74
TaskReady, 27-87
TaskSigAlloc, 27-75
TaskTrapAlloc, 27-76
TaskTrapCode, 27-72
TaskWait, 27-88
TermArray0, 66-27
TermArray1, 66-28
ThisTask, 27-59
ToolTypeArray, 36-41
TopBorder, 41-408
UCOUNT, 75-46
UCopIns, 76-37
UCopperListInit, 34-35
UNDERFLOW, 52-50
US_LEGAL, 41-1146
US_LETTER, 41-1145
UndoBuffer, 41-439
UndoPos, 41-445
Unit, 9-40, 17-35, 43-25, 43-34, 61-126, 61-140
UserData, 41-242, 41-785, 41-962
VP_HIDE, 76-62
VPotRes, 41-406
VertBody, 41-401
VertPot, 41-381
WBDISK, 77-34
WBDRAWER, 77-35
WBGARBAGE, 77-38
WBObject, 36-38, 77-61, 77-93, 77-98
WBPROJECT, 77-37
WBTOOL, 77-36
WDOWNBACK, 41-348
WDRAWING, 41-344
WORDBITS, 75-30
WUBFRONT, 41-346
WTRACTOR, 41-1148
WarmCapture, 27-42
Window, 8-43, 12-35, 41-172, 41-615, 41-693, 41-695, 41-742, 41-920, 42-44, 77-60, 77-120
XY, 41-512
_cliprects, 8-51
_fperr, 52-85
_p1, 8-56, 8-66
_p2, 8-66
aBMS, 61-108
aCAM, 61-111
aDEN1, 61-66

```

aDEN2, 61-65
 aDEN3, 61-64
 aDEN4, 61-63
 aDEN5, 61-62
 aDEN6, 61-61
 aFNT0, 61-76
 aFNT1, 61-77
 aFNT10, 61-86
 aFNT2, 61-78
 aFNT3, 61-79
 aFNT4, 61-80
 aFNT5, 61-81
 aFNT6, 61-82
 aFNT7, 61-83
 aFNT8, 61-84
 aFNT9, 61-85
 aHTS, 61-113
 aIND, 61-39
 aJFY0, 61-95
 aJFY1, 61-97
 aJFY3, 61-96
 aJFY5, 61-92
 aJFY6, 61-94
 aJFY7, 61-93
 aLMS, 61-105
 aNEL, 61-40
 aPERF, 61-102
 aPERF0, 61-103
 aPLD, 61-74
 aPLU, 61-73
 aPROP0, 61-90
 aPROP1, 61-89
 aPROP2, 61-88
 aRI, 61-41
 aRIN, 61-38
 aRIS, 61-37
 aRMS, 61-106
 aSBC, 61-51
 aSEC, 61-50
 aSCR0, 61-43
 aSCR1, 61-48
 aSCR22, 61-49
 aSCR23, 61-45
 aSCR24, 61-47
 aSCR3, 61-44
 aSCR4, 61-46
 aSHORP0, 61-53
 aSHORP1, 61-55
 aSHORP2, 61-54
 aSHORP3, 61-57
 aSHORP4, 61-56
 aSHORP5, 61-59
 aSHORP6, 61-58
 aSLPP, 61-101
 aSLRM, 61-110
 aSTBM, 61-109

aSUS0, 61-72
 aSUS1, 61-69
 aSUS2, 61-68
 aSUS3, 61-71
 aSUS4, 61-70
 aTBC0, 61-115
 aTBC1, 61-117
 aTBC3, 61-116
 aTBC4, 61-118
 aTMS, 61-107
 aTSS, 61-91
 aVERP0, 61-99
 aVERP1, 61-100
 aVTS, 61-114
 abc, 53-31, 69-65
 ac_len, 15-88
 ac_ptr, 15-89
 ac_ptr, 15-87
 ac_vol, 15-90
 adkcon, 15-85
 adkconr, 15-29
 afp, 53-44
 arg1, 52-38
 arg2, 52-38
 atan2, 52-95
 atof, 52-92
 atoi, 52-90
 atol, 52-91
 back, 8-27
 bltadat, 15-69
 bltafm, 15-54
 bltalva, 15-55
 bltamod, 15-64
 bltapt, 15-58
 bltbdat, 15-68
 bltbmod, 15-63
 bltbpt, 15-57
 bltcdat, 15-67
 bltcmmod, 15-62
 bltcon0, 15-52
 bltcon1, 15-53
 bltcpt, 15-56
 bltddat, 15-21
 bltdmod, 15-65
 bltdpt, 15-59
 bltsize, 15-60
 bounds, 8-30, 8-65, 64-15, 64-20
 bptr, 22-80, 22-80
 ccode, 35-18
 ceil, 52-93
 ciaa.resource, 7-6
 ciab.resource, 7-7
 clxcon, 15-82
 clxdat, 15-28
 ccode, 35-17
 copllc, 15-72

```

cop2lc, 15-73
copcon, 15-43
copins, 15-76
copjmp1, 15-74
copjmp2, 15-75
cr, 8-55
cr2, 8-55
crnew, 8-55
cu_UnitNum, 9-31
cu_UnitNum, 9-32
custom.dmacon, 34-20, 34-21, 34-22, 34-23
custom.intena, 34-25, 34-26
dbf, 53-44
ddfstop, 15-80
ddfstret, 15-79
device.h, 26-9
divstop, 15-78
divstret, 15-77
dl_DiskType, 23-213
dl_Name, 23-215
dl_unused, 23-214
dmacon, 15-81
dmaconr, 15-22
dos.library, 22-13
drand48, 52-96
ds_Days, 22-43
ds_Minute, 22-44
ds_Tick, 22-45
dskbytr, 15-34
dskdat, 15-39
dskdatr, 15-25
dsklen, 15-38
dskpt, 15-37
dsksync, 15-71
ecvt, 52-88
erand48, 52-96
errno, 52-86
except, 52-95
exception, 52-34
exec.library, 28-2
fib_Comment, 22-60
fib_Date, 22-59
fib_DirEntryType, 22-52
fib_DiskKey, 22-51
fib_EntryType, 22-56
fib_FileName, 22-54
fib_NumBlocks, 22-58
fib_Protection, 22-55
fib_Size, 22-57
fl_Access, 23-228
fl_Key, 23-227
fl_Link, 23-226
fl_Task, 23-229
fl_Volume, 23-230
float, 75-43
floor, 52-93

```

```

fnod, 52-93
for, 41-1210
frexp, 52-93
front, 8-27
gfx.h, 8-5, 41-22, 63-5, 64-5, 76-5
graphics, 8-5, 19-20, 34-17, 41-22, 41-26, 41-30, 41-34, 41-38, 41-42, 42-21, 63-5,
64-5, 76-5
i, 53-25, 53-25
icon.library, 36-30
id_BytesPerBlock, 22-91
id_DiskState, 22-88
id_DiskType, 22-92
id_InUse, 22-94
id_NumBlocks, 22-89
id_NumBlocksUsed, 22-90
id_NumSoftErrors, 22-86
id_UnitNumber, 22-87
id_VolumeNode, 22-93
intena, 15-83
intemar, 15-35
intreq, 15-84
intreqr, 15-36
io_Actual, 9-44, 43-38
io_Baud, 66-65
io_BrkTime, 66-66
io_ClipID, 9-48
io_Command, 9-41, 43-26, 43-35, 61-127, 61-141
io_CtlChar, 66-62
io_Data, 9-46, 43-40
io_Device, 9-39, 43-24, 43-33, 61-125, 61-139
io_Error, 9-43, 43-28, 43-37, 61-129, 61-143
io_ExtFlags, 66-64
io_Flags, 9-42, 43-27, 43-36, 61-128, 61-142
io_Length, 9-45, 43-39
io_Message, 9-38, 43-23, 43-32, 61-124, 61-138
io_Offset, 9-47, 43-41
io_RBulLen, 66-63
io_ReadLen, 66-68
io_SerFlags, 66-71
io_StopBits, 66-70
io_TermArray, 66-67
io_Unit, 9-40, 43-25, 43-34, 61-126, 61-140
io_WriteLen, 66-69
iotd_Count, 73-111
iotd_Req, 73-110
iotd_SecLabel, 73-112
is_Node, 35-15, 40-27
isalnum, 14-42, 14-47
isascii, 14-46
iscntrl, 14-45
iscsym, 14-47
iscsymf, 14-48
isdigit, 14-38
isgraph, 14-44
islower, 14-37, 14-50
isprint, 14-43

```

```

ispunct, 14-41
isspace, 14-40
isxdigit, 14-39
itof, 53-25
joy0dat, 15-26
joyldat, 15-27
joytest, 15-47
lrand48, 52-91
l_LockMessage, 8-48
lists.h, 9-17, 18-26, 19-17, 26-3, 33-5, 40-22, 47-13, 59-22, 61-24, 62-19, 70-22,
77-23
lobs, 8-63
lrand48, 52-91
m, 34-30, 34-30
matherr, 52-90
memory.h, 26-5
mn_Length, 59-54
mn_Node, 59-52
mn_ReplyPort, 59-53
mrand48, 52-91
name, 52-37
nodes.h, 9-14, 19-14, 26-2, 35-9, 40-18, 48-18, 50-18, 54-18, 59-18, 61-20, 62-16,
65-18, 70-18, 77-19
nrand48, 52-91
p, 34-29, 34-29, 34-31, 34-31, 69-49, 69-49, 69-49, 69-49, 69-51, 69-51, 69-51,
69-51, 69-53, 69-53, 69-54, 69-54, 69-55, 69-55
pad2d, 15-61
pad3d, 15-66
pad3b, 15-70
padding, 41-1096
ports.h, 8-8, 9-20, 12-12, 17-22, 18-30, 23-17, 26-6, 41-46, 43-18, 47-9, 61-28,
62-22, 68-14, 71-13
pot0dat, 15-30
potldat, 15-31
potgo, 15-46
potgo.resource, 60-7
potinp, 15-32
prev, 8-62
rastport.h, 34-17, 41-34
refptr, 15-40
register, 75-21
reserved, 8-36, 8-67, 33-56, 63-84, 76-41
reservedl, 8-37
retval, 52-39
round, 53-24
rp, 8-29
rt_EndSkip, 65-24
rt_Flags, 65-25
rt_IdString, 65-30
rt_Init, 65-31
rt_MatchTag, 65-23
rt_MatchWord, 65-22
rt_Name, 65-29
rt_Pri, 65-28
rt_Type, 65-27
rt_Version, 65-26

```

```

seed48, 52-89
serdat, 15-44
serdatr, 15-33
serper, 15-45
sizeof, 27-99, 77-67, 77-67
sm_Bids, 59-61
sm_LockMsg, 59-64
sm_MsgPort, 59-60
static, 75-20
str0qu, 15-48
strhor, 15-50
strlong, 15-51
strtol, 52-91
strvbl, 15-49
tanh, 52-26, 52-94
timer.h, 38-14, 41-50, 62-38
toascii, 14-52
tolover, 14-51
toupper, 14-50
trackdisk.device, 73-73
trunc, 53-23
type, 52-36, 77-134, 77-134, 77-135, 77-135
vhposr, 15-24
vhposv, 15-42
vposr, 15-23
vposv, 15-41
x, 41-1211, 41-1211, 41-1211, 51-6, 51-6, 51-6, 51-6, 53-23, 53-23, 53-24, 53-24,
67-14, 69-65, 69-65, 69-65, 69-65

```



```

1 /* Commodore-Amiga, Inc. */
2 /* shortcuts used by c code */
3
4 #define MAX(a,b) ((a)>(b)?(a):(b))
5 #define MIN(a,b) ((a)<(b)?(a):(b))
6 #define ABS(x) ((x<0)?(-x):(x))

```

```

1 #ifndef DEVICES_AUDIO_H
2 #define DEVICES_AUDIO_H
3 /****** Commodore-Amiga, Inc. *****/
4 /* audio.h *****/
5 /******
6
7
8 #ifndef EXEC_IO_H
9 #include "exec/io.h"
10 #endif
11
12 #define AUDIONAME "audio.device"
13
14 #define ADHARD_CHANNELS 4
15
16 #define ADALLOC_MINPREC -128
17 #define ADALLOC_MAXPREC 127
18
19 #define ADCMD_FREE (CMD_NONSTD+0)
20 #define ADCMD_SETPREC (CMD_NONSTD+1)
21 #define ADCMD_FINISH (CMD_NONSTD+2)
22 #define ADCMD_PERVOL (CMD_NONSTD+3)
23 #define ADCMD_LOCK (CMD_NONSTD+4)
24 #define ADCMD_WAITCYCLE (CMD_NONSTD+5)
25 #define ADCMDB_NOUNIT (1<<5)
26 #define ADCDFE_NOUNIT (ADCFE_NOUNIT+0)
27 #define ADCMD_ALLOCATE (1<<4)
28
29 #define ADIOB_PERVOL (1<<4)
30 #define ADIOF_PERVOL (1<<4)
31 #define ADIOB_SYNCYCLE (1<<5)
32 #define ADIOF_SYNCYCLE (1<<5)
33 #define ADIOB_NOWAIT (1<<6)
34 #define ADIOF_NOWAIT (1<<6)
35 #define ADIOB_WRITEMESSAGE (1<<7)
36 #define ADIOF_WRITEMESSAGE (1<<7)
37
38 #define ADIOERR_NOALLOCATION -10
39 #define ADIOERR_ALLOCFAILED -11
40 #define ADIOERR_CHANNELSTOLEN -12
41
42 struct IOAudio {
43     IORequest ioa_Request;
44     IOAlloKey ioa_AlloKey;
45     UBYTE *ioa_Data;
46     ULONG ioa_Length;
47     UMWORD ioa_Period;
48     UMWORD ioa_Volume;
49     UMWORD ioa_Cycles;
50     struct Message ioa_WriteMsg;
51 };
52 #endif

```



```

1 #ifndef DEVICES_CLIPBOARD_H
2 #define DEVICES_CLIPBOARD_H
3 /*
4 /* Commodore-Amiga, Inc.
5 /* clipboard.h
6 /*
7 /*
8 *
9 * clipboard device command definitions
10 *
11 *
12 *
13 #ifndef EXEC_NODES_H
14 #include "exec/nodes.h"
15 #endif
16 #ifndef EXEC_LISTS_H
17 #include "exec/lists.h"
18 #endif
19 #ifndef EXEC_PORTS_H
20 #include "exec/ports.h"
21 #endif
22 #define CBD_POST (CMD_NONSTD+0)
23 #define CBD_CURRENTREADID (CMD_NONSTD+1)
24 #define CBD_CURRENTWRITEID (CMD_NONSTD+2)
25 #define CBERR_OBSOLETEID 1
26
27 struct ClipboardUnitPartial {
28     struct Node cu_Node; /* list of units */
29     ULONG cu_UnitNum; /* unit number for this unit */
30     /* the remaining unit data is private to the device */
31 };
32
33 struct IOClipReq {
34     struct Message io_Message; /* device node pointer */
35     struct Device *io_Device; /* unit (driver private) */
36     struct Unit *io_Unit; /* device command */
37     UMORDB io_Command; /* including QUICK and SATISFY */
38     UBYTE io_Flags; /* error or warning num */
39     BYTE io_Error; /* number of bytes transferred */
40     ULONG io_Actual; /* number of bytes requested */
41     ULONG io_Length; /* either clip stream or post port */
42     STRPTR io_Data; /* offset in clip stream */
43     ULONG io_Offset; /* ordinal clip identifier */
44     LONG io_ClipID;
45 };
46
47 #define PRIMARY_CLIP 0 /* primary clip unit */
48
49 struct SatisfyMsg {
50     struct Message sm_Msg; /* the length will be 6 */
51     UMORDB sm_Unit; /* which clip unit this is */
52     LONG sm_ClipID; /* the clip identifier of the post */
53 };
54
55
56

```

```

1 /*
2 /* Commodore-Amiga, Inc.
3 /* bootblock.h
4 /*
5 /*
6 /*
7 *
8 * Source Control
9 *
10 *
11 * $Header: bootblock.h,v 27.2 85/07/10 01:55:47 neil Exp $
12 *
13 * $Locker: $
14 *
15 *
16 *
17 *
18 *
19 * BootBlock definition: */
20
21 struct BootBlock {
22     UBYTE bb_id[4]; /* 4 character identifier */
23     LONG bb_chksum; /* boot block checksum (balance) */
24     LONG bb_dosblock; /* reserved for DOS patch */
25 };
26
27 #define BOOTSECTS 2 /* 1K bootstrap */
28
29 #define BBID_DOS { 'D', 'O', 'S', '\0' }
30 #define BBID_KICK { 'K', 'I', 'C', 'K' }
31 #define BENAME_DOS (('D'<<24) | ('O'<<16) | ('S'<<8))
32 #define BENAME_KICK (('K'<<24) | ('I'<<16) | ('C'<<8) | ('K'))

```

57 };
58
59 #endif

```

1 #ifndef DEVICES_CONSOLE_H
2 #define DEVICES_CONSOLE_H
3 /******
4 /* Commodore-Amiga, Inc.
5 /* console.h
6 /******
7 /******
8 *
9 * Console device command definitions
10 *
11 * Source Control
12 * -----
13 * $Header: console.h,v 1.4 85/11/13 15:13:14 kodiak Exp $
14 *
15 * $Locker:  $
16 *
17 ******
18
19 #ifndef EXEC_IO_H
20 #include "exec/io.h"
21 #endif
22
23 /****** Console commands *****/
24 #define CD_ASKEYMAP (CMD_NONSTD+0)
25 #define CD_SETKEYMAP (CMD_NONSTD+1)
26
27
28 /****** SCR parameters *****/
29
30 #define SCR_PRIMARY 0
31 #define SCR_BOLD 1
32 #define SCR_ITALIC 3
33 #define SCR_UNDERSCORE 4
34 #define SCR_NEGATIVE 7
35
36 /* these names refer to the ANSI standard, not the implementation */
37 #define SCR_BLACK 30
38 #define SCR_RED 31
39 #define SCR_GREEN 32
40 #define SCR_YELLOW 33
41 #define SCR_BLUE 34
42 #define SCR_MAGENTA 35
43 #define SCR_CYAN 36
44 #define SCR_WHITE 37
45 #define SCR_DEFAULT 39
46
47 #define SCR_BLACKBG 40
48 #define SCR_REDBG 41
49 #define SCR_GREENBG 42
50 #define SCR_YELLOWBG 43
51 #define SCR_BLUEBG 44
52 #define SCR_MAGENTABG 45
53 #define SCR_CYANBG 46
54 #define SCR_WHITEBG 47
55 #define SCR_DEFAULTBG 49
56

```

```

57 /* these names refer to the implementation, they are the preferred */
58 /* names for use with the Amiga console device. */
59 #define SCR_CLR0 30
60 #define SCR_CLR1 31
61 #define SCR_CLR2 32
62 #define SCR_CLR3 33
63 #define SCR_CLR4 34
64 #define SCR_CLR5 35
65 #define SCR_CLR6 36
66 #define SCR_CLR7 37
67
68 #define SCR_CLR0BG 40
69 #define SCR_CLR1BG 41
70 #define SCR_CLR2BG 42
71 #define SCR_CLR3BG 43
72 #define SCR_CLR4BG 44
73 #define SCR_CLR5BG 45
74 #define SCR_CLR6BG 46
75 #define SCR_CLR7BG 47
76
77 /****** DSR parameters *****/
78 #define DSR_CPR 6
79
80 /****** CTC parameters *****/
81 #define CTC_HSETTAB 0
82 #define CTC_HCLRTAB 2
83 #define CTC_HCLRTABSALL 5
84
85 /****** TBC parameters *****/
86 #define TBC_HCLRTAB 0
87 #define TBC_HCLRTABSALL 3
88
89 /****** SM and RM parameters *****/
90 #define MLAM 20 /* linefeed newline mode */
91 #define MASH ">1" /* auto scroll mode */
92 #define MAMM "77" /* auto wrap mode */
93
94 #endif

```

```

1 /****** Commodore-Amiga, Inc. *****/
2 /* conunit.h *****/
3 /****** Console device unit definitions *****/
4 *
5 *
6 *
7 *
8 *
9 *
10 *
11 #ifndef EXEC_PORTS_H
12 #include "exec/ports.h"
13 #endif
14
15 #ifndef DEVICES_CONSOLE_H
16 #include "devices/console.h"
17 #endif
18
19 #ifndef DEVICES_KEYMAP_H
20 #include "devices/keymap.h"
21 #endif
22
23 #ifndef DEVICES_INPUTEVENT_H
24 #include "devices/inputevent.h"
25 #endif
26
27 #define PMB_ASM (MLAM+1) /* internal storage bit for AS flag */
28 #define PMB_AAM (PMB_ASM+1) /* internal storage bit for AM flag */
29 #define MAXTABS 80
30
31 struct ConUnit {
32     struct MsgPort cu_MP;
33     /* ---- read only variables */
34     struct Window *cu_Window; /* intuition window bound to this unit */
35     WORD cu_XCP; /* character position */
36     WORD cu_YCP; /* max character position */
37     WORD cu_XMax; /* character raster size */
38     WORD cu_YMax; /* raster origin */
39     WORD cu_XRSize; /* raster maxima */
40     WORD cu_YRSize; /* raster extent */
41     WORD cu_XROrigin; /* smallest area intact from resize process */
42     WORD cu_YROrigin; /* cursor position */
43     WORD cu_XRExtant;
44     WORD cu_YRExtant;
45     WORD cu_MinShrink;
46     WORD cu_MaxShrink;
47     WORD cu_XOCP;
48     WORD cu_YOCP;
49
50     /* ---- read/write variables (writes must be protected) */
51     /* storage for AskKeyMap and SetKeyMap */
52     struct KeyMap cu_KeyMapStruct;
53     /* ---- tab stops */
54     UWORD cu_TabStops[MAXTABS]; /* 0 at start, 0xffff at end of list */
55
56

```

```

57 /* ----- console rastport attributes */
58 BYTE cu_Mask;
59 BYTE cu_FgPen;
60 BYTE cu_BgPen;
61 BYTE cu_AOLPen;
62 BYTE cu_DrawMode;
63 BYTE cu_AreaPtSz;
64 APTR cu_AreaPtrn;
65 UBYTE cu_Minterms[9]; /* cursor area pattern */
66 struct TextFont *cu_Font; /* console minterms */
67 UBYTE cu_AlgoStyle;
68 UBYTE cu_TxFlags;
69 UWORD cu_TxHeight;
70 UWORD cu_TxWidth;
71 UWORD cu_TxBaseline;
72 UWORD cu_TxSpacing;
73
74 /* ----- console MODES and RAW EVENTS switches */
75 UBYTE cu_Modes[(PMB_AMM*7)/8]; /* one bit per mode */
76 UBYTE cu_RawEvents[(TECLASS_MAX*7)/8];
77 };

```

```

1 #ifndef DEVICES_GAMEPORT_H
2 #define DEVICES_GAMEPORT_H
3 /*----- Commodore-Amiga, Inc. -----*/
4 /* gameport.h */
5 /*----- gameport.h -----*/
6 /*----- gameport.h -----*/
7 /*----- gameport.h -----*/
8 *
9 * GamePort public definitions
10 *
11 *----- GamePort structures -----*/
12
13 /*----- GamePort commands -----*/
14 #define GPD_READEVENT (CMD_NONSTD+0)
15 #define GPD_ASKCTYPE (CMD_NONSTD+1)
16 #define GPD_SECTYPE (CMD_NONSTD+2)
17 #define GPD_ASKTRIGGER (CMD_NONSTD+3)
18 #define GPD_SETTRIGGER (CMD_NONSTD+4)
19
20 /*----- GamePort structures -----*/
21
22 /* gpt_Keys */
23 #define GPTB_DOWNKEYS 0
24 #define GPTD_DOWNKEYS (1<<0)
25 #define GPTB_UPKEYS 1
26 #define GPTD_UPKEYS (1<<1)
27
28 struct GamePortTrigger {
29     UWORD gpt_Keys; /* key transition triggers */
30     UWORD gpt_Timeout; /* time trigger (vertical blank units) */
31     UWORD gpt_XDelta; /* X distance trigger */
32     UWORD gpt_YDelta; /* Y distance trigger */
33 };
34
35 /*----- Controller Types -----*/
36 #define GPCT_ALLOCATED -1 /* allocated by another user */
37 #define GPCT_NOCONTROLLER 0
38
39 #define GPCT_MOUSE 1
40 #define GPCT_RELJOYSTICK 2
41 #define GPCT_ABSJOYSTICK 3
42
43
44 /*----- Errors -----*/
45 #define GPERR_SECTYPE 1 /* this controller not valid at this time */
46
47 #endif

```

```

1 #ifndef DEVICES_INPUT_H
2 #define DEVICES_INPUT_H
3 /*----- Commodore-Amiga, Inc. -----*/
4 /*
5 Commodore-Amiga, Inc.
6 input.h
7 */
8 *
9 * Input device command definitions
10 *
11 *
12 #ifndef EXEC_IO_H
13 #include "exec/10.h"
14 #endif
15
16 #define IND_ADDHANDLER (CMD_NONSTD*0)
17 #define IND_REMHANDLER (CMD_NONSTD*1)
18 #define IND_WRITEVENT (CMD_NONSTD*2)
19 #define IND_SETTRESH (CMD_NONSTD*3)
20 #define IND_SETPERIOD (CMD_NONSTD*4)
21 #define IND_SETMPORT (CMD_NONSTD*5)
22 #define IND_SETMYPE (CMD_NONSTD*6)
23 #define IND_SETMIAIG (CMD_NONSTD*7)
24
25 #endif

```

```

1 #ifndef DEVICES_INPUTEVENT_H
2 #define DEVICES_INPUTEVENT_H
3 /*----- Commodore-Amiga, Inc. -----*/
4 /*
5 Commodore-Amiga, Inc.
6 inputevent.h
7 */
8 *
9 * Input event definitions
10 *
11 *
12 #ifndef DEVICES_TIMER_H
13 #include "devices/timer.h"
14 #endif
15
16 /*----- constants -----*/
17
18 /* --- InputEvent.ie_Class --- */
19 /* A NOP input event */
20 #define IECLASS_NULL 0x00
21 /* A raw keycode from the keyboard device */
22 #define IECLASS_RAWKEY 0x01
23 /* The raw mouse report from the game port device */
24 #define IECLASS_RAWMOUSE 0x02
25 /* A private console event */
26 #define IECLASS_EVENT 0x03
27 /* A Pointer Position report */
28 #define IECLASS_POINTERPOS 0x04
29 /* A timer event */
30 #define IECLASS_TIMER 0x06
31 /* select button pressed down over a Gadget (address in ie_EventAddress) */
32 #define IECLASS_GADGETDOWN 0x07
33 /* select button released over the same Gadget (address in ie_EventAddress) */
34 #define IECLASS_GADGETUP 0x08
35 /* some Requester activity has taken place. See Codes REQCLEAR and REQSET */
36 #define IECLASS_REQUESTER 0x09
37 /* this is a Menu Number transmission (Menu number is in ie_Code) */
38 #define IECLASS_MENULIST 0x0A
39 /* User has selected the active Window's Close Gadget */
40 #define IECLASS_CLOSEWINDOW 0x0B
41 /* this Window has a new size */
42 #define IECLASS_SIZEWINDOW 0x0C
43 /* the Window pointed to by ie_EventAddress needs to be refreshed */
44 #define IECLASS_REFRESHWINDOW 0x0D
45 /* new preferences are available */
46 #define IECLASS_PREFEREFS 0x0E
47 /* the disk has been removed */
48 #define IECLASS_DISKREMOVED 0x0F
49 /* the disk has been inserted */
50 #define IECLASS_DISKINSERTED 0x10
51 /* the window is about to be been made active */
52 #define IECLASS_ACTIVIEWINDOW 0x11
53 /* the window is about to be made inactive */
54 #define IECLASS_INACTIVIEWINDOW 0x12
55
56

```

```

57 /* the last class */
58 #define IECLASS_MAX 0x12
59
60
61
62
63 /* --- InputEvent.ie_Code --- */
64 /* IECLASS_BANKEY */
65 #define IECODE_UP_PREFIX 0x80
66 #define IECODE_KEY_CODE_FIRST 0x00
67 #define IECODE_KEY_CODE_LAST 0x77
68 #define IECODE_CMD_CODE_FIRST 0x78
69 #define IECODE_CMD_CODE_LAST 0x7E
70
71 /* IECLASS_ANSI */
72 #define IECODE_C0_FIRST 0x00
73 #define IECODE_C0_LAST 0x1F
74 #define IECODE_ASCII_FIRST 0x20
75 #define IECODE_ASCII_LAST 0x7E
76 #define IECODE_ASCII_DEL 0x7E
77 #define IECODE_C1_FIRST 0x80
78 #define IECODE_C1_LAST 0x9F
79 #define IECODE_LATIN1_FIRST 0xA0
80 #define IECODE_LATIN1_LAST 0xFF
81
82 /* IECLASS_MOUSE */
83 #define IECODE_LBUTTON 0x68 /* also uses IECODE_UP_PREFIX */
84 #define IECODE_RBUTTON 0x69
85 #define IECODE_MBUTTON 0x6A
86 #define IECODE_MOBUTTON 0xFF
87
88 /* IECLASS_EVENT */
89 #define IECODE_NEWACTIVE 0x01 /* active input window changed */
90
91 /* IECLASS_REQUESTER_CODES */
92 #define REQSET is broadcast when the first Requester (not subsequent ones) opens
93 * in the Window
94
95 #define IECODE_REQSET 0x01
96 /* REQCLEAR is broadcast when the last Requester clears out of the Window */
97 #define IECODE_REQCLEAR 0x00
98
99
100 /* --- InputEvent.ie_Qualifier --- */
101 #define IEQUALIFIER_LSHIFT 0x0001
102 #define IEQUALIFIER_RSHIFT 0x0002
103 #define IEQUALIFIER_CAPSLOCK 0x0004
104 #define IEQUALIFIER_CONTROL 0x0008
105 #define IEQUALIFIER_ALT 0x0010
106 #define IEQUALIFIER_RALT 0x0020
107 #define IEQUALIFIER_COMMAND 0x0040
108 #define IEQUALIFIER_ROOMMAND 0x0080
109 #define IEQUALIFIER_NUMERICPAD 0x0100
110 #define IEQUALIFIER_REPEAT 0x0200
111 #define IEQUALIFIER_INTERRUPT 0x0400
112 #define IEQUALIFIER_MULTIBROADCAST 0x0800

```

```

113 #define IEQUALIFIER_LBUTTON 0x1000
114 #define IEQUALIFIER_RBUTTON 0x2000
115 #define IEQUALIFIER_MBUTTON 0x4000
116 #define IEQUALIFIER_RELATIVEMOUSE 0x8000
117
118 /* ----- InputEvent ----- */
119
120 struct InputEvent {
121     struct InputEvent *ie_NextEvent; /* the chronologically next event */
122     UBYTE ie_Class; /* the input event class */
123     UBYTE ie_SubClass; /* optional subclass of the class */
124     UWORD ie_Code; /* the input event code */
125     UWORD ie_Qualifier; /* qualifiers in effect for the event */
126     union {
127         WORD ie_X; /* the pointer position for the event */
128         WORD ie_Y;
129     } ie_XY;
130     APTR ie_addr;
131     } ie_position;
132     struct timeval ie_TimeStamp; /* the system tick at the event */
133 };
134
135 #define ie_X ie_position.ie_xy.ie_x
136 #define ie_Y ie_position.ie_xy.ie_y
137 #define ie_EventAddress ie_position.ie_addr
138
139
140 #endif

```

```

1 #ifndef DEVICES_KEYBOARD_H
2 #define DEVICES_KEYBOARD_H
3 /******
4 /* Commodore-Amiga, Inc.
5 /* keyboard.h
6 /******
7 /******
8 *
9 * Keyboard device command definitions
10 *
11 *
12 *
13 #ifndef EXEC_IO_H
14 #include "exec/io.h"
15 #endif
16 #define KBD_READEVENT (CMD_NONSTD*0)
17 #define KBD_READMATRIX (CMD_NONSTD*1)
18 #define KBD_ADDRESSETHANDLER (CMD_NONSTD*2)
19 #define KBD_RESETHANDLER (CMD_NONSTD*3)
20 #define KBD_RESETHANDLERDONE (CMD_NONSTD*4)
21 #endif

```

```

1 #ifndef DEVICES_KEYMAP_H
2 #define DEVICES_KEYMAP_H
3 /******
4 /* Commodore-Amiga, Inc.
5 /* keymap.h
6 /******
7 /******
8 *
9 * console.device key map definitions
10 *
11 *
12 *
13 struct KeyMap {
14     APTR km_LoKeyMapTypes;
15     APTR km_LoKeyMap;
16     APTR km_LoCapsable;
17     APTR km_HiKeyMapTypes;
18     APTR km_HiKeyMap;
19     APTR km_HiCapsable;
20     APTR km_HIRepeatable;
21 };
22
23 #define KCB_NOP 7
24 #define KCF_NOP 0x80
25
26 #define KC_NOQUAL 0
27 #define KC_VANILLA 7
28 #define KCF_SHIFT 0x01
29 #define KCF_ALT 0x02
30 #define KCB_CONTROL 2
31 #define KCF_CONTROL 0x04
32 #define KCB_DOWNUP 3
33 #define KCF_DOWNUP 0x08
34
35 #define KCB_STRING 6
36 #define KCF_STRING 0x40
37 #endif

```

/* note that SHIFT+ALT+CTRL is VANILLA */

```

1  #ifndef DEVICES_NARRATOR_H
2  #define DEVICES_NARRATOR_H
3  /*
4  /* Commodore-Amiga, Inc.
5  /* narrator.h
6  /*
7  #endif EXEC_IO_H
8  #include "exec/io.h"
9  #endif
10 #endif
11
12 /* Error Codes */
13
14 #define ND_NoMem -2 /* Can't allocate memory */
15 #define ND_NoAudLib -3 /* Can't open audio device */
16 #define ND_MakeBad -4 /* Error in MakeLibrary call */
17 #define ND_UnitErr -5 /* Unit other than 0 */
18 #define ND_CantAlloc -6 /* Can't allocate audio channel(s) */
19 #define ND_Unimpl -7 /* Unimplemented command */
20 #define ND_NoWlts -8 /* Read for mouth without write first */
21 #define ND_Expunged -9 /* Can't open, deferred expunge bit set */
22 #define ND_PhonErr -20 /* Phoneme code spelling error */
23 #define ND_RateErr -21 /* Rate out of bounds */
24 #define ND_PitchErr -22 /* Pitch out of bounds */
25 #define ND_SexErr -23 /* Sex not valid */
26 #define ND_ModeErr -24 /* Mode not valid */
27 #define ND_FreqErr -25 /* Sampling frequency out of bounds */
28 #define ND_VolErr -26 /* Volumes out of bounds */
29
30
31
32
33 /* Input parameters and defaults */
34 #define DEFPITCH 110 /* Default pitch */
35 #define DEFRRATE 150 /* Default speaking rate (wpm) */
36 #define DEFVOL 64 /* Default volumes (full) */
37 #define DEFREQ 22200 /* Default sampling frequency (Hz) */
38 #define DEFMALE 0 /* Male vocal tract */
39 #define DEFMALE 1 /* Female vocal tract */
40 #define NATURALE0 0 /* Natural pitch contours */
41 #define ROBOTICE0 1 /* Monotone */
42 #define DEFSEX MALE /* Default sex */
43 #define DEFMODE NATURALE0 /* Default mode */
44
45
46
47 /* Parameter bounds */
48
49 #define MINRATE 40 /* Minimum speaking rate */
50 #define MAXRATE 400 /* Maximum speaking rate */
51 #define MINPITCH 65 /* Minimum pitch */
52 #define MAXPITCH 320 /* Maximum pitch */
53 #define MINREQ 5000 /* Minimum sampling frequency */
54 #define MAXREQ 28000 /* Maximum sampling frequency */
55 #define MINVOL 0 /* Minimum volumes */
56 #define MAXVOL 64 /* Maximum volumes */

```

```

57
58
59
60 /* Standard Write request */
61
62 struct narrator_rb {
63     struct IOStdReq message; /* Standard IO RB
64     /* Speaking rate (words/minute)
65     /* Baseline pitch in Hertz
66     /* Pitch mode
67     /* Sex of voice
68     /* Pointer to audio alloc maps
69     /* Number of audio alloc maps
70     /* Volumes. 0 (off) thru 64
71     /* Audio sampling freq
72     /* If non-zero, generate mouths
73     /* Which ch mask used (internal)
74     /* Num ch masks used (internal)
75     /* For alignment
76     };
77
78
79
80 /* Standard Read request */
81
82
83
84 struct mouth_rb {
85     struct narrator_rb voice; /* Speech IO RB
86     /* Width (returned value)
87     /* Height (returned value)
88     /* Internal use, do not modify
89     /* For alignment
90     };
91
92 #endif DEVICES_NARRATOR_H

```



```

1  /*****
2  /** Commodore-Amiga, Inc.
3  /** parallel.h
4  /**
5  /**
6  /**
7  /**
8  /** external declarations for Parallel Port Driver
9  /**
10 /** SOURCE CONTROL
11 /** -----
12 /** $Header: parallel.h,v 25.0 85/03/27 19:14:15 temp Exp $
13 /**
14 /** $Locker:  $
15 /**
16 /**
17 /**
18 #ifndef DEVICES_PARALLEL_H
19 #define DEVICES_PARALLEL_H
20
21 #ifndef EXEC_IO_H
22 #include "exec/io.h"
23 #endif
24
25 struct IOArray {
26     ULONG PTermArray0;
27     ULONG PTermArray1;
28 };
29
30 /*****
31 /** CAUTION !! IF YOU ACCESS the parallel.device, you MUST (!!!!) use
32 an IOExtPar-sized structure or you may overlay innocent memory !! */
33 /*****
34
35 struct IOExtPar {
36     struct IOStcReq IOPar;
37
38     struct
39     {
40         0 APTR Succ
41         4 APTR Pred
42         8 UBYTE Type
43         9 UBYTE Pr-1
44         A APTR Name
45         E APTR ReplyPort
46         12 UMORD MMLength
47         14 APTR io_Device
48         18 APTR io_Unit
49         1C UMORD io_Command
50         1E UBYTE io_Flags
51         1F UBYTE io_Error
52         STRUCT IOStcReq
53         20 ULONG io_Actual
54         24 ULONG io_Length
55         28 APTR io_Data
56         2C ULONG io_Offset

```

```

57 * 30 */
58 ULONG io_PExtFlags; /* (not used) flag extension area */
59 UBYTE io_Status; /* status of parallel port and registers
60 io_ParFlags; /* see PARFLAGS bit definitions below */
61 struct IOArray io_PTermArray; /* termination character array */
62 };
63
64 #define PARB_SHARED 5 /* ParFlags non-exclusive access bit */
65 #define PARB_RAD_BOOGIE (1<<5) /* non-exclusive access mask */
66 #define PARB_RAD_BOOGIE 3 /* (not yet implemented) */
67 #define PARB_EOFMODE (1<<3) /* (not yet implemented) */
68 #define PARB_EOFMODE 1 /* EOF mode enabled bit */
69 #define IOFLACS EOF mode enabled mask */
70 #define IOFLACS rqst-queued bit */
71 #define IOFLACS rqst-queued mask */
72 #define IOFLACS rqst-aborted bit */
73 #define IOFLACS rqst-aborted mask */
74 #define IOFLACS rqst-que-or-current bit */
75 #define IOFLACS rqst-que-or-current mask */
76 #define IO_STATUS read=0,write=1 bit */
77 #define IO_STATUS read=0,write=1 mask */
78 #define IO_STATUS printer in busy toggle bit */
79 #define IO_STATUS printer in busy toggle mask */
80 #define IO_STATUS paper out bit */
81 #define IO_STATUS paper out mask */
82 #define IO_STATUS printer selected bit */
83 #define IO_STATUS printer selected mask */
84 #define IOPTB_PSEL (1<<0)
85 #define PARALLELNAME "parallel.devices"
86
87 #define PDCMD_QUERY (CMD_NONSTD)
88 #define PDCMD_SETPARAMS (CMD_NONSTD+1)
89
90 #define ParErr_DevBusy 1
91 #define ParErr_BufTooBig 2
92 #define ParErr_InvParam 3
93 #define ParErr_LineErr 4
94 #define ParErr_PortOpen 5
95 #define ParErr_PortReset 6
96 #define ParErr_InitErr 7
97
98 #endif DEVICES_PARALLEL_H

```

```

1 #ifndef DEVICES_PRINTER_H
2 #define DEVICES_PRINTER_H
3 /****** Commodore-Amiga, Inc. *****/
4 /****** printer.h *****/
5 /****** *****/
6 /****** *****/
7 /****** *****/
8 *
9 * printer device command definitions
10 *
11 * Source Control
12 *
13 * $Header: printer.h,v 1.2 85/10/09 16:16:10 kodlak Exp $
14 *
15 * $Locker: $
16 *
17 *
18 *
19 #ifndef EXEC_NODES_H
20 #include "exec/nodes.h"
21 #endif
22
23 #ifndef EXEC_LISTS_H
24 #include "exec/lists.h"
25 #endif
26
27 #ifndef EXEC_PORTS_H
28 #include "exec/ports.h"
29 #endif
30
31 #define PRD_RAMWRITE (CMD_NONSTD*0)
32 #define PRD_PRICOMMAND (CMD_NONSTD*1)
33 #define PRD_DUMPSPORT (CMD_NONSTD*2)
34
35 /* printer command definitions */
36
37 #define aRIS 0 /* ESCc reset ISO */
38 #define aRIN 1 /* ESC#1 initialize +++ */
39 #define aIND 2 /* ESCD lf ISO */
40 #define aNEL 3 /* ESCc return,lf ISO */
41 #define aRI 4 /* ESCM reverse lf ISO */
42
43 #define aSCR0 5 /* ESC[0m normal char set ISO */
44 #define aSCR3 6 /* ESC[3m italics on ISO */
45 #define aSCR23 7 /* ESC[23m italics off ISO */
46 #define aSCR4 8 /* ESC[4m underline on ISO */
47 #define aSCR24 9 /* ESC[24m underline off ISO */
48 #define aSCR1 10 /* ESC[1m boldface on ISO */
49 #define aSCR22 11 /* ESC[22m boldface off ISO */
50 #define aSC 12 /* SCR30-39 set foreground color ISO */
51 #define aSBC 13 /* SCR40-49 set background color ISO */
52
53 #define aSHORP0 14 /* ESC[0w normal pitch DEC */
54 #define aSHORP2 15 /* ESC[2w elite on DEC */
55 #define aSHORP1 16 /* ESC[1w elite off DEC */
56 #define aSHORP4 17 /* ESC[4w condensed line on DEC */

```

```

57 #define aSHORP3 18 /* ESC[3w condensed off DEC */
58 #define aSHORP6 19 /* ESC[6w enlarged on DEC */
59 #define aSHORP5 20 /* ESC[5w enlarged off DEC */
60
61 #define aDEN6 21 /* ESC[6"z shadow print on DEC (sort of) */
62 #define aDEN5 22 /* ESC[5"z shadow print off DEC */
63 #define aDEN4 23 /* ESC[4"z doublestrike on DEC */
64 #define aDEN3 24 /* ESC[3"z doublestrike off DEC */
65 #define aDEN2 25 /* ESC[2"z NLQ on DEC */
66 #define aDEN1 26 /* ESC[1"z NLQ off DEC */
67
68 #define aSUS2 27 /* ESC[2v superscript on +++ */
69 #define aSUS1 28 /* ESC[1v superscript off +++ */
70 #define aSUS4 29 /* ESC[4v subscript on +++ */
71 #define aSUS3 30 /* ESC[3v subscript off +++ */
72 #define aSUS0 31 /* ESC[0v normalize the line +++ */
73 #define aPLU 32 /* ESCL partial line up ISO */
74 #define aPLD 33 /* ESCL partial line down ISO */
75
76 #define aFNT0 34 /* ESC(B US char set DEC */
77 #define aFNT1 35 /* ESC(R French char set DEC */
78 #define aFNT2 36 /* ESC(K German char set DEC */
79 #define aFNT3 37 /* ESC(A UK char set DEC */
80 #define aFNT4 38 /* ESC(E Danish I char set DEC */
81 #define aFNT5 39 /* ESC(H Sweden char set DEC */
82 #define aFNT6 40 /* ESC(Y Italian char set DEC */
83 #define aFNT7 41 /* ESC(Z Spanish char set DEC */
84 #define aFNT8 42 /* ESC(J Japanese char set +++ */
85 #define aFNT9 43 /* ESC(6 Norwegian char set DEC */
86 #define aFNT10 44 /* ESC(C Danish II char set +++ */
87
88 #define aPROP2 45 /* ESC[2p proportional on +++ */
89 #define aPROP1 46 /* ESC[ip proportional clear +++ */
90 #define aPROP0 47 /* ESC[0p proportional clear +++ */
91 #define aTSS 48 /* ESC[n E set proportional offset ISO */
92 #define aJFY5 49 /* ESC[5 F auto left justify ISO */
93 #define aJFY7 50 /* ESC[7 F auto right justify ISO */
94 #define aJFY6 51 /* ESC[6 F auto full justify ISO */
95 #define aJFY0 52 /* ESC[0 F auto justify off ISO */
96 #define aJFY3 53 /* ESC[3 F letter space (justify) ISO (special) */
97 #define aJFY1 54 /* ESC[1 F word fill(auto center) ISO (special) */
98
99 #define aVERP0 55 /* ESC[0z 1/8" line spacing +++ */
100 #define aVERP1 56 /* ESC[1z 1/6" line spacing +++ */
101 #define aLPP 57 /* ESC[nt set form length n DEC */
102 #define aPERF 58 /* ESC[nt perf skip n (n>0) +++ */
103 #define aPERF0 59 /* ESC[0q perf skip off +++ */
104
105 #define aLMS 60 /* ESC#9 Left margin set +++ */
106 #define aRMS 61 /* ESC#0 Right margin set +++ */
107 #define aIMS 62 /* ESC#8 Top margin set +++ */
108 #define aBMS 63 /* ESC#2 Bottom marg set +++ */
109 #define aSTRM 64 /* ESC[Pn1:Pn2 T&B margins DEC */
110 #define aSLRM 65 /* ESC[Pn1:Pn2s L&R margin DEC */
111 #define aCAN 66 /* ESC#3 Clear margins +++ */
112

```

```

113 #define aHTS 67 /* ESCH Set horiz tab
114 #define aVTS 68 /* ESCJ Set vertical tabs
115 #define aTBC0 69 /* ESC[0g Clr horiz tab
116 #define aTBC3 70 /* ESC[3g Clr all h tab
117 #define aTBC1 71 /* ESC[1g Clr vertical tabs
118 #define aTBC4 72 /* ESC[4g Clr all v tabs
119 #define aTBCALL 73 /* ESC#4 Clr all h & v tabs
120 #define aTBSALL 74 /* ESC#5 Set default tabs
121 #define aEXTEND 75 /* ESC[Pn" x extended commands
122
123 struct IOPrtCmdReq {
124     struct Message io_Message;
125     struct Device *io_Device;
126     struct Unit *io_Unit;
127     UMWORD io_Command;
128     UBYTE io_Flags;
129     BYTE io_Error;
130     UMWORD io_PrtCommand;
131     UBYTE io_Parm0;
132     UBYTE io_Parm1;
133     UBYTE io_Parm2;
134     UBYTE io_Parm3;
135 };
136
137 struct IOERPRReq {
138     struct Message io_Message;
139     struct Device *io_Device;
140     struct Unit *io_Unit;
141     UMWORD io_Command;
142     UBYTE io_Flags;
143     BYTE io_Error;
144     struct RastPort *io_RastPort;
145     struct ColorMap *io_ColorMap;
146     ULONG io_Modes;
147     UMWORD io_SrcX;
148     UMWORD io_SrcY;
149     UMWORD io_SrcWidth;
150     UMWORD io_SrcHeight;
151     LONG io_DestCols;
152     LONG io_DestRows;
153     UMWORD io_Special;
154 };
155
156 #define SPECIAL_MILCOOLS 0x001
157 #define SPECIAL_MILROWS 0x002
158 #define SPECIAL_FULCOOLS 0x004
159 #define SPECIAL_FULROWS 0x008
160 #define SPECIAL_FRACOOLS 0x010
161 #define SPECIAL_FRAROWS 0x020
162 #define SPECIAL_ASPECT 0x080
163 #define SPECIAL_DENSITYMASK 0xf00
164 #define SPECIAL_DENSITY1 0x100
165 #define SPECIAL_DENSITY2 0x200
166 #define SPECIAL_DENSITY3 0x300
167 #define SPECIAL_DENSITY4 0x400
168
169 #define ISO */
170 #define PDERR_CANCEL 1
171 #define PDERR_NOTGRAPHICS 2
172 #define PDERR_INVERTHAM 3
173 #define PDERR_BADDIMENSION 4
174 #define PDERR_DIMENSIONTOLOW 5
175 #define PDERR_INTERNALMEMORY 6
176 #define PDERR_BUFFERMEMORY 7
177 #endif
178
179 /* user canceled a printer timeout */
180 /* printer cannot output graphics */
181 /* cannot invert hold & modify print */
182 /* print dimensions illegal */
183 /* print dimensions too large */
184 /* no memory for internal variables */
185 /* no memory for print buffer */

```

```

169 #define PDERR_CANCEL 1
170 #define PDERR_NOTGRAPHICS 2
171 #define PDERR_INVERTHAM 3
172 #define PDERR_BADDIMENSION 4
173 #define PDERR_DIMENSIONTOLOW 5
174 #define PDERR_INTERNALMEMORY 6
175 #define PDERR_BUFFERMEMORY 7
176 #endif
177
178 /* user canceled a printer timeout */
179 /* printer cannot output graphics */
180 /* cannot invert hold & modify print */
181 /* print dimensions illegal */
182 /* print dimensions too large */
183 /* no memory for internal variables */
184 /* no memory for print buffer */

```

```

1  /*****
2  /* Commodore-Amiga, Inc.
3  /* prtbase.h
4  /*
5  /*
6  *
7  * printer device data definition
8  *
9  *****/
10
11 #ifndef DEVICES_PRTBASE_H
12 #define DEVICES_PRTBASE_H
13
14
15 #ifndef EXEC_NODES_H
16 #include "exec/nodes.h"
17 #endif
18 #ifndef EXEC_LISTS_H
19 #include "exec/lists.h"
20 #endif
21 #ifndef EXEC_PORTS_H
22 #include "exec/ports.h"
23 #endif
24 #ifndef EXEC_LIBRARIES_H
25 #include "exec/libraries.h"
26 #endif
27 #ifndef EXEC_TASKS_H
28 #include "exec/tasks.h"
29 #endif
30
31 #ifndef DEVICES_PARALLEL_H
32 #include "devices/parallel.h"
33 #endif
34 #ifndef DEVICES_SERIAL_H
35 #include "devices/serial.h"
36 #endif
37 #ifndef DEVICES_TIMER_H
38 #include "devices/timer.h"
39 #endif
40 #ifndef LIBRARIES_DOSEXTENS_I
41 #include "libraries/dosextns.h"
42 #endif
43 #ifndef INTUITION_INTUITION_H
44 #include "intuition/intuition.h"
45 #endif
46
47
48 struct DeviceData {
49     struct Library dd_Device; /* standard library node */
50     APTR dd_Segment; /* A0 when initialized */
51     APTR dd_ExecBase; /* A6 for exec */
52     APTR dd_CmdVectors; /* command table for device commands */
53     APTR dd_CmdBytes; /* bytes describing which command queue */
54     UWORD dd_NumCommands; /* the number of commands supported */
55 };
56

```

```

57 #define P_STKSIZE 0x800
58
59 struct PrinterData {
60     struct DeviceData pd_Device;
61     struct MsgPort pd_Unit; /* the one and only unit */
62     struct pd_PrinterSegment; /* the printer specific segment */
63     UWORD pd_PrinterType; /* the segment printer type */
64     struct PrinterSegment *pd_SegmentData; /* the segment data structure */
65     UBYTE *pd_PrintBuf; /* the raster print buffer */
66     VOID (*pd_PWrite)(); /* the write function */
67     VOID (*pd_PBothReady)(); /* port I/O request 0 */
68     union {
69         struct IOExtPar pd_p0;
70         struct IOExtSer pd_p1;
71     } pd_IOR0;
72     #define pd_PIOR0 pd_IOR0.pd_p0
73     #define pd_PSIOR0 pd_IOR0.pd_p1
74     union {
75         struct IOExtPar pd_p1;
76         struct IOExtSer pd_p2;
77     } pd_IOR1;
78     #define pd_PIOR1 pd_IOR1.pd_p1
79     #define pd_PSIOR1 pd_IOR1.pd_p2
80     struct timerrequest pd_TIMER; /* timer I/O request */
81     struct MsgPort pd_IORPort; /* and messages reply port */
82     struct Task pd_TC; /* write task */
83     UBYTE pd_Stk[P_STKSIZE]; /* and stack space */
84     UBYTE pd_Flags; /* device flags */
85     UBYTE pd_pads;
86     struct Preferences pd_Preferences; /* the latest preferences */
87     UBYTE pd_PWaitEnabled; /* wait function switch */
88 };
89
90 #define PPCB_GEX 0
91 #define PPCB_GEX 0x01
92 #define PPCB_COLOR 1
93 #define PPCB_COLOR 0x02
94 #define PPCB_ALPHA 0
95 #define PPCB_ALPHA 1
96 #define PPCB_BW 1
97 #define PPCB_BW 2
98 #define PPCB_YMC 3
99 #define PPCB_YMC 4
100 #define POC_BW 1
101 #define POC_YMC 2
102 #define POC_YMC_BW 3
103 #define POC_YMC_BW 4
104
105 struct PrinterExtendedData {
106     char *ped_PrinterName; /* printer name, null terminated */
107     VOID (*ped_Init)(); /* called after LoadSeg */
108     VOID (*ped_Expunge)(); /* called before UnLoadSeg */
109     VOID (*ped_Open)(); /* called at OpenDevice */
110     VOID (*ped_Close)(); /* called at CloseDevice */
111     UBYTE ped_PrinterClass; /* printer class */
112     UBYTE ped_ColorClass; /* color class */

```

```

113 UBYTE ped_MaxColumns; /* number of print columns available */
114 UBYTE ped_NumCharSets; /* number of character sets */
115 UWORD ped_NumRows; /* number of raster rows in a raster dump */
116 ULONG ped_MaxDots; /* number of dots maximum in a raster dump */
117 UWORD ped_XDotsInch; /* number of dots maximum in a raster dump */
118 UWORD ped_YDotsInch; /* horizontal dot density */
119 char **ped_Commands; /* vertical dot density */
120 VOID (*ped_DeSpecial)(); /* printer text command table */
121 VOID (*ped_Render)(); /* special command handler */
122 LONG ped_TimeoutSecs; /* raster render function */
123 /* good write timeout */
124 };
125
126 struct PrinterSegment {
127     ULONG ps_NextSegment; /* (actually a BPTR) */
128     ULONG ps_runAlert; /* MOVEQ #0,D0 : RTS */
129     UWORD ps_Version; /* segment version */
130     struct PrinterExtendedData ps_PED; /* segment revision */
131     /* printer extended data */
132 };
133 #endif

```

```

1 /******
2 /* Commodore-Amiga, Inc.
3 /* serial.h
4 /******
5 /******
6 /* external declarations for Serial Port Driver
7
8
9 * SOURCE CONTROL
10 * -----
11 * $Header: serial.h,v 25.0 85/03/27 19:14:15 temp Exp $
12
13 * $Locker:  $
14
15 /******
16 #ifndef DEVICES_SERIAL_H
17 #define DEVICES_SERIAL_H
18
19 #ifndef EXEC_IO_H
20 #include "exec/io.h"
21 #endif !EXEC_IO_H
22
23 /* array of termination char's */
24 /* to use, see serial.doc setparams */
25
26 struct IOArray {
27     ULONG TermArray0;
28     ULONG TermArray1;
29 };
30
31 /******
32 /* CAUTION !! IF YOU ACCESS the serial.device, you MUST (!!!!) use
33 /* an IOExtSer-sized structure or you may overlay innocent memory !!!
34 /******
35 struct IOExtSer {
36     struct IOStdReq IOStdReq;
37
38     STRUCT MsgNode
39     * 0 APTR Succ
40     * 4 APTR Pred
41     * 8 UBYTE Type
42     * 9 UBYTE Pri
43     * A APTR Name
44     * E APTR ReplyPort
45     * 12 UWORD MNLlength
46     * STRUCT IOExt
47     * 14 APTR io_Device
48     * 18 APTR io_Unit
49     * 1C UWORD io_Command
50     * 1E UBYTE io_Flags
51     * 1F UBYTE io_Error
52     * STRUCT IOStdExt
53     * 20 ULONG io_Actual
54     * 24 ULONG io_Length
55     * 28 APTR io_Data
56     * 2C ULONG io_Offset

```

```

57 *
58 *
59 * IMPORTANT !! DON'T CHANGE the long-word alignment of ANY of these fields !!
60 * You can add to the end if you must do something.
61 *
62 * 30 */
63 ULONG io_CtlChar; /* control char's (order = xON, xOFF, IRQ, ACK) */
64 ULONG io_RBufLen; /* length in bytes of serial port's read buffer */
65 ULONG io_ExtFlags; /* (not used) flag extension area */
66 ULONG io_Baud; /* baud rate requested (true baud) */
67 ULONG io_BkTlms; /* duration of break signal in MICROseconds */
68 struct IOTArray io_TermArray; /* termination character array */
69 UBYTE io_ReadLen; /* bits per read character (bit count) */
70 UBYTE io_WriteLen; /* bits per write character (bit count) */
71 UBYTE io_StopBits; /* stopbits for read (count) */
72 UWORD io_SerFlags; /* see SerFlags bit definitions below */
73 UWORD io_Status;
74 ); /* status of serial port, as follows:
75 BIT ACTIVE FUNCTION
76 0 low busy
77 1 low paper out
78 2 low select
79 3 low Data Set Ready
80 4 low Clear To Send
81 5 low Carrier Detect
82 6 low Ready To Send
83 7 low Data Terminal Ready
84 8 high read overrun
85 9 high break sent.
86 10 high break received
87 11 high transmit x-OFFed
88 12 high receive x-OFFed
89 13-15 (not) reserved
90 */
91
92 #define SDCMD_QUERY CMD_NONSTD
93 #define SDCMD_BREAK (CMD_NONSTD+1)
94 #define SDCMD_SETPARAMS (CMD_NONSTD+2)
95
96 #define SERB_XDISABLED 7 /* SerFlags xOn-xOff feature disabled bit */
97 #define SERB_XDISABLED (1<<7) /* xOn-xOff feature disabled mask */
98 #define SERB_EOFMODE 6 /* EOF mode enabled bit */
99 #define SERB_EOFMODE (1<<6) /* EOF mode enabled mask */
100 #define SERB_SHARED 5 /* non-exclusive access bit */
101 #define SERB_SHARED (1<<5) /* non-exclusive access mask */
102 #define SERB_RAD_BOOGIE 4 /* high-speed mode active bit */
103 #define SERB_RAD_BOOGIE (1<<4) /* high-speed mode active mask */
104 #define SERB_QUEUEDERRK 3 /* queue this Break ioRqst */
105 #define SERB_QUEUEDERRK (1<<3) /* queue this Break ioRqst mask */
106 #define SERB_7WIRE 2 /* RS232 7-wire protocol */
107 #define SERB_7WIRE (1<<2) /* RS232 7-wire protocol mask */
108 #define SERB_PARTY_ODD 1 /* parity feature enabled bit */
109 #define SERB_PARTY_ODD (1<<1) /* parity feature enabled mask */
110 #define SERB_PARTY_ON 0 /* parity-enabled bit */
111 #define SERB_PARTY_ON (1<<0) /* parity-enabled mask */
112 #define IOSERB_BUFREAD 7 /* io_Flags from read buffer bit */

```

```

113 #define IOSERB_BUFREAD (1<<7) /* from read buffer mask */
114 #define IOSERB_QUEUED 6 /* rqst-queued bit */
115 #define IOSERB_QUEUED (1<<6) /* rqst-queued mask */
116 #define IOSERB_ABORT 5 /* rqst-aborted bit */
117 #define IOSERB_ABORT (1<<5) /* rqst-aborted mask */
118 #define IOSERB_ACTIVE 4 /* rqst-quad-or-current bit */
119 #define IOSERB_ACTIVE (1<<4) /* rqst-quad-or-current mask */
120 #define IOSTB_XOFFERREAD 4 /* receive currently xOFF'ed bit */
121 #define IOSTB_XOFFERREAD (1<<4) /* receive currently xOFF'ed mask */
122 #define IOSTB_XOFFWRITE 3 /* transmit currently xOFF'ed bit */
123 #define IOSTB_XOFFWRITE (1<<3) /* transmit currently xOFF'ed mask */
124 #define IOSTB_READBREAK 2 /* break was latest input bit */
125 #define IOSTB_READBREAK (1<<2) /* break was latest input mask */
126 #define IOSTB_WRITEBREAK 1 /* break was latest output bit */
127 #define IOSTB_WRITEBREAK (1<<1) /* break was latest output mask */
128 #define IOSTB_OVERRUN 0 /* status word RBE overrun bit */
129 #define IOSTB_OVERRUN (1<<0) /* status word RBE overrun mask */
130
131 #define SerErr_DevBusy 1
132 #define SerErr_BaudMismatch 2
133 #define SerErr_InvBaud 3
134 #define SerErr_BufErr 4
135 #define SerErr_InvParam 5
136 #define SerErr_LineErr 6
137 #define SerErr_NotOpen 7
138 #define SerErr_PortReset 8
139 #define SerErr_ParityErr 9
140 #define SerErr_InitErr 10
141 #define SerErr_TimerErr 11
142 #define SerErr_BufOverflow 12
143 #define SerErr_NoDSR 13
144 #define SerErr_NoCTS 14
145 #define SerErr_DetectedBreak 15
146
147 #define SERIALNAME "serial.device"
148
149 #endif IDEVICES_SERIAL_H

```

```

1 /******
2 /* Commodore-Amiga, Inc.
3 /* timer.h
4 /******
5 /******
6 *
7 * SOURCE CONTROL
8 * -----
9 * $Header: timer.h,v 27.1 85/06/24 13:32:37 neil Exp $
10 *
11 * $Locker: $
12 *
13 *
14
15 #ifndef DEVICES_TIMER_H
16 #define DEVICES_TIMER_H
17
18 #ifndef EXEC_IO_H
19 #include "exec/io.h"
20 #endif EXEC_IO_H
21
22 /* unit definitions */
23 #define UNIT_MICROHZ 0
24 #define UNIT_VBLANK 1
25
26 #define TIMERNAME "timer.device"
27
28 struct timeval {
29     ULONG tv_secs;
30     ULONG tv_micro;
31 };
32
33 struct timerrequest {
34     struct IOrequest tr_node;
35     struct timeval tr_time;
36 };
37
38 /* IO_COMMAND to use for adding a timer */
39 #define TR_ADDRESSREQUEST CMD_NONSTD
40 #define TR_GETSYSTIME (CMD_NONSTD+1)
41 #define TR_SETSYSTIME (CMD_NONSTD+2)
42
43 #endif DEVICES_TIMER_H

```

```

1 /******
2 /* Commodore-Amiga, Inc.
3 /* trackdisk.h
4 /******
5 /******
6 *
7 * trackdisk.h
8 *
9 * Source Control
10 * -----
11 *
12 * $Header: trackdisk.h,v 27.3 85/07/12 23:16:05 neil Exp $
13 *
14 * $Locker: $
15 *
16 *
17 *
18 #ifndef DEVICES_TRACKDISK_H
19 #define DEVICES_TRACKDISK_H
20
21 #ifndef EXEC_IO_H
22 #include "exec/io.h"
23 #endif EXEC_IO_H
24
25 /*
26 *
27 *
28 * Physical drive constants
29 *
30 *
31 *
32 */
33
34 #define NUMCYLS 80 /* normal # of cylinders */
35 #define MAXCYLS (NUMCYLS+20) /* max # cyls to look for during cal */
36 #define NUMSECS 11
37 #define NUMHEADS 2
38 #define MAXRETRY 10
39 #define NUMTRACKS (NUMCYLS*NUMHEADS)
40 #define NUMUNITS 4
41
42 /*
43 *
44 * Useful constants
45 *
46 *
47 *
48 */
49
50 /*-- sizes before mfm encoding */
51 #define TD_SECTOR 512 /* log TD_SECTOR */
52 #define TD_SECSHIFT 9
53
54 /*
55 *
56 *

```

```

57 * Driver Specific Commands
58 *
59 *-----
60 */
61 /*
62 *--- TD_NAME is a generic macro to get the name of the driver. This
63 *--- way if the name is ever changed you will pick up the change
64 *--- automatically.
65 *--- Normal usage would be:
66 *--- char InternalName[] = TD_NAME;
67 *---
68 *---
69 *---
70 *---
71 */
72 #define TD_NAME "trackdisk.device"
73 #define TDE_EXTOOM (1<<15) /* for internal use only! */
74
75 #define TD_MOTOR (CMD_NONSTD+0) /* control the disk's motor */
76 #define TD_SEEK (CMD_NONSTD+1) /* explicit seek (for testing) */
77 #define TD_FORMAT (CMD_NONSTD+2) /* format disk */
78 #define TD_REMOVE (CMD_NONSTD+3) /* notify when disk changes */
79 #define TD_CHANGENUM (CMD_NONSTD+4) /* number of disk changes */
80 #define TD_CHANGESTATE (CMD_NONSTD+5) /* is there a disk in the drive? */
81 #define TD_PROTSTATUS (CMD_NONSTD+6) /* is the disk write protected? */
82 #define TD_LASTOOM TD_PROTSTATUS
83
84 /*
85 *
86 * The disk driver has an "extended command" facility. These commands
87 * take a superset of the normal IO Request block.
88 *
89 *
90 *
91 *
92 *
93 */
94 #define ETD_WRITE (CMD_WRITE|TDE_EXTOOM)
95 #define ETD_READ (CMD_READ|TDE_EXTOOM)
96 #define ETD_MOTOR (TD_MOTOR|TDE_EXTOOM)
97 #define ETD_SEEK (TD_SEEK|TDE_EXTOOM)
98 #define ETD_FORMAT (TD_FORMAT|TDE_EXTOOM)
99 #define ETD_UPDATE (CMD_UPDATE|TDE_EXTOOM)
100 #define ETD_CLEAR (CMD_CLEAR|TDE_EXTOOM)
101
102 /*
103 *
104 * extended IO has a larger than normal io request block.
105 *
106 *
107 */
108 struct IOExtTD {
109     struct IOStdReq iotd_Req;
110     ULONG iotd_Count;
111     ULONG iotd_SecLabel;
112

```

```

113 };
114
115 /* labels are TD_LABELSIZE bytes per sector */
116 #define TD_LABELSIZE 16
117
118 *-----
119 *
120 *
121 * Driver error defines
122 *
123 *-----
124 */
125
126 #define TDERR_NotSpecified 20
127 #define TDERR_NoSecHdr 21
128 #define TDERR_BadSecPreamble 22
129 #define TDERR_BadSecID 23
130 #define TDERR_BadHdrSum 24
131 #define TDERR_BadSecSum 25
132 #define TDERR_TooFewSecs 26
133 #define TDERR_BadSecHdr 27
134 #define TDERR_WriteProt 28
135 #define TDERR_DiskChanged 29
136 #define TDERR_SeekError 30
137 #define TDERR_NoMem 31
138 #define TDERR_BadUnitNum 32
139 #define TDERR_BadDriveType 33
140 #define TDERR_DriveInUse 34
141
142
143 #endif DEVICES_TRACKDISK_H

```



```

1 #ifndef EXEC_ALERTS_H
2 #define EXEC_ALERTS_H
3 /*****
4 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
5 *
6 *
7 *
8 *
9 * Source Control:
10 *
11 * $Header: alerts.h,v 1.0 85/08/28 15:05:44 carl Exp $
12 *
13 * $Locker: $
14 *
15 *
16 *
17 #define SE_ALERTWACK (1<<1) /* in ExecBase.SysFlag */
18
19 /*****
20 *
21 *
22 * Format of the alert error number:
23 *
24 * +-----+-----+-----+-----+
25 * |D| SubSysId | General Error | SubSystem Specific Error |
26 * +-----+-----+-----+-----+
27 *
28 * D: DeadEnd alert
29 * SubSysId: indicates ROM subsystem number.
30 * General Error: roughly indicates what the error was
31 * Specific Error: indicates more detail
32 *
33 /*****
34 *
35 *
36 * General Dead-End Alerts
37 *
38 /*****
39 *
40 * alert types */
41 #define AT_DeadEnd 0x80000000
42 #define AT_Recovery 0x00000000
43
44 /***** general purpose alert codes */
45 #define AG_NoMemory 0x00010000
46 #define AG_MakeLib 0x00020000
47 #define AG_OpenLib 0x00030000
48 #define AG_OpenDev 0x00040000
49 #define AG_OpenRes 0x00050000
50 #define AG_IOError 0x00060000
51
52 /***** alert objects: */
53 #define AO_ExecLib 0x00008001
54 #define AO_GraphicsLib 0x00008002
55 #define AO_LayersLib 0x00008003
56 #define AO_Intuition 0x00008004

```

```

57 #define AO_MathLib 0x00008005
58 #define AO_DOSLib 0x00008006
59 #define AO_DOSTLib 0x00008007
60 #define AO_BAWLib 0x00008008
61 #define AO_IconLib 0x00008009
62 #define AO_AudioDev 0x00008010
63 #define AO_ConsoleDev 0x00008011
64 #define AO_GamePortDev 0x00008012
65 #define AO_KeyboardDev 0x00008013
66 #define AO_TrackDiskDev 0x00008014
67 #define AO_TimerDev 0x00008015
68 #define AO_DiskRsrc 0x00008020
69 #define AO_MiscRsrc 0x00008021
70 #define AO_MiscRsrc 0x00008022
71 #define AO_BootStrap 0x00008030
72 #define AO_Workbench 0x00008031
73
74 /*****
75 *
76 *
77 * Specific Dead-End Alerts:
78 *
79 /*****
80 *
81 *
82 * exec.library */
83 #define AN_ExecLib 0x10000000
84 #define AN_ExecVect 0x81000001
85 #define AN_BaseChkSum 0x81000002
86 #define AN_LibChkSum 0x81000003
87 #define AN_LibMem 0x81000004
88 #define AN_MemCorrupt 0x81000005
89 #define AN_IntrMem 0x81000006
90 #define AN_InitAPtr 0x81000007
91
92 /***** graphics.library */
93 #define AN_GraphicsLib 0x20000000
94 #define AN_CopDisplay 0x82010001
95 #define AN_CopInstr 0x82010002
96 #define AN_CopListOver 0x82000003
97 #define AN_CopListHead 0x82010004
98 #define AN_LongFrame 0x82010005
99 #define AN_ShortFrame 0x82010006
100 #define AN_FloodFill 0x82010007
101 #define AN_TextTempRas 0x02010008
102 #define AN_BitBMap 0x82010009
103
104 /***** layers.library */
105 #define AN_LayersLib 0x03000000
106
107 /***** intuition.library */
108 #define AN_Intuition 0x04000000
109 #define AN_GadgetType 0x84000001
110 #define AN_BadGadget 0x04000002
111 #define AN_CreatePort 0x84010002
112 #define AN_ItemAlloc 0x04010003

```

```

/* 68000 exception vector checksum */
/* execbase checksum */
/* library checksum failure */
/* no memory to make library */
/* corrupted memory list */
/* no memory for interrupt servers */
/* InitStruct() of an APTX source */
/* copper display list, no memory */
/* copper instruction list, no memory */
/* copper list overload */
/* copper intermediate list overload */
/* copper list head, no memory */
/* long frame, no memory */
/* short frame, no memory */
/* flood fill, no memory */
/* text, no memory for ImpRas */
/* BitBMap, no memory */
/* unknown gadget type */
/* Recovery form of AN_GadgetType */
/* create port, no memory */
/* item plane alloc, no memory */

```

```

113 #define AN_SubAlloc      0x04010004 /* sub alloc, no memory */
114 #define AN_PlaneAlloc   0x84010005 /* plane alloc, no memory */
115 #define AN_ItemBoxTop   0x84000006 /* item box top < RealZero */
116 #define AN_OpenScreen   0x84010007 /* open screen, no memory */
117 #define AN_OpenScrnrast 0x84010008 /* open screen, raster alloc, no memory */
118 #define AN_SysScrType   0x84000009 /* open sys screen, unknown type */
119 #define AN_AddSMcadget  0x8401000A /* add SW gadgets, no memory */
120 #define AN_OpenWindow   0x8401000B /* open window, no memory */
121 #define AN_BadState     0x8400000C /* Bad State Return entering Intuition */
122 #define AN_BadMessage   0x8400000D /* Bad Message received by IDCMP */
123 #define AN_WeirdEcho    0x8400000E /* Weird echo causing incomprehension */
124 #define AN_NoConsole    0x8400000F /* couldn't open the Console Device */
125
126
127 /*----- math.library */
128 #define AN_MathLib      0x05000000
129
130 /*----- clist.library */
131 #define AN_CListLib    0x06000000
132
133 /*----- dos.library */
134 #define AN_DOSLib     0x07000000
135 #define AN_ScartMem   0x07010001 /* no memory at startup */
136 #define AN_EndTask    0x07000002 /* EndTask didn't */
137 #define AN_OpktFail   0x07000003 /* Opkt failure */
138 #define AN_AsyncPkt   0x07000004 /* Unexpected packet received */
139 #define AN_FreeVec    0x07000005 /* Freevec failed */
140 #define AN_DiskBlkSeq 0x07000006 /* Disk block sequence error */
141 #define AN_BitMap     0x07000007 /* Bitmap corrupt */
142 #define AN_KeyFree    0x07000008 /* Key already free */
143 #define AN_BadChkSum  0x07000009 /* Invalid checksum */
144 #define AN_DiskError  0x0700000A /* Disk Error */
145 #define AN_KeyRange   0x0700000B /* Key out of range */
146 #define AN_BadOverlay 0x0700000C /* Bad overlay */
147
148 /*----- ramlib.library */
149 #define AN_RAMLib     0x08000000
150
151 /*----- ramlib.library */
152 #define AN_IconLib    0x09000000
153
154 /*----- audio.device */
155 #define AN_AudioDev   0x10000000
156
157 /*----- console.device */
158 #define AN_ConsoleDev 0x11000000
159
160 /*----- gameport.device */
161 #define AN_GamePortDev 0x12000000
162
163 /*----- keyboard.device */
164 #define AN_KeyboardDev 0x13000000
165
166 /*----- trackdisk.device */
167 #define AN_TrackDiskDev 0x14000000
168 #define AN_IDCallbSeek 0x14000001 /* calibrate: seek error */

```

```

169 #define AN_TTDelay     0x14000002 /* delay: error on timer wait */
170 /*----- timer.device */
171 #define AN_TimerDev    0x15000000
172 #define AN_TMBadReq    0x15000001 /* bad request */
173
174 /*----- cia.resource */
175 #define AN_CIARsrc     0x20000000
176
177 /*----- disk.resource */
178 #define AN_DiskRsrc    0x21000000
179 #define AN_DRHasDisk  0x21000001 /* get unit: already has disk */
180 #define AN_DRIntNoAct 0x21000002 /* interrupt: no active unit */
181
182 /*----- misc.resource */
183 #define AN_MiscRsrc    0x22000000
184
185 /*----- bootstrap */
186 #define AN_BootStrap   0x30000000
187 #define AN_BootError   0x30000001 /* boot code returned an error */
188
189 /*----- Woribench */
190 #define AN_Woribench   0x31000000
191
192 #endif !EXEC_ALERTS_H

```

```

1  #ifndef EXEC_DEVICES_H
2  #define EXEC_DEVICES_H
3  /*****
4  *
5  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: devices.h,v 1.0 85/08/28 15:06:50 carl Exp $
12 *
13 * $Locker:  $
14 *
15 *
16 *
17 #ifndef EXEC_LIBRARIES_H
18 #include "exec/libraries.h"
19 #endif !EXEC_LIBRARIES_H
20
21 #ifndef EXEC_PORTS_H
22 #include "exec/ports.h"
23 #endif !EXEC_PORTS_H
24
25 /***** Device *****/
26
27 struct Device {
28     struct Library dd_Library;
29 };
30
31 /***** Unit *****/
32
33 struct Unit {
34     struct MsgPort *unit_MsgPort; /* queue for unprocessed messages */
35     UBYTE unit_flags;
36     UBYTE unit_pad;
37     UWORD unit_OpenCnt; /* number of active opens */
38 };
39
40 #define UNITE_ACTIVE (1<<0)
41 #define UNITE_INTASK (1<<1)
42
43 #endif

```

```

1  /*****
2  *
3  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
4  *
5  *
6  *
7  * Source Control:
8  *
9  * $Header: errors.h,v 1.0 85/08/28 15:07:14 carl Exp $
10 *
11 * $Locker:  $
12 *
13 *
14 *
15 *
16 #define IOERR_OPENFAIL -1 /* device/unit failed to open */
17 #define IOERR_ABORTED -2 /* request aborted */
18 #define IOERR_NOCMD -3 /* command not supported */
19 #define IOERR_BADLENGTH -4 /* not a valid length */

```

Dec 8 16:37 1985 exec/exec.h Page 1

```
1 #include "exec/nodes.h"
2 #include "exec/lists.h"
3 #include "exec/interrupts.h"
4 #include "exec/memory.h"
5 #include "exec/ports.h"
6 #include "exec/tasks.h"
7 #include "exec/libraries.h"
8 #include "exec/devices.h"
9 #include "exec/lo.h"
10
11
```

Dec 8 16:37 1985 exec/execbase.h Page 1

```
1 #ifndef EXEC_EXECBASE_H
2 #define EXEC_EXECBASE_H
3 /*
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *
8 * Source Control:
9 *
10 * $Header: execbase.h,v 1.1 85/11/12 16:10:26 carl Exp $
11 *
12 * $Locker: carl $
13 *
14 *
15 *
16 *
17 #ifndef EXEC_LISTS_H
18 #include "exec/libraries.h"
19 #endif !EXEC_LISTS_H
20
21 #ifndef EXEC_INTERRUPTS_H
22 #include "exec/interrupts.h"
23 #endif !EXEC_INTERRUPTS_H
24
25 #ifndef EXEC_LIBRARIES_H
26 #include "exec/libraries.h"
27 #endif !EXEC_LIBRARIES_H
28
29 #ifndef EXEC_TASKS_H
30 #include "exec/tasks.h"
31 #endif !EXEC_TASKS_H
32
33
34 struct ExecBase {
35     struct Library LibNode;
36
37     UWORD SoftVer; /* kickstart release number */
38     UWORD LowMemChkSum; /* system base pointer complement */
39     ULONG ChkBase; /* coldstart soft vector */
40     APTR ColdCapture;
41     APTR CoolCapture;
42     APTR WarmCapture;
43     APTR SysStkUpper; /* system stack base (upper bound) */
44     APTR SysStkLower; /* top of system stack (lower bound) */
45     ULONG MaxLocMem;
46     APTR DebugEntry;
47     APTR DebugData;
48     APTR AlertData;
49     APTR RsvdExt;
50
51     UWORD ChkSum;
52
53     /****** Interrupt Related *****/
54     struct IntVector IntVects[16];
55
56
```

```

57 /***** System Variables *****/
58 struct Task *ThisTask; /* pointer to current task */
59 ULONG IdleCount; /* idle counter */
60 ULONG DispCount; /* dispatch counter */
61 UMORD Quantum; /* time slice quantum */
62 UMORD SysFlags; /* current quantum ticks */
63 UMORD Elapsed; /* misc system flags */
64 BYTE IDNestCnt; /* interrupt disable nesting count */
65 BYTE IDNestCnt; /* task disable nesting count */
66 BYTE TDNestCnt;
67 UMORD AttnFlags; /* special attention flags */
68 UMORD AttnResched; /* rescheduling attention */
69 APTR ResModules; /* resident module array pointer */
70 APTR ResModules;
71 APTR TaskTrapCode;
72 APTR TaskExceptCode;
73 APTR TaskExitCode;
74 UMORD TaskSigAlloc;
75 UMORD TaskTrapAlloc;
76
77 /***** System Lists *****/
78 struct List MemList;
79 struct List ResourceList;
80 struct List DevicelList;
81 struct List IntrList;
82 struct List LibList;
83 struct List PortList;
84 struct List TaskReady;
85 struct List TaskWait;
86 struct SoftIntList SoftInts[5];
87
88 /***** Other Globals *****/
89 LONG LastAlert[4];
90 LONG ExecBaseReserved[9];
91
92 /***** AttnFlags *****/
93 /* Processors and Co-processors: */
94 #define AFB_68010 0 /* (will remain set for 68020 as well) */
95 #define AFB_68020 1
96 #define AFB_68981 4
97 #define AFB_PAL 8
98 #define AFB_50HZ 9
99 #define PAL_NTSC
100 #endif

```

```

1 /* Commodore-Amiga, Inc. */
2 #define EXECNAME "exec.library"

```

```

1 #ifndef EXEC_INTERRUPTS_H
2 #define EXEC_INTERRUPTS_H
3 /*****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *
8 *
9 * Source Control:
10 *
11 * $Header: interrupts.h,v 1.0 85/08/28 15:09:53 carl Exp $
12 *
13 * $Locker:  $
14 *
15 *****/
16
17 #ifndef EXEC_NODES_H
18 #include "exec/nodes.h"
19 #endif !EXEC_NODES_H
20
21 #ifndef EXEC_LISTS_H
22 #include "exec/lists.h"
23 #endif !EXEC_LISTS_H
24
25
26 struct Interrupt {
27     struct Node is_Node;
28     APTR is_Data;
29     VOID (*is_Code) ();
30 };
31
32
33 struct IntVector {
34     APTR iv_Data;
35     VOID (*iv_Code) ();
36     struct Node *iv_Node;
37 };
38
39
40 struct SoftIntList {
41     struct List sh_List;
42     UMWORD sh_Pad;
43 };
44
45 #define SIH_PRIMASK (0xf0)
46
47 #endif

```

```

1 #ifndef EXEC_IO_H
2 #define EXEC_IO_H
3 /*****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *
8 *
9 * Source Control:
10 *
11 * $Header: io.h,v 1.0 85/08/28 15:10:30 carl Exp $
12 *
13 * $Locker:  $
14 *
15 *****/
16
17 #ifndef EXEC_PORTS_H
18 #include "exec/ports.h"
19 #endif !EXEC_PORTS_H
20
21
22 struct IORequest {
23     struct Message io_Message;
24     struct Device *io_Device;
25     struct Unit *io_Unit;
26     UMWORD io_Command;
27     UBYTE io_Flags;
28     BYTE io_Error;
29 };
30
31 struct IOStdReq {
32     struct Message io_Message;
33     struct Device *io_Device;
34     struct Unit *io_Unit;
35     UMWORD io_Command;
36     UBYTE io_Flags;
37     BYTE io_Error;
38     ULONG io_Actual;
39     ULONG io_Length;
40     APTR io_Data;
41     ULONG io_Offset;
42     ULONG io_Reserved1;
43     ULONG io_Reserved2;
44 };
45
46
47 #define IOB_QUICK 0
48 #define IOF_QUICK (1<<0)
49
50
51 #define CMD_INVALID 0
52 #define CMD_RESET 1
53 #define CMD_READ 2
54 #define CMD_WRITE 3
55 #define CMD_UPDATE 4

```

```

57 #define CMD_CLEAR      5
58 #define CMD_STOP      6
59 #define CMD_START     7
60 #define CMD_FLUSH     8
61 #define CMD_NONSTD    9
62 #endif
63
64

```

```

1 #ifndef EXEC_LIBRARIES_H
2 #define EXEC_LIBRARIES_H
3 /******
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *
8 *
9 * Source Control:
10 *
11 * $Header: libraries.h,v 1.0 85/08/28 15:10:56 carl Exp $
12 *
13 * $Locker:  $
14 *
15 *
16 #ifndef EXEC_NODES_H
17 #include "exec/nodes.h"
18 #endif EXEC_NODES_H
19
20
21
22 #define LIB_VECTSIZE      6
23 #define LIB_RESERVED     4
24 #define LIB_BASE         (-LIB_VECTSIZE)
25 #define LIB_USERDEF      (LIB_BASE-(LIB_RESERVED+LIB_VECTSIZE))
26 #define LIB_NONSTD      (LIB_USERDEF)
27
28 #define LIB_OPEN         (-6)
29 #define LIB_CLOSE       (-12)
30 #define LIB_EXPUNCE     (-18)
31 #define LIB_EXTIFUNC    (-24)
32
33
34 extern struct Library {
35     struct Node lib_Node;
36     UBYTE lib_Flags;
37     UBYTE lib_Pad;
38     UWORD lib_NegSize;
39     UWORD lib_PosSize;
40     UWORD lib_Version;
41     UWORD lib_Revision;
42     APTR lib_IdString;
43     ULONG lib_Sum;
44     UWORD lib_OpenCnt;
45 };
46
47 /* number of bytes before library */
48 /* number of bytes after library */
49
50 /* the checksum itself */
51 /* number of current opens */
52
53 /* we are currently checksumming */
54 /* we have just changed the lib */
55 /* set if we should bother to sum */
56 /* delayed expunge */
57
58 /* Temporary Compatibility */
59 #define lh_Node lib_Node
60 #define lh_Flags lib_Flags
61 #define lh_Pad lib_Pad
62 #define lh_NegSize lib_NegSize

```

```
57 #define lh_PosSize lib_PosSize
58 #define lh_Version lib_Version
59 #define lh_Revision lib_Revision
60 #define lh_IdString lib_IdString
61 #define lh_Sum lib_Sum
62 #define lh_OpenCnt lib_OpenCnt
63
64 #endif
```

```
1 #ifndef EXEC_LISTS_H
2 #define EXEC_LISTS_H
3 /*****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *
8 *
9 * Source Control:
10 *
11 * $Header: lists.h,v 1.0 95/08/28 15:11:23 carl Exp $
12 *
13 * $Locker: $
14 *
15 *
16 #ifndef EXEC_NODES_H
17 #include "exec/nodes.h"
18 #endif !EXEC_NODES_H
19
20
21
22 struct List {
23     struct Node *lh_Head;
24     struct Node *lh_Tail;
25     struct Node *lh_TailPred;
26     UBYTE lh_Type;
27     UBYTE l_pad;
28 };
29
30 #endif
```



```

1 #ifndef EXEC_MEMORY_H
2 #define EXEC_MEMORY_H
3 /*****
4 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
5 *
6 *
7 *
8 * Source Control:
9 *
10 * $Header: memory.h,v 1.0 85/08/28 15:11:49 carl Exp $
11 *
12 * $Locker:  $
13 *
14 *
15 *
16 #ifndef EXEC_NODES_H
17 #include "exec/nodes.h"
18 #endif EXEC_NODES_H
19
20
21 /***** MemChunk *****/
22 struct MemChunk {
23     struct MemChunk *mc_Next; /* pointer to next chunk */
24     ULONG mc_Bytes; /* chunk byte size */
25 };
26
27 /***** MemHeader *****/
28 struct MemHeader {
29     struct MemHeader *mh_Attributes; /* characteristics of this region */
30     ULONG mh_First; /* first free region */
31     ULONG mh_Lower; /* lower memory bound */
32     ULONG mh_Upper; /* upper memory bound+1 */
33     ULONG mh_Free; /* total number of free bytes */
34 };
35
36 /***** MemEntry *****/
37 struct MemEntry {
38     ULONG meu_Reqs; /* the AllocMem requirements */
39     ULONG meu_Addr; /* the address of this memory region */
40     ULONG me_Length; /* the length of this memory region */
41 };
42
43 #define ma_Un /* compatability */
44 #define ma_Un_Reqs ma_Un
45 #define ma_Reqs ma_Un_Reqs
46 #define ma_Addr ma_Un_Addr
47
48
49
50
51
52
53
54
55
56

```

```

57 /***** MemList *****/
58 struct MemList {
59     struct Node ml_Node;
60     ULONG ml_NumEntries; /* number of entries in this struct */
61     struct MemEntry ml_ME[1]; /* the first entry */
62 };
63
64 #define ml_me ml_ME /* compatability */
65
66 /***** Memory Requirement Types *****/
67 #define MEME_PUBLIC (1<<0)
68 #define MEME_CHIP (1<<1)
69 #define MEME_FAST (1<<2)
70 #define MEME_CLEAR (1<<16)
71 #define MEME_LARGEST (1<<17)
72 #define MEM_BLOCKSIZE 8
73 #define MEM_BLOCKMASK 7
74
75 #endif
76
77
78
79
80

```

```

1 #ifndef EXEC_NODES_H
2 #define EXEC_NODES_H
3 /*****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *
8 *
9 * Source Control:
10 *
11 * $Header: nodes.h,v 1.1 85/11/12 18:22:53 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****/
16
17
18 struct Node {
19     struct Node *ln_Succ;
20     struct Node *ln_Pred;
21     UBYTE ln_Type;
22     BYTE ln_Pri;
23     char *ln_Name;
24 };
25
26 /***** Node Types -----*/
27 #define NT_UNKNOWN 0
28 #define NT_TASK 1
29 #define NT_INTERRUPT 2
30 #define NT_DEVICE 3
31 #define NT_MSPORT 4
32 #define NT_MESSAGE 5
33 #define NT_FREEMSG 6
34 #define NT_REPLYMSG 7
35 #define NT_RESOURCE 8
36 #define NT_LIBRARY 9
37 #define NT_MEMORY 10
38 #define NT_SOFTINT 11
39 #define NT_FONT 12
40 #define NT_PROCESS 13
41 #define NT_SEMAPHORE 14
42
43 #endif

```

```

1 #ifndef EXEC_PORTS_H
2 #define EXEC_PORTS_H
3 /*****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *
8 *
9 * Source Control:
10 *
11 * $Header: ports.h,v 1.1 85/11/12 18:11:45 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****/
16
17 #ifndef EXEC_NODES_H
18 #include "exec/nodes.h"
19 #endif
20
21 #ifndef EXEC_LISTS_H
22 #include "exec/lists.h"
23 #endif
24
25 #ifndef EXEC_TASKS_H
26 #include "exec/tasks.h"
27 #endif
28
29 /***** MsgPort *****/
30
31 struct MsgPort {
32     struct Node mp_Node;
33     UBYTE mp_Flags;
34     UBYTE mp_Sigbit;
35     struct Task *mp_SigTask;
36     struct List mp_MsgList;
37 };
38
39 #define mp_SoftInt mp_SigTask
40
41 #define PF_ACTION 3
42
43 #define PA_SIGNAL 0
44 #define PA_SOFTINT 1
45 #define PA_IGNORE 2
46
47 /***** Message *****/
48
49 struct Message {
50     struct Node mn_Node;
51     struct MsgPort *mn_ReplyPort;
52     UWORD mn_Length;
53 };
54
55 /* message reply port */
56 /* message len in bytes */

```

```

1 #ifndef EXEC_RESIDENT_H
2 #define EXEC_RESIDENT_H
3 /*****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *
8 *
9 * Source Control:
10 *
11 * $Header: resident.h,v 1.0 85/08/28 15:13:28 carl Exp $
12 *
13 * $Locker: $
14 *
15 *
16 *
17 #ifndef EXEC_NODES_H
18 #include "exec/nodes.h"
19 #endif !EXEC_NODES_H
20
21 struct Resident {
22     UWORD rt_MatchWord; /* word to match on (ILLEGAL) */
23     struct Resident *rt_MatchFlag; /* pointer to the above */
24     APTR rt_EndSkip; /* address to continue scan */
25     UBYTE rt_Flags; /* various tag flags */
26     UBYTE rt_Version; /* release version number */
27     UBYTE rt_Type; /* type of module (NT mumble) */
28     BYTE rt_Pri; /* initialization priority */
29     char *rt_Name; /* pointer to node name */
30     APTR rt_IdString; /* pointer to ident string */
31     APTR rt_Init; /* pointer to init code */
32 };
33
34 #define RTC_MATCHWORD 0x4AFC
35
36 #define RTE_AUTOINIT (1<<7)
37 #define RTE_COLDSTART (1<<0)
38
39 /* Compatibility: */
40 #define RTM_WHEN 3
41 #define RTM_NEVER 0
42 #define RTM_COLDSTART 1
43
44 #endif

```

```

57 /***** Semaphore *****/
58 struct Semaphore {
59     struct MsgPort sm_MsgPort;
60     WORD sm_Bids;
61 };
62
63 #define sm_LockMsg mp_SigTask
64 #endif

```

```

1 #ifndef EXEC_TASKS_H
2 #define EXEC_TASKS_H
3 /*-----*/
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *-----*/
7 *
8 *
9 * Source Control:
10 *
11 * $Header: tasks.h,v 1.0 85/08/28 15:14:19 carl Exp $
12 *
13 * $Locker:  $
14 *
15 *-----*/
16
17 #ifndef EXEC_NODES_H
18 #include "exec/nodes.h"
19 #endif !EXEC_NODES_H
20
21 #ifndef EXEC_LISTS_H
22 #include "exec/lists.h"
23 #endif !EXEC_LISTS_H
24
25
26 extern struct Task {
27     struct Node tc_Node;
28     UBYTE tc_Flags;
29     UBYTE tc_State;
30     BYTE tc_IDNestCnt;
31     BYTE tc_IDNestCnt;
32     ULONG tc_SigAlloc;
33     ULONG tc_SigMalt;
34     ULONG tc_SigRecvd;
35     ULONG tc_SigExcept;
36     UNORD tc_TrapAlloc;
37     UNORD tc_TrapAble;
38     APTR tc_ExceptData;
39     APTR tc_ExceptCode;
40     APTR tc_TrapData;
41     APTR tc_TrapCode;
42     APTR tc_SPReg;
43     APTR tc_SPLower;
44     APTR tc_SPUpper;
45     VOID (*tc_Switch) ();
46     VOID (*tc_Launch) ();
47     struct List tc_MemEntry;
48     APTR tc_UserData;
49 };
50
51 /*----- Flag Bits -----*/
52 #define TB_PROCTIME 0
53 #define TB_STACKCHK 4
54 #define TB_EXCEPT 5
55 #define TB_SWITCH 6
56 #define TB_LAUNCH 7

```

```

57 /*----- Task States -----*/
58 #define TS_INVALID 0
59 #define TS_ADDED 1
60 #define TS_RUN 2
61 #define TS_READY 3
62 #define TS_WAIT 4
63 #define TS_EXCEPT 5
64 #define TS_REMOVED 6
65
66 /*----- Predefined Signals -----*/
67
68 #define SIG_ABORT (1<<0)
69 #define SIG_CHILD (1<<1)
70 #define SIG_ELIT (1<<4)
71 #define SIG_DOS (1<<8)
72
73 #endif
74
75

```

```

1 #ifndef EXEC_TYPES_H
2 #define EXEC_TYPES_H
3 /*****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *****/
7
8
9 * Source Control:
10 *
11 * $Header: types.h,v 1.2 85/11/15 17:43:37 carl Exp $
12 *
13 * $Locker:  $
14 *
15 *****/
16
17
18 #define GLOBAL extern
19 #define IMPORT extern
20 #define STATIC static
21 #define REGISTER register
22
23 #define VOID void
24
25 typedef long LONG;
26 typedef unsigned long ULONG;
27 typedef unsigned long LONGBITS;
28 typedef short WORD;
29 typedef unsigned short UWORD;
30 typedef unsigned short WORDBITS;
31 typedef char BYTE;
32 typedef unsigned char UBYTE;
33 typedef unsigned char BYTEBITS;
34 typedef unsigned char *STRPTR;
35 typedef STRPTR *APTR;
36
37 /* For compatibility only: (don't use in new code) */
38 typedef short SHORT;
39 typedef unsigned short USHORT;
40
41
42 /* Types with specific semantics */
43 typedef float FLOAT;
44 typedef double DOUBLE;
45 typedef short COUNT;
46 typedef unsigned short UCOUNT;
47 typedef short BOOL;
48 typedef unsigned char TEXT;
49
50 #define TRUE 1
51 #define FALSE 0
52 #define NULL 0
53
54 #define BYTEMASK 0xFF
55
56 #define LIBRARY_VERSION 31

```

```

57 #endif
58

```

```

1 #ifndef GRAPHICS_CLIP_H
2 #define GRAPHICS_CLIP_H
3
4 #ifndef GRAPHICS_CFX_H
5 #include <graphics/gfx.h>
6 #endif
7 #ifndef EXEC_PORTS_H
8 #include <exec/ports.h>
9 #endif
10
11 /******
12 /* Commodore-Amiga, Inc.
13 /* clip.h
14 /******
15
16 # Modification History
17 # date : author : Comments
18 # -----
19 # 02-04-85 Dale created file from graph.h
20 #*****
21
22 /* structures used by and constructed by windowlib.a */
23 /* understood by rom software */
24
25 struct Layer
26 {
27     struct Layer *front,*back; /* ignored by roms */
28     struct ClipRect *ClipRect; /* read by roms to find first cliprect */
29     struct RastPort *rp; /* ignored by roms, I hope */
30     struct Rectangle bounds; /* ignored by roms */
31     UBYTE Locking; /* roms, obey locking/unlocking
32                       convention */
33     UBYTE LockCount; /* roms can nest their own locks and
34                       still work */
35     UBYTE LayerLockCount; /* lock counter used by layer software */
36     UBYTE reserved;
37     UWORD Flags; /* obscured ?, Virtual BitMap? */
38     struct BitMap *SuperBitMap;
39     struct ClipRect *SuperClipRect; /* super bitmap cliprects if
40                                     VBitMap != 0 */
41
42     APTR Window; /* else damage cliprect list for refresh */
43     SHORT Scroll_X,Scroll_Y; /* reserved for user interface use */
44     struct MsgPort LockPort;
45     struct Message LockMessage;
46     struct MsgPort ReplyPort;
47     struct Message l.LockMessage;
48     struct Region *DamageList; /* list of rectangles to refresh
49                                 through */
50
51     ClipRect *_cliprects; /* system use during refresh */
52     Layer_Info *LayerInfo; /* points to head of the list */
53     struct Task *LayerLocker; /* points to task that has layerlock */
54     struct ClipRect *SuperSaveClipRects; /* preallocated cr's */
55     struct ClipRect *cr,*cr2,*crnew; /* used by dedice */
56     APTR _pl; /* system use, reserved */

```

```

57 };
58
59 struct ClipRect
60 {
61     struct ClipRect *Next; /* roms used to find next ClipRect */
62     struct ClipRect *prev; /* ignored by roms, used by windowlib */
63     struct Layer *lobs; /* ignored by roms, used by windowlib */
64     struct BitMap *BitMap;
65     struct Rectangle bounds; /* set up by windowlib, used by roms */
66     struct ClipRect *_pl,*_p2; /* system reserved */
67     LONG reserved; /* system use */
68 #ifdef NEWCLIPRECTS_1_1
69     LONG Flags; /* only exists in layer allocation */
70 #endif
71 };
72
73 /* internal cliprect flags */
74 #define CR_NEEDS_NO_CONCEALED_RASTERS 1
75
76 /* defines for code values for getcode */
77 #define ISLESSX 1
78 #define ISLESSY 2
79 #define ISCRTRX 4
80 #define ISCRTRY 8
81 #endif

```

```

1 #ifndef GRAPHICS_COLLIDE_H
2 #define GRAPHICS_COLLIDE_H
3 /***** collide.h *****/
4 /*
5 /* Commodore-Amiga, Inc.
6 /*
7 /* Modification History
8 /* author : Comments
9 /* -----
10 /* 8-24-84 Dale added this header file
11 /*
12 /*
13 /*****
14 /* include file for collision detection and control */
15
16 /* These bit descriptors are used by the GEL collide routines.
17 * These bits are set in the hitMask and mask variables of
18 * a GEL to describe whether or not these types of collisions
19 * can affect the GEL. ENDRY_HIT is described further below;
20 * this bit is permanently assigned as the boundary-hit flag.
21 * The other bit GEL_HIT is meant only as a default to cover
22 * any GEL hitting any other; the user may redefine this bit.
23 */
24 #define BORDERHIT 0
25
26 /* These bit descriptors are used by the GEL boundry hit routines.
27 * When the user's boundry-hit routine is called (via the argument
28 * set by a call to SetCollision) the first argument passed to
29 * the user's routine is the address of the GEL involved in the
30 * boundry-hit, and the second argument has the appropriate bit(s)
31 * set to describe which boundry was surpassed
32 */
33 #define TOPHIT 1
34 #define BOTTOMHIT 2
35 #define LEFTHIT 4
36 #define RIGHTHIT 8
37
38 #endif

```

```

1 #ifndef GRAPHICS_COPPER_H
2 #define GRAPHICS_COPPER_H
3 /***** copper.h *****/
4 /*
5 /* Commodore-Amiga, Inc.
6 /*
7 /* Modification History
8 /* author : Comments
9 /* -----
10 /* 8-24-84 Dale added this header file
11 /* 9-11-84 Dale redefined with unions
12 /* 2-09-85 Dale made #defines for union ignorance
13 /*****
14
15 #define COPPER_MOVE 0 /* pseudo opcode for move #XXXX.dir */
16 #define COPPER_WAIT 1 /* pseudo opcode for wait y,x */
17 #define CPRNXTBUF 2 /* continue processing with next buffer */
18 #define CPR_NXT_LOF 0x8000 /* copper instruction only for short frames */
19 #define CPR_NXT_SHT 0x4000 /* copper instruction only for long frames */
20 struct CopIns
21 {
22     short OpCode; /* 0 = move, 1 = wait */
23     union
24     {
25         struct CopList *nxtlist;
26         struct
27         {
28             union
29             {
30                 SHORT VWaitPos; /* vertical beam wait */
31                 SHORT DestAddr; /* destination address of copper move */
32             } u1;
33             union
34             {
35                 SHORT HWaitPos; /* horizontal beam wait position */
36                 SHORT DestData; /* destination immediate data to send */
37             } u2;
38             } u3;
39         } u3;
40     };
41 /* shorthand for above */
42 #define NXTLIST u3.nxtlist
43 #define VWAITPOS u3.u4.u1.VWaitPos
44 #define DESTADDR u3.u4.u1.DestAddr
45 #define HWAITPOS u3.u4.u2.HWaitPos
46 #define DESTDATA u3.u4.u2.DestData
47
48 /* structure of cprlist that points to list that hardware actually executes
49 struct cprlist
50 {
51     struct cprlist *Next; /* start of copper list */
52     USHORT *start; /* number of long instructions */
53     SHORT max;
54 }
55 */

```

```

57 struct CopList
58 {
59     struct CopList *Next; /* next block for this copper list */
60     struct CopList *CopList; /* system use */
61     struct ViewPort *ViewPort; /* system use */
62     struct CopList *CopList; /* start of this block */
63     struct CopList *CopList; /* intermediate ptr */
64     WORD *CopListStart; /* argcop fills this in for Long Frame */
65     WORD *CopListStart; /* argcop fills this in for Short Frame */
66     SHORT Count; /* intermediate counter */
67     SHORT MaxCount; /* max # of coplins for this block */
68     SHORT DyOffset; /* offset this copper list vertical waits */
69 };
70
71 struct UCopList
72 {
73     struct CopList *Next;
74     struct CopList *FirstCopList; /* head node of this copper list */
75     struct CopList *CopList; /* node in use */
76 };
77
78 struct copinit
79 {
80     WORD diagstrt[4]; /* copper list for first bitplane */
81     WORD sprstrtop[(2*8*2)+2*(2*2)+2];
82     WORD sprstop[2];
83 };
84
85 #endif

```

```

1  /***** display.h *****/
2  /*
3  /* Commodore-Amiga, Inc.
4  /*
5  /* Modification History
6  /*
7  /* date : author : Comments
8  /* -----
9  /* 8-24-84 Dale added this header file
10 /*
11 /*
12 /*
13 /* Include define file for display control registers */
14 /* bplcon0 defines */
15 #define MODE_640 0x8000
16 #define PLANCNMSK 0x7
17 #define PLNCNMSHFT 12
18 #define PF2PRI 0x40
19 #define COLORON 0x0200
20 #define DELPF 0x400
21 #define HOLDMODIFY 0x800
22 #define INTERLACE 4
23
24 /* bplcon1 defines */
25 #define PFA_FINE_SCROLL 0xF
26 #define PFB_FINE_SCROLL_SHIFT 4
27 #define PF_FINE_SCROLL_MASK 0xF
28
29 /* display window start and stop defines */
30 #define DIW_HORIZ_POS 0x7F /* horizontal start/stop */
31 #define DIW_VRTCL_POS 0x1FF /* vertical start/stop */
32 #define DIW_VRTCL_POS_SHIFT 7
33
34 /* Data fetch start/stop horizontal position */
35 #define DFVCL_MASK 0xFF
36
37 /* vposr bits */
38 #define VPOSRLOF 0x8000
39

```



```

1 #ifndef GRAPHICS_GELS_H
2 #define GRAPHICS_GELS_H
3
4 /*****
5 *
6 * Include file for AMIGA GELS (Graphics Elements)
7 *
8 * Commodore-Amiga, Inc.
9 *
10 * Modification History
11 * author : Comments
12 * -----
13 * 8-24-84 Dale added this header file
14 * 9-28-84 --RJ-- for GELS16 added Bob.h to this file
15 * made name and declaration changes
16 *
17 * *****/
18
19
20 /* VSprite flags */
21 #user-set VSprite flags:
22 #define SUSERFLAGS 0x00FF /* mask of all user-settable VSprite-flags */
23 #define VSPRITE 0x0001 /* set if VSprite, clear if Bob */
24 #define SAVEBACK 0x0002 /* set if background is to be saved/restored */
25 #define OVERLAY 0x0004 /* set to mask image of Bob onto background */
26 #define MUSTDRAW 0x0008 /* set if VSprite absolutely must be drawn */
27
28 /* system-set VSprite flags:
29 #define BACKSAVED 0x0100 /* this Bob's background has been saved */
30 #define BOBUPDATE 0x0200 /* temporary flag, useless to outside world */
31 #define CELCONE 0x0400 /* set if gel is completely clipped (offscreen) */
32 #define VSOVERFLOW 0x0800 /* VSprite overflow (if MUSTDRAW set we draw!) */
33
34 /* Bob flags */
35 #define USERFLAGS 0x00FF /* mask of all user-settable Bob-flags */
36 #define SAVEBOB 0x0001 /* set to not erase Bob */
37 #define BOBSCOPE 0x0002 /* set to identify Bob as AnimComp */
38
39 #define EMATING 0x0100 /* set while Bob is waiting on 'after' */
40 #define BURANN 0x0200 /* set when Bob is drawn this DrawC pass */
41 #define BOBSAWAY 0x0400 /* set to initiate removal of Bob */
42 #define BOBNIX 0x0800 /* set when Bob is completely removed */
43 #define SAVEPRESERVE 0x1000 /* for back-restore during double-buffer */
44 #define OUTSTEP 0x2000 /* for double-clearing if double-buffer */
45
46 /* defines for the animation procedures */
47 #define ANERACSIZE 6
48 #define ANIMHALF 0x0020
49 #define RINGTRICER 0x0001
50
51 /* UserStuff definitions
52 * the user can define these to be a single variable or a sub-structure
53 * if undefined by the user, the system turns these into innocuous variables
54 * see the manual for a thorough definition of the UserStuff definitions
55 *
56

```

```

57 */
58 #ifndef VUserStuff /* VSprite user stuff */
59 #define VUserStuff SHORT
60 #endif
61
62 #ifndef BUserStuff /* Bob user stuff */
63 #define BUserStuff SHORT
64 #endif
65
66 #ifndef AUserStuff /* AnimOb user stuff */
67 #define AUserStuff SHORT
68 #endif
69
70
71
72
73 /***** CEL STRUCTURES *****/
74
75 struct VSprite
76 {
77     /*----- SYSTEM VARIABLES -----*/
78     /* CEL linked list forward/backward pointers sorted by y.x value */
79     struct VSprite *NextVSprite;
80     struct VSprite *PrevVSprite;
81
82     /* CEL draw list constructed in the order the Bobs are actually drawn, then
83      * list is copied to clear list
84      * must be here in VSprite for system boundary detection
85      */
86     struct VSprite *DrawPath; /* pointer of overlay drawing */
87     struct VSprite *ClearPath; /* pointer for overlay clearing */
88
89     /* the VSprite positions are defined in (y,x) order to make sorting
90      * sorting easier, since (y,x) as a long integer
91      */
92     WORD OldY, OldX; /* previous position */
93
94     /*----- COMMON VARIABLES -----*/
95     WORD Flags; /* VSprite flags */
96
97
98     /*----- USER VARIABLES -----*/
99     /* the VSprite positions are defined in (y,x) order to make sorting
100      * sorting easier, since (y,x) as a long integer
101      */
102     WORD Y, X; /* screen position */
103
104     WORD Height; /* number of words per row of image data */
105     WORD Width; /* number of planes of data */
106     WORD Depth;
107
108     WORD MemMask; /* which types can collide with this VSprite
109     WORD HitMask; /* which types this VSprite can collide with
110     WORD *ImageData; /* pointer to VSprite image */
111
112

```

```

113 /* borderLine is the one-dimensional logical OR of all
114 * the VSprite bits, used for fast collision detection of edge
115 */
116 WORD *BorderLine; /* logical OR of all VSprite bits */
117 WORD *CollMask; /* similar to above except this is a matrix */
118
119 /* pointer to this VSprite's color definitions (not used by Bobs) */
120 WORD *SprColors;
121
122 struct Bob *VSBob; /* points home if this VSprite is part of
123 * a Bob */
124
125 /* planePick flag: set bit selects a plane from image, clear bit selects
126 * use of shadow mask for that plane
127 * OnOff flag: if using shadow mask to fill plane, this bit (corresponding
128 * to bit in planePick) describes whether to fill with 0's or 1's
129 * There are two uses for these flags:
130 * - if this is the VSprite of a Bob, these flags describe how the Bob
131 * is to be drawn into memory
132 * - if this is a simple VSprite and the user intends on setting the
133 * MUSTDRAW flag of the VSprite, these flags must be set too to describe
134 * which color registers the user wants for the image
135 */
136 BYTE PlanePick;
137 BYTE PlaneOnOff;
138
139 UserStuff VUserExt; /* user definable: see note above */
140 };
141
142 struct Bob
143 /* blitter-objects */
144 {
145 /*
146 ----- SYSTEM VARIABLES -----
147
148 ----- COMMON VARIABLES -----
149 /* general purpose flags (see definitions below) */
150
151 ----- USER VARIABLES -----
152 /* pointer to the buffer for background save */
153
154 /* used by Bobs for "cookie-cutting" and multi-plane masking */
155 WORD *ImageShadow;
156
157 /* pointer to BOBs for sequenced drawing of Bobs
158 * for correct over-laying of multiple component animations
159
160 struct Bob *Before; /* draw this Bob before Bob pointed to by before */
161 struct Bob *After; /* draw this Bob after Bob pointed to by after */
162
163 struct VSprite *BobVSprite; /* this Bob's VSprite definition */
164
165 struct AnimComp *BobComp; /* pointer to this Bob's AnimComp def */
166
167 struct DBufPacket *DBuffer; /* pointer to this Bob's dBuf packet */
168
169 UserStuff BUUserExt; /* Bob user extension */

```

```

169 };
170
171 struct AnimComp
172 {
173 /*
174 ----- SYSTEM VARIABLES -----
175
176 ----- COMMON VARIABLES -----
177 /* AnimComp flags for system & user */
178 WORD Flags;
179
180 /* timer defines how long to keep this component active:
181 * if set non-zero, timer decrements to zero then switches to nextSeq
182 * if set to zero, AnimComp never switches
183
184 WORD Timer;
185
186 ----- USER VARIABLES -----
187 /* Initial value for timer when the AnimComp is activated by the system */
188 WORD TimesSet;
189
190 /* pointer to next and previous components of animation object */
191 struct AnimComp *NextComp;
192 struct AnimComp *PrevComp;
193
194 /* pointer to component definition of next image in sequence */
195 struct AnimComp *NextSeq;
196 struct AnimComp *PrevSeq;
197
198 WORD (*AnimCRoutine)(); /* address of special animation procedure */
199
200 WORD YTrans; /* initial y translation (if this is a component) */
201 WORD XTrans; /* initial x translation (if this is a component) */
202
203 struct AnimOb *HeadOb;
204
205 struct Bob *AnimBob;
206
207 struct AnimOb
208 {
209 /*
210 ----- SYSTEM VARIABLES -----
211 /* number of calls to Animate this AnimOb has endured */
212 LONG Clock;
213
214 WORD AnOldY, AnOldX; /* old y,x coordinates */
215
216 /*
217 ----- COMMON VARIABLES -----
218 /* y,x coordinates of the AnimOb */
219 WORD AnY, AnX;
220
221 ----- USER VARIABLES -----
222 /* velocities of this object */
223 WORD YVel, XVel;
224 /* accelerations of this object */
225 WORD YAccel, XAccel;
226
227 /* ring translation values */
228 WORD RingYTrans, RingXTrans;

```

```

225 WORD (*AnimORoutine) (); /* address of special animation
226 procedure */
227
228 struct AnimComp *HeadComp; /* pointer to first component */
229
230 AUserStuff AUserExt; /* AnimOb user extension */
231 };
232
233 /* dBufPacket defines the values needed to be saved across buffer to buffer
234 * when in double-buffer mode
235 */
236 struct DBufPacket
237 {
238 WORD BufY, BufX; /* save the other buffers screen coordinates
239 struct VSprite *BufPath; /* carry the draw path over the gap */
240
241 /* these pointers must be filled in by the user */
242 /* pointer to other buffer's background save buffer */
243 WORD *BufBuffer;
244 };
245
246
247
248
249
250 /* these are GEL functions that are currently simple enough to exist as a
251 * definition. It should not be assumed that this will always be the case
252 */
253 #define InitAnimate(animKey) (*(animKey) = NULL;)
254 #define RemBob(b) {(b)->Flags |= BOBSMAY;}
255
256
257
258
259 #define B2NORM 0
260 #define B2SWAP 1
261 #define B2BORBER 2
262
263
264
265 /* a structure to contain the 16 collision procedure addresses */
266 struct collTable
267 {
268 int (*collPtrs[16]) ();
269 };
270
271
272 #endif

```

```

1 #ifndef GRAPHICS_GFX_H
2 #define GRAPHICS_GFX_H
3 /***** gfx.h *****/
4
5 /* Commodore-Amiga, Inc.
6
7
8
9
10 /* 8-24-84 Dale added this header file
11 /* Feb 85 Dale added Rectangle, BitMap structures
12 /*****
13
14 /* general include file for application programs */
15 #define BITSET 0x8000
16 #define BITCLR 0
17
18 #define AGRUS
19 #ifdef AGRUS
20 #define TOBB(a) ((long)(a))
21 #else
22 #define TOBB(a) ((long)(a)>>1) /* convert Chip adr to Bread Board Adr
23 #endif
24
25 struct Rectangle
26 {
27     SHORT MinX,MinY;
28     SHORT MaxX,MaxY;
29 };
30
31 typedef UBYTE *PLANEPTR;
32
33 struct BitMap
34 {
35     UWORD BytesPerRow;
36     UWORD Rows;
37     UBYTE Flags;
38     UWORD Depth;
39     UWORD pad;
40     PLANEPTR Planes[8];
41 };
42
43 #define RASSIZE(w,h) ((h)*(w+15)>>360xFFFF)
44
45 #endif

```

```

1  #ifndef GRAPHICS_GFXBASE_H
2  #define GRAPHICS_GFXBASE_H
3
4  #ifndef EXEC_LISTS_H
5  #include <exec/lists.h>
6  #endif
7  #ifndef EXEC_LIBRARIES_H
8  #include <exec/libraries.h>
9  #endif
10 #ifndef EXEC_INTERRUPTS_H
11 #include <exec/interrupts.h>
12 #endif
13
14 /***** gfxbase.h *****/
15 /* Commodore-Amiga, Inc.
16 /*
17 /*
18 /* Modification History
19 /* author : Comments
20 /* -----
21 /* 10-20-84 Kodiak added this header file & TextFonts
22 /*
23 /*****
24 struct GfxBase
25 {
26     struct Library LibNode;
27     struct View *ActiveView;
28     struct copinit *copinit; /* ptr to copper start up list */
29     long *cia; /* for 8520 resource use */
30     long *blitter; /* for future blitter resource use */
31     UWORD *LOEList;
32     UWORD *SHEList;
33     struct bitnode *blthd,*blttl;
34     struct bitnode *bsblthd,*bsblttl;
35     struct Interrupt vbsrv,tlmsrv,bltsrv;
36     struct List TextFonts;
37     struct TextFont *DefaultFont;
38     UWORD Modes; /* copy of current first bplcon0 */
39     BYTE VBlank;
40     BYTE Debug;
41     SHORT Beamsync;
42     SHORT system_bplcon0; /* this is initialized to 0 */
43     /* it is ored into each bplcon0 for display */
44     UBYTE SpritesReserved;
45     UBYTE bytesReserved;
46     /* candidates for removal */
47     USHORT Flags;
48     USHORT BlitLock;
49     short BlitNest;
50
51     struct List BlitWaitQ;
52     struct Task *BlitOwner;
53     struct List TOE_WaitQ;
54     UWORD DisplayFlags; /* NTSC PAL GENLOC etc */
55     /* Display flags are determined at power on
56     /* reserved[2]: /* for future use */

```

```

57 );
58
59 #define NTSC 1
60 #define GENLOC 2
61 #define PAL 4
62
63 #define BLITMSG_FAULT 4
64 #endif

```

```

1 #ifndef GRAPHICS_GFXMACROS_H
2 #define GRAPHICS_GFXMACROS_H
3 /****** gfxmacros.h *****/
4 /*
5 /* Commodore-Amiga, Inc.
6 /*
7 /*
8 /* Modification History
9 /* author : Comments
10 /* date : -----
11 /* 8-24-84 Dale added this header file
12 /* 9-06-84 Dale fixed macros using v-> to use (v)->
13 /* 9-07-84 Dale fixed macros to use new RastPort
14 /*
15 /******
16 #ifndef GRAPHICS_RASTPORT_H
17 #include <graphics/rastport.h>
18 #endif
19
20 #define ON_DISPLAY custom.dmacon = BITSET|DMAE_RASTER;
21 #define OFF_DISPLAY custom.dmacon = BITCLR|DMAE_RASTER;
22 #define ON_SPRITE custom.dmacon = BITSET|DMAE_SPRITE;
23 #define OFF_SPRITE custom.dmacon = BITCLR|DMAE_SPRITE;
24
25 #define ON_VBLANK custom.intena = BITSET|INTE_VERTB
26 #define OFF_VBLANK custom.intena = BITCLR|INTE_VERTB
27
28 #define SetPen(v,c) {(v)->AOPen = c; (v)->Flags |= AREAOUTLINE;}
29 #define SetDrPt(v,p) {(v)->LinePtrn = p; (v)->Flags |= FRST_DOT;}
30 #define SetWrMask(v,m) {(v)->Mask = m;}
31 #define SetAIPt(v,p,n) {(v)->AreaPtrn = p; (v)->AreaPtSz = n;}
32
33 #define ENDRYOFF {(v)->Flags &= "AREAOUTLINE"}
34
35 #define CINIT(c,n) { UCopperListInit(c,n); }
36 #define CMOVE(c,a,b) { CMove(c,a,b); CBump(c); }
37 #define CWAIT(c,a,b) { CWait(c,a,b); CBump(c); }
38 #define CEND(c) { CWAIT(c,10000,255); }
39
40 #endif

```

```

1 #ifndef GRAPHICS_GRAPHINT_H
2 #define GRAPHICS_GRAPHINT_H
3 /****** GRAPHICS_GRAPHINT_H *****/
4 /*
5 /* Commodore-Amiga, Inc.
6 /*
7 /* graphint.h
8 /*
9 #ifndef EXEC_NODES_H
10 #include <exec/nodes.h>
11 #endif
12 /* structure used by AddTOFTask */
13 struct Isrvstr
14 {
15     struct Node is_Node;
16     struct Isrvstr *lptr; /* passed to srvr by os */
17     int (*code)();
18     int (*ccode)();
19     int Carg;
20 };
21
22 #endif

```

```

1  /***** Commodore-Amiga, Inc. *****/
2  /* *****/
3  /***** Commodore-Amiga, Inc. *****/
4  /***** *****/
5  #ifndef GRAPHICS_LAYERS_H
6  #define GRAPHICS_LAYERS_H
7
8  #ifndef EXEC_PORTS_H
9  #include <exec/ports.h>
10 #endif
11
12 #ifndef EXEC_LISTS_H
13 #include <exec/lists.h>
14 #endif
15
16 #define LAYERSIMPLE 1
17 #define LAYERSMART 2
18 #define LAYERSUPER 4
19 #define LAYERBACKDROP 0x40
20 #define LAYERREFRESH 0x80
21
22 struct Layer_Info
23 {
24     struct Layer *top_layer;
25     struct Layer *check_ip;
26     struct Layer *obs;
27     struct MsgPort RP_ReplyPort;
28     struct MsgPort LockPort;
29     UBYTE Lock;
30     UBYTE broadcast;
31     UBYTE LockNest;
32     UBYTE Flags;
33     struct Task *Locker;
34     BYTE fatten_count;
35     UBYTE bytesreserved;
36     UWORD wordreserved; /* used to be a node in here someplace */
37     UWORD LayerInfo_extra_size;
38     ULONG longreserved;
39     struct LayerInfo_extra *LayerInfo_extra;
40 };
41
42 #define NEWLAYERINFO_CALLED 1
43 #define ALERTLAYERSNOWEM 0x83010000
44
45 #endif

```

```

1  #ifndef GRAPHICS_RASTPORT_H
2  #define GRAPHICS_RASTPORT_H
3
4  #ifndef GRAPHICS_GFX_H
5  #include <graphics/gfx.h>
6  #endif
7
8  /***** rastport.h *****/
9  *
10 * Commodore-Amiga, Inc.
11 *
12 * Modification History
13 * date : author : Comments
14 * -----
15 * 02-04-85 Dale created from graph.h
16 * *****/
17
18 struct AreaInfo
19 {
20     SHORT *VctrTbl; /* ptr to start of vector table */
21     SHORT *VctrPtr; /* ptr to current vertex */
22     BYTE *FlagTbl; /* ptr to start of vector flag table */
23     BYTE *FlagPtr; /* ptrs to areafill flags */
24     SHORT Count; /* number of vertices in list */
25     SHORT MaxCount; /* AreaMove/Draw will not allow Count>MaxCount */
26     SHORT FirstX,FirstY; /* first point for this polygon */
27 };
28
29 struct TmpRas
30 {
31     BYTE *RasPtr;
32     LONG Size;
33 } /* other misc junk for freelist etc. */
34
35 /* unoptimized for 32bit alignment of pointers */
36 struct CalsInfo
37 {
38     BYTE sprRsrvd; /* flag of which sprites to reserve from
39                    vsprite system */
40     UBYTE Flags; /* system use */
41     struct VSprite *gslHead, *gslTail; /* dummy vSprites for list management!
42                    /* pointer to array of 8 WORDS for sprite available lines */
43     WORD *nextLines;
44     /* pointer to array of 8 pointers for color-last-assigned to vSprites */
45     WORD **lastColor;
46     struct collTable *collHandler; /* addresses of collision routines */
47     short leftmost, rightmost, topmost, bottommost;
48     APTer firstBlissObj, lastBlissObj; /* system use only */
49 };
50
51 struct RastPort
52 {
53     struct Layer *Layer;
54     struct BitMap *BitMap;
55     USHORT *AreaPtrn; /* ptr to areafill pattern */
56

```

```

57 struct TmpRas *TmpRas;
58 struct AreaInfo *AreaInfo;
59 struct CellsInfo *CellsInfo;
60 UBYTE Mask; /* write mask for this raster */
61 BYTE FgPen; /* foreground pen for this raster */
62 BYTE BgPen; /* background pen */
63 BYTE AOIPen; /* areafill outline pen */
64 BYTE DrawMode; /* drawing mode for fill, lines, and text */
65 BYTE AreaPtSz; /* 2'n words for areafill pattern */
66 BYTE lInpatcnt; /* current line drawing pattern preshift */
67 BYTE dummy;
68 USHORT Flags; /* miscellaneous control bits */
69 USHORT LinePtrn; /* 16 bits for textured lines */
70 SHORT cp_x, cp_y; /* current pen position */
71 UBYTE minterms[8];
72 SHORT PenWidth;
73 SHORT PenHeight;
74 struct TextFont *Font; /* current font address */
75 UBYTE AlgoStyle; /* the algorithmically generated style */
76 UBYTE TxFlags; /* text specific flags */
77 UWORD TxHeight; /* text height */
78 UWORD TxWidth; /* text nominal width */
79 UWORD TxBaseline; /* text baseline */
80 UWORD TxSpacing; /* text spacing (per character) */
81 APTR *RP_User;
82 UWORD wordreserved[7]; /* used to be a node */
83 UWORD longreserved[2];
84 UBYTE reserved[8]; /* for future use */
85 };
86 /* drawing modes */
87 #define JAM1 0 /* jam 1 color into raster */
88 #define JAM2 1 /* jam 2 colors into raster */
89 #define COMPLEMENT 2 /* XOR bits into raster */
90 #define INVERSVID 4 /* inverse video for drawing modes */
91
92 /* these are the flag bits for RastPort flags */
93 #define FRST_DOT 0x01 /* draw the first dot of this line ? */
94 #define ONE_DOT 0x02 /* use one dot mode for drawing lines */
95 #define DBUFFER 0x04 /* flag set when RastPorts
96 are double-buffered */
97
98 /* only used for bobs */
99
100 #define AREAOUTLINE 0x08 /* used by areafiller */
101 #define NOCROSSFILL 0x20 /* areafills have no crossovers */
102
103 /* there is only one style of clipping: raster clipping */
104 /* this preserves the continuity of jaggles regardless of clip window */
105 /* When drawing into a RastPort, if the ptr to ClipRect is nil then there
106 is no clipping done, this is dangerous but useful for speed */
107
108 #endif

```

```

1 #ifndef GRAPHICS_REGIONS_H
2 #define GRAPHICS_REGIONS_H
3
4 #ifndef GRAPHICS_GFX_H
5 #include <graphics/gfx.h>
6 #endif
7
8 /*
9  * Commodore-Amiga, Inc.
10  */
11
12 struct RegionRectangle
13 {
14     struct RegionRectangle *Next, *Prev;
15     struct RegionRectangle bounds;
16 };
17
18 struct Region
19 {
20     struct RegionRectangle *RegionRectangle;
21 };
22
23 #endif

```

```

1  #ifndef GRAPHICS_SPRITE_H
2  #define GRAPHICS_SPRITE_H
3  /*
4  /* Commodore-Amiga, Inc.
5  /* sprite.h
6  /*
7
8  #define SPRITE_ATTACHED 0x80
9
10 struct SimpleSprite
11 {
12     UWORD *posctldata;
13     UWORD height;
14     UWORD x,y; /* current position */
15     UWORD num;
16 };
17 #endif

```

```

1  #ifndef GRAPHICS_TEXT_H
2  #define GRAPHICS_TEXT_H
3  /*
4  /* Commodore-Amiga, Inc.
5  /* file.h
6  /*
7  /* graphics library text structures
8  *
9  *
10 *****
11 #ifndef EXEC_PORTS_H
12 #include "exec/ports.h"
13 #endif
14
15 /*----- Font Styles -----*/
16 #define FS_NORMAL 0 /* normal text (no style bits set) */
17 #define FSE_EXTENDED 3 /* extended face (wider than normal) */
18 #define FSE_ITALIC 2 /* italic (slanted 1:2 right) */
19 #define FSB_BOLD 1 /* bold face text (ORed w/ slanted) */
20 #define FSE_UNDERLINED 0 /* underlined (under baseline) */
21 #define FSE_UNDERLINED (1<<0)
22
23 /*----- Font Flags -----*/
24 #define FPB_ROMFONT 0 /* font is in rom */
25 #define FPB_DISKFONT 1 /* font is from diskfont.library */
26 #define FPD_DISKFONT (1<<1) /* designed path is reversed (e.g. left) */
27 #define FPR_REVPATH 2 /* designed for hires non-interlaced */
28 #define FPD_TALLOTT 3 /* designed for lores interlaced */
29 #define FPD_WIDEDOT 4 /* character sizes can vary from nominal */
30 #define FPD_PROPORTIONAL 5 /* size is "designed", not constructed */
31 #define FPD_DESIGNED 6 /* the font has been removed */
32 #define FPD_REMOVED 7
33 #define FPD_REMOVED (1<<7)
34
35 ***** TextAttr node, matches text attributes in RastPort *****
36 struct TextAttr {
37     STRPTR ta_Name; /* name of the font */
38     UWORD ta_YSIZE; /* height of the font */
39     UBYTE ta_Style; /* intrinsic font style */
40     UBYTE ta_Flags; /* font preferences and flags */
41 };
42
43 ***** TextFonts node *****
44 struct TextFont {
45     struct Message tf_Message; /* reply message for font removal */

```



```

57      /* font name in LN          \ used in this */
58      /* font height             | order to best */
59      /* font style              | match a font */
60      /* preferences and flags   | request. */
61      /* nominal font width      */
62      /* distance from the top of char to baseline */
63      /* smear to affect a bold enhancement */
64
65      /* access count */
66
67      /* the first character described here */
68      /* the last character described here */
69      /* the bit character data */
70
71      /* the row modulo for the strike font data */
72      /* ptr to location data for the strike font */
73      /* 2 words: bit offset then size */
74      /* ptr to words of proportional spacing data */
75      /* ptr to words of kerning data */
76
77
78 #endif

```

```

1  #ifndef GRAPHICS_VIEW_H
2  #define GRAPHICS_VIEW_H
3
4  #ifndef GRAPHICS_GFX_H
5  #include <graphics/gfx.h>
6  #endif
7
8  /****** Commodore-Amiga, Inc. *****/
9  /* View.h *****/
10 /****** *****/
11 /****** *****/
12 /****** *****/
13 * Modification History
14 * date : author : Comments
15 * -----
16 * 2-4-85 Dale created from graph.h
17 * 2-8-85 Dale conversion to 24 View->ViewPort
18 *
19 *****
20 struct ColorMap
21 {
22     UBYTE Flags;
23     UBYTE Type;
24     UWORD Count;
25     APTR ColorTable;
26 };
27 /* if Type == 0 then ColorTable is a table of UWORDS xRGB */
28
29 struct ViewPort
30 {
31     struct ViewPort *Next;
32     struct ColorMap *ColorMap; /* table of colors for this viewport */
33     /* if this is nil, MakeVPort assumes default values */
34     struct CopList *DapIns; /* user by MakeView() */
35     struct CopList *SprIns; /* used by sprite stuff */
36     struct CopList *CirIns; /* used by sprite stuff */
37     struct UCopList *UCopIns; /* User copper list */
38     SHORT DWidth,DHeight;
39     SHORT DxOffset,DyOffset;
40     UWORD Modes;
41     UWORD reserved;
42     struct RasInfo *RasInfo;
43 };
44
45 struct View
46 {
47     struct ViewPort *ViewPort; /* used for interlaced and noninterlaced */
48     struct CopList *LOFCopList; /* only used during interlace */
49     struct CopList *SHFCopList; /* for complete View positioning */
50     short DyOffset,DxOffset; /* offsets are +- adjustments to standard */
51     UWORD Modes; /* such as INTERLACE, GENLOC */
52 };
53
54
55 /* defines used for Modes in IVPargs */
56 #define PFBA 0x40

```

```

57 #define DUALPF 0x400
58 #define HIRRES 0x8000
59 #define LACE 4
60 #define HAM 0x800
61 #define SFRITES 0x4000
62 #define VP_HIDE 0x2000
63 #define GENLOCK_AUDIO 0x100
64 #define GENLOCK_VIDEO 2
65 #define EXTRA_HALFBRITE 0x80
66
67 struct RasInfo /* used by callers to and InitDepC() */
68 {
69     struct RasInfo *Next; /* used for dualpf */
70     struct BitMap *BitMap; /* scroll offsets in this BitMap */
71     SHORT RxOffset,RyOffset;
72 };
73
74 #endif

```

/* reuse one of plane ctr bits */
 /* reuse another plane crt bit */

```

1 /******
2 * Commodore-Amiga, Inc.
3 * adkbits.h -- bit definitions for adkcon register
4 *
5 * $Header: adkbits.h,v 27.1 85/06/24 14:42:34 nell Exp $
6 *
7 * $Locker: $
8 *
9 *****/
10
11 #ifndef HARDWARE_ADKBITS_H
12 #define HARDWARE_ADKBITS_H
13
14 #define ADKB_SETCCLR 15 /* standard set/clear bit */
15 #define ADKB_PRECOMP1 14 /* two bits of precompensation */
16 #define ADKB_PRECOMP0 13 /* use mfm style precompensation */
17 #define ADKB_MEMPREC 12 /* force uart output to zero */
18 #define ADKB_UARTTRK 11 /* enable DSKSYNC register matching */
19 #define ADKB_WORDSYNC 10 /* (Apple OCR Only) sync on MSB for reading */
20 #define ADKB_MSBSYNC 9 /* 1 -> 2 us/bit (mfm), 2 -> 4 us/bit (gcr) */
21 #define ADKB_FAST 8 /* use aud chan 3 to modulate period of ?? */
22 #define ADKB_USE3PN 7 /* use aud chan 2 to modulate period of 3 */
23 #define ADKB_USE2P3 6 /* use aud chan 1 to modulate period of 2 */
24 #define ADKB_USE1P2 5 /* use aud chan 0 to modulate period of 1 */
25 #define ADKB_USE0P1 4 /* use aud chan 3 to modulate volume of ?? */
26 #define ADKB_USE3VN 3 /* use aud chan 2 to modulate volume of 3 */
27 #define ADKB_USE2V3 2 /* use aud chan 1 to modulate volume of 2 */
28 #define ADKB_USE1V2 1 /* use aud chan 0 to modulate volume of 1 */
29 #define ADKB_USE0V1 0 /* use aud chan 0 to modulate volume of 1 */
30
31 #define ADKF_SETCCLR (1<<15)
32 #define ADKF_PRECOMP1 (1<<14)
33 #define ADKF_PRECOMP0 (1<<13)
34 #define ADKF_MEMPREC (1<<12)
35 #define ADKF_UARTTRK (1<<11)
36 #define ADKF_WORDSYNC (1<<10)
37 #define ADKF_MSBSYNC (1<<9)
38 #define ADKF_FAST (1<<8)
39 #define ADKF_USE3PN (1<<7)
40 #define ADKF_USE2P3 (1<<6)
41 #define ADKF_USE1P2 (1<<5)
42 #define ADKF_USE0P1 (1<<4)
43 #define ADKF_USE3VN (1<<3)
44 #define ADKF_USE2V3 (1<<2)
45 #define ADKF_USE1V2 (1<<1)
46 #define ADKF_USE0V1 (1<<0)
47
48 #define ADKE_PRE00NS 0 /* 000 ns of precomp */
49 #define ADKE_PRE140NS (ADKE_PRECOMP0) /* 140 ns of precomp */
50 #define ADKE_PRE280NS (ADKE_PRECOMP1) /* 280 ns of precomp */
51 #define ADKE_PRE560NS (ADKE_PRECOMP0|ADKE_PRECOMP1) /* 560 ns of precomp */
52
53 #endif HARDWARE_ADKBITS_H

```

```

1  /******
2  * blit.h
3  * Commodore-Amiga, Inc.
4  *
5  * $Header: blit.h,v 27.1 85/06/24 14:42:40 nell Exp $
6  *
7  * $Locker:  $
8  *
9  *
10 *****
11 #ifndef HARDWARE_BLIT_H
12 #define HARDWARE_BLIT_H
13
14 /* include file for blitter */
15 #define HSIZEBITS 6
16 #define VSIZEBITS 16-HSIZEBITS /* 2^6 -- 1 */
17 #define HSIZEMASK 0x3f /* 2^10 - 1 */
18 #define VSIZEMASK 0x3ff
19
20 #define MAXBYTESPERROW 128
21
22 /* definitions for blitter control register 0 */
23
24 #define ABC 0x80
25 #define ABNC 0x40
26 #define ANBC 0x20
27 #define ANBNC 0x10
28 #define NABC 0x8
29 #define NABNC 0x4
30 #define NANBC 0x2
31 #define NANBNC 0x1
32
33 /* some commonly used operations */
34 #define A_OR_B ABC|ANBC|NABC | ABNC|ANBNC|NABNC
35 #define A_OR_C ABC|ANBC|ABNC | ANBC|NANBC|ANBNC
36 #define A_XOR_C NABC|ABNC | NANBC|ANBNC
37 #define A_TO_D ABC|ANBC|ABNC|ANBNC
38
39 #define BCOB_DEST 8
40 #define BCOB_SROC 9
41 #define BCOB_SRCB 10
42 #define BCOB_SRCB 11
43 #define BCOF_DEST 0x100
44 #define BCOF_SROC 0x200
45 #define BCOF_SRCB 0x400
46 #define BCOF_SRCB 0x800
47
48 #define BCIF_DESC 2 /* blitter descend direction */
49
50 #define DEST 0x100
51 #define SROC 0x200
52 #define SRCB 0x400
53 #define SRCA 0x800
54
55 #define ASHIFTSHIFT 12 /* bits to right align ashift value */
56 #define BSHIFTSHIFT 12 /* bits to right align bshift value */

```

```

57 /* definitions for blitter control register 1 */
58 #define LINEMODE 0x1
59 #define FILL_OR 0x8
60 #define FILL_XOR 0x10
61 #define FILL_CARRYIN 0x4
62 #define FILL_CARRYIN 0x4
63 #define ONEDOT 0x2
64 #define OVELAG 0x20
65 #define SGNELAG 0x40
66 #define BLITREVERSE 0x2
67
68 #define SUD 0x10
69 #define SUL 0x8
70 #define AUL 0x4
71
72 #define OCTANT8 24
73 #define OCTANT7 4
74 #define OCTANT6 12
75 #define OCTANT5 28
76 #define OCTANT4 20
77 #define OCTANT3 8
78 #define OCTANT2 0
79 #define OCTANT1 16
80
81 /* stuff for blit geuer */
82 struct bltnode
83 {
84     struct bltnode *n;
85     int (*function) ();
86     char stat;
87     short blitsize;
88     short beamsync;
89     int (*cleanup) ();
90 };
91
92 /* defined bits for bltstat */
93 #define CLEANUP 0x40
94 #define CLEANW CLEANUP
95 #define !HARDWARE_BLIT_H
96

```

```

1 /******
2 /* Commodore-Amiga, Inc.
3 /* cia.h
4 /******
5 #define CIAANAME "ciaa.resource"
6 #define CIABNAME "ciab.resource"
7
8

```

```

1 /******
2 * Commodore-Amiga, Inc.
3 * custom.h
4 *
5 * $Header: custom.h,v 27.1 85/06/24 14:42:53 neil Exp $
6 *
7 * $Locker:  $
8 *
9 *****
10
11 #ifndef HARDWARE_CUSTOM_H
12 #define HARDWARE_CUSTOM_H
13
14 /*
15 * do this to get base of custom registers:
16 * extern struct Custom custom;
17 */
18
19
20 struct Custom {
21     UMORD bitddat;
22     UMORD dmaconr;
23     UMORD vposr;
24     UMORD vposw;
25     UMORD diskdatr;
26     UMORD joy0dat;
27     UMORD joy1dat;
28     UMORD cixdat;
29     UMORD adkconr;
30     UMORD pot0dat;
31     UMORD pot1dat;
32     UMORD pot1np;
33     UMORD serdatr;
34     UMORD disdytr;
35     UMORD intonar;
36     UMORD intreqr;
37     APTX  diskpt;
38     UMORD dsklen;
39     UMORD dskdat;
40     UMORD reiptr;
41     UMORD vposv;
42     UMORD vposw;
43     UMORD copcon;
44     UMORD serdat;
45     UMORD serpar;
46     UMORD potgo;
47     UMORD joytest;
48     UMORD strequ;
49     UMORD strvbl;
50     UMORD strhor;
51     UMORD strlong;
52     UMORD bitcon0;
53     UMORD bitcon1;
54     UMORD bitafwm;
55     UMORD bitalwm;
56     APTX  bitcpt;

```

```

57  APTR  bltbpt;
58  APTR  bltapt;
59  APTR  bltdpt;
60  UMORD bitsize;
61  UMORD pad2d[3];
62  UMORD bltcm0d;
63  UMORD bltbmod;
64  UMORD bltamod;
65  UMORD bltdmod;
66  UMORD pad34[4];
67  UMORD bitcodat;
68  UMORD bltbdat;
69  UMORD bltadat;
70  UMORD pad3b[4];
71  UMORD dsksync;
72  ULONG copllc;
73  ULONG cop2lc;
74  UMORD copjmp1;
75  UMORD copjmp2;
76  UMORD copins;
77  UMORD divstrt;
78  UMORD divst0p;
79  UMORD ddfstprt;
80  UMORD ddfst0p;
81  UMORD dmacon;
82  UMORD clxcon;
83  UMORD intena;
84  UMORD intrq;
85  UMORD adkcon;
86  struct AudChannel {
87  UMORD *ac_ptr; /* ptr to start of waveform data */
88  UMORD ac_len; /* length of waveform in words */
89  UMORD ac_per; /* sample period */
90  UMORD ac_vol; /* volume */
91  UMORD ac_dat; /* sample pair */
92  UMORD ac_pad[2]; /* unused */
93  } aud[4];
94  APTR  bplpt[5];
95  UMORD pad7c[4];
96  UMORD bplcom0;
97  UMORD bplcom1;
98  UMORD bplcom2;
99  UMORD pad83;
100 UMORD bpl1mod;
101 UMORD bpl2mod;
102 UMORD pad86[2];
103 UMORD bpldat[6];
104 UMORD pad8e[2];
105 APTR  sprpt[8];
106 struct SpritesDef {
107 UMORD pos;
108 UMORD cti;
109 UMORD dataa;
110 UMORD datab;
111 } spr[8];
112 UMORD color[32];

```

```

113 };
114 #endif HARDWARE_CUSTOM_H

```

```

1 /*****
2 * Commodore-Amiga, Inc.
3 * dmabits.h
4 *
5 * $Header: dmabits.h,v 27.1 85/06/24 14:42:59 neil Exp $
6 *
7 * $Locker: $
8 *
9 *****/
10 #ifndef HARDWARE_DMABITS_H
11 #define HARDWARE_DMABITS_H
12
13 /* include file for defining dma control stuff */
14
15 /* write definitions for dmaconr */
16 #define DMAE_SETCCLR 0x8000
17 #define DMAE_AUDIO 0x000E /* 4 bit mask */
18 #define DMAE_AUD0 0x0001
19 #define DMAE_AUD1 0x0002
20 #define DMAE_AUD2 0x0004
21 #define DMAE_AUD3 0x0008
22 #define DMAE_DISK 0x0010
23 #define DMAE_SPRITE 0x0020
24 #define DMAE_BLITTER 0x0040
25 #define DMAE_COPPER 0x0080
26 #define DMAE_RASTER 0x0100
27 #define DMAE_MASTER 0x0200
28 #define DMAE_BLITTHOG 0x0400
29 #define DMAE_ALL 0x01FF /* all dma channels */
30
31 /* read definitions for dmaconr */
32 /* bits 0-8 correspond to dmaconr definitions */
33 #define DMAE_BLITDONE 0x4000
34 #define DMAE_BLITZERO 0x2000
35
36 #define DMAB_SETCCLR 15
37 #define DMAB_AUD0 0
38 #define DMAB_AUD1 1
39 #define DMAB_AUD2 2
40 #define DMAB_AUD3 3
41 #define DMAB_DISK 4
42 #define DMAB_SPRITE 5
43 #define DMAB_BLITTER 6
44 #define DMAB_COPPER 7
45 #define DMAB_RASTER 8
46 #define DMAB_MASTER 9
47 #define DMAB_BLITTHOG 10
48 #define DMAB_BLITDONE 14
49 #define DMAB_BLITZERO 13
50
51 #endif /*HARDWARE_DMABITS_H

```

```

1 /*****
2 * Commodore-Amiga, Inc.
3 * intnabits.h -- definitions for the bits in the interrupt enable
4 * (and interrupt request) register
5 *
6 * $Header: intbits.h,v 27.1 85/06/24 14:43:04 neil Exp $
7 *
8 * $Locker: $
9 *
10 *****/
11 #ifndef HARDWARE_INTBITS_H
12 #define HARDWARE_INTBITS_H
13
14 #define INTB_SETCCLR (15) /* Set/Clear control bit. Determines if bits
15 /* written with a 1 get set or cleared. Bits */
16 #define INTB_INTEN (14) /* Master interrupt (enable only) */
17 #define INTB_EXTEN (13) /* External interrupt */
18 #define INTB_DSKSYN (12) /* Disk re-SYNChronized */
19 #define INTB_RBF (11) /* serial port Receive Buffer Full */
20 #define INTB_AUD3 (10) /* Audio channel 3 block finished */
21 #define INTB_AUD2 (9) /* Audio channel 2 block finished */
22 #define INTB_AUD1 (8) /* Audio channel 1 block finished */
23 #define INTB_AUD0 (7) /* Audio channel 0 block finished */
24 #define INTB_BLIT (6) /* Blitter finished */
25 #define INTB_VERTB (5) /* start of Vertical Blank */
26 #define INTB_COOPER (4) /* Coprocessor */
27 #define INTB_PORTS (3) /* I/O Ports and timers */
28 #define INTB_SOFTINT (2) /* software interrupt request */
29 #define INTB_DSKBLK (1) /* Disk Block done */
30 #define INTB_TBE (0) /* serial port Transmit Buffer Empty */
31
32 #define INTF_SETCCLR (1<<15)
33 #define INTF_INTEN (1<<14)
34 #define INTF_EXTEN (1<<13)
35 #define INTF_DSKSYN (1<<12)
36 #define INTF_RBF (1<<11)
37 #define INTF_AUD3 (1<<10)
38 #define INTF_AUD2 (1<<9)
39 #define INTF_AUD1 (1<<8)
40 #define INTF_AUD0 (1<<7)
41 #define INTF_BLIT (1<<6)
42 #define INTF_VERTB (1<<5)
43 #define INTF_COOPER (1<<4)
44 #define INTF_PORTS (1<<3)
45 #define INTF_SOFTINT (1<<2)
46 #define INTF_DSKBLK (1<<1)
47 #define INTF_TBE (1<<0)
48
49 #endif /*HARDWARE_INTBITS_H

```

```

1 #ifndef INTUITION_INTUITION_H
2 #define INTUITION_INTUITION_H TRUE
3
4 /** Intuition.h *****
5 * Commodore-Amiga, Inc.
6 *
7 * Intuition.h main include for c programmers
8 *
9 * Modification History
10 * date : author : Comments
11 * -----
12 * 1-30-85 --RJ-- created this file!
13 * 10-03-85 Support for HP printers
14 *
15 *****
16
17 #ifndef INTUITION_INTUITIONBASE_H
18 #include "intuition/intuitionbase.h"
19 #endif
20
21 #ifndef GRAPHICS_GFX_H
22 #include "graphics/gfx.h"
23 #endif
24
25 #ifndef GRAPHICS_CLIP_H
26 #include "graphics/clip.h"
27 #endif
28
29 #ifndef GRAPHICS_VIEW_H
30 #include "graphics/view.h"
31 #endif
32
33 #ifndef GRAPHICS_RASPORT_H
34 #include "graphics/rasport.h"
35 #endif
36
37 #ifndef GRAPHICS_LAYERS_H
38 #include "graphics/layers.h"
39 #endif
40
41 #ifndef GRAPHICS_TEXT_H
42 #include "graphics/text.h"
43 #endif
44
45 #ifndef EXEC_PORTS_H
46 #include "exec/ports.h"
47 #endif
48
49 #ifndef DEVICES_TIMER_H
50 #include "devices/timer.h"
51 #endif
52
53 #ifndef DEVICES_INPUTEVENT_H
54 #include "devices/inputevent.h"
55 #endif
56

```

```

57 /**
58 == Menu
59 /**
60 struct Menu
61 {
62     struct Menu *NextMenu; /* same level */
63     SHORT LeftEdge, TopEdge; /* position of the select box */
64     SHORT Width, Height; /* dimensions of the select box */
65     USHORT Flags; /* see flag definitions below */
66     BYTE *MenuName; /* text for this Menu Header */
67     struct MenuItem *FirstItem; /* pointer to first in chain */
68
69     /* these mysteriously-named variables are for internal use only */
70     SHORT JazzX, JazzY, BeatX, BeatY;
71 };
72
73
74
75 /* FLAGS SET BY BOTH THE APPLIPROG AND INTUITION */
76 #define MENUENABLED 0x0001 /* whether or not this menu is enabled */
77
78 /* FLAGS SET BY INTUITION */
79 #define MIDRAWN 0x0100 /* this menu's items are currently drawn */
80
81
82
83
84
85
86
87 == MenuItem
88 /**
89 struct MenuItem
90 {
91     struct MenuItem *NextItem; /* pointer to next in chained list */
92     SHORT LeftEdge, TopEdge; /* position of the select box */
93     SHORT Width, Height; /* dimensions of the select box */
94     USHORT Flags; /* see the defines below */
95
96     LONG MutualExclude; /* set bits mean this item
97     /* excludes that */
98     APTR ItemFill; /* points to Image, IntuiText, or NULL
99
100     /* when this item is pointed to by the cursor and the items highlight
101     * mode HIGHIMACE is selected, this alternate image will be displayed
102     */
103     APTR SelectFill; /* points to Image, IntuiText, or NULL */
104
105     BYTE Command; /* only if applipro sets the COMSEQ f)
106
107     struct MenuItem *SubItem; /* if non-zero, DrawMenu shows "->" */
108
109     /* The NextSelect field represents the menu number of next selected
110     * item (when user has drag-selected several items)
111     */
112     USHORT NextSelect;

```

```

113 );
114
115 /* FLAGS SET BY THE APPLIPROG */
116 #define CHECKIT 0x0001 /* whether to check this item if selected */
117 #define ITEMTEXT 0x0002 /* set if textual, clear if graphical item */
118 #define COMSEQ 0x0004 /* set if there's a command sequence */
119 #define MENTUOGGLE 0x0008 /* set to toggle the check of a menu item */
120 #define ITENABLED 0x0010 /* set if this item is enabled */
121
122 /* these are the SPECIAL HIGHLIGHT FLAG state meanings */
123 #define HIGHEFLAS 0x00C0 /* see definitions below for these bits */
124 #define HIGHIWACE 0x0000 /* use the user's "select image" */
125 #define HIGHCOMP 0x0040 /* highlight by complementing the selectbox */
126 #define HIGHBOX 0x0080 /* highlight by "boxing" the selectbox */
127 #define HIGHWONE 0x00C0 /* don't highlight */
128
129 /* FLAGS SET BY BOTH APPLIPROG AND INTUITION */
130 #define CHECKED 0x0100 /* if CHECKIT, then set this when selected */
131
132 /* FLAGS SET BY INTUITION */
133 #define ISDRAWN 0x1000 /* this item's subs are currently drawn */
134 #define HIGHTEM 0x2000 /* this item is currently highlighted */
135 #define MENTUOGGLED 0x4000 /* this item was already toggled */
136
137
138
139
140
141
142 // Requester
143 // Requester
144 // Requester
145 struct Requester
146 {
147     /* the Cliprect and BitMap and used for rendering the requester */
148     struct Requester *OlderRequest; /* dimensions of the entire box */
149     SHORT LeftEdge, TopEdge; /* dimensions of the entire box */
150     SHORT Width, Height; /* dimensions of the entire box */
151     SHORT RelLeft, RelTop; /* for Pointer relativity offsets */
152
153     struct Gadget *ReqGadget; /* pointer to a list of Gadgets */
154     struct Border *ReqBorder; /* the box's border */
155     struct IntuiText *ReqText; /* the box's text */
156     USHORT Flags; /* see definitions below */
157
158     /* pen number for back-plane fill before draws */
159     UBYTE BackFill;
160     /* Layer in place of clip rect */
161     struct Layer *ReqLayer;
162
163     UBYTE ReqPad1[32];
164
165     /* If the BitMap plane pointers are non-zero, this tells the system
166     * that the image comes pre-drawn (if the appliprogram wants to define
167     * it's own box, in any shape or size it wants!); this is OK by
168     * Intuition as long as there's a good correspondence between

```

```

169 * the image and the specified Gadgets
170 */
171 struct BitMap *ImageMap; /* points to the BitMap of PREDRAWN Imagery
172 struct Window *RWindow; /* added. points back to Window */
173 UBYTE ReqPad2[36]
174 };
175
176 /* FLAGS SET BY THE APPLIPROG */
177 #define POINTREL 0x0001 /* if POINTREL set, TopLeft is relative to point.
178 #define PREDRAWN 0x0002 /* if ReqMap points to predrawn Requester
179 Imagery */
180 /* FLAGS SET BY BOTH THE APPLIPROG AND INTUITION */
181
182 /* FLAGS SET BY INTUITION */
183 #define REQOFFWINDOW 0x1000 /* part of one of the Gadgets was offwindow
184 #define REQACTIVE 0x2000 /* this requester is active */
185 #define SYSREQUEST 0x4000 /* this requester caused by system */
186 #define DEFERREFRESH 0x8000 /* this Requester stops a Refresh broadcast
187
188
189
190
191
192
193
194 // Gadget
195 // Gadget
196 // Gadget
197 struct Gadget
198 {
199     struct Gadget *NextGadget; /* next gadget in the list */
200
201     SHORT LeftEdge, TopEdge; /* "hit box" of gadget */
202     SHORT Width, Height; /* "hit box" of gadget */
203
204     USHORT Flags; /* see below for list of defines */
205
206     USHORT Activation; /* see below for list of defines */
207
208     USHORT GadgetType; /* see below for defines */
209
210     /* appliprogram can specify that the Gadget be rendered as either as Bord-
211     * or an Image. This variable points to which (or equals NULL if there
212     * nothing to be rendered about this Gadget)
213     */
214     APTR GadgetRender;
215
216     /* appliprogram can specify "highlighted" imagery rather than algorithmic
217     * this can point to either Border or Image data
218     */
219     APTR SelectRender;
220
221     struct IntuiText *GadgetText; /* text for this gadget */
222
223     /* by using the MutualExclude word, the appliprogram can describe
224     * which gadgets mutually-exclude which other ones. The bits

```



```

225 * In MutualExclude correspond to the gadgets in object containing
226 * the gadget list. If this gadget is selected and a bit is set
227 * in this gadget's MutualExclude and the gadget corresponding to
228 * that bit is currently selected (e.g. bit 2 set and gadget 2
229 * is currently selected) that gadget must be unselected.
230 * Intuition does the visual unselecting (with checkmarks) and
231 * leaves it up to the program to unselect internally
232 */
233 LONG MutualExclude; /* set bits mean this gadget excludes
234 * that gadget */
235
236 /* pointer to a structure of special data required by Proportional,
237 * String and Integer Gadgets
238 */
239 APTR SpecialInfo;
240
241 USHRT GadgetID; /* user-definable ID field */
242 APTR UserData; /* ptr to general purpose User data
243 * (ignored by In) */
244 };
245
246
247 /* --- FLAGS SET BY THE APPLIPROC ---
248 * combinations in these bits describe the highlight technique to be used */
249 #define GADGHIGHBITS 0x0003
250 #define GADGCHOMP 0x0000 /* Complement the select box */
251 #define GADGCHBOX 0x0001 /* Draw a box around the image */
252 #define GADGCHIMAGE 0x0002 /* Blast in this alternate image */
253 #define GADGCHNONE 0x0003 /* don't highlight */
254
255 /* set this flag if the GadgetRender and SelectRender point to Image imagery,
256 * clear if it's a Border
257 */
258 #define GADGIMAGE 0x0004
259
260 /* combinations in these next two bits specify to which corner the gadget's
261 * Left & Top coordinates are relative. If relative to Top/Left,
262 * these are "normal" coordinates (everything is relative to something in
263 * this universe)
264 */
265 #define GRELBOTTOM 0x0008 /* set if rel to bottom, clear if rel top */
266 #define GRELRIGHT 0x0010 /* set if rel to right, clear if to left */
267 /* set the RELWIDTH bit to spec that Width is relative to width of screen */
268 #define GRELWIDTH 0x0020
269 /* set the RELHEIGHT bit to spec that Height is rel to height of screen */
270 #define GRELHEIGHT 0x0040
271
272 /* the SELECTED flag is initialized by you and set by Intuition. It
273 * specifies whether or not this Gadget is currently selected/highlighted
274 */
275 #define SELECTED 0x0080
276
277
278 /* the GADGDISABLED flag is initialized by you and later set by Intuition
279 * according to your calls to On/OffGadget(). It specifies whether or not
280 * this Gadget is currently disabled from being selected

```

```

281 */
282 #define GADGDISABLED 0x0100
283
284
285 /* --- These are the Activation flag bits ---
286 * RELVERIFY is set if you want to verify that the pointer was still over
287 * the gadget when the select button was released
288 */
289 #define RELVERIFY 0x0001
290
291 /* the flag GADGIMMEDIATE, when set, informs the caller that the gadget
292 * was activated when it was activated. this flag works in conjunction wit
293 * the RELVERIFY flag
294 */
295 #define GADGIMMEDIATE 0x0002
296
297 /* the flag ENDGADGET, when set, tells the system that this gadget, when
298 * selected, causes the Requester or AbsMessages to be ended. Requesters or
299 * AbsMessages that are ended are erased and unlinked from the system */
300 #define ENDGADGET 0x0004
301
302 /* the FOLLOWMOUSE flag, when set, specifies that you want to receive
303 * reports on mouse movements (ie, you want the REPORTMOUSE function for
304 * your Window). When the Gadget is deselected (immediately if you have
305 * no RELVERIFY) the previous state of the REPORTMOUSE flag is restored
306 * you probably want to set the GADGIMMEDIATE flag when using FOLLOWMOUSE,
307 * since that's the only reasonable way you have of learning why Intuition
308 * is suddenly sending you a stream of mouse movement events. If you don't
309 * set RELVERIFY, you'll get at least one Mouse Position event.
310 */
311 #define FOLLOWMOUSE 0x0008
312
313 /* If any of the BORDER flags are set in a Gadget that's included in the
314 * Gadget list when a Window is opened, the corresponding Border will
315 * be adjusted to make room for the Gadget
316 */
317 #define RIGHTBORDER 0x0010 /* this bit for toggle-select mode */
318 #define LEFTBORDER 0x0020 /* should be a StringInfo flag, but it's On
319 #define TOPBORDER 0x0040 /* should be a StringInfo flag, but it's
320 #define BOTTOMBORDER 0x0080
321
322 #define TOGGLESELECT 0x0100 /* this bit for toggle-select mode */
323
324 #define STRINGCENTER 0x0200 /* should be a StringInfo flag, but it's On
325 #define STRINGRIGHT 0x0400 /* should be a StringInfo flag, but it's
326 #define LONGINT 0x0800 /* this String Gadget is actually LONG Int
327 #define ALTKEYMAP 0x1000 /* this String has an alternate keypad */
328
329
330
331 /* --- GADGET TYPES ---
332 * These are the Gadget Type definitions for the variable GadgetType
333 * gadget number type MUST start from one. NO TYPES OF ZERO ALLOWED.
334 * first comes the mask for Gadget flags reserved for Gadget typing
335 */
336

```

```

337 #define GADGETTYPE      0x8000 /* all Gadget Global Type flags (padded) */
338 #define SYSGADGET      0x8000 /* 1 = SysGadget, 0 = AppliGadget */
339 #define SCRGADGET      0x4000 /* 1 = ScreenGadget, 0 = WindowGadget */
340 #define GZCGADGET      0x2000 /* 1 = Gadget for GIM#ZEROZERO borders */
341 #define REQCGADGET      0x1000 /* 1 = this is a Requester Gadget */
342 /* system gadgets */
343 #define SIZING          0x0010
344 #define WRAGGING        0x0020
345 #define SRAGGING        0x0030
346 #define WUPFRONT        0x0040
347 #define SUPFRONT        0x0050
348 #define WDOWNBACK       0x0060
349 #define SDOWNBACK       0x0070
350 #define CLOSE           0x0080
351 /* application gadgets */
352 #define BOULGADGET      0x0001
353 #define GADGET0002      0x0002
354 #define PRPGADGET       0x0003
355 #define STRGADGET       0x0004
356
357
358
359
360
361
362
363 // // //
364 // // //
365 // // //
366 // this is the special data required by the proportional Gadget
367 // typically, this data will be pointed to by the Gadget variable SpecialInfo
368 struct PropInfo
369 {
370     USHORT Flags; /* general purpose flag bits (see defines below) */
371
372     /* You initialize the Pot variables before the Gadget is added to
373     * the system. Then you can look here for the current settings
374     * any time, even while User is playing with this Gadget. To
375     * adjust these after the Gadget is added to the System, use
376     * ModifyProp(); The Pots are the actual proportional settings,
377     * where a value of zero means zero and a value of MAXPOT means
378     * that the Gadget is set to its maximum setting.
379     */
380     USHORT HorizPot; /* 16-bit FixedPoint horizontal quantity percentage */
381     USHORT VertPot; /* 16-bit FixedPoint vertical quantity percentage */
382
383     /* the 16-bit FixedPoint Body variables describe what percentages of
384     * the entire body of stuff referred to by this Gadget is actually
385     * shown at one time. This is used with the AUTOKNOB routines,
386     * to adjust the size of the AUTOKNOB according to how much of
387     * the data can be seen. This is also used to decide how far
388     * to advance the Pots when User hits the Container of the Gadget.
389     * For instance, if you were controlling the display of a 5-line
390     * Window of text with this Gadget, and there was a total of 15
391     * lines that could be displayed, you would set the VertBody value to
392     * (MAXBODY / (TotalLines / DisplayLines)) = MAXBODY / 3.

```

```

393
394
395
396
397
398
399
400     USHORT HorizBody; /* horizontal Body */
401     USHORT VertBody; /* vertical Body */
402
403     /* these are the variables that Intuition sets and maintains */
404     USHORT CWidth; /* Container width (with any relativity absolved)
405     USHORT CHeight; /* Container height (with any relativity absolved)
406     USHORT HPotRes; /* pot increments */
407     USHORT LeftBorder; /* Container borders */
408     USHORT TopBorder; /* Container borders */
409
410
411
412
413     #define AUTOKNOB      0x0001 /* this flag sez: gimms that old auto-kno
414     #define FREEHORIZ     0x0002 /* if set, the knob can move horizontally
415     #define FREEVERT      0x0004 /* if set, the knob can move vertically */
416     #define PROPORDERLESS 0x0008 /* if set, no border will be rendered */
417     #define KNOBHIT       0x0100 /* set when this Knob is hit */
418
419     #define KNOBMIN        6 /* minimum horizontal size of the Knob */
420     #define KNOBWHIN      4 /* minimum vertical size of the Knob */
421     #define MAXBODY       0xFFFF /* maximum body value */
422     #define MAXPOT        0xFFFF /* maximum pot value */
423
424
425
426
427
428
429 // // //
430 // // //
431 // // //
432 // this is the special data required by the string Gadget
433 // typically, this data will be pointed to by the Gadget variable SpecialIn
434 //
435 struct StringInfo
436 {
437     /* you initialize these variables, and then Intuition maintains them
438     * UBYTE *Buffer; /* the buffer containing the start and final string
439     * UBYTE *UndoBuffer; /* optional buffer for undoing current entry */
440     USHORT BufferPos; /* character position in Buffer */
441     USHORT MaxChars; /* max number of chars in Buffer (including NULL)
442     USHORT Disppos; /* Buffer position of first displayed character */
443
444     /* Intuition initializes and maintains these variables for you */
445     USHORT UndoPos; /* character position in the undo buffer */
446     USHORT NumChars; /* number of characters currently in Buffer */
447     USHORT DispCount; /* number of whole characters visible in Container
448     USHORT CLeft, CTop; /* topleft offset of the container */

```

```

449 struct Layer *LayerPtr; /* the RastPort containing this Gadget */
450
451 /* you can initialize this variable before the gadget is submitted to
452  * Intuition, and then examine it later to discover what Integer
453  * the user has entered (if the user never plays with the gadget,
454  * the value will be unchanged from your initial setting)
455  */
456 LONG LongInt;
457
458 /* If you want this Gadget to use your own Console keymapping, you
459  * set the ALTKEYMAP bit in the Activation flags of the Gadget, and then
460  * set this variable to point to your keymap. If you don't set the
461  * ALTKEYMAP, you'll get the standard ASCII keymapping.
462  */
463 struct KeyMap *AltKeyMap;
464
465 };
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504

```

```

505 struct Border
506 {
507     SHORT LeftEdge, TopEdge; /* initial offsets from the origin */
508     SHORT FrontPen, BackPen; /* pens numbers for rendering */
509     UBYTE DrawMode; /* mode for rendering */
510     BYTE Count; /* number of XY pairs */
511     SHORT *XY; /* vector coordinate pairs relative
512                * to LeftTop */
513     struct Border *NextBorder; /* pointer to any other Border too */
514 };
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560

```

```

561      * (pick no planes of data) and set PlaneOnOff to describe the pen
562      * color of the rectangle.
563      */
564      UBYTE   PlanePick, PlaneOnOff;
565
566      /* If the NextImage variable is not NULL, Intuition presumes that
567      * it points to another Image structure with another Image to be
568      * rendered
569      */
570      struct Image *NextImage;
571 };
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616

```

```

617      /* system-use variable */
618      struct IntuiMessage *SpecialLink;
619 };
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672

```

```

673 * find out fast whether or not this Message is available for me to send.
674 */
675 #define LONELYMESSAGE 0x80000000
676
677
678 /* --- IDCMP Codes ----- */
679 /* This group of codes is for the MENUVERIFY function */
680 #define MENUHOT 0x0001 /* Intu!wants verification or MENCANCEL */
681 #define MENCANCEL 0x0002 /* HOT Reply of this cancels Menu operation */
682 #define MENUWAITING 0x0003 /* Intuition simply wants a Reply!msg() ASAP */
683
684 /* This group of codes is for the WBEINCMESAGE messages */
685 #define WBEINCHOPEN 0x0001
686 #define WBEINCHCLOSE 0x0002
687
688
689
690 //==== Window =====
691 //====
692 //====
693 struct Window
694 {
695     struct Window *NextWindow; /* for the linked list in a screen */
696     SHORT LeftEdge, TopEdge; /* screen dimensions of window */
697     SHORT Width, Height; /* screen dimensions of window */
698     SHORT MouseX, MouseY; /* relative to upper-left of window */
699     SHORT MinWidth, MinHeight; /* minimum sizes */
700     SHORT MaxWidth, MaxHeight; /* maximum sizes */
701     ULONG Flags; /* see below for defines */
702     struct Menu *MenuStrip; /* the strip of Menu headers */
703     UBYTE *Title; /* the title text for this window */
704     struct Requester *FirstRequest; /* all active Requesters */
705     struct Requester *DMRequest; /* double-click Requester */
706     SHORT ReqCount; /* count of reqs blocking Window */
707     struct Screen *NScreen; /* this Window's Screen */
708     struct RastPort *RPort; /* this Window's very own RastPort */
709
710     /* the border variables describe the window border. If you specify
711     * GIMMEZERO when you open the window, then the upper-left of the
712     * ClipRect for this window will be upper-left of the BitMap (with correct
713     * offsets when in SuperBitMap mode; you MUST select GIMMEZERO when
714     * using SuperBitMap). If you don't specify ZeroZero, then you save
715     * memory (no allocation of RastPort, Layer, ClipRect and associated
716     * Bitmaps), but you also must offset all your writes by BorderTop,
717     * BorderLeft and do your own mini-clipping to prevent writing over the
718     * system gadgets
719
720
721
722
723
724
725
726
727
728

```

```

729 */
730 BYTE BorderLeft, BorderTop, BorderRight, BorderBottom;
731 struct RastPort *BorderRPort;
732
733
734
735 /* You supply a linked-list of Gadgets for your Window.
736 * This list DOES NOT include system gadgets. You get the standard
737 * window system gadgets by setting flag-bits in the variable Flags (see
738 * the bit definitions below)
739 */
740 struct Gadget *FirstGadget;
741
742 /* these are for opening/closing the windows */
743 struct Window *Parent, *Descendant;
744
745 /* sprite data information for your own Pointer
746 * set these AFTER you Open the Window by calling SetPointer()
747 */
748 USHORT *Pointer; /* sprite data */
749 BYTE PtrHeight; /* sprite height (not including
750 /* sprite padding) */
751 BYTE PtrWidth; /* sprite width (must be less than or
752 /* equal to 16) */
753 BYTE XOffset, YOffset; /* sprite offsets */
754
755 /* the IDCMP Flags and User's and Intuition's Message Ports */
756 ULONG IDCMPFlags; /* User-selected flags */
757 struct MsgPort *UserPort, *WindowPort;
758 struct IntuiMessage *MessageKey;
759
760 UBYTE DetailPen, BlockPen; /* for bar/border/gadget rendering */
761
762 /* the CheckMark is a pointer to the Imagery that will be used when
763 * rendering MenuItems of this Window that want to be checked
764 * if this is equal to NULL, you'll get the default imagery
765 */
766 struct Image *CheckMark;
767
768 UBYTE *ScreenTitle; /* if non-null, Screen title when Window is active
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

785 BYTE *UserData; /* general-purpose pointer to User data extension */
786
787 /** jimm: NEW: 11/18/85: this pointer keeps a duplicate of what
788 * Window.RPort->Layer is supposed_ to be pointing at
789 */
790 struct Layer *MLayer;
791 };
792
793
794 /* --- FLAGS REQUESTED (NOT DIRECTLY SET THOUGH) BY THE APPLIPROG --- */
795 #define WINDOWSIZING 0x0001 /* include sizing system-gadget? */
796 #define WINDOWDRAG 0x0002 /* include dragging system-gadget? */
797 #define WINDOWDEPTH 0x0004 /* include depth arrangement gadget? */
798 #define WINDOWCLOSE 0x0008 /* include close-box system-gadget? */
799
800 #define SIZEERIGHT 0x0010 /* size gadget uses right border */
801 #define SIZEBOTTOM 0x0020 /* size gadget uses bottom border */
802
803 /* --- refresh modes
804 /* combinations of the REFRESHBITS select the refresh type */
805 #define REFRESHBITS 0x00C0
806 #define SMART_REFRESH 0x0000
807 #define SIMPLE_REFRESH 0x0040
808 #define SUPER_BITMAP 0x0080
809 #define OTHER_REFRESH 0x00C0
810
811 #define BACKDROP 0x0100 /* this is an ever-popular BACKDROP window */
812
813 #define REPORTMOUSE 0x0200 /* set this to hear about every mouse move */
814
815 #define GIMMEZEROZERO 0x0400 /* make extra border stuff */
816
817 #define BORDERLESS 0x0800 /* set this to get a Window sans border */
818
819 #define ACTIVATE 0x1000 /* when Window opens, it's the Active one */
820
821 /* FLAGS SET BY INTUITION */
822 #define WINDOWACTIVE 0x2000 /* this window is the active one */
823 #define INREQUEST 0x4000 /* this window is in request mode */
824 #define MENUSTATE 0x8000 /* this Window is active with its Menu on */
825
826 /* --- Other User Flags
827 #define RMBTRAP 0x00010000 /* Catch RMB events for your own */
828 #define NOCAREREFRESH 0x00020000 /* not to be bothered with REFRESH */
829
830
831 /* --- Other Intuition Flags
832 #define WINDOWREFRESH 0x01000000 /* Window is currently refreshing */
833 #define WBEACHWINDOW 0x02000000 /* WorkBench tool ONLY Window */
834 #define WINDOWTICKED 0x04000000 /* only one timer tick at a time */
835
836 #define SUPER_UNUSED 0xF0000000 /* bits of Flag unused yet */
837
838
839 /* --- see struct IntuiMessage for the IDCMP Flag definitions --- */
840

```

```

841
842
843
844
845 // == NewWindow
846 //
847 struct NewWindow
848 {
849     LeftEdge, TopEdge; /* screen dimensions of window */
850     SHORT Width, Height; /* screen dimensions of window */
851
852     UBYTE DetailPen, BlockPen; /* for bar/border/gadget rendering */
853     ULONG IDCMPFlags; /* User-selected IDCMP flags */
854
855     ULONG Flags; /* see Window struct for defines */
856
857     /* You supply a linked-list of Gadgets for your Window.
858     * This list DOES NOT include system Gadgets. You get the standard
859     * system Window Gadgets by setting flag-bits in the variable Flags (s
860     * the bit definitions under the Window structure definition)
861     */
862     struct Gadget *FirstGadget;
863
864     /* the CheckMark is a pointer to the imagery that will be used when
865     * rendering Menus of this Window that want to be checkmarked
866     * if this is equal to NULL, you'll get the default imagery
867     */
868     struct Image *CheckMark;
869
870     UBYTE *Title; /* the title text for this window */
871
872     /* the Screen pointer is used only if you've defined a CUSTOMSCREEN and
873     * want this Window to open in it. If so, you pass the address of the
874     * Custom Screen structure in this variable. Otherwise, this variable
875     * is ignored and doesn't have to be initialized.
876     */
877     struct Screen *Screen;
878
879     /* SUPER_BITMAP Window? If so, put the address of your BitMap struct
880     * in this variable. If not, this variable is ignored and doesn't have
881     * to be initialized
882     */
883     struct BitMap *BitMap;
884
885     /* the values describe the minimum and maximum sizes of your Windows.
886     * these matter only if you've chosen the WINDOWSIZING Gadget option,
887     * which means that you want to let the User to change the size of
888     * this Window. You describe the minimum and maximum sizes that the
889     * Window can grow by setting these variables. You can initialize
890     * any one these to zero, which will mean that you want to duplicate
891     * the setting for that dimension (if MinWidth == 0, MinWidth will b
892     * set to the opening Width of the Window).
893     * You can change these settings later using SetWindowLimits().
894     * If you haven't asked for a SIZING Gadget, you don't have to
895     * initialize any of these variables.
896

```

```

897  *//
898  SHORT  MinWidth, MinHeight; /* minimums */
899  SHORT  MaxWidth, MaxHeight; /* maximums */
900
901  /* the type variable describes the Screen in which you want this Window to
902  * open. The type value can either be CUSTOMSCREEN or one of the
903  * system standard Screen Types such as WENCHSCREEN. See the
904  * type definitions under the Screen structure
905  */
906  USHORT Type;
907  };
908
909  /* --- FLACS SET BY INTUITION ---
910  /* The SCREENTYPE bits are reserved for describing various Screen types
911  * available under Intuition.
912  */
913  #define SCREENTYPE 0x000F /* all the screens types available */
914  /* --- the definitions for the Screen Type ---
915  #define WENCHSCREEN 0x0001 /* Ta Da! The Workbench */
916  #define CUSTOMSCREEN 0x000E /* for that special look */
917  struct Screen
918  {
919  struct Screen *NextScreen; /* linked list of screens */
920  struct Window *FirstWindow; /* linked list Screen's Windows */
921
922  SHORT LeftEdge, TopEdge; /* parameters of the screen */
923  SHORT Width, Height; /* parameters of the screen */
924
925  SHORT MouseX, MouseY; /* position relative to upper-left */
926
927  USHORT Flags; /* see definitions below */
928
929  UBYTE *Title; /* null-terminated Title text */
930  UBYTE *DefaultTitle; /* for Windows without ScreenTitle */
931
932  /* Bar sizes for this Screen and all Window's in this Screen */
933  BYTE BarHeight, BarVBorder, BarHBorder, ManuVBorder, ManuHBorder;
934  BYTE WBotTop, WBotLeft, WBotRight, WBotBottom;
935
936  struct TextAttr *Font; /* this screen's default font */
937
938  /* the display data structures for this Screen */
939  struct ViewPort ViewPort; /* describing the Screen's display */
940  struct RastPort RastPort; /* describing Screen rendering */
941  struct BitMap BitMap; /* auxiliary graphics baggage */
942  struct Layer_Info LayerInfo; /* each screen gets a LayerInfo */
943
944  /* You supply a linked-list of Gadgets for your Screen.
945  * This list DOES NOT include system Gadgets. You get the standard
946  * system Screen Gadgets by default
947  */
948  struct Gadget *FirstGadget;
949
950  UBYTE DetailPen, BlockPen; /* for bar/border/gadget rendering */
951
952  /* the following variable(s) are maintained by Intuition to support the

```

```

953  * DisplayBeep() color flashing technique
954  */
955  USHORT SaveColor0;
956
957  /* This layer is for the Screen and Menu bars */
958  struct Layer *BarLayer;
959
960  UBYTE *ExtData;
961
962  UBYTE *UserData; /* general-purpose pointer to User data extension */
963  };
964
965
966  /* --- FLACS SET BY INTUITION ---
967  /* The SCREENTYPE bits are reserved for describing various Screen types
968  * available under Intuition.
969  */
970  #define SCREENTYPE 0x000F /* all the screens types available */
971  /* --- the definitions for the Screen Type ---
972  #define WENCHSCREEN 0x0001 /* Ta Da! The Workbench */
973  #define CUSTOMSCREEN 0x000E /* for that special look */
974
975  #define SHOWTITLE 0x0010 /* this gets set by a call to
976  ShowTitle() */
977
978  #define BEEPING 0x0020 /* set when Screen is beeping */
979
980  #define CUSTOMBITMAP 0x0040 /* if you are supplying your own Bitmap
981
982
983
984
985  == NewScreen
986  ==
987  ==
988  struct NewScreen
989  {
990  SHORT LeftEdge, TopEdge, Width, Height, Depth; /* screen dimensions
991
992  UBYTE DetailPen, BlockPen; /* for bar/border/gadget rendering */
993
994  USHORT ViewModes; /* the Modes for the ViewPort (and View)
995
996  USHORT Type; /* the Screen type (see defines below)
997
998  struct TextAttr *Font; /* this Screen's default text attrib
999
1000  UBYTE *DefaultTitle; /* the default title for this Screen */
1001
1002  struct Gadget *Gadgets; /* your own Gadgets for this Screen */
1003
1004  /* if you are opening a CUSTOMSCREEN and already have a Bitmap
1005  * that you want used for your Screen, you set the flags CUSTOMBITMAP
1006  * the types variable and you set this variable to point to your Bitmap
1007  * structure. The structure will be copied into your Screen structure,
1008  * after which you may discard your own Bitmap if you want

```

```

1009  */
1010  struct BitMap *CustomBitMap;
1011  };
1012
1013
1014
1015
1016  // == Preferences ==
1017  // ==
1018  // ==
1019
1020  /* these are the definitions for the printer configurations */
1021  #define FILENAME_SIZE 30 /* File name size */
1022
1023  #define POINTERSIZE (1 + 16 + 1) * 2 /* Size of Pointer data buffer */
1024
1025  /* These defines are for the default font size. These actually describe the
1026  * height of the default fonts. The default font type is the topaz
1027  * font, which is a fixed width font that can be used in either
1028  * eighty-column or sixty-column mode. The Preferences structure reflects
1029  * which is currently selected by the value found in the variable FontSize,
1030  * which may have either of the values defined below. These values actually
1031  * are used to select the height of the default font. By changing the
1032  * height, the resolution of the font changes as well.
1033  */
1034  #define TOPAZ_EIGHTY 8
1035  #define TOPAZ_SIXTY 9
1036
1037
1038  struct Preferences
1039  {
1040  /* the default font height */ /* height for system default font */
1041  BYTE FontHeight;
1042
1043  /* constant describing what's hooked up to the port */
1044  UBYTE Pr-InterPort; /* printer port connection */
1045
1046  /* the baud rate of the port */ /* baud rate for the serial port */
1047  USHORT BaudRate;
1048
1049  /* various timing rates */
1050  struct timeval KeyRptSpeed; /* repeat speed for keyboard */
1051  struct timeval KeyRptDelay; /* Delay before keys repeat */
1052  struct timeval DoubleClick; /* Interval allowed between clicks */
1053
1054  /* Intuition Pointer data */
1055  USHORT PointerMatrix[POINTSIZESIZE]; /* Definition of pointer sprite */
1056  BYTE XOffset; /* X-Offset for active 'bit' */
1057  BYTE YOffset; /* Y-Offset for active 'bit' */
1058  USHORT color17; /* Colours for active pointer */
1059  USHORT color18; /* Colours for sprite pointer */
1060  USHORT color19; /* Colours for pointer */
1061  USHORT PointerTicks; /* Sensitivity of the pointer */
1062
1063  /* Workbench Screen colors */
1064  USHORT color0; /* Workbench colors */

```

```

1065  USHORT color1; /* Standard default colours */
1066  USHORT color2; /* Used in the Workbench */
1067  USHORT color3; /* ***** */
1068
1069  /* positioning data for the Intuition View */
1070  BYTE ViewXOffset; /* Offset for top lefthand corner */
1071  BYTE ViewYOffset; /* X and Y dimensions */
1072  WORD ViewInitX, ViewInitY; /* View initial offset values */
1073
1074  BOOL EnableCLI; /* CLI availability switch */
1075
1076  /* printer configurations */
1077  USHORT Pr-InterType; /* printer type */
1078  UBYTE Pr-InterFilename[FILENAME_SIZE]; /* file for printer */
1079
1080  /* print format and quality configurations */
1081  USHORT Pr-IntPitch; /* print pitch */
1082  USHORT Pr-IntQuality; /* print quality */
1083  USHORT Pr-IntSpacing; /* number of lines per inch */
1084  UWORD Pr-IntLeftMargin; /* left margin in characters */
1085  UWORD Pr-IntRightMargin; /* right margin in characters */
1086  USHORT Pr-IntImage; /* positive or negative */
1087  USHORT Pr-IntAspect; /* horizontal or vertical */
1088  USHORT Pr-IntShade; /* b/w, half-tone, or color */
1089  WORD Pr-IntThreshold; /* darkness ctrl for b/w dumps */
1090
1091  /* print paper descriptors */
1092  USHORT PaperSize; /* paper size */
1093  USHORT PaperLength; /* paper length in number of lines */
1094  USHORT PaperType; /* continuous or single sheet */
1095
1096  BYTE padding[50]; /* For further system expansion */
1097  };
1098
1099
1100  /* PrinterPort */
1101  #define PARALLEL_PRINTER 0x00
1102  #define SERIAL_PRINTER 0x01
1103
1104  /* BaudRate */
1105  #define BAUD_110 0x00
1106  #define BAUD_300 0x01
1107  #define BAUD_1200 0x02
1108  #define BAUD_2400 0x03
1109  #define BAUD_4800 0x04
1110  #define BAUD_9600 0x05
1111  #define BAUD_19200 0x06
1112  #define BAUD_MIDI 0x07
1113
1114  /* PaperType */
1115  #define FANFOLD 0x00
1116  #define SINGLE 0x80
1117
1118  /* PrintPitch */
1119  #define PICA 0x000
1120  #define ELITE 0x400

```



```

1177 */
1178 struct Remember
1179 {
1180     struct Remember *NextRemember;
1181     ULONGC RememberSize;
1182     UBYTE *Memory;
1183 };
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232

```

```

1121 #define FINE 0x800
1122
1123 /* PrintQuality */
1124 #define DRAFT 0x000
1125 #define LETTER 0x100
1126
1127 /* PrintSpacing */
1128 #define SIX_LPI 0x000
1129 #define EIGHT_LPI 0x200
1130
1131 /* Print Image */
1132 #define IMAGE_POSITIVE 0x00
1133 #define IMAGE_NEGATIVE 0x01
1134
1135 /* PrintAspect */
1136 #define ASPECT_HORIZ 0x00
1137 #define ASPECT_VERT 0x01
1138
1139 /* PrintShade */
1140 #define SHADE_BW 0x00
1141 #define SHADE_GREYSCALE 0x01
1142 #define SHADE_COLOR 0x02
1143
1144 /* PaperSize */
1145 #define US_LETTER 0x00
1146 #define US_LEGAL 0x10
1147 #define N_TRACTOR 0x20
1148 #define W_TRACTOR 0x30
1149 #define CUSTOM 0x40
1150
1151 /* PrinterType */
1152 #define CUSTOM_NAME 0x00
1153 #define ALPHA_P_101 0x01
1154 #define BROTHER_15XL 0x02
1155 #define CER_MPS1000 0x03
1156 #define DIAB_630 0x04
1157 #define DIAB_ADV_D25 0x05
1158 #define DIAB_C_150 0x06
1159 #define EPSON 0x07
1160 #define EPSON_JX_80 0x08
1161 #define OKIMATE_20 0x09
1162 #define QUME_LP_20 0x0A
1163 /* new printer entries, 3 October 1985 */
1164 #define HP_LASERJET 0x0B
1165 #define HP_LASERJET_PLUS 0x0C
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176

```

```

1233 /* When you're defining IntuiText for the Positive and Negative Gadgets
1234 * created by a call to AutoRequest(), these defines will get you
1235 * reasonable-looking text. The only field without a define is the IText
1236 * field; you decide what text goes with the Gadget
1237 */
1238 #define AUTOFROMPEN 0
1239 #define AUTOBACKPEN 1
1240 #define AUTODRAMMODE JAMZ
1241 #define AUTOLEFTEDGE 6
1242 #define AUTOTOPEDGE 3
1243 #define AUTOITEXTFONT NULL
1244 #define AUTOWEXTTEXT NULL
1245
1246
1247 /* --- RAMMOUSE Codes and Qualifiers (Console OR IDCMP) ----- */
1248 #define SELECTUP (IECODE_LBUTTON | IECODE_UP_PREFIX)
1249 #define SELECTDOWN (IECODE_LBUTTON)
1250 #define MENUUP (IECODE_RBUTTON | IECODE_UP_PREFIX)
1251 #define MENUDOWN (IECODE_RBUTTON)
1252 #define ALTLEFT (IEQUALIFIER_LALT)
1253 #define ALTRIGHT (IEQUALIFIER_RALT)
1254 #define AMIGALEFT (IEQUALIFIER_LCOMMAND)
1255 #define AMIGARIGHT (IEQUALIFIER_RCOMMAND)
1256 #define AMIGAKEYS (AMIGALEFT | AMIGARIGHT)
1257
1258 #define CURSORUP 0x4C
1259 #define CURSORLEFT 0x4F
1260 #define CURSORRIGHT 0x4E
1261 #define CURSORDOWN 0x4D
1262 #define KEYCODE_Q 0x10
1263 #define KEYCODE_X 0x32
1264 #define KEYCODE_N 0x36
1265 #define KEYCODE_M 0x37
1266
1267
1268
1269 #endif

```

```

1 #ifndef INTUITION_INTUITIONBASE_H
2 #define INTUITION_INTUITIONBASE_H 1
3
4 /*** intuitionbase.h ****
5 * Commodore-Amiga, Inc.
6 *
7 * the IntuitionBase structure and supporting structures
8 *
9 * Modification History
10 * date : author : Comments
11 * -----
12 * 3-1-85 --RJ-- created this file!
13 *
14 *
15 *****
16 #ifndef EXEC_LIBRARIES_H
17 #include "exec/libraries.h"
18 #endif
19
20 #ifndef GRAPHICS_VIEW_H
21 #include "graphics/view.h"
22 #endif
23
24 /*
25 * Be sure to protect yourself against someone modifying these data as
26 * you look at them. This is done by calling:
27 *
28 * lock = LockIBase(0), which returns a ULONG. When done call
29 * UnlockIBase(lock) where lock is what LockIBase() returned.
30 *
31 * NOTE: these library functions are simply stubs now, but should be called
32 * to be compatible with future releases.
33 */
34
35 //===== IntuitionBase
36 //=====
37 //=====
38 struct IntuitionBase
39 {
40     struct Library LibNode;
41     struct View ViewLord;
42     struct Window 'ActiveWindow;
43     struct Screen 'ActiveScreen;
44
45     /* the FirstScreen variable points to the frontmost Screen. Screens are
46     * then maintained in a front to back order using Screen.NextScreen
47     */
48     struct Screen 'FirstScreen; /* for linked list of all screens */
49 };
50
51 #endif

```

```

1  /*
2  *
3  * This header file defines various ASCII character manipulation macros,
4  * as follows:
5  *
6  * isalpha(c) non-zero if c is alpha
7  * isupper(c) non-zero if c is upper case
8  * islower(c) non-zero if c is lower case
9  * isdigit(c) non-zero if c is a digit (0 to 9)
10 * isxdigit(c) non-zero if c is a hexadecimal digit (0 to 9, A to F,
11 * a to f)
12 * isspace(c) non-zero if c is white space
13 * ispunct(c) non-zero if c is punctuation
14 * isalnum(c) non-zero if c is alpha or digit
15 * isprint(c) non-zero if c is printable (including blank)
16 * isgraph(c) non-zero if c is graphic (excluding blank)
17 * iscntrl(c) non-zero if c is control character
18 * isascii(c) non-zero if c is ASCII
19 * iscsym(c) non-zero if valid character for C symbols
20 * iscsymf(c) non-zero if valid first character for C symbols
21 *
22 **/
23 #define U 1
24 #define L 2
25 #define N 4
26 #define S 8
27 #define P 16
28 #define C 32
29 #define B 64
30 #define X 128
31
32 extern char _ctype[]; /* character type table */
33
34 #define isalpha(c) (_ctype[(c)+1]&(_U|L))
35 #define isupper(c) (_ctype[(c)+1]&U)
36 #define islower(c) (_ctype[(c)+1]&L)
37 #define isdigit(c) (_ctype[(c)+1]&D)
38 #define isxdigit(c) (_ctype[(c)+1]&X)
39 #define isspace(c) (_ctype[(c)+1]&S)
40 #define ispunct(c) (_ctype[(c)+1]&P)
41 #define isalnum(c) (_ctype[(c)+1]&(_U|L|N))
42 #define isprint(c) (_ctype[(c)+1]&(_U|L|N|_B))
43 #define isgraph(c) (_ctype[(c)+1]&(_U|L|N))
44 #define iscntrl(c) ((unsigned)(c) <= 127)
45 #define isascii(c) ((c) <= 127)
46 #define iscsym(c) (((c) & 127) == 0x5f)
47 #define iscsymf(c) (((c) & 127) == 0x5f)
48 #define toupper(c) ((c) - ('a' - 'A'))
49 #define tolower(c) ((c) + ('a' - 'A'))
50 #define toascii(c) ((c) & 127)
51 #define toascii(c) ((c) & 127)
52

```

```

1  /*
2  *
3  * This file contains information used by the decimal arithmetic package.
4  *
5  * A floating decimal number is a byte array consisting of a two-byte
6  * header followed by a byte for each two digits. The header has the
7  * following format:
8  *
9  * Byte 0, bit 7: Set if negative number
10 * Byte 0, bits 0 to 6: Number of digit bytes (array length - 2)
11 * Byte 1: Decimal exponent (-128 to +127)
12 *
13 **/
14
15 #define D_DIG 8 /* Maximum number of digit bytes */
16 #define D_MAX (D_DIG+2) /* Maximum number of bytes */
17
18 extern char D0[], D1[], D2[]; /* Integer constants 0, 1, 2 */
19 extern char D5[], D05[], D005[]; /* Decimal constants 0.5, 0.05, 0.005 */
20 extern char PI[], PID2[], PIM2[]; /* Constants PI, PI/2, PI*2 */
21 extern char E[]; /* Constant E (base of natural logs) */
22 extern char M[]; /* Constant log10(E) */
23 extern char DPR[], RPD[]; /* Degrees per radian, radians per degree */
24 extern char SR10[]; /* Square root of 10 */
25 extern char X10[], Y10[], Z10[]; /* Work areas */
26 extern char X1[], Y1[], Z1[]; /* Work areas */
27
28 extern char FEEDIT; /* Set to include leading dollar sign */
29 extern char EDTYPE; /* Set if last cvfd input was exponential */
30 extern char FDFECP; /* decimal point character */
31 extern char FDMONY; /* money symbol */
32
33 extern char *cvfd(), *cvfdx(), *vcfd(), *vcfdi(), *vcfde(), *vcfdcd();
34
35

```

```

1 #ifndef LIBRARIES_DOS_H
2 #define LIBRARIES_DOS_H
3 /******
4 /* Commodore-Amiga, Inc.
5 /* dos.h
6 /* Standard C header for AmigaDOS on the MCS8000
7 /******
8
9 #ifndef EXEC_TYPES_H
10 #include "exec/types.h"
11 #endif
12
13 #define DOSNAME "dos.library"
14
15 /* Predefined Amiga DOS global constants */
16
17 /* Mode parameter to Open() */ 1005 /* Open existing file read/write
18 #define MODE_OLDFILE 1006 /* Open freshly created file (delete
19 #define MODE_NEWFILE 1006 /* old file) read/write
20
21
22
23 /* Relative position to Seek() */
24 #define OFFSET_BEGINNING -1 /* relative to Beginning of File */
25 #define OFFSET_CURRENT 0 /* relative to Current file position */
26 #define OFFSET_END 1 /* relative to End Of File */
27
28 #define OFFSET_BEGINNING OFFSET_BEGINNING /* ancient compatibility */
29
30 #define BITS_PER_BYTE 8
31 #define BYTES_PER_LONG 4
32 #define BITS_PER_LONG 32
33 #define MAXINT 0x7FFFFFFF
34 #define MININT 0x80000000
35
36 /* Passed as type to Lock() */
37 #define SHARED_LOCK -2 /* File is readable by others */
38 #define ACCESS_READ -2 /* Synonym */
39 #define EXCLUSIVE_LOCK -1 /* No other access allowed */
40 #define ACCESS_WRITE -1 /* Synonym */
41
42 struct DateStamp {
43     LONG ds_Days; /* Number of days since Jan. 1, 1978 */
44     LONG ds_Minute; /* Number of minutes past midnight */
45     LONG ds_Tick; /* Number of ticks past minute */
46 }; /* DateStamp */
47 #define TICKS_PER_SECOND 50 /* Number of ticks in one second */
48
49 /* Returned by Examine() and ExInfo(), must be on a 4 byte boundary */
50 struct FileInfoBlock {
51     LONG fib_DiskKey;
52     LONG fib_DirEntryType; /* Type of Directory. If < 0, then a plain file.
53     /* If > 0 a directory */
54     char fib_FileName[108]; /* Null terminated. Max 30 chars used for now */
55     LONG fib_Protection; /* bit mask of protection, rwx are 3-0.
56     LONG fib_EntryType;

```

```

57     LONG fib_Size; /* Number of bytes in file */
58     LONG fib_NumBlocks; /* Number of blocks in file */
59     struct DateStamp fib_Date; /* Date file last changed */
60     char fib_Comment[116]; /* Null terminated.
61     /* Comment associated with file */
62 }; /* FileInfoBlock */
63
64 /* FIB stands for FileInfoBlock */
65 /* FIBB are bit definitions, FIBF are field definitions */
66 #define FIBB_READ 3
67 #define FIBB_WRITE 2
68 #define FIBB_EXECUTE 1
69 #define FIBB_DELETE 0
70 #define FIBF_READ (1<<FIBB_READ)
71 #define FIBF_WRITE (1<<FIBB_WRITE)
72 #define FIBF_EXECUTE (1<<FIBB_EXECUTE)
73 #define FIBF_DELETE (1<<FIBB_DELETE)
74
75
76 /* All BCPL data must be long word aligned. BCPL pointers are the long word
77 * address (i.e. byte address divided by 4 (>>2)) */
78 typedef long BPTR; /* Long word pointer */
79 typedef long BSTR; /* Long word pointer to BCPL string
80 #define BADDR ( bptr ) /* Convert BPTR to typical C pointer
81 /* BCPL strings have a length in the first byte and then the characters.
82 * For example: s[0]=3 s[1]=S s[2]=Y s[3]=S
83
84 /* returned by Info(), must be on a 4 byte boundary */
85 struct InfoData {
86     LONG id_NumSoftErrors; /* number of soft errors on disk */
87     LONG id_UnitNumber; /* Which unit disk is (was) mounted on */
88     LONG id_DiskState; /* See defines below */
89     LONG id_NumBlocks; /* Number of blocks below */
90     LONG id_NumBlocksUsed; /* Number of blocks on disk */
91     LONG id_BytesPerBlock; /* Number of block in use */
92     LONG id_DiskType; /* Disk Type code */
93     BPTR id_VolumeNode; /* BCPL pointer to volume node */
94     LONG id_InUse; /* Flag, zero if not in use */
95 }; /* InfoData */
96
97 /* ID stands for InfoData */
98 /* Disk states */
99 #define ID_WRITE_PROTECTED 80 /* Disk is write protected */
100 #define ID_VALIDATING 81 /* Disk is currently being validated */
101 #define ID_VALIDATED 82 /* Disk is consistent and writeable */
102
103 /* Disk types */
104 #define ID_NO_DISK_PRESENT (-1)
105 #define ID_UNREADABLE_DISK (('B'<<24) | ('A'<<16) | ('D'<<8))
106 #define ID_DOS_DISK (('D'<<24) | ('O'<<16) | ('S'<<8))
107 #define ID_NOT_REALY_DOS (('N'<<24) | ('D'<<16) | ('O'<<8) | ('S'))
108 #define ID_KICKSTART_DISK (('K'<<24) | ('I'<<16) | ('C'<<8) | ('K'))
109
110 /* Errors from IoErr(), etc. */
111 #define ERROR_NO_FREE_STORE 103
112 #define ERROR_NO_DEFAULT_DIR 201

```

```

113 #define ERROR_OBJECT_IN_USE
114 #define ERROR_OBJECT_EXISTS
115 #define ERROR_DIR_NOT_FOUND
116 #define ERROR_OBJECT_NOT_FOUND
117 #define ERROR_BAD_STREAM_NAME
118 #define ERROR_OBJECT_TOO_LARGE
119 #define ERROR_ACTION_NOT_KNOWN
120 #define ERROR_INVALID_COMPONENT_NAME
121 #define ERROR_INVALID_LOCK
122 #define ERROR_OBJECT_WRONG_TYPE
123 #define ERROR_DISK_NOT_VALIDATED
124 #define ERROR_DISK_WRITE_PROTECTED
125 #define ERROR_RENAME_ACROSS_DEVICES
126 #define ERROR_DIRECTORY_NOT_EMPTY
127 #define ERROR_TOO_MANY_LEVELS
128 #define ERROR_DEVICE_NOT_MOUNTED
129 #define ERROR_SEEK_ERROR
130 #define ERROR_COMMENT_TOO_BIG
131 #define ERROR_DISK_FULL
132 #define ERROR_DELETE_PROTECTED
133 #define ERROR_WRITE_PROTECTED
134 #define ERROR_READ_PROTECTED
135 #define ERROR_NOT_A_DOS_DISK
136 #define ERROR_NO_DISK
137 #define ERROR_NO_MORE_ENTRIES
138
139 /* These are the return codes used by convention by AmigaDOS commands */
140 /* See FAILAT and IF for relevance to EXECUTE files */
141 #define RETURN_OK 0 /* No problems, success */
142 #define RETURN_WARN 5 /* A warning only */
143 #define RETURN_ERROR 10 /* Something wrong */
144 #define RETURN_FAIL 20 /* Complete or severe failure */
145
146 /* Bit numbers that signal you that a user has issued a break */
147 #define SIGBREAK_CTRL_C 12
148 #define SIGBREAK_CTRL_D 13
149 #define SIGBREAK_CTRL_E 14
150 #define SIGBREAK_CTRL_F 15
151
152 /* Bit fields that signal you that a user has issued a break */
153 /* for example: if (SetSignal(0,0) & BREAK_CTRL_CF) cleanup_and_exit(); */
154 #define SIGBREAK_CTRL_C (1<<SIGBREAK_CTRL_C)
155 #define SIGBREAK_CTRL_D (1<<SIGBREAK_CTRL_D)
156 #define SIGBREAK_CTRL_E (1<<SIGBREAK_CTRL_E)
157 #define SIGBREAK_CTRL_F (1<<SIGBREAK_CTRL_F)
158
159 #endif LIBRARIES_DOS_H

```

```

1 /**
2 *
3 * The file "/include/libraries/dos.h" contains all the error messages.
4 * Do not use this file.
5 *
6 */
7 #include "include/libraries/dos.h"

```

Dec 8 16:38 1985 lattice/fcntl.h Page 1

```
1 /**
2 *
3 * The following symbols are used for the "open" and "creat" functions.
4 *
5 **/
6 #define O_RDONLY 0 /* Read-only value (right byte of mode word) */
7 #define O_WRONLY 1 /* Write-only value */
8 #define O_RDWR 2 /* Read-write value */
9
10 #define O_NDELAY 4 /* Non-blocking I/O flag */
11 #define O_APPEND 8 /* Append mode flag */
12 #define O_CREAT 0x0100 /* File creation flag */
13 #define O_TRUNC 0x200 /* File truncation flag */
14 #define O_EXCL 0x400 /* Exclusive access flag */
15
16 #define O_RAW 0x8000 /* Raw I/O flag (Lattice feature) */
17 /**
18 *
19 * The following symbols are used for the "fcntl" function.
20 *
21 *
22 **/
23 #define F_DUPED 0 /* Duplicate file descriptor */
24 #define F_GETFD 1 /* Get file descriptor flags */
25 #define F_SETFD 2 /* Set file descriptor flags */
26 #define F_GETFL 3 /* Get file flags */
27 #define F_SETFL 4 /* Set file flags */
```

Dec 8 16:38 1985 lattice/ios1.h Page 1

```
1 /**
2 *
3 * The following structure is a UNIX file block that retains information
4 * a file being accessed via the level 1 I/O functions.
5 */
6 struct UFB
7 {
8     char ufbflg; /* flags */
9     char ufbtyp; /* file handle */
10     int ufbfh; /* file handle */
11 };
12 #define NUFB 20 /* number of UFBs defined */
13
14 /*
15 * UFB.ufbflg definitions
16 *
17 */
18 #define UFB_OP 0x80 /* file is open */
19 #define UFB_RA 0x40 /* reading is allowed */
20 #define UFB_WA 0x20 /* writing is allowed */
21 #define UFB_NT 0x10 /* access file with no translation */
22 #define UFB_AP 8 /* append mode flag */
23 #define UFB_NC 4 /* no-close flag */
24
25 /*
26 *
27 *
28 * UFB.ufbtyp definitions
29 *
30 */
31 #if MSDOS1
32 #define D_DISK 0
33 #define D_CON 1
34 #define D_PRN 2
35 #define D_AUX 3
36 #define D_NULL 4
37 #endif
```

```
1 #define HUCE_VAL 1.797693E+308
```

```
1 /**
2 *
3 * Redefine secondary simulation function names to become primary names
4 * for systems without a Numeric Data Processor.
5 *
6 */
7 #ifndef MONDP
8 #define _acos acos
9 #define _asin asin
10 #define _atan atan
11 #define _cos cos
12 #define _cosh cosh
13 #define _cot cot
14 #define _exp exp
15 #define _fabs fabs
16 #define _ldexp ldexp
17 #define _log log
18 #define _log10 log10
19 #define _modf modf
20 #define _pow pow
21 #define _pow2 pow2
22 #define _sin sin
23 #define _sinh sinh
24 #define _sqrt sqrt
25 #define _tan tan
26 #define _tanh tanh
27 #endif
28
29 /**
30 *
31 * Structure to hold information about math exceptions
32 *
33 */
34 struct exception
35 {
36     int type;
37     char *name;
38     double arg1, arg2;
39     double retval;
40 };
41
42 /**
43 *
44 * Exception type codes, found in exception.type
45 *
46 */
47 #define DOMAIN 1 /* domain error */
48 #define SING 2 /* singularity */
49 #define OVERFLOW 3 /* overflow */
50 #define UNDERFLOW 4 /* underflow */
51 #define TLOSS 5 /* total loss of significance */
52 #define PLOSS 6 /* partial loss of significance */
53
54 /**
55 *
56 * Error codes generated by basic arithmetic operations (+ - * /)
```

```

57 *
58 */
59 #define EPEUND 1 /* underflow */
60 #define EPEOVF 2 /* overflow */
61 #define EPEZDV 3 /* zero divisor */
62 #define EPEANAN 4 /* not a number (Invalid operation) */
63
64 /**
65 * Constants
66 *
67 *
68 */
69 #define PI 3.14159265358979323846
70 #define PID2 1.57079632679489661923 /* PI divided by 2 */
71 #define PID4 0.78539816339744830962 /* PI divided by 4 */
72 #define I_PI 0.31830988618379067154 /* Inverse of PI */
73 #define I_PID2 0.63661977236758134308 /* Inverse of PID2 */
74
75 #define HUGE 1.797693e308 /* huge value */
76 #define TINY 2.2e-308 /* tiny value */
77 #define LOCHUGE 709.778 /* natural log of huge value */
78 #define LOCTINY -708.396 /* natural log of tiny value */
79
80 /**
81 * External declarations
82 *
83 *
84 */
85 extern int _fperr; /* floating point arithmetic error */
86 extern int errno; /* UNIX error code */
87
88 extern char *ecvt();
89 extern short *seed48();
90 extern int atoi(), matherr();
91 extern long atol(), strtol(), lrand48(), nrand48(), mrand48(), jrand48();
92 extern double atof(), exp(), log(), log10(), pow(), sqrt();
93 extern double floor(), ceil(), fmod(), fabs(), frexp(), ldexp(), modf();
94 extern double sinh(), cosh(), tanh(), sin(), cos(), tan(), cot(), asin(), acos();
95 extern double atan(), atan2(), except();
96 extern double drand48(), erand48();

```

```

1 /**
2 *
3 * This header file defines the information used by the standard I/O
4 * package.
5 *
6 **/
7 #define _BUFSIZ 512 /* standard buffer size */
8 #define BUFSIZ 512 /* standard buffer size */
9 #define _NFILE 20 /* maximum number of files */
10
11 struct _lobuf
12 {
13     unsigned char *_ptr; /* current buffer pointer */
14     int _rcnt; /* current byte count for reading */
15     int _wcnt; /* current byte count for writing */
16     unsigned char *_base; /* base address of I/O buffer */
17     char _flag; /* control flags */
18     char _file; /* file number */
19     int _size; /* size of buffer */
20     unsigned char *_cbuf; /* single char buffer */
21     char _pad; /* (pad to even number of bytes) */
22 };
23
24 extern struct _lobuf _lob[_NFILE];
25
26 #define _IOREAD 1 /* read flag */
27 #define _IOWRT 2 /* write flag */
28 #define _IONBF 4 /* non-buffered flag */
29 #define _IOMBUF 8 /* private buffer flag */
30 #define _IOEOF 16 /* end-of-file flag */
31 #define _IOERR 32 /* error flag */
32 #define _IOSTRG 64 /* read-write (update) flag */
33 #define _IORN 128
34
35 #ifndef NULL
36 #if SPTX
37 #define NULL 0 /* null pointer value */
38 #else
39 #define NULL 0L
40 #endif
41 #endif
42 #define FILE struct _lobuf /* shorthand */
43 #define EOF (-1) /* end-of-file code */
44
45 #define stdin (&_lob[0]) /* standard input file pointer */
46 #define stdout (&_lob[1]) /* standard output file pointer */
47 #define stderr (&_lob[2]) /* standard error file pointer */
48
49 #definegetc(p) (--(p)->_rcnt>=0? *(p)->_ptr++:_filbf(p))
50 #definegetchar() getc(stdin)
51 #defineputc(c,p) (--(p)->_wcnt>=0? ((int) (*p)->_ptr++=(c)):_filbf(c))
52 #defineputchar(c) putc(c,stdout)
53 #definefeof(p) (((p)->_flag&_IOEOF)!=0)
54 #defineferror(p) (((p)->_flag&_IOERR)!=0)
55 #definefileno(p) ((p)->_file)
56 #definerwld(fp) fseek(fp,0L,0)

```



```

57 #define fflush(fp) _fllsbf(-1, fp)
58 #define clearerr(fp) clrerr(fp)
59
60 FILE *fopen();
61 FILE *freopen();
62 long ftell();
63 char *fgets();
64
65 #define abs(x) ((x) < 0 ? -(x) : (x))
66 #define max(a,b) ((a) > (b) ? (a) : (b))
67 #define min(a,b) ((a) <= (b) ? (a) : (b))
68

```

```

1 #ifndef LIBRARIES_DISKFONT_H
2 #define LIBRARIES_DISKFONT_H
3 /****** Commodore-Amiga, Inc. *****/
4 /*
5 /* diskfont.h
6 /******
7 /******
8 *
9 * diskfont library definitions
10 *
11 *
12 *
13 #ifndef EXEC_NODES_H
14 #include "exec/nodes.h"
15 #endif
16 #ifndef EXEC_LISTS_H
17 #include "exec/lists.h"
18 #endif
19 #ifndef GRAPHICS_TEXT_H
20 #include "graphics/text.h"
21 #endif
22
23 #define MAXFONTPATH 256 /* including null terminator */
24
25 struct FontContents {
26     char fc_FileName[MAXFONTPATH];
27     UMWORD fc_Size;
28     UBYTE fc_Style;
29     UBYTE fc_Flags;
30 };
31
32 #define FCH_ID 0x0f00
33
34 struct FontContentsHeader {
35     UMWORD fch_FileID; /* FCH_ID */
36     UMWORD fch_NumEntries; /* the number of FontContents elements */
37     /* struct FontContents fch_FC[]; */
38 };
39
40 #define DEFH_ID 0x0f80
41 #define MAXFONTNAME 32 /* font name including ".font\0" */
42
43 struct DiskFontHeader {
44     /* the following 8 bytes are not actually considered a part of the
45     /* DiskFontHeader, but immediately preceded it. The NextSegment is
46     /* supplied by the linker/loader, and the ReturnCode is the code
47     /* at the beginning of the font in case someone runs it...
48     /* ULONG dfh_NextSegment; /* actually a BPTR */
49     /* ULONG dfh_ReturnCode; /* MOVEQ #0, D0 : RTS */
50     /* here then is the official start of the DiskFontHeader...
51     struct Node dfh_DF; /* node to link disk fonts */
52     UMWORD dfh_FileID; /* DFH_ID */
53     UMWORD dfh_Revision; /* the font revision */
54     LONG dfh_Segment; /* the font name (null terminated) */
55     char dfh_Name[MAXFONTNAME]; /* loaded TextFont structure */
56     struct TextFont dfh_TF; /* loaded TextFont structure */

```

```

57 };
58
59 #define AFB_MEMORY 0
60 #define AFE_MEMORY 1
61 #define AFB_DISK 1
62 #define AFE_DISK 2
63 #define AFB_DISK 2
64
65 struct AvailFonts { /* MEMORY or DISK */
66   UWORD af_Type;
67   struct TextAttr af_Atr; /* text attributes for font */
68 };
69
70 struct AvailFontsHeader { /* number of AvailFonts elements */
71   UWORD afh_NumEntries;
72   /* struct AvailFonts afh_AF[]; */
73 };
74 #endif
75

```

```

#define LIBRARIES_DOS_H
#define LIBRARIES_DOS_H
/* ***** Commodore-Amiga, Inc. ***** */
/* dos.h */
/* Standard C header for AmigaDOS on the MC68000 */
/* ***** */

#ifdef EXEC_TYPES_H
#include "exec/types.h"
#endif

#define DOSNAME "dos.library"

/* Predefined Amiga DOS global constants */

/* Mode parameter to Open() */
#define MODE_OLDFILE 1005 /* Open existing file read/write
                          * positioned at beginning of file. */
#define MODE_NEWFILE 1006 /* Open freshly created file (delete
                          * old file) read/write */

/* Relative position to Seek() */
#define OFFSET_BEGINNING -1 /* relative to Beginning Of File */
#define OFFSET_CURRENT 0 /* relative to Current file position */
#define OFFSET_END 1 /* relative to End Of File */

#define OFFSET_BEGINNING OFFSET_BEGINNING /* ancient compatibility */

#define BITSPERBYTE 8
#define BYTESPERLONG 4
#define BITSPERLONG 32
#define MAXINT 0x7FFFFFFF
#define MININT 0x80000000

/* Passed as type to Lock() */
#define SHARED_LOCK -2 /* File is readable by others */
#define ACCESS_READ -1 /* Synonym */
#define EXCLUSIVE_LOCK -1 /* No other access allowed */
#define ACCESS_WRITE -1 /* Synonym */

struct DateStamp {
  LONG ds_Days; /* Number of days since Jan. 1, 1978 */
  LONG ds_Minute; /* Number of minutes past midnight */
  LONG ds_Tick; /* Number of ticks past minute */
}; /* DateStamp */

#define TICKS_PER_SECOND 50 /* Number of ticks in one second */

/* Returned by Examine() and ExInfo(), must be on a 4 byte boundary */
struct FileInfoBlock {
  LONG fib_DiskKey;
  LONG fib_DirEntryType; /* Type of Directory. If < 0, then a plain file.
                          * If > 0 a directory */
  char fib_FileName[108]; /* Null terminated. Max 30 chars used for now */
  LONG fib_Protection; /* bit mask of protection, rwxrd are 3-0. */
  LONG fib_EntryType;
};

```

```

LONG fib_Size; /* Number of bytes in file */
LONG fib_NumBlocks; /* Number of blocks in file */
struct DateStamp fib_Date; /* Date file last changed */
char fib_Comment[116]; /* Null terminated.
*/; /* FileInfoBlock */

/* FIB stands for FileInfoBlock */
/* FIBB are bit definitions, FIBF are field definitions */
#define FIBB_READ 3
#define FIBB_WRITE 2
#define FIBB_EXECUTE 1
#define FIBB_DELETE 0
#define FIBB_READ (1<<FIBB_READ)
#define FIBB_WRITE (1<<FIBB_WRITE)
#define FIBB_EXECUTE (1<<FIBB_EXECUTE)
#define FIBB_DELETE (1<<FIBB_DELETE)

/* All BCPL data must be long word aligned. BCPL pointers are the long word
* address (i.e byte address divided by 4 (>>2)) */
typedef long BPTR; /* Long word pointer */
#define BADOR (bptr << 2) /* Convert BPTR to typical C pointer */
/* BCPL strings have a length in the first byte and then the characters.
* For example: s[0]=3 s[1]=S s[2]=Y s[3]=S
*/ returned by Info(), must be on a 4 byte boundary */
struct InfoData {
LONG id_NumSoftErrors; /* number of soft errors on disk */
LONG id_UnitNumber; /* Which unit disk is (was) mounted on */
LONG id_DiskState; /* See defines below */
LONG id_NumBlocks; /* Number of blocks on disk */
LONG id_BytesPerBlock; /* Number of blocks in use */
LONG id_DiskType; /* Disk type code */
BPTR id_VolumeNode; /* BCPL pointer to volume node */
LONG id_InUse; /* Flag, zero if not in use */
}; /* InfoData */

/* ID stands for InfoData */
/* Disk states */
#define ID_WRITE_PROTECTED 80 /* Disk is write protected */
#define ID_VALIDATING 81 /* Disk is currently being validated */
#define ID_INVALIDATED 82 /* Disk is consistent and writeable */

/* Disk types */
#define ID_NO_DISK_PRESENT (-1) ('B'<<24) | ('A'<<16) | ('D'<<8)
#define ID_UNREADABLE_DISK ('D'<<24) | ('O'<<16) | ('S'<<8)
#define ID_DOS_DISK ('N'<<24) | ('I'<<16) | ('O'<<8) | ('S')
#define ID_NOT_REALLY_DOS ('K'<<24) | ('I'<<16) | ('C'<<8) | ('K')
#define ID_KICKSTART_DISK

/* Errors from IoErr() etc. */
#define ERROR_NO_FREE_STORE 103
#define ERROR_NO_DEFAULT_DIR 201

```

```

#define ERROR_OBJECT_IN_USE 202
#define ERROR_OBJECT_EXISTS 203
#define ERROR_DIR_NOT_FOUND 204
#define ERROR_OBJECT_NOT_FOUND 205
#define ERROR_BAD_STREAM_NAME 206
#define ERROR_OBJECT_TOO_LARGE 207
#define ERROR_ACTION_NOT_KNOWN 209
#define ERROR_INVALID_COMPONENT_NAME 210
#define ERROR_INVALID_LOCK 211
#define ERROR_OBJECT_WRONG_TYPE 212
#define ERROR_DISK_NOT_VALIDATED 213
#define ERROR_DISK_WRITE_PROTECTED 214
#define ERROR_RENAME_ACROSS_DEVICES 215
#define ERROR_DIRECTORY_NOT_EMPTY 216
#define ERROR_TOO_MANY_LEVELS 217
#define ERROR_DEVICE_NOT_MOUNTED 218
#define ERROR_SEEK_ERROR 219
#define ERROR_COMMENT_TOO_BIG 220
#define ERROR_DISK_FULL 221
#define ERROR_DELETE_PROTECTED 222
#define ERROR_WRITE_PROTECTED 223
#define ERROR_READ_PROTECTED 224
#define ERROR_NOT_A_DOS_DISK 225
#define ERROR_NO_DISK 226
#define ERROR_NO_MORE_ENTRIES 232

/* These are the return codes used by convention by AmigaDOS commands */
/* See FAILAT and IF for relevance to EXECUTE files */
#define RETURN_OK 0 /* No problems, success */
#define RETURN_WARN 5 /* A warning only */
#define RETURN_ERROR 10 /* Something wrong */
#define RETURN_FAIL 20 /* Complete or severe failure */

/* Bit numbers that signal you that a user has issued a break */
#define SIGBREAKB_CTRL_C 12
#define SIGBREAKB_CTRL_D 13
#define SIGBREAKB_CTRL_E 14
#define SIGBREAKB_CTRL_F 15

/* Bit fields that signal you that a user has issued a break */
/* for example: if (SetSignal(0,0) & BREAK_CTRL_CF) cleanup_and_exit(); */
#define SIGBREAKB_CTRL_C (1<<SIGBREAKB_CTRL_C)
#define SIGBREAKB_CTRL_D (1<<SIGBREAKB_CTRL_D)
#define SIGBREAKB_CTRL_E (1<<SIGBREAKB_CTRL_E)
#define SIGBREAKB_CTRL_F (1<<SIGBREAKB_CTRL_F)

#endif LIBRARIES_DOS_H

```

```

1  #ifndef LIBRARIES_DOSEXTENS_H
2  #define LIBRARIES_DOSEXTENS_H 1
3  /****** Commodore-Amiga, Inc. *****/
4  /*
5  /* dosextens.h
6  /*
7  /* DOS structures not needed for the casual DOS user */
8
9
10 #ifndef EXEC_TYPES_H
11 #include "exec/types.h"
12 #endif
13 #ifndef EXEC_TASKS_H
14 #include "exec/tasks.h"
15 #endif
16 #ifndef EXEC_PORTS_H
17 #include "exec/ports.h"
18 #endif
19 #ifndef EXEC_LIBRARIES_H
20 #include "exec/libraries.h"
21 #endif
22
23 #ifndef LIBRARIES_DOS_H
24 #include "libraries/dos.h"
25 #endif
26
27 /* All DOS processes have this structure */
28 /* Create and Device Proc returns pointer to the MsgPort in this structure */
29 /* dev_proc = (struct Process *) (DeviceProc(..) - sizeof(struct Task)); */
30
31 struct Process {
32     struct Task pr_Task;
33     struct MsgPort pr_MsgPort; /* This is EPTer address from DOS functions */
34     WORD pr_Pad; /* Remaining variables on 4 byte boundaries */
35     EPTer pr_SegList; /* Array of seg lists used by this process */
36     LONG pr_StackSize; /* Size of process stack in bytes */
37     APTer pr_GlobVec; /* Global vector for this process (BCPL) */
38     LONG pr_TaskNum; /* CLI task number of zero if not a CLI */
39     EPTer pr_StackBase; /* Ptr to high memory end of process stack */
40     LONG pr_Result2; /* Value of secondary result from last call */
41     EPTer pr_CurrentDir; /* Lock associated with current directory */
42     EPTer pr_CIS; /* Current CLI Input Stream */
43     APTer pr_COS; /* Current CLI Output Stream */
44     APTer pr_ConsoleTask; /* Console handler process for the
45     * current window*/
46     APTer pr_FileSystemTask; /* File handler process for current drive
47     * pointer to ConsoleInterpreter
48     * pr_ReturnAddr; pointer to previous stack frame
49     * APTer pr_PktWait; Function to be called when awaiting msg
50     * APTer pr_WindowPtr; /* Window for error printing */
51     }; /* Process */
52
53 /* The long word address (EPTer) of this structure is returned by
54 * Open() and other routines that return a file. You need only worry
55 * about this struct to do async io's via PutMsg() instead of
56 * standard file system calls */

```

```

57 struct FileHandle {
58     struct Message *fh_Link; /* EXEC message */
59     struct MsgPort *fh_Port; /* Reply port for the packet */
60     struct MsgPort *fh_Type; /* Port to do PutMsg() to
61     * Address is negative if a plain file */
62
63     LONG fh_Buf;
64     LONG fh_Pos;
65     LONG fh_End;
66     LONG fh_Funcs;
67     #define fh_Func1 fh_Funcs
68     LONG fh_Func2;
69     LONG fh_Func3;
70     LONG fh_Args;
71     #define fh_Arg1 fh_Args
72     LONG fh_Arg2;
73     }; /* FileHandle */
74
75 /* This is the extension to EXEC Messages used by DOS */
76
77 struct DosPacket {
78     struct Message *dp_Link; /* EXEC message */
79     struct MsgPort *dp_Port; /* Reply port for the packet */
80     LONG dp_Type; /* Must be filled in each send. */
81     /* See ACTION_... below and
82     * 'R' means Read, 'W' means Write to the
83     * file system */
84     LONG dp_Res1; /* For file system calls this is the result
85     * that would have been returned by the
86     * function, e.g. Write ('W') returns actua
87     * length written */
88     LONG dp_Res2; /* For file system calls this is what would
89     * have been returned by IoErr() */
90     /* Device packets common equivalents */
91     #define dp_Action dp_Type
92     #define dp_Status dp_Res1
93     #define dp_Status2 dp_Res2
94     #define dp_BufAddr dp_Arg1
95     LONG dp_Arg1;
96     LONG dp_Arg2;
97     LONG dp_Arg3;
98     LONG dp_Arg4;
99     LONG dp_Arg5;
100    LONG dp_Arg6;
101    LONG dp_Arg7;
102    }; /* DosPacket */
103
104 /* A Packet does not require the Message to be before it in memory, but
105 * for convenience it is useful to associate the two.
106 * Also see the function init_std_pkt for initializing this structure */
107
108 struct StandardPacket {
109     struct Message sp_Msg;
110     struct DosPacket sp_Pkt;
111     }; /* StandardPacket */
112

```

```

113 /* Packet types */
114 #define ACTION_NIL 0
115 #define ACTION_GET_BLOCK 2
116 #define ACTION_SET_MAP 4
117 #define ACTION_DIE 5
118 #define ACTION_EVENT 6
119 #define ACTION_CURRENT_VOLUME 7
120 #define ACTION_LOCATE_OBJECT 8
121 #define ACTION_RENAME_DISK 9
122 #define ACTION_WRITE 'w'
123 #define ACTION_READ 'r'
124 #define ACTION_FREE_LOCK 15
125 #define ACTION_DELETE_OBJECT 16
126 #define ACTION_RENAME_OBJECT 17
127
128 #define ACTION_COPY_DIR 19
129 #define ACTION_WAIT_CHAR 20
130 #define ACTION_SET_PROTECT 21
131 #define ACTION_CREATE_DIR 22
132 #define ACTION_EXAMINE_OBJECT 23
133 #define ACTION_EXAMINE_NEXT 24
134 #define ACTION_DISK_INFO 25
135 #define ACTION_INFO 26
136
137 #define ACTION_SET_COMMENT 28
138 #define ACTION_PARENT 29
139 #define ACTION_TIMER 30
140 #define ACTION_INHIBIT 31
141 #define ACTION_DISK_TYPE 32
142 #define ACTION_DISK_CHANGE 33
143
144 /* DOS library node structure.
145 * This is the data at positive offsets from the library node.
146 * Negative offsets from the node is the jump table to DOS functions
147 * node = (struct DosLibrary *) OpenLibrary( "dos.library" .. ) */
148
149 struct DosLibrary {
150     struct Library dl_lib; /* Pointer to RootNode, described below */
151     APTR dl_Root; /* Pointer to BCPL global vector */
152     APTR dl_GV; /* Private register dump of DOS */
153     LONG dl_A2;
154     LONG dl_A5;
155     LONG dl_A6; /* DosLibrary */
156 }; /*
157
158
159
160 struct RootNode {
161     BPTR rn_TaskArray; /* [0] is max number of CLI's
162     /* [1] is APTR to process id of CLI n
163     /* [n] is APTR to process id of CLI n */
164     BPTR rn_ConsoleSegment; /* SegList for the CLI
165     struct DateStamp rn_Flme; /* Current time
166     LONG rn_RestartSeg; /* SegList for the disk validator process
167     BPTR rn_Info; /* Pointer of the Info structure
168 }; /* RootNode */

```

```

169 struct DosInfo {
170     BPTR dl_McName; /* Network name of this machine; currently 0
171     BPTR dl_DevInfo; /* Device List
172     BPTR dl_Devices; /* Currently zero
173     BPTR dl_Handlers; /* Currently zero
174     APTR dl_NetHand; /* Network handler processid; currently zero
175     /* DosInfo */
176 };
177
178 /* DOS Processes started from the CLI via RUN or NEMCLI have this additional
179 * set to data associated with them */
180
181 struct CommandLineInterface {
182     LONG cli_Result2; /* Value of IoErr from last command
183     BSTR cli_SetName; /* Name of current directory
184     BPTR cli_CommandDir; /* Lock associated with command directory
185     LONG cli_ReturnCode; /* Return code from last command
186     BSTR cli_CommandName; /* Name of current command
187     LONG cli_FailLevel; /* Fail level (set by FAILAT)
188     BSTR cli_Prompt; /* Current prompt (set by PROMPT)
189     BPTR cli_StandardInput; /* Default (terminal) CLI input
190     BPTR cli_CurrentInput; /* Current CLI input
191     BSTR cli_CommandFile; /* Name of EXECUTE command file
192     LONG cli_Interactive; /* Boolean; True if prompts required
193     LONG cli_Background; /* Boolean; True if CLI created by RUN
194     BPTR cli_CurrentOutput; /* Current CLI output
195     LONG cli_DefaultStack; /* Stack size to be obtained in long words
196     BPTR cli_StandardOutput; /* Default (terminal) CLI output
197     BPTR cli_Module; /* SegList of currently loaded command
198     /* CommandLineInterface */
199 };
200
201 /* this structure needs some work. It should really be a union, because
202 * it can take on different values depending on whether it is a device,
203 * an assigned directory, or a volume.
204 * For now, it reflects a volume.
205 */
206 struct DeviceList {
207     BPTR dl_Next; /* bptr to next device list */
208     LONG dl_Type; /* see DLT below */
209     struct MsgPort * dl_Task; /* ptr to handler task */
210     BPTR dl_Lock; /* not for volumes */
211     struct DateStamp dl_VolumeDate; /* creation date */
212     BPTR dl_LockList; /* outstanding locks */
213     LONG dl_DiskType; /* 'DOS', etc */
214     LONG dl_Unused;
215     BSTR * dl_Name; /* bptr to bopl name */
216 };
217
218 /* definitions for dl_Type */
219 #define DLT_DEVICE 0
220 #define DLT_DIRECTORY 1
221 #define DLT_VOLUME 2
222
223
224 /* a lock structure, as returned by Lock() or DupLock() */

```

```

225 struct FileLock {
226     BPTR fl_Link;
227     LONG fl_Key;
228     LONG fl_Access;
229     struct MsgPort * fl_Task;
230     BPTR fl_Volume;
231 };
232
233 #endif LIBRARIES_DOSEXTENS_H

```

```

/* bop1 pointer to next lock */
/* disk block number */
/* exclusive or shared */
/* handler task's port */
/* bptr to a DeviceList */

```

```

1 #ifndef LIBRARIES_MATHFP_H
2 #define LIBRARIES_MATHFP_H
3 /*****
4 ** Commodore-Amiga, Inc.
5 ** mathfp.h
6 **
7 **
8 **
9 ** * general floating point declarations
10 */
11 #define PI ((FLOAT) 3.1415192653857)
12 #define TWO_PI ((FLOAT) 2) * PI
13 #define PI2 (PI / ((FLOAT) 2))
14 #define PI4 (PI / ((FLOAT) 4))
15 #define E ((FLOAT) 2.7182818284590453)
16 #define LOG10 ((FLOAT) 2.3025850929940456)
17
18 #define EPTEN ((FLOAT) 10.0)
19 #define EPONE ((FLOAT) 1.0)
20 #define EPHALF ((FLOAT) 0.5)
21 #define EPZERO ((FLOAT) 0.0)
22
23 #define trunc(x) ((int) (x))
24 #define round(x) ((int) ((x) + 0.5))
25 #define itof(i) ((FLOAT) (i))
26
27 int SFFix();
28 FLOAT SFFit();
29 int SFCmp();
30 int SFTst();
31 FLOAT SPAbs();
32 FLOAT SPNeg();
33 FLOAT SPAdd();
34 FLOAT SPSub();
35 FLOAT SPMin();
36 FLOAT SPDiv();
37
38 FLOAT SPAsin();
39 FLOAT SPCos();
40 FLOAT SPSinh();
41 FLOAT SPExp();
42 FLOAT SPSqrt();
43
44 FLOAT afp(), dbf();
45
46 #endif LIBRARIES_MATHFP_H

```

/* Basic math functions */

/* Transcendental math functions */

/* Math conversion functions */

Dec 8 16:38 1985 libraries/translator.h Page 1

```
1 #ifndef LIBRARIES_TRANSLATOR_H
2 #define LIBRARIES_TRANSLATOR_H
3 /******
4 /* Commodore-Amiga, Inc.
5 /* translator.h
6 /******
7
8 /* Translator error return codes */
9
10 #define TR_NotUsed -1 /* This is an oft used system rc */
11 #define TR_NoMem -2 /* Can't allocate memory */
12 #define TR_MakeBad -4 /* Error in MakeLibrary call */
13
14
15 #endif LIBRARIES_TRANSLATOR_H
```

Dec 12 18:20 1985 resources/cia.h Page 1

```
/******
/* Commodore-Amiga, Inc.
/* cia.h
/******
#define CIAANAME "ciaa.resource"
#define CIABNAME "ciab.resource"
```

```

1  #ifndef RESOURCES_DISK_H
2  #define RESOURCES_DISK_H
3  /*
4  * Commodore-Amiga, Inc.
5  * disk.h
6  */
7  /*
8  *
9  * external declarations for disk resources
10 *
11 *
12 * SOURCE CONTROL
13 * -----
14 * $Header: disk.h,v 27.2 85/07/12 23:12:44 nell Exp $
15 *
16 * $Locker:  $
17 *
18 *
19 *
20 #ifndef EXEC_TYPES_H
21 #include "exec/types.h"
22 #endif !EXEC_TYPES_H
23
24 #ifndef EXEC_LISTS_H
25 #include "exec/lists.h"
26 #endif !EXEC_LISTS_H
27
28 #ifndef EXEC_PORTS_H
29 #include "exec/ports.h"
30 #endif !EXEC_PORTS_H
31
32 #ifndef EXEC_INTERRUPTS_H
33 #include "exec/interrupts.h"
34 #endif !EXEC_INTERRUPTS_H
35
36 #ifndef EXEC_LIBRARIES_H
37 #include "exec/libraries.h"
38 #endif !EXEC_LIBRARIES_H
39
40
41
42
43 * Resource structures
44 *
45 *
46
47
48
49 struct DiscResourceUnit {
50     struct Message dru_Message;
51     struct Interrupt dru_DiscBlock;
52     struct Interrupt dru_DiscSync;
53     struct Interrupt dru_Index;
54 };
55
56 struct DiscResource {

```

```

57     struct Library dr_Library;
58     struct DiscResourceUnit *dr_Current;
59     UBYTE dr_Flags;
60     UBYTE dr_pad;
61     struct Library *dr_SysLib;
62     struct Library *dr_ClarResource;
63     ULONG dr_UnitID[4];
64     struct List dr_Waiting;
65     struct Interrupt dr_DiscBlock;
66     struct Interrupt dr_DiscSync;
67     struct Interrupt dr_Index;
68 };
69
70 /* dr_Flags entries */
71 #define DRB_ALLOC0 0 /* unit zero is allocated */
72 #define DRB_ALLOC1 1 /* unit one is allocated */
73 #define DRB_ALLOC2 2 /* unit two is allocated */
74 #define DRB_ALLOC3 3 /* unit three is allocated */
75 #define DRB_ACTIVE 7 /* is the disk currently busy? */
76
77 #define DRF_ALLOC0 (1<<0) /* unit zero is allocated */
78 #define DRF_ALLOC1 (1<<1) /* unit one is allocated */
79 #define DRF_ALLOC2 (1<<2) /* unit two is allocated */
80 #define DRF_ALLOC3 (1<<3) /* unit three is allocated */
81 #define DRF_ACTIVE (1<<7) /* is the disk currently busy? */
82
83
84
85 /*
86 * Hardware Magic
87 *
88 *
89 *
90
91 #define DSKWAOFF 0x4000 /* idle command for dsklen register */
92
93
94
95
96
97 * Resource specific commands
98 *
99
100
101
102
103 * DISKNAME is a generic macro to get the name of the resource.
104 * This way if the name is ever changed you will pick up the
105 * change automatically.
106
107 #define DISKNAME "disk.resource"
108
109 #define DR_ALLOCUNIT (LIB_BASE - 0*LIB_VECTSIZE)
110 #define DR_FREEUNIT (LIB_BASE - 1*LIB_VECTSIZE)
111 #define DR_GETUNIT (LIB_BASE - 2*LIB_VECTSIZE)
112

```



```

113 #define DR_GIVEUNIT (LIB_BASE - 3*LIB_VECSIZE)
114 #define DR_GETUNITID (LIB_BASE - 4*LIB_VECSIZE)
115
116 #define DR_LASTOON (DR_GIVEUNIT)
117
118 /*
119 * drive types
120 *
121 *
122 *
123 *
124 #define DRT_AMICA (0x00000000)
125 #define DRT_37422D2S (0x55555555)
126 #define DRT_EMPTY (0xFFFFFFFF)
127
128 #endif RESOURCES_DISK_H
129

```

```

1 #ifndef RESOURCES_MISC_I
2 #define RESOURCES_MISC_I
3 /*
4 /* Commodore-Amiga, Inc.
5 /* misc.h
6 /*
7 /*
8 *
9 * external declarations for misc system resources
10 *
11 *
12 * SOURCE CONTROL
13 * -----
14 * $Header: misc.h,v 27.3 85/07/12 16:28:29 neil Exp $
15 *
16 * $Locker: $
17 *
18 *
19
20 #ifndef EXEC_TYPES_H
21 #include "exec/types.h"
22 #endif !EXEC_TYPES_H
23
24 #ifndef EXEC_LIBRARIES_H
25 #include "exec/libraries.h"
26 #endif !EXEC_LIBRARIES_H
27
28
29 /*
30 * Resource structures
31 *
32 *
33 *
34 #define MR_SERIALPORT 0
35 #define MR_SERIALBITS 1
36 #define MR_PARALLELPORT 2
37 #define MR_PARALLELBITS 3
38 #define NUMRTYPES 4
39
40 struct MiscResource {
41     struct Library mr_Library;
42     ULONG mr_AllocArray[NUMRTYPES];
43 };
44
45 #define MR_ALLOCMISCRESOURCE (LIB_BASE)
46 #define MR_FREEMISCRESOURCE (LIB_BASE + LIB_VECSIZE)
47
48 #define MISCRNAME "misc.resource"
49
50 #endif !RESOURCES_MISC_H
51

```

```
1 #ifndef RESOURCES_POTGO_H
2 #define RESOURCES_POTGO_H
3 /******
4 /* Commodore-Amiga, Inc.
5 /* potgo.h
6 /******
7 #define POTGO_NAME "potgo.resource"
8 #endif
```

```
1 #ifndef LIBRARIES_ICON_H
2 #define LIBRARIES_ICON_H
3 /******
4 /* Commodore-Amiga, Inc.
5 /* icon.h
6 /******
7 /******
8 /******
9 /******
10 /******
11 *
12 * icon.h -- external declarations for workbench support library
13 *
14 * SOURCE CONTROL
15 * -----
16 * $Header: icon.h,v 31.1 85/08/31 09:10:56 neil Exp $
17 * $Locker: $
18 *
19 *
20 *
21 *
22 *
23 /******
24 * library structures
25 *
26 *
27 *
28 *
29 #define ICONNAME "icon.library"
30 /******
31 *
32 *
33 *
34 * function types
35 *
36 *
37 struct MBOBject *GetMBOBject(), *AllocMBOBject();
38 LONG PutMBOBject(), PutIcon(), GetIcon(), MatchToolValue();
39 VOID FreeFreeList(), FreeMBOBject(), AddressFreeList();
40 char *ToolTypeArray();
41
42
43
44 #endif LIBRARIES_ICON_H
```

```

1  /*****
2  /* Commodore-Amiga, Inc.
3  /* startup.h
4  /*****
5  /* NOTE: This file is NOT used to generate lib/Astartup.obj or */
6  /* lib/Lstartup.obj. */
7
8  #ifndef EXEC_TYPES_H
9  #include "exec/types.h"
10 #endif EXEC_TYPES_H
11
12 #ifndef EXEC_PORTS_H
13 #include "exec/ports.h"
14 #endif EXEC_PORTS_H
15
16 #ifndef LIBRARIES_DOS_H
17 #include "libraries/dos.h"
18 #endif LIBRARIES_DOS_H
19
20 struct WStartup {
21     struct Message sm_Message; /* a standard message structure */
22     struct MsgPort sm_Port; /* the process descriptor for you */
23     BPTR sm_Segment; /* a descriptor for your code */
24     LONG sm_Args; /* the number of elements in ArgList */
25     char * sm_ToolWindow; /* description of window */
26     struct WArg * sm_ArgList; /* the arguments themselves */
27 };
28
29 struct WArg {
30     BPTR wa_Lock; /* a lock descriptor */
31     BYTE wa_Name; /* a string relative to that lock */
32 };
33
34

```

```

1  /*****
2  /* Commodore-Amiga, Inc.
3  /* workbench.h
4  /*
5  /* Commodore-Amiga, Inc.
6  /*
7  /* $Header: workbench.h,v 31.4 85/10/27 13:50:28 neil Exp $
8  /*
9  /* $Locker:  $
10 /*
11 /*
12 *****/
13
14 #ifndef EXEC_TYPES_H
15 #include "exec/types.h"
16 #endif EXEC_TYPES_H
17
18 #ifndef EXEC_NODES_H
19 #include "exec/nodes.h"
20 #endif EXEC_NODES_H
21
22 #ifndef EXEC_LISTS_H
23 #include "exec/lists.h"
24 #endif EXEC_LISTS_H
25
26 #ifndef EXEC_TASKS_H
27 #include "exec/tasks.h"
28 #endif EXEC_TASKS_H
29
30 #ifndef INTUITION_INTUITION_H
31 #include "intuition/intuition.h"
32 #endif INTUITION_INTUITION_H
33
34 #define WEDISK 1
35 #define WEDRAWER 2
36 #define WETOOL 3
37 #define WEPROJECT 4
38 #define WBCARBAGE 5
39 #define WEDOVICE 6
40 #define WEKICK 7
41
42 struct DrawerData {
43     struct NewWindow dd_NewWindow; /* args to open window */
44     LONG dd_CurrentX; /* current x coordinate of origin */
45     LONG dd_CurrentY; /* current y coordinate of origin */
46     LONG dd_MinX; /* smallest x coordinate in window */
47     LONG dd_MinY; /* smallest y coordinate in window */
48     LONG dd_MaxX; /* largest x coordinate in window */
49     LONG dd_MaxY; /* largest y coordinate in window */
50     struct Gadget dd_HorizScroll;
51     struct Gadget dd_VertScroll;
52     struct Gadget dd_UpMove;
53     struct Gadget dd_DownMove;
54     struct Gadget dd_LeftMove;
55     struct Gadget dd_RightMove;
56     struct Image dd_HorizImage;

```

```

57 struct Image
58 struct PropInfo
59 struct PropInfo
60 struct Window *
61 struct WObject *
62 struct List
63 LONG
64 };
65
66 /* the amount of DrawerData actually written to disk */
67 #define DRAWERDATAFILESIZE (sizeof( struct NewWindow ) * 2* sizeof(LONG))
68
69 struct DiskObject {
70  WORD
71  WORD
72  WORD
73  struct Gadget
74  struct Gadget
75  char *
76  char **
77  LONG
78  LONG
79  struct DrawerData * do_DrawerData;
80  char *
81  LONG
82 };
83
84 #define WB_DISKMAGIC 0xe310 /* a magic number, not easily impersonated */
85 #define WB_DISKVERSION 1 /* our current version number */
86
87 struct FreeList {
88  WORD
89  struct List
90  fl_NumFree;
91  fl_MemList;
92 };
93
94 struct WObject {
95  struct Node
96  struct Node
97  struct Node
98  struct Node
99  struct WObject *
100 /* object flags */
101 #ifdef SMARTCOMPILER
102  UBYTE
103  UBYTE
104  UBYTE
105  UBYTE
106 #else
107 /* lattice is not full system V compatible (yet)... */
108  UBYTE
109 #endif
110  wo_Type;
111  wo_UseCount;
112

```

```

113  object */
114  wo_Name; /* this object's textual name */
115  wo_NameOffset; /* where to put the name */
116  wo_NameYOffset;
117
118  char *
119  wo_DefaultTool;
120  struct DrawerData * wo_DrawerData;
121  struct Window * wo_IconWin;
122  LONG
123  wo_CurrentX; /* if this is a drawer or disk */
124  wo_CurrentY; /* each object's icon lives here */
125  wo_ToolTypes; /* virtual X in drawer */
126  wo_Gadget; /* virtual Y in drawer */
127  wo_FreeList; /* the types for this tool */
128  wo_ToolWindow; /* NOT a pointer, but an instance of a gadget structure */
129  wo_StackSize; /* this objects free list */
130  wo_Lock; /* character string for tool's window */
131
132  };
133 #define TMalloc( size, type ) ((type)Malloc( size ))
134 #define ObjMalloc( obj, size, type ) ((type)Malloc( obj, size ))
135 #define STREQ( a, b ) (strcmp( a, b ))
136
137 /* each message that comes into the WorkbenchPort must have a type field
138 * in the preceding short. These are the defines for this type */
139
140 #define MTYPE_PSTD 1 /* a "standard Potion" message */
141 #define MTYPE_TOOLEXIT 2 /* exit message from our tools */
142 #define MTYPE_DISKCHANGE 3 /* dos telling us of a disk change */
143 #define MTYPE_TIMER 4 /* we got a timer tick */
144 #define MTYPE_CLOSEDOWN 5 /* <unimplemented> */
145 #define MTYPE_IOPROC 6 /* <unimplemented> */
146
147 /* we use the gadget id field to encode some special information */
148 #define GID_WBOBJECT 0 /* a normal workbench object */
149 #define GID_HORIZSCROLL 1 /* the horizontal scroll gadget for a drawer */
150 #define GID_VERTSCROLL 2 /* the vertical scroll gadget for a drawer */
151 #define GID_LEFTSCROLL 3 /* move one window left */
152 #define GID_RIGHTSCROLL 4 /* move one window right */
153 #define GID_UPSCROLL 5 /* move one window up */
154 #define GID_DOWNSCROLL 6 /* move one window down */
155 #define GID_NAME 7 /* the name field for an object */
156
157 /* workbench does different complement modes for its gadgets.
158 * It supports separate images, complement mode, and backfill mode.
159 * The first two are identical to intuitions GADIMAGE and GADGROOMP.
160 * backfill is similar to GADGROOMP, but the region outside of the
161 * image (which normally would be color three when complemented)
162 * is flood-filled to color zero.
163
164 #define GADCBACKFILL 0x0001
165

```

Dec 8 16:39 1985 workbench/workbench.h Page 4

```
169 /* if an icon does not really live anywhere, set its current position
170 * to here
171 */
172 #define NO_ICON_POSITION (0x80000000)
```



Appendix E

Assembly Include Files — “.i” Files

This appendix contains the assembly language include files that define the system data structures used by the ROM (or kickstart) routines and the disk-loadable libraries.

As with the documentation files, these include-files are organized on a functional basis. In other words, things pertinent to the exec are listed under “exec/something.i”, things pertinent to graphics are listed under “graphics/graphicsitem.i” and so on.

This appendix is a hard-copy of the ”SYS:includes” directory on the Amiga Macro Assembler disk.

Cross Reference Utility (C) 1984, 1985 Commodore-Amiga, Inc.

1:dictionary 2:ables.i 3:adkbits.i 4:alerts.i 5:audio.i
6:blit.i 7:bootblock.i 8:cia.i 9:ciabase.i 10:clip.i
11:clipboard.i 12:console.i 13:copper.i 14:custom.i 15:devices.i
16:disk.i 17:diskfont.i 18:display.i 19:dmabits.i 20:dos.i
21:dos_lib.i 22:dosextns.i 23:errors.i 24:exec.i 25:exec_lib.i
26:execbase.i 27:execname.i 28:gameport.i 29:gels.i 30:gfx.i
31:gfxbase.i 32:icon.i 33:initializers.i 34:input.i 35:inputevent.i
36:inbbits.i 37:interrupts.i 38:intuition.i 39:intuitionbase.i 40:io.i
41:keyboard.i 42:keymap.i 43:layers.i 44:libraries.i 45:lists.i
46:memory.i 47:misc.i 48:narrator.i 49:nodes.i 50:parallel.i
51:ports.i 52:petgo.i 53:printer.i 54:prtbase.i 55:prtport.i
56:regions.i 57:resident.i 58:serial.i 59:sprite.i 60:startup.i
61:strings.i 62:tasks.i 63:text.i 64:timer.i 65:trackdisk.i
66:translator.i 67:types.i 68:view.i 69:workbench.i

ABC, 6-38
ABNC, 6-39
ABORT, 50-75, 58-72
ABORTIO, 40-81
ACTIVATE, 38-807
ACTIVE, 15-51, 16-72, 50-76, 58-73
ADALLOC_MAXPREC, 5-19
ADALLOC_MINPREC, 5-18
ADCMDB_NOUNIT, 5-27
ADCMDB_NOUNIT, 5-28, 5-29
ADCMD_ALLOCATE, 5-29
ADCMD_FINISH, 5-23
ADCMD_FREE, 5-21
ADCMD_LOCK, 5-25
ADCMD_PERVOL, 5-24
ADCMD_SETPREC, 5-22
ADCMD_WAITCYCLE, 5-26
ADDO_B, 2-40, 2-45, 2-80
ADHARD_CHANNELS, 5-16
ADIOB_NOWAIT, 5-35
ADIOB_PERVOL, 5-31
ADIOB_SYNCYCLE, 5-33
ADIOB_WRITEMESSAGE, 5-37
ADIOERR_ALLOCFAILED, 5-41
ADIOERR_CHANNELSTOLEN, 5-42
ADIOERR_NOALLOCATION, 5-40
ADIOF_NOWAIT, 5-36
ADIOF_PERVOL, 5-32
ADIOF_SYNCYCLE, 5-34
ADIOF_WRITEMESSAGE, 5-38
ADKB_FAST, 3-22
ADKB_MFMPREC, 3-18
ADKB_MSBSYNC, 3-21
ADKB_PRECOMP0, 3-17
ADKB_PRECOMP1, 3-16
ADKB_SETCLR, 3-15
ADKB_UARTBRK, 3-19
ADKB_USEOP1, 3-26

ADKB_USEOV1, 3-30
ADKB_USE1P2, 3-25
ADKB_USE1V2, 3-29
ADKB_USE2P3, 3-24
ADKB_USE2V3, 3-28
ADKB_USE3PN, 3-23
ADKB_USE3VN, 3-27
ADKB_WORDSYNC, 3-20
ADKE_FAST, 3-39
ADKE_MFMPREC, 3-35
ADKE_MSBSYNC, 3-38
ADKE_PRE000NS, 3-49
ADKE_PRE140NS, 3-50
ADKE_PRE280NS, 3-51
ADKE_PRE560NS, 3-52
ADKE_PRECOMP0, 3-34, 3-50, 3-52
ADKE_PRECOMP1, 3-33, 3-51, 3-52
ADKE_SETCLR, 3-32
ADKE_UARTBRK, 3-36
ADKE_USEOP1, 3-43
ADKE_USEOV1, 3-47
ADKE_USE1P2, 3-42
ADKE_USE1V2, 3-46
ADKE_USE2P3, 3-41
ADKE_USE2V3, 3-45
ADKE_USE3PN, 3-40
ADKE_USE3VN, 3-44
ADKE_WORDSYNC, 3-37
AF, 17-60, 17-61, 17-63
AG_IOError, 4-71
AG_MakeLib, 4-67
AG_NoMemory, 4-66
AG_OpenDev, 4-69
AG_OpenLib, 4-68
AG_OpenRes, 4-70
ALERT, 4-39
ALERTWACK, 4-17
ALLOCO, 16-68
ALLOCL, 16-69
ALLOCC, 16-70
ALLOCC, 16-71
ALTKEYMAP, 38-339
ANBC, 6-40
ANBC, 6-41
AN_BootError, 4-212
AN_BootStrap, 4-211
AN_CIARsrc, 4-200
AN_DRHsdDisk, 4-204
AN_DRIntNoAct, 4-205
AN_DiskRsrc, 4-203
AN_MiscRsrc, 4-208
AN_TMBsdReq, 4-197
AN_TimerDev, 4-196
AN_Workbench, 4-215
AREAOURLINE, 55-46
AT_DeadEnd, 4-62

AT_Recovery, 4-63
 AUDIONAME, 5-12
 AUTOITEXTFONT, 38-1259
 AUTONEXTTEXT, 38-1260
 AbortIO, 25-80
 AddDevice, 25-72
 AddLibrary, 25-66
 AddPort, 25-59
 AddResource, 25-81
 AlertData, 26-45
 AllocTrap, 25-57
 AreaInfo, 55-93
 AttnFlags, 26-83
 AttnResched, 26-84
 BACKDROP, 38-799
 BEGINIO, 40-77
 BGE_S, 2-53, 2-60
 BOLD, 63-21
 BORDERLESS, 38-805
 BOTTOMBORDER, 38-330
 CD_ASKKEYMAP, 12-26
 CD_SETKEYMAP, 12-27
 CHECKED, 38-142
 CHECKIT, 38-128
 CLEANME, 6-36
 CLEANMEN, 6-25, 6-26
 CMD, 33-48, 33-50, 33-54
 CMD_CLEAR, 40-117, 65-98
 CMD_COUNT, 40-95, 40-98, 40-103, 40-104, 40-104
 CMD_FLUSH, 40-120
 CMD_INVALID, 40-112
 CMD_NONSTD, 5-21, 5-22, 5-23, 5-24, 5-25, 5-26, 40-95, 40-125, 50-48, 50-49, 58-42, 58-43, 58-44, 58-46
 CMD_READ, 40-114, 65-93
 CMD_RESET, 40-113
 CMD_START, 40-119
 CMD_STOP, 40-118
 CMD_UPDATE, 40-116, 65-97
 CMD_WRITE, 40-115, 65-92
 COMMSEQ, 38-130
 COPPER_MOVE, 13-9
 COPPER_WAIT, 13-10
 COUNT, 33-43, 33-46, 33-48
 CPRNXTBUE, 13-11
 CPR_NT_LOE, 13-12
 CPR_NT_SHT, 13-13
 CTC_HCLRTAB, 12-86
 CTC_HCLRTABSALL, 12-87
 CTC_HSETTAB, 12-85
 CTRL_C, 20-150
 CTRL_D, 20-151
 CTRL_E, 20-152
 CTRL_F, 20-153
 CheckIO, 25-78
 ChkBase, 26-36
 ChkSum, 26-48

Close, 21-19
 CloseDevice, 25-75
 CloseLibrary, 25-69
 ColdCapture, 26-37
 CoolCapture, 26-38
 CopIn, 13-15
 CopList, 13-33
 CreateDir, 21-33
 CreateProc, 21-36
 CurrentDir, 21-34
 DBUFFER, 55-44
 DD, 15-32
 DD_SIZE, 15-33
 DEFERREFRESH, 38-210
 DESIGNED, 63-31
 DEVICES_AUDIO_I, 5-1, 5-2
 DEVICES_CONSOLE_I, 12-1, 12-2
 DEVICES_INPUTEVENT_I, 35-1, 35-2, 38-51
 DEVICES_INPUT_I, 34-1, 34-2
 DEVICES_KEYBOARD_I, 41-1, 41-2
 DEVICES_KEYMAP_I, 42-1, 42-2
 DEVICES_PARALLEL_I, 50-17, 50-18, 50-126, 54-30
 DEVICES_PRINTER_I, 53-1, 53-2
 DEVICES_TIMER_I, 35-13, 38-47, 54-36, 64-15, 64-16
 DEVICES_TRACKDISK_I, 65-18, 65-19
 DEV_ABORTIO, 40-68, 40-82
 DEV_BEGINIO, 40-67, 40-78
 DFH_ID, 17-40
 DISABLE, 2-37
 DISCRESOURCE, 16-55
 DISCRESOURCEUNIT, 16-47
 DISK, 17-61
 DISKFONT, 63-26
 DISKNAME, 16-100
 DR, 16-68, 16-69, 16-70, 16-71, 16-72
 DRT_37422D2S, 16-122
 DRT_AMIGA, 16-121
 DRT_EMPTY, 16-123
 DRU_DISCLOCK, 16-48
 DRU_DISCSYNC, 16-49
 DRU_INDEX, 16-50
 DRU_SIZE, 16-51
 DR_ALLOCUNIT, 16-106
 DR_CIARESOURCE, 16-60
 DR_CURRENT, 16-56
 DR_DISCLOCK, 16-63
 DR_DISCSYNC, 16-64
 DR_FLAGS, 16-57
 DR_FREEUNIT, 16-107
 DR_GETUNIT, 16-108
 DR_GETUNITID, 16-110
 DR_GIVEUNIT, 16-109, 16-112
 DR_INDEX, 16-65
 DR_LASTCOMM, 16-112
 DR_SIZE, 16-66
 DR_SYSLIB, 16-59

DR_UNITID, 16-61
 DR_WAITING, 16-62
 DR_pad, 16-58
 DS_W, 16-102, 33-41, 52-10, 64-28, 65-71
 DSKDMAOFF, 16-82
 DSR_CPR, 12-82
 DebugData, 26-44
 DebugEntry, 26-43
 Delay, 21-46
 DeleteFile, 21-25
 DeviceList, 26-98
 DeviceProc, 21-42
 DiskFontHeader, 17-43
 DispCount, 26-76
 DoIO, 25-76
 DoaPacket, 22-76
 DupLock, 21-29
 ENABLE, 2-50, 2-53, 2-55, 2-60, 2-62
 ENDGADGET, 38-312
 EOFMODE, 50-73, 58-64
 ETD_CLEAR, 65-98
 ETD_FORMAT, 65-96
 ETD_MOTOR, 65-94
 ETD_READ, 65-93
 ETD_SEEK, 65-95
 ETD_UPDATE, 65-97
 ETD_WRITE, 65-92
 EXEC_ABLES_I, 2-1, 2-2, 2-88
 EXEC_ALERTS_I, 4-1, 4-2, 4-217
 EXEC_DEVICES_I, 15-1, 15-2, 15-54
 EXEC_EXECBASE_I, 2-21, 2-23, 26-1, 26-2, 26-120
 EXEC_INITIALIZERS_I, 33-1, 33-2, 33-59
 EXEC_INTERRUPTS_I, 16-32, 16-34, 26-21, 26-23, 31-15, 37-1, 37-2, 37-71
 EXEC_IO_I, 5-8, 11-22, 12-19, 28-13, 34-13, 40-1, 40-2, 40-127, 41-13, 48-8, 50-24, 50-26, 53-31, 58-24, 58-26, 64-18, 64-20, 65-21, 65-23
 EXEC_LIBRARIES_I, 15-17, 15-19, 16-36, 16-38, 22-20, 26-25, 26-27, 31-12, 39-16, 40-21, 40-23, 44-1, 44-2, 44-125, 47-9, 47-11, 54-23
 EXEC_STRINGS_I, 50-20, 50-22, 58-20, 58-22, 61-1, 61-2, 61-64
 EXEC_TASKS_I, 22-14, 54-26, 62-1, 62-2, 62-92, 69-26, 69-28
 EXEC_TYPES_I, 2-17, 2-19, 16-20, 16-22, 22-11, 47-5, 47-7, 60-14, 60-16, 67-1, 67-2, 67-119, 69-14, 69-16
 EXTCOM, 65-74
 EXTENDED, 63-19
 Elapsed, 26-78
 ExNext, 21-31
 Examine, 21-30
 ExecBase, 26-32
 ExecBaseReserved, 26-109
 Execute, 21-50
 Exit, 21-37
 FC, 17-25
 FCH, 17-25
 FCH_IN, 17-32
 FOLLOWMOUSE, 38-322
 FORBID, 2-79
 FP, 63-25, 63-26, 63-27, 63-28, 63-29, 63-30, 63-31, 63-32

FRST_DOT, 55-42, 55-107
 FS, 63-19, 63-20, 63-21, 63-22
 FS_NORMAL, 63-18
 FileHandle, 22-59
 FindPort, 25-65
 FreeTrap, 25-58
 GADGDISABLED, 38-296
 GADGETTYPE, 38-346
 GADCHBOX, 38-269
 GADCHCOMP, 38-268
 GADCHIGHBITS, 38-267
 GADGHIMAGE, 38-270
 GADCHNONE, 38-271
 GADGIMAGE, 38-275
 GADGIMMEDIATE, 38-307
 GIMMEZEROZERO, 38-803
 GRAPHICS_COPPER_I, 13-2, 13-3
 GRAPHICS_GFXBASE_I, 31-6, 31-7
 GRAPHICS_TEXT_I, 17-19, 38-39, 63-1, 63-2
 GRELBOTTOM, 38-281
 GRELHEIGHT, 38-286
 GRELRIGHT, 38-282
 GRELWIDTH, 38-284
 Gadget, 38-219
 GelsInfo, 55-22
 GetCC, 25-88
 GetMsg, 25-62
 GetPacket, 21-40
 GfxBase, 31-19
 HARDWARE_ADKBITS_I, 3-12, 3-13, 3-54
 HARDWARE_BLIT_I, 6-11, 6-12, 6-89
 HIGHBOX, 38-138
 HIGHCOMP, 38-137
 HIGHLAGS, 38-135
 HIGHIMAGE, 38-136
 HIGHITEM, 38-147
 HIGHNONE, 38-139
 HSIZEBITS, 6-29, 6-30
 HSIZEMASK, 6-31
 ICONNAME, 32-30
 IDNestCnt, 2-40, 2-45, 2-52, 2-59, 26-80
 IECLASS_ACTIVIEWINDOW, 35-53
 IECLASS_CLOSEWINDOW, 35-41
 IECLASS_DISKINSERTED, 35-51
 IECLASS_DISKREMOVED, 35-49
 IECLASS_EVENT, 35-27
 IECLASS_GADGETDOWN, 35-33
 IECLASS_GADGETUP, 35-35
 IECLASS_MENULIST, 35-39
 IECLASS_NEWFREES, 35-47
 IECLASS_NULL, 35-21
 IECLASS_POINTERPOS, 35-29
 IECLASS_RANKEY, 35-23
 IECLASS_RANMOUSE, 35-25
 IECLASS_REFRESHWINDOW, 35-45
 IECLASS_REQUESTER, 35-37

```

IECLASS_SIZEWINDOW, 35-43
IECLASS_TIMER, 35-31
IND_ADDHANDLER, 34-19
IND_REMHANDLER, 34-20
IND_SETMPORT, 34-24
IND_SETMTRIG, 34-26
IND_SETMTYPE, 34-25
IND_SETPERIOD, 34-23
IND_SETTHRESH, 34-22
IND_WRITEEVENT, 34-21
INITBYTE, 33-18
INITLONG, 33-33
INITSTRUCT, 33-40
INITWORD, 33-26
INREQUEST, 38-811
INTASK, 15-52
INTUITION_INTUITIONBASE_I, 39-1, 39-2
INTABLES, 2-32
    IO, 40-34, 40-56, 54-62, 54-63, 54-64, 54-65
    IOAudio, 5-44
    IOEXTPAR, 50-95
    IOEXTPAR_SIZE, 50-123, 54-89, 54-90, 54-91, 54-94
    IOEXTID, 65-105
    IOPAR, 50-74, 50-75, 50-76
    IOPT, 50-77, 50-78, 50-79, 50-80
    IOSTD_SIZE, 40-51, 48-58, 50-95, 58-91, 65-105
    IOTD_COUNT, 65-106
    IOTD_SECLABEL, 65-107
    IOTD_SIZE, 65-108
    IOTV_TIME, 64-37
    IO_ACTUAL, 40-45
    IO_COMMAND, 40-37
    IO_DATA, 40-47
    IO_DEVICE, 40-35, 40-78, 40-82
    IO_ERROR, 40-39
    IO_FLAGS, 40-38
    IO_LENGTH, 40-46
    IO_OFFSET, 40-48
    IO_PARFLAGS, 50-121
    IO_PARSTATUS, 50-120
    IO_PEXTELAGS, 50-119
    IO_PTERMARRAY, 50-122
    IO_SIZE, 5-44, 40-40, 53-129, 53-137, 64-36
    IO_UNIT, 40-36
    ISDRAWN, 38-146
    ITALIC, 63-26
    ITEMENABLED, 38-132
    ITEMTEXT, 38-129
    IVAUD0, 26-61
    IVAUD1, 26-62
    IVAUD2, 26-63
    IVAUD3, 26-64
    IVBLIT, 26-60
    IVCOPEL, 26-58
    IVDSKBLK, 26-55
    IVDSKSYNC, 26-66

```

```

IVEXTER, 26-67
IVINTEN, 26-68
IVNMI, 26-69
IVPORTS, 26-57
IVRBE, 26-65
IVSOFTINT, 26-56
IVTBE, 26-54
IVVERTB, 26-59
IdleCount, 26-75
Info, 21-32
Input, 21-22
IntVects, 26-53
IntrList, 26-99
IntuitionBase, 39-37
IoErr, 21-35
IsInteractive, 21-49
    JSR, 2-85, 44-109
KBD_ADDRESETHANDLER, 41-21
KBD_READEVENT, 41-19
KBD_READMATRIX, 41-20
KBD_REMRESETHANDLER, 41-22
KBD_RESETHANDLERDONE, 41-23
KCB_CONTROL, 42-32
KCB_DOWNUP, 42-34
KCB_NOP, 42-25
KCB_STRING, 42-37
KCF_ALT, 42-31
KCF_CONTROL, 42-33
KCF_DOWNUP, 42-35
KCF_NOP, 42-26
KCF_SHIFT, 42-30
KCF_STRING, 42-38
KC_NOQUAL, 42-28
KC_VANILLA, 42-29
    KeyMap, 42-13
LEFTBORDER, 38-328
LIBINIT, 16-105, 40-65, 44-38, 44-62, 47-48
LIBRARIES_DISKFONT_I, 17-1, 17-2
LIBRARIES_DOSXTENS_I, 22-1, 22-3, 22-234, 54-39
LIB_BASE, 16-105, 44-26, 44-27, 44-62, 47-48
LINKLIB, 40-78, 40-82, 44-115, 44-117
LMN_REGION, 43-23
LN, 49-24
LN_NAME, 49-29
LN_PRI, 49-28, 54-59
LN_SUCC, 49-25
LN_TYPE, 49-27
LONGINT, 38-337
LastAlert, 26-107
LayerInfo_extra, 43-16
Layer_Info, 43-32
LibList, 26-100
LoadSeq, 21-38
Lock, 21-27
LowMemChkSum, 26-35
MACRO, 2-32, 2-37, 2-50, 2-73, 2-79, 2-84, 5-12, 8-6, 8-10, 12-95, 12-98,

```

16-100, 20-15, 20-81, 20-84, 21-11, 29-49, 29-54, 32-30, 33-18, 33-26, 33-33,
 33-40, 40-77, 40-81, 40-93, 40-102, 44-38, 44-50, 44-105, 44-107, 44-115, 44-117,
 45-37, 45-44, 45-53, 45-57, 45-61, 45-66, 45-71, 45-76, 45-82, 45-90, 45-100,
 45-107, 45-126, 45-133, 47-53, 50-67, 52-7, 58-59, 61-35, 61-42, 61-50, 61-57,
 64-26, 65-69, 67-17, 67-21, 67-26, 67-31, 67-36, 67-41, 67-46, 67-51, 67-56,
 67-61, 67-66, 67-71, 67-76, 67-81, 67-86, 67-91, 67-107, 67-113
 MAXBYTESPERROW, 6-34
 MAXCYLS, 65-33
 MAXFONTNAME, 17-41, 17-55
 MAXFONTPATH, 17-23, 17-26
 MAXRETRY, 65-37
 MEMORY, 17-60
 MENUSTATE, 38-812
 MENUTOGGLE, 38-131
 MENUTOGGLED, 38-148
 MEXIT, 33-52
 MN, 51-63
 MN_LENGTH, 51-65
 MN_REPLYPORT, 51-64
 MP, 51-32
 MP_FLAGS, 51-33
 MP_MSGLIST, 51-36
 MP_SIGBIT, 51-34
 MP_SIGTASK, 51-35, 51-42, 51-82
 MP_SOFTINT, 51-42
 MRB_SIZE, 48-80
 M_ASM, 12-95
 M_AWM, 12-98
 M_LNM, 12-94
 MaxLocMem, 26-42
 MemList, 26-96
 NABC, 6-42
 NOCAREREFRESH, 38-816
 NOCROSSEILL, 55-47
 NT_DEVICE, 49-38
 NT_FONT, 49-47
 NT_FREEMSG, 49-41
 NT_INTERRUPT, 49-37
 NT_LIBRARY, 49-44
 NT_MEMORY, 49-45
 NT_MESSAGE, 49-40
 NT_MSGPORT, 49-39
 NT_PROCESS, 49-48
 NT_REPLYMSG, 49-42
 NT_RESOURCE, 49-43
 NT_SEMAPHORE, 49-49
 NT_SOFTINT, 49-46
 NT_TASK, 49-36
 NT_UNKNOWN, 49-35
 NUMCYLS, 65-32, 65-33
 NUMHEADS, 65-36
 NUMSECS, 65-35
 NUMTRACKS, 65-38
 NUMUNITS, 65-39
 NewWindow, 38-841
 ONE_DOT, 55-43, 55-105

OTHER_REFRESH, 38-797
 OWNBLITTERn, 31-55
 OldOpenLibrary, 25-68
 Open, 21-18
 OpenDevice, 25-74
 OpenLibrary, 25-92
 OpenResource, 25-83
 Output, 21-23
 PAPEROUT, 50-79
 PAR, 50-71, 50-72, 50-73
 PARALLELNAME, 50-67
 PA_IGNORE, 51-54
 PA_SIGNAL, 51-52
 PA_SOFTINT, 51-53
 PBUSY, 50-78
 PDCMD_QUERY, 50-48
 PDCMD_SETPARAMS, 50-49
 PERMIT, 2-84
 PF_ACTION, 51-47
 POINTREL, 38-203
 POINTNAME, 52-7
 PRD_DUMPFORMAT, 53-39
 PRD_PRICOMMAND, 53-38
 PRD_RAWWRITE, 53-37
 FREDRAWN, 38-204
 PRIMARY_CLIP, 11-57
 PROPORTIONAL, 63-30
 PSEL, 50-80
 PTERMARRAY, 50-85
 PTERMARRAY_0, 50-86
 PTERMARRAY_1, 50-87
 PTERMARRAY_SIZE, 50-88, 50-122
 ParErr_BuTTooBig, 50-35
 ParErr_DevBusy, 50-36
 ParErr_InitErr, 50-40
 ParErr_InvParam, 50-36
 ParErr_LineErr, 50-37
 ParErr_NotOpen, 50-38
 ParErr_PortReset, 50-39
 Par_DEVFINISH, 50-50
 ParentDir, 21-48
 PortList, 26-101
 PrinterExtendedData, 54-121
 PrinterSegment, 54-142
 Process, 22-33
 Procure, 25-90
 PutMag, 25-61
 QBOWNER, 31-58
 QBOWNERn, 31-56, 31-58
 QUEUED, 50-74, 54-62, 58-71
 QUICK, 40-56
 Quantum, 26-77
 QueuePacket, 21-41
 RAD_BOOGIE, 50-72, 58-66
 REFRESHBITS, 38-793
 RELVERIFY, 38-302

REMOVED, 63-32
 REPORTMOUSE, 38-801
 REQACTIVE, 38-208
 REQOFFWINDOW, 38-207
 RESOURCES_DISK_I, 16-1, 16-2
 RESOURCES_POTGO_I, 52-1, 52-2
 RETURN_FAIL, 20-147
 REVPATH, 63-27
 RIGHTBORDER, 38-327
 RMBTRAP, 38-815
 ROMEONT, 63-25
 RP, 55-42, 55-43, 55-44, 55-46, 55-47, 55-56
 RP_COMPLEMENT, 55-52
 RP_INVERSVID, 55-53
 RP_JAMI, 55-50
 RWDIR, 50-77
 RastPort, 55-58
 RavDoFmt, 25-87
 RavIOInit, 25-84
 RawMayGetChar, 25-85
 RawPutChar, 25-86
 Read, 21-20
 ReadDevice, 25-73
 ReadLibrary, 25-67
 ReadPort, 25-60
 ReadResource, 25-82
 Rename, 21-26
 ReplyMsg, 25-63
 Requester, 38-158
 ResModules, 26-85
 ResourceList, 26-97
 RsvdExt, 26-46
 S, 4-17, 7-47, 20-114, 20-115, 37-53, 37-54, 37-55
 SELECTED, 38-290
 SCR_BLACK, 12-39
 SCR_BLACKBG, 12-49
 SCR_BLUE, 12-43
 SCR_BLUEBG, 12-53
 SCR_BOLD, 12-33
 SCR_CLRO, 12-61
 SCR_CLR0BG, 12-70
 SCR_CLR1, 12-62
 SCR_CLR1BG, 12-71
 SCR_CLR2, 12-63
 SCR_CLR2BG, 12-72
 SCR_CLR3, 12-64
 SCR_CLR3BG, 12-73
 SCR_CLR4, 12-65
 SCR_CLR4BG, 12-74
 SCR_CLR5, 12-66
 SCR_CLR5BG, 12-75
 SCR_CLR6, 12-67
 SCR_CLR6BG, 12-76
 SCR_CLR7, 12-68
 SCR_CLR7BG, 12-77
 SCR_CYAN, 12-45

SCR_CYANBG, 12-55
 SCR_DEFAULT, 12-47
 SCR_DEFAULTBG, 12-57
 SCR_GREEN, 12-41
 SCR_GREENBG, 12-51
 SCR_ITALIC, 12-34
 SCR_MAGENTA, 12-44
 SCR_MAGENTABG, 12-54
 SCR_NEGATIVE, 12-36
 SCR_PRIMARY, 12-32
 SCR_RED, 12-40
 SCR_REDBG, 12-50
 SCR_UNDERSCORE, 12-35
 SCR_WHITE, 12-46
 SCR_WHITEBG, 12-56
 SCR_YELLOW, 12-42
 SCR_YELLOWBG, 12-52
 SHARED, 50-71, 58-65
 SIGBREAK, 20-150, 20-151, 20-152, 20-153
 SIMPLE_REFRESH, 38-795
 SIZEBOTTOM, 38-789
 SIZEBRIGHT, 38-788
 SIZEOF_VIEW, 39-40
 SM, 51-75
 SMART_REFRESH, 38-794
 SM_BIDS, 51-76
 SM_LOCKMSG, 51-82
 SM_SIZE, 51-77
 STRING, 50-68, 58-60, 61-35
 STRINGCENTER, 38-334
 STRINGRIGHT, 38-335
 SUBQ.B, 2-52, 2-59
 SUPER_BITMAP, 38-796
 SUPER_UNUSED, 38-823
 SYSREQUEST, 38-209
 SatisfyMsg, 11-59
 Screen, 38-909
 Seek, 21-24
 SendIO, 25-77
 SetComment, 21-43
 SetFunction, 25-70
 SetProtection, 21-44
 SoftInts, 26-105
 SoftVer, 26-34
 SunLibrary, 25-71
 SysFlags, 26-79
 SysStkLower, 26-41
 SysStkUpper, 26-40
 TALLDOT, 63-28
 TASKABLES, 2-73
 TBC_HCLRTAB, 12-90
 TBC_HCLRTABALL, 12-91
 TC_SIZE, 22-34, 54-101, 62-54
 TD, 65-74
 TDERR_BadHdrSum, 65-124
 TDERR_BadSecHdr, 65-127

TDERR_BadSecID, 65-123
 TDERR_BadSecPreamble, 65-122
 TDERR_BadSecSum, 65-125
 TDERR_DiskChanged, 65-129
 TDERR_NoSecHdr, 65-121
 TDERR_NotSpecified, 65-120
 TDERR_TooFewSecs, 65-126
 TDERR_WriteProt, 65-128
 TDE_EXTCOM, 65-92, 65-93, 65-94, 65-95, 65-96, 65-97, 65-98
 TDnestCnt, 2-80, 26-81
 TD_CHANGEENUM, 65-81
 TD_CHANGESTATE, 65-82
 TD_FORMAT, 65-79, 65-96
 TD_LABELSIZE, 65-112
 TD_LASTCOMM, 65-85
 TD_MOTOR, 65-77, 65-94
 TD_NAME, 65-69
 TD_PROTSTATUS, 65-83, 65-85
 TD_REMOVE, 65-80
 TD_SECSHIFT, 65-50
 TD_SECTOR, 65-49
 TD_SEEK, 65-78, 65-95
 TIMEREQUEST, 64-36
 TIMERNAME, 64-26
 TIMEVAL, 64-31
 TOGGLESELECT, 38-332
 TOPBORDER, 38-329
 TR_ADDRREQUEST, 64-42
 TR_GETSYSTIME, 64-43
 TR_SETSYSTIME, 64-44
 TV_MICRO, 64-33
 TV_SECS, 64-32
 TXSCALE, 55-56
 TaskExceptCode, 26-88
 TaskExitCode, 26-89
 TaskReady, 26-102
 TaskSigAlloc, 26-90
 TaskTrapAlloc, 26-91
 TaskTrapCode, 26-87
 TaskWait, 26-103
 TextAttr, 63-36
 TextFont, 63-45
 ThisTask, 26-74
 TapRas, 55-15
 TypeOfMem, 25-89
 UCopList, 13-46
 UNDERLINED, 63-22
 UNIT, 15-42, 15-51, 15-52
 UNIT_FLAGS, 15-43
 UNIT_MICROHZ, 64-23
 UNIT_OPENCNT, 15-45
 UNIT_SIZE, 15-46
 UNIT_VBLANK, 64-24
 UNIT_pad, 15-44
 UnLoadSeg, 21-39
 UnLock, 21-28

VSIZEBITS, 6-30
 VSIZEMASK, 6-32
 Vacate, 25-91
 ViewPort, 13-36, 68-28
 WBENCHWINDOW, 38-820
 WIDEDOT, 63-29
 WINDOWACTIVE, 38-810
 WINDOWCLOSE, 38-786
 WINDOWDEPTH, 38-785
 WINDOWDRAG, 38-784
 WINDOWREFRESH, 38-819
 WINDOWSIZING, 38-783
 WINDOWTICKED, 38-821
 WORKBENCH_ICON_Y, 32-2, 32-3, 32-34
 WaitForChar, 21-47
 WaitIO, 25-79
 WaitPort, 25-64
 WarnCapture, 26-39
 Write, 21-21
 XREF, 2-33, 2-75, 67-18
 LVO, 21-12, 67-18
 LVOAlert, 4-46
 LVOPermit, 2-78, 2-85
 _intena, 2-33, 2-39, 2-44, 2-54, 2-61
 a5, 4-40, 4-43, 4-47
 a6, 4-40, 4-45, 4-46, 4-47
 aIND, 53-44
 aNEL, 53-45
 aRI, 53-46
 aRIN, 53-43
 aRIS, 53-42
 aSCR0, 53-48
 aSCR23, 53-50
 aSCR24, 53-52
 aSCR3, 53-49
 aSCR4, 53-51
 af_Type, 17-64
 ai_Count, 55-98
 ai_FlagPtr, 55-97
 ai_FlagTbl, 55-96
 ai_MaxCount, 55-99
 ai_VctrPtr, 55-95
 ai_VctrTbl, 55-94
 alertNumber, 4-39
 already, 4-204
 audio.device, 5-13
 bitnode, 6-14
 bn_SIZEOF, 6-22
 bn_beamsync, 6-20
 bn_bitsize, 6-19
 bn_clearup, 6-21
 bn_dummy, 6-18
 bn_function, 6-16
 bn_n, 6-15
 bn_stat, 6-17
 ci_DestAddr, 13-19

```

ci_DestData, 13-22
ci_HWaitPos, 13-21
ci_OpCode, 13-16
ci_SIZEOF, 13-24
ci_VWaitPos, 13-18
ci_nxtlist, 13-17
ci, 13-36
ci_SIZEOF, 13-44
color, 14-122
copinit, 13-53
copinit_diagstrt, 13-54
copinit_sprstop, 13-56
copinit_sprstrtop, 13-55
count, 21-10, 21-12, 21-13, 21-13
cprlist, 13-27
crl_Next, 13-28
crl_SIZEOF, 13-31
crl_max, 13-30
crl_start, 13-29
d7, 4-40, 4-41, 4-47
devices, 35-14, 38-48, 38-52, 54-31, 54-34, 54-37
dfh_DE, 17-51
dfh_FileID, 17-52
dfh_Name, 17-55
dfh_Revision, 17-53
dfh_SIZEOF, 17-57
dfh_Segment, 17-54
dfh_TT, 17-56
disk, 4-204
disk_resource, 16-101
dos.i, 22-25, 60-23
dp_Action, 22-90
dp_Arg1, 22-88, 22-93
dp_Link, 22-77
dp_Port, 22-78
dp_Res1, 22-82, 22-91
dp_Res2, 22-86, 22-92
dp_Status, 22-91
dp_Status2, 22-92
dp_Type, 22-80, 22-90
execbase.i, 2-22
fc_FileName, 17-26
fc_Flags, 17-29
fc_SIZEOF, 17-30
fc_Style, 17-28
fc_YSize, 17-27
fch_FC, 17-37
fch_FileID, 17-35
fch_NumEntries, 17-36
fh_Arg1, 22-71
fh_Arg2, 22-72
fh_Args, 22-70, 22-71
fh_Buf, 22-63
fh_End, 22-65
fh_Func1, 22-67
fh_Func2, 22-68

```

```

fh_Func3, 22-69
fh_Funcs, 22-66, 22-67
fh_Interactive, 22-61
fh_Link, 22-60
fh_Pos, 22-64
fh_SIZEOF, 22-73
fh_Type, 22-62
gb_ActivView, 31-20
gb_BeamSync, 31-38
gb_BlitLock, 31-44
gb_BlitNest, 31-45
gb_BlitOwner, 31-47
gb_BlitWaitQ, 31-46
gb_Debug, 31-37
gb_DefaultFont, 31-34
gb_DisplayFlags, 31-49
gb_Flags, 31-43
gb_LOFlist, 31-24
gb_Modes, 31-35
gb_SHElist, 31-25
gb_SIZE, 31-52
gb_SpriteReserved, 31-40
gb_TOF_WaitQ, 31-48
gb_TextFonts, 31-33
gb_VBlank, 31-36
gb_blitter, 31-23
gb_bltld, 31-26
gb_bltlrv, 31-32
gb_blttl, 31-27
gb_bbltld, 31-28
gb_bbltll, 31-29
gb_bytereserved, 31-41
gb_cia, 31-22
gb_copinit, 31-21
gb_reserved, 31-51
gb_system_bplcon0, 31-39
gb_timer, 31-31
gb_vbrv, 31-30
get, 4-204
gg_Activation, 38-230
gg_Flags, 38-228
gg_GadgetID, 38-260
gg_GadgetRender, 38-237
gg_GadgetText, 38-243
gg_GadgetType, 38-232
gg_Height, 38-226
gg_LeftEdge, 38-223
gg_MutualExclude, 38-254
gg_NextGadget, 38-221
gg_SIZEOF, 38-263, 69-54, 69-55, 69-56, 69-57, 69-58, 69-59, 69-77, 69-120
gg_SelectRender, 38-241
gg_SpecialInfo, 38-258
gg_TopEdge, 38-224
gg_UserData, 38-261
gg_Width, 38-225
gl_Flags, 55-25

```

```

    gi_SIZEOF, 55-39
    gi_bottommost, 55-36
    gi_collHandler, 55-32
    gi_firstBlissObj, 55-37
    gi_gelHead, 55-26
    gi_gelTail, 55-27
    gi_lastBlissObj, 55-38
    gi_leftmost, 55-33
    gi_rightmost, 55-34
    gi_sprRervd, 55-23
    gi_topmost, 55-35
    has, 4-204
    ib_ActiveScreen, 39-42
    ib_ActiveWindow, 39-41
    ib_FirstScreen, 39-47
    ib_LibNode, 39-39
    ib_ViewLord, 39-40
    icon.library, 32-31
    interrupts.i, 16-33, 24-4, 26-22, 31-16
    io.i, 5-9, 11-23, 12-20, 24-10, 28-14, 34-14, 41-14, 48-9, 50-25, 53-32, 58-25,
    64-19, 65-22
    io_ClipID, 11-52
    io_Data, 11-50
    io_Length, 11-49
    io_Offset, 11-51
    ioa_AllocKey, 5-45
    ioa_Cycles, 5-50
    ioa_Data, 5-46
    ioa_Length, 5-47
    ioa_Period, 5-48
    ioa_SIZEOF, 5-52
    ioa_Volume, 5-49
    ioa_WriteMag, 5-51
    iocr_SIZEOF, 11-53
    jar, 4-46
    km_HiCapsable, 42-20
    km_HiKeyMap, 42-19
    km_HiKeyMapTypes, 42-18
    km_HiRepeatable, 42-21
    km_LoCapsable, 42-16
    km_LoKeyMap, 42-15
    km_LoKeyMapTypes, 42-14
    km_LoRepeatable, 42-17
    km_SIZEOF, 42-22
    lea, 4-43
    li_Lock, 43-38
    li_LockPort, 43-37
    li_Locker, 43-42
    li_RP_ReplyPort, 43-36
    li_broadcast, 43-39
    li_bytereserved, 43-43
    li_check_lp, 43-34
    li_locknest, 43-40
    li_obs, 43-35
    li_pad, 43-41
    li_top_layer, 43-33

```

```

    li_wordreserved, 43-44
    libraries, 22-25, 54-40, 60-23
    libraries.i, 15-18, 16-37, 22-21, 24-8, 26-26, 31-13, 39-17, 40-22, 47-10, 54-24
    lie_FreeClipRects, 43-19
    lie_SIZEOF, 43-21
    lie_blitbuff, 43-20
    lie_env, 43-17
    lie_mem, 43-18
    macro, 4-39, 7-38, 7-42, 27-4
    mem_node, 43-25
    memnode_SIZEOF, 43-30
    memnode_how_big, 43-29
    memnode_pred, 43-27
    memnode_succ, 43-26
    memnode_where, 43-28
    mi_Command, 38-108
    mi_ItemFill, 38-102
    mi_KludgeFill00, 38-117
    mi_NextSelect, 38-123
    mi_SIZEOF, 38-125
    mi_SelectFill, 38-106
    mi_SubItem, 38-119
    move.l, 4-41, 4-45
    movem.l, 4-40, 4-47
    nw_BitMap, 38-877
    nw_BlockPen, 38-849
    nw_CheckMark, 38-864
    nw_DetailPen, 38-848
    nw_FirstGadget, 38-859
    nw_Flags, 38-853
    nw_Height, 38-846
    nw_IDCMPFlags, 38-851
    nw_LeftEdge, 38-843
    nw_MaxHeight, 38-893
    nw_MaxWidth, 38-892
    nw_MinHeight, 38-891
    nw_MinWidth, 38-890
    nw_SIZE, 38-901, 69-47, 69-71
    nw_Screen, 38-872
    nw_Title, 38-866
    nw_TopEdge, 38-844
    nw_Type, 38-899
    nw_Width, 38-845
    parallel.device, 50-68
    paramArray, 4-39
    ped_Close, 54-126
    ped_ColorClass, 54-128
    ped_Commands, 54-136
    ped_DoSpecial, 54-137
    ped_Expunge, 54-124
    ped_Init, 54-123
    ped_MaxColumns, 54-129
    ped_MaxDots, 54-132
    ped_MaxYDots, 54-133
    ped_NumCharSets, 54-130
    ped_NumRows, 54-131

```



```

ped_Open, 54-125
ped_PrinterClass, 54-127
ped_PrinterName, 54-122
ped_Render, 54-138
ped_SIZEOF, 54-140
ped_TimeoutSecs, 54-139
ped_XDotsInch, 54-134
ped_YDotsInch, 54-135
potgo_resource, 52-8
pr_CIS, 22-44
pr_CLI, 22-48
pr_COS, 22-45
pr_ConsoleTask, 22-46
pr_CurrentDir, 22-43
pr_FileSystemTask, 22-47
pr_GlobVec, 22-39
pr_MsgPort, 22-35
pr_Pad, 22-36
pr_PktWait, 22-50
pr_Result2, 22-42
pr_ReturnAddr, 22-49
pr_SIZEOF, 22-52
pr_SegList, 22-37
pr_StackBase, 22-41
pr_StackSize, 22-38
pr_Task, 22-34
pr_TaskRun, 22-40
pr_WindowPtr, 22-51
ps_NextSegment, 54-143
ps_PED, 54-147
ps_Revision, 54-146
ps_Version, 54-145
ps_runAlert, 54-144
reserve, 21-8
rp_AOLPen, 55-68
rp_AlgoStyle, 55-81
rp_AreaInfo, 55-63
rp_AreaPtSz, 55-70
rp_AreaPtrn, 55-61
rp_BgPen, 55-67
rp_BitMap, 55-60
rp_DrawMode, 55-69
rp_Dummy, 55-71
rp_EgPen, 55-66
rp_Flags, 55-73
rp_Font, 55-80
rp_GelsInfo, 55-64
rp_Layer, 55-59
rp_LinePtrn, 55-74
rp_Mask, 55-65
rp_PenHeight, 55-79
rp_PenWidth, 55-78
rp_RP_User, 55-87
rp_TmpRas, 55-62
rp_TxBaseline, 55-85
rp_TxFlags, 55-82

```

```

rp_TxHeight, 55-83
rp_TxSpacing, 55-86
rp_TxWidth, 55-84
rp_cp_x, 55-75
rp_cp_y, 55-76
rp_linpatcnt, 55-72
rp_longreserved, 55-89
rp_minterms, 55-77
rp_reserved, 55-90
rp_wordreserved, 55-88
rq_BackFill, 38-176
rq_Flags, 38-174
rq_Height, 38-165
rq_KludgeFill00, 38-185
rq_LeftEdge, 38-162
rq_OlderRequest, 38-161
rq_RWindow, 38-197
rq_RelLeft, 38-167
rq_RelTop, 38-168
rq_ReqBMap, 38-195
rq_ReqBorder, 38-171
rq_ReqCadget, 38-170
rq_ReqLayer, 38-187
rq_ReqPad1, 38-188
rq_ReqPad2, 38-198
rq_ReqText, 38-172
rq_SIZEOF, 38-200
rq_TopEdge, 38-163
rq_Width, 38-164
satisfyMsg_SIZEOF, 11-63
sc_FirstWindow, 38-912
sc_Height, 38-918
sc_LeftEdge, 38-914
sc_MouseY, 38-920
sc_NextScreen, 38-911
sc_TopEdge, 38-915
sc_Width, 38-917
scratch, 4-39
sd_ctl, 14-118
sd_dataa, 14-119
sd_datab, 14-120
sd_pos, 14-117
sa_ClipID, 11-62
sa_Msg, 11-60
sa_Unit, 11-61
sp, 4-40, 4-47
strings.i, 50-21, 58-21
ta_Flags, 63-40
ta_Name, 63-37
ta_Style, 63-39
ta_YSize, 63-38
tasks.i, 22-15, 24-7, 54-27, 69-27
text.i, 17-20, 38-40
tf_Accessors, 63-54
tf_Baseline, 63-51
tf_BoldSmear, 63-52

```

```

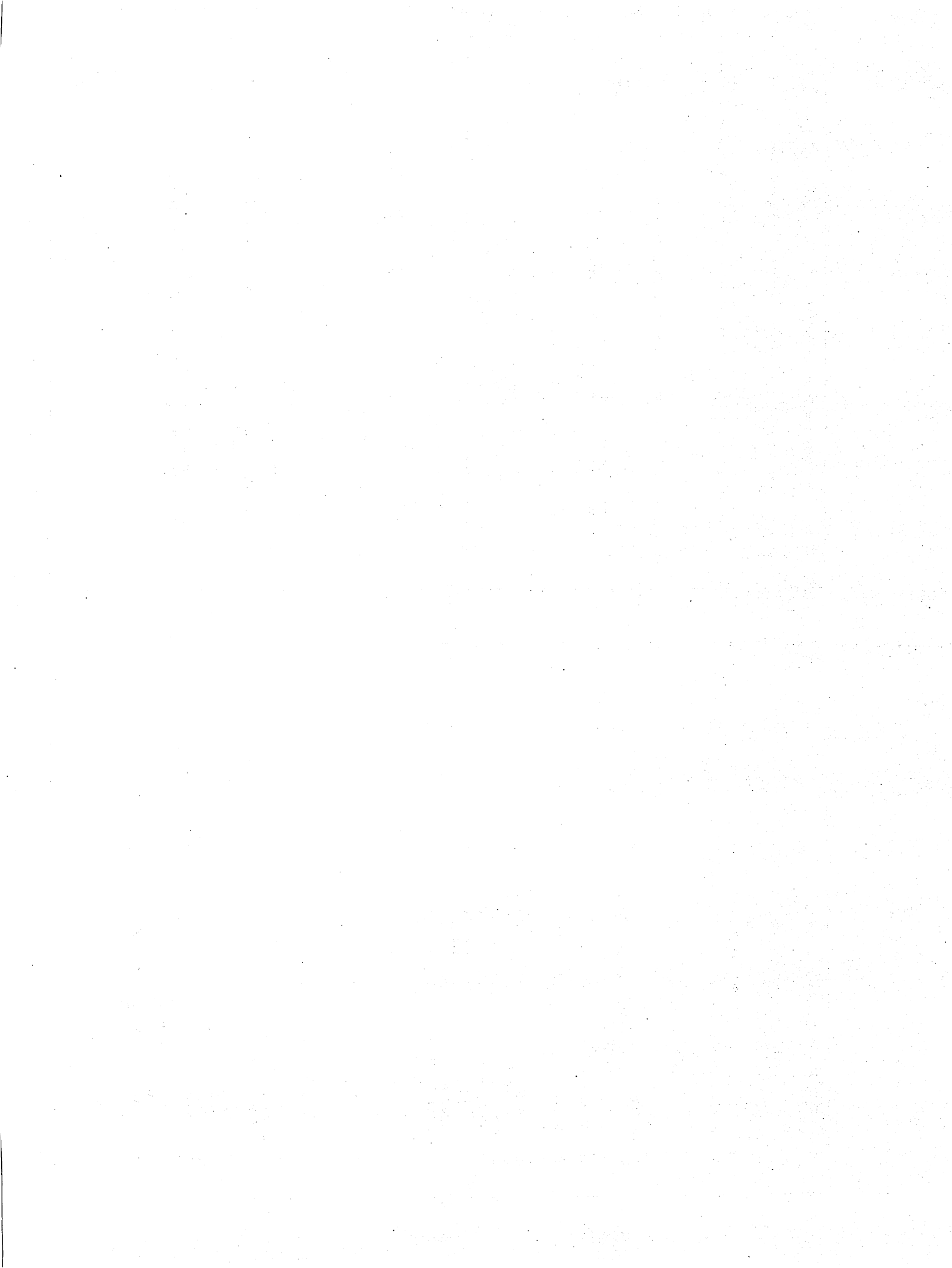
tf_CharData, 63-58
tf_CharKern, 63-64
tf_CharLoc, 63-61
tf_CharSpace, 63-63
tf_Flags, 63-49
tf_HiChar, 63-57
tf_LoChar, 63-56
tf_Modulo, 63-60
tf_SIZEOF, 17-56, 63-65
tf_Style, 63-48
tf_XSize, 63-50
tf_YSize, 63-47
timer.device, 64-37
timer.i, 35-14, 38-48, 54-37
tr_RasPtr, 55-16
tr_SIZEOF, 55-18
tr_Size, 55-17
trackdisk.device, 65-70
types.i, 2-18, 16-21, 22-12, 47-6, 60-15, 69-15
ucl_SIZEOF, 13-50
unit, 4-204
vsize, 21-9, 21-13
vd_BlockPen, 38-752
vd_BorderBottom, 38-724
vd_BorderLeft, 38-721
vd_BorderRPort, 38-725
vd_BorderRight, 38-723
vd_BorderTop, 38-722
vd_CheckMark, 38-757
vd_DMRequest, 38-707
vd_Descendant, 38-735
vd_DetailPen, 38-751
vd_ExtData, 38-774
vd_FirstGadget, 38-731
vd_FirstRequest, 38-706
vd_Flags, 38-700
vd_GZZHeight, 38-772
vd_GZZMouseX, 38-767
vd_GZZMouseY, 38-768
vd_GZZWidth, 38-771
vd_IDCMPFlags, 38-746
vd_MaxHeight, 38-698
vd_MaxWidth, 38-697
vd_MenuStrip, 38-702
vd_MessageKey, 38-749
vd_Parent, 38-734
vd_Pointer, 38-739
vd_PtrHeight, 38-740
vd_PtrWidth, 38-741
vd_RPort, 38-710
vd_ReqCount, 38-708
vd_ScreenTitle, 38-760
vd_Size, 38-780
vd_Title, 38-704
vd_UserData, 38-777
vd_UserPort, 38-747

```

```

vd_WLayer, 38-778
vd_WScreen, 38-709
vd_WindowPort, 38-748
vd_XOffset, 38-742
vd_YOffset, 38-743

```



```

1 IFND DEVICES_AUDIO_I
2 DEVICES_AUDIO_I SET 1
3 *****
4 * Commodore-Amiga, Inc.
5 * audio.1
6 *****
7
8 IFND EXEC_IO_I
9 INCLUDE "exec/io.1"
10 ENDC
11
12 AUDIONAME MACRO
13 DC.B 'audio.device', 0
14 ENDM
15
16 ADHARD_CHANNELS EQU 4
17
18 ADALOC_MINPREC EQU -128
19 ADALOC_MAXPREC EQU 127
20
21 ADCMD_FREE EQU CMD_NONSTD+0
22 ADCMD_SETPREC EQU CMD_NONSTD+1
23 ADCMD_FINISH EQU CMD_NONSTD+2
24 ADCMD_PERVOL EQU CMD_NONSTD+3
25 ADCMD_LOCK EQU CMD_NONSTD+4
26 ADCMD_WAITCYCLE EQU CMD_NONSTD+5
27 ADCMDB_NOUNIT EQU 5
28 ADCMDF_NOUNIT EQU 1<<5
29 ADCMD_ALLOCATE EQU ADCMDF_NOUNIT+0
30
31 ADIOB_PERVOL EQU 4
32 ADIOE_PERVOL EQU 1<<4
33 ADIOB_SYMCYCLE EQU 5
34 ADIOE_SYMCYCLE EQU 1<<5
35 ADIOB_NOWAIT EQU 6
36 ADIOE_NOWAIT EQU 1<<6
37 ADIOB_WRITEMESSAGE EQU 7
38 ADIOE_WRITEMESSAGE EQU 1<<7
39
40 ADIOERR_NOALLOCATION EQU -10
41 ADIOERR_ALLOCFAILED EQU -11
42 ADIOERR_CHANNELSTOLEN EQU -12
43
44 STRUCTURE IOAudio, IO_SIZE
45 WORD ioa_AllocKey
46 APTR ioa_Data
47 ULONG ioa_Length
48 UMWORD ioa_Period
49 UMWORD ioa_Volume
50 UMWORD ioa_Cycles
51 STRUCT ioa_WriteMsg, MN_SIZE
52 LABEL ioa_SIZEOF
53
54 ENDC

```

```

1 *****
2 * Commodore-Amiga, Inc.
3 * bootblock.1
4 *****
5 * Source Control
6 * -----
7 * $Header: bootblock.1,v 27.1 85/06/24 13:15:16 neil Exp $
8 * $Locker: $
9 *
10 * $Log: bootblock.1,v $
11 * Revision 27.1 85/06/24 13:15:16 neil
12 * *** empty log message ***
13 *
14 * Revision 26.2 85/06/18 23:55:38 neil
15 * Added BBNAMES definitions
16 *
17 * Revision 26.1 85/06/17 20:08:25 neil
18 * *** empty log message ***
19 *****
20 ***** BootBlock definition:
21
22 STRUCTURE BB, 0
23 STRUCT BB_ID, 4
24 LONG BB_CHKSUM
25 LABEL BB_ENTRY
26 LABEL BB_SIZE
27
28 BOOTSECTS equ 2
29
30 BBID_DOS macro
31 dc.b 'DOS', 0
32
33 endm
34
35 BBID_KICK macro
36 dc.b 'KICK'
37
38 endm
39
40 BBNAMES_DOS EQU ((('D'<<24)|('O'<<16)|('S'<<8)))
41 BBNAMES_KICK EQU ((('K'<<24)|('I'<<16)|('C'<<8)|('K'))

```

```

1  IFND  DEVICES_CLIPBOARD_I
2  DEVICES_CLIPBOARD_I EQU 1
3  *****
4  * Commodore-Amiga, Inc.
5  * clipboard.i
6  *****
7  *****
8  *
9  * clipboard device command definitions
10 *****
11 *****
12 *****
13 IFND  EXEC_NODES_I
14 INCLUDE "exec/nodes.i"
15 ENDC
16 IFND  EXEC_LISTS_I
17 INCLUDE "exec/lists.i"
18 ENDC
19 IFND  EXEC_PORTS_I
20 INCLUDE "exec/ports.i"
21 ENDC
22 IFND  EXEC_IO_I
23 INCLUDE "exec/io.i"
24 ENDC
25
26 DEVINIT
27
28 DEVCHD CBD_POST
29 DEVCHD CBD_CURRENTREADID
30 DEVCHD CBD_CURRENTWRITEID
31
32 CBERR_OBSOLETEID EQU 1
33
34
35 STRUCTURE ClipboardUnitPartial, 0
36 STRUCT cu_Node, LN_SIZE; ; list of units
37 ULONG cu_UnitNum; ; unit number for this unit
38 ; the remaining unit data is private to the device
39
40
41 STRUCTURE IOClipReq, 0
42 STRUCT io_Message, MN_SIZE ; device node pointer
43 APTR io_Device ; unit (driver private)
44 APTR io_Unit ; device command
45 UMWORD io_Command ; including QUICK and SATISFY
46 BYTE io_Flags ; error or warning num
47 BYTE io_Error ; number of bytes transferred
48 ULONG io_Actual ; number of bytes requested
49 ULONG io_Length ; either clip stream or post port
50 APTR io_Data ; offset in clip stream
51 ULONG io_Offset ; ordinal clip identifier
52 LONG io_ClipID
53 LABEL ioocr_SIZEOF
54
55
56

```

```

57 PRIMARY_CLIP EQU 0 ; primary clip unit
58
59 STRUCTURE SatisfyMsg, 0
60 STRUCT sm_Msg, MN_SIZE ; the length will be 6
61 UMWORD sm_Unit ; which clip unit this is
62 LONG sm_ClipID ; the clip identifier of the post
63 LABEL satisfyMsg_SIZEOF
64
65 ENDC

```

```

1  IFND DEVICES_CONSOLE_I
2  DEVICES_CONSOLE_I SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  * console.i
6  *****
7  *
8  *
9  * Console device command definitions
10 *
11 * Source Control
12 * -----
13 * $Header: console.i,v 1.4 85/11/13 15:13:21 kodiak Exp $
14 *
15 * $Locker:  $
16 *
17 *****
18
19 IFND EXEC_IO_I
20 INCLUDE "exec/10.1"
21 ENDC
22
23 ***** Console commands *****
24 DEVINIT
25
26 DEVCMD CD_ASKKEYMAP
27 DEVCMD CD_SETKEYMAP
28
29
30 ***** SCR parameters
31
32 SCR_PRIMARY EQU 0
33 SCR_BOLD EQU 1
34 SCR_ITALIC EQU 3
35 SCR_UNDERSCORE EQU 4
36 SCR_NEGATIVE EQU 7
37
38 * these names refer to the ANSI standard, not the implementation
39 SCR_BLACK EQU 30
40 SCR_RED EQU 31
41 SCR_GREEN EQU 32
42 SCR_YELLOW EQU 33
43 SCR_BLUE EQU 34
44 SCR_MAGENTA EQU 35
45 SCR_CYAN EQU 36
46 SCR_WHITE EQU 37
47 SCR_DEFAULT EQU 39
48
49 SCR_BLACKBG EQU 40
50 SCR_REDEBG EQU 41
51 SCR_GREENBG EQU 42
52 SCR_YELLOWBG EQU 43
53 SCR_BLUEBG EQU 44
54 SCR_MAGENTABG EQU 45
55 SCR_CYANBG EQU 46
56 SCR_WHITEBG EQU 47

```

```

57 SCR_DEFAULTTBC EQU 49
58
59 * these names refer to the implementation, they are the preferred
60 * names for use with the Amiga console device.
61 SCR_CLR0 EQU 30
62 SCR_CLR1 EQU 31
63 SCR_CLR2 EQU 32
64 SCR_CLR3 EQU 33
65 SCR_CLR4 EQU 34
66 SCR_CLR5 EQU 35
67 SCR_CLR6 EQU 36
68 SCR_CLR7 EQU 37
69
70 SCR_CLR0BG EQU 40
71 SCR_CLR1BG EQU 41
72 SCR_CLR2BG EQU 42
73 SCR_CLR3BG EQU 43
74 SCR_CLR4BG EQU 44
75 SCR_CLR5BG EQU 45
76 SCR_CLR6BG EQU 46
77 SCR_CLR7BG EQU 47
78
79 ***** DSR parameters
80
81 DSR_CPR EQU 6
82
83 ***** CTC parameters
84 CTC_HSETTAB EQU 0
85 CTC_HCLR7TAB EQU 2
86 CTC_HCLR7ABSALL EQU 5
87
88 ***** TBC parameters
89 TBC_HCLR7TAB EQU 0
90 TBC_HCLR7ABSALL EQU 3
91
92 ***** SM and RM parameters
93 M_LNM EQU 20 ; linefeed newline mode
94 M_LASH MACRO ; auto scroll mode
95 M_LASH DC.B '>1'
96 M_LASH ENDM
97 M_LASH MACRO ; auto wrap mode
98 M_LASH DC.B '&77'
99 M_LASH ENDM
100
101 ENDC
102

```

```

1  IFND DEVICES_GAMEPORT_I
2  DEVICES_GAMEPORT_I SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  * gameport.i
6  *****
7  *
8  *
9  * Game Port device command definitions
10 *****
11 *****
12 *****
13 IFND EXEC_IO_I
14 INCLUDE "exec/10.1"
15 ENDC
16 *****
17 *****
18 ***** GamePort commands *****
19 DEVINIT
20 *****
21 DEVCMD GPD_READEVENT
22 DEVCMD GPD_ASKTYPE
23 DEVCMD GPD_SEICTYPE
24 DEVCMD GPD_ASKTRIGGER
25 DEVCMD GPD_SEITRIGGER
26 ***** GamePort structures *****
27 *****
28 * gpt_Keys
29 BITDEF GPT_DOWNKEYS, 0
30 BITDEF GPT_UPKEYS, 1
31 *****
32 STRUCTURE GamePortTrigger, 0
33 UMORD gpt_Keys ;key transition triggers
34 UMORD gpt_Timeout ;time trigger (vertical blank units)
35 UMORD gpt_XDelta ;X distance trigger
36 UMORD gpt_YDelta ;Y distance trigger
37 LABEL gpt_SIZEOF
38 ***** Controller Types *****
39 GPT_ALLOCATED EQU -1 ; allocated by another user
40 GPT_NOCONTROLLER EQU 0
41 *****
42 GPT_MOUSE EQU 1
43 GPT_RELJOYSTICK EQU 2
44 GPT_ABSJOYSTICK EQU 3
45 ***** Errors *****
46 GPDERR_SEICTYPE EQU 1 ; this controller not valid at this time
47 *****
48 *****
49 *****
50 *****
51 *****
52 *****

```

```

1  IFND DEVICES_INPUT_I
2  DEVICES_INPUT_I SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  * input.i
6  *****
7  *
8  *
9  * input device command definitions
10 *****
11 *****
12 *****
13 IFND EXEC_IO_I
14 INCLUDE "exec/10.1"
15 ENDC
16 *****
17 *****
18 ***** GamePort structures *****
19 *****
20 *****
21 *****
22 *****
23 *****
24 *****
25 *****
26 *****
27 *****
28 *****
29 *****
30 *****
31 *****
32 *****
33 *****
34 *****
35 *****
36 *****
37 *****
38 *****
39 *****
40 *****
41 *****
42 *****
43 *****
44 *****
45 *****
46 *****
47 *****
48 *****
49 *****
50 *****
51 *****
52 *****

```

```

1  IFND DEVICES_INPULTEVENT_I
2  DEVICES_INPULTEVENT_I SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  * Inpultevent.i
6  *****
7  *
8  * Input event definitions
9  *
10 *****
11 *****
12 *****
13 IFND DEVICES_TIMER_I
14 INCLUDE "devices/timer.i"
15 ENDC
16
17 *----- constants -----
18
19 * --- InputEvent.ie_Class ---
20 * A NOP input event EQU $00
21 IECLASS_NULL EQU $00
22 * A raw keycode from the keyboard device
23 IECLASS_RAWKEY EQU $01
24 * A raw mouse report from the game port device
25 IECLASS_RAWMOUSE EQU $02
26 * A private console event EQU $03
27 IECLASS_EVENT EQU $03
28 * A Pointer Position report EQU $04
29 IECLASS_POINTERPOS EQU $04
30 * A timer event
31 IECLASS_TIMER EQU $06
32 * select button pressed down over a Gadget (address in ie_EventAddress)
33 IECLASS_GADGETDOWN EQU $07
34 * select button released over the same Gadget (address in ie_EventAddress)
35 IECLASS_GADGETUP EQU $08
36 * some Requester activity has taken place. See Codes REQCLEAR and REQSET
37 IECLASS_REQUESTER EQU $09
38 * this is a Menu Number transmission (Menu number is in ie_Code)
39 IECLASS_MENULIST EQU $0A
40 * User has selected the active Window's Close Gadget
41 IECLASS_CLOSEWINDOW EQU $0B
42 * this Window has a new size
43 IECLASS_SIZEWINDOW EQU $0C
44 * the Window pointed to by ie_EventAddress needs to be refreshed
45 IECLASS_REFRESHWINDOW EQU $0D
46 * new preferences are available
47 IECLASS_NEWPREFS EQU $0E
48 * the disk has been removed
49 IECLASS_DISKREMOVED EQU $0F
50 * the disk has been inserted
51 IECLASS_DISKINSERTED EQU $10
52 * the window is about to be been made active
53 IECLASS_ACTIVEMINDOW EQU $11
54 * the window is about to be made inactive
55 IECLASS_INACTIVEMINDOW EQU $12
56

```

```

57 * the last class EQU $12
58 IECLASS_MAX
59
60 * --- InputEvent.ie_Code ---
61 * IECLASS_RAWKEY EQU $80
62 IECODE_UP_PREFIX EQU 7
63 IECODEB_UP_PREFIX EQU $00
64 IECODE_KEY_CODE_FIRST EQU $77
65 IECODE_KEY_CODE_LAST EQU $78
66 IECODE_COMM_CODE_FIRST EQU $7E
67 IECODE_COMM_CODE_LAST EQU $7F
68
69 * IECLASS_ANSI
70 IECODE_C0_FIRST EQU $00
71 IECODE_C0_LAST EQU $1F
72 IECODE_ASCII_FIRST EQU $20
73 IECODE_ASCII_LAST EQU $7E
74 IECODE_ASCII_DEL EQU $7F
75 IECODE_C1_FIRST EQU $80
76 IECODE_C1_LAST EQU $9F
77 IECODE_LATIN1_FIRST EQU $A0
78 IECODE_LATIN1_LAST EQU $FF
79
80 * IECLASS_RAWMOUSE
81 IECODE_LBUTTON EQU $68 ; also uses IECODE_UP_PREFIX
82 IECODE_RBUTTON EQU $69 ;
83 IECODE_MBUTTON EQU $6A ;
84 IECODE_MOBUTTON EQU $6F
85
86 * IECLASS_EVENT
87 IECODE_NEWACTIVE EQU $01 ; active input window changed
88
89 * IECLASS_REQUESTER Codes
90 * REQSET is broadcast when the first Requester (not subsequent ones) opens
91 * in the Window EQU $01
92 IECODE_REQSET
93 * REQCLEAR is broadcast when the last Requester clears out of the Window
94 IECODE_REQCLEAR EQU $00
95
96
97 * --- InputEvent.ie_Qualifier ---
98 IEQUALIFIER_LSHIFT EQU $0001
99 IEQUALIFIER_RSHIFT EQU 0
100 IEQUALIFIER_RSHIFT EQU $0002
101 IEQUALIFIER_RSHIFT EQU 1
102 IEQUALIFIER_CAPSLOCK EQU $0004
103 IEQUALIFIER_CAPSLOCK EQU 2
104 IEQUALIFIER_CONTROL EQU $0008
105 IEQUALIFIER_CONTROL EQU 3
106 IEQUALIFIER_LALT EQU $0010
107 IEQUALIFIER_LALT EQU 4
108 IEQUALIFIER_RALT EQU $0020
109 IEQUALIFIER_RALT EQU 5
110 IEQUALIFIER_JCOMMAND EQU $0040
111 IEQUALIFIER_JCOMMAND EQU 6
112 IEQUALIFIER_ROOMMAND EQU $0080

```



```

113 IEQUALIFIERB_COMMAND EQU $0100
114 IEQUALIFIERB_NUMERICPAD EQU $0200
115 IEQUALIFIERB_NUMERICPAD EQU $0200
116 IEQUALIFIERB_REPEAT EQU $0400
117 IEQUALIFIERB_REPEAT EQU $0400
118 IEQUALIFIERB_INTERRUPT EQU $0800
119 IEQUALIFIERB_INTERRUPT EQU $0800
120 IEQUALIFIERB_MULTIBROADCAST EQU $1000
121 IEQUALIFIERB_MULTIBROADCAST EQU $1000
122 IEQUALIFIERB_LEUTTON EQU $2000
123 IEQUALIFIERB_LEUTTON EQU $2000
124 IEQUALIFIERB_REUTTON EQU $4000
125 IEQUALIFIERB_REUTTON EQU $4000
126 IEQUALIFIERB_MBUTTON EQU $8000
127 IEQUALIFIERB_MBUTTON EQU $8000
128 IEQUALIFIERB_RELATIVEHOUSE EQU $8000
129 IEQUALIFIERB_RELATIVEHOUSE EQU $8000

```

----- InputEvent -----

STRUCTURE InputEvent, 0

```

130 APTR ie_NextEvent
131 UBYTE ie_Class
132 UBYTE ie_SubClass
133 UWORD ie_Code
134 UWORD ie_Qualifier
135 WORD ie_X
136 WORD ie_Y
137 STRUCT ie_TimeStamp, TV_SIZE
138 LABEL ie_SIZEOF
139
140
141
142
143
144
145 ENDC

```

; the chronologically next event
; the input event class
; optional subclass of the class
; the input event code
; qualifiers in effect for the event
; a pointer parameter for an event
; the pointer position for the event,
; usually in canvas relative coords
; usually in canvas relative coords
; the system tick at the event

```

1 IFND DEVICES_KEYBOARD_I
2 DEVICES_KEYBOARD_I SET 1
3 *****
4 * Commodore-Amiga, Inc.
5 * keyboard.i
6 *****
7 *
8 * Keyboard device command definitions
9 *
10 *****
11 *****
12 *****
13 IFND EXEC_IO_I
14 INCLUDE "exec/io.i"
15 ENDC
16
17 DEVINIT
18
19 DEVCMD KBD_READEVENT
20 DEVCMD KBD_READMATRIX
21 DEVCMD KBD_ADRESETHANDLER
22 DEVCMD KBD_REMRESETHANDLER
23 DEVCMD KBD_RESETHANDLERDONE
24
25 ENDC

```

```

1  IFND DEVICES_KEYMAP_I
2  DEVICES_KEYMAP_I SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  * keymap.i
6  *
7  *
8  *
9  * console.device key map definitions
10 *
11 *****
12 *****
13 STRUCTURE KeyMap_0
14 APTR km_LoKeyMapTypes
15 APTR km_LoKeyMap
16 APTR km_LoCapsable
17 APTR km_LoRepeatable
18 APTR km_HLKeyMapTypes
19 APTR km_HLKeyMap
20 APTR km_HLCapsable
21 APTR km_HLRepeatable
22 LABEL km_SIZEOF
23
24
25 KCB_NOP EQU 7
26 KCE_NOP EQU $80
27
28 KC_NOEQUAL EQU 0
29 KC_VANILLA EQU 7
30 KCF_SHIFT EQU $01
31 KCE_ALT EQU $02
32 KCB_CONTROL EQU 2
33 KCF_DOWNUP EQU $04
34 KCB_DOWNUP EQU 3
35 KCE_DOWNUP EQU $08
36
37 KCB_STRING EQU 6
38 KCE_STRING EQU $40
39
40 ENDC

```

: note that SHIFT+ALT+CTRL is VANILLA

```

1  IFND DEVICES_NARRATOR_I
2  DEVICES_NARRATOR_I SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  * narrator.i
6  *
7  *
8  * EXEC_IO_I
9  INCLUDE "exec/io.i"
10 ENDC
11
12 ***** DEFAULT VALUES, USER PARMS, AND GENERAL CONSTANTS
13
14 DEFPITCH EQU 110 ;DEFAULT PITCH
15 DEFRRATE EQU 150 ;DEFAULT RATE
16 DEFVOL EQU 64 ;DEFAULT VOLUME (FULL)
17 DEFFREQ EQU 22200 ;DEFAULT SAMPLING FREQUENCY
18 NATURALF0 EQU 0 ;NATURAL F0 CONTOURS
19 ROBOTICE0 EQU 1 ;MONOTONE F0
20 MALE EQU 0 ;MALE SPEAKER
21 FEMALE EQU 1 ;FEMALE SPEAKER
22 DEFSEX EQU MALE ;DEFAULT SEX
23 DEFMODE EQU NATURALF0 ;DEFAULT MODE
24
25 * Parameter bounds
26
27 MINRATE EQU 40 ;MINIMUM SPEAKING RATE
28 MAXRATE EQU 400 ;MAXIMUM SPEAKING RATE
29 MINPITCH EQU 65 ;MINIMUM PITCH
30 MAXPITCH EQU 320 ;MAXIMUM PITCH
31 MINREQ EQU 5000 ;MINIMUM SAMPLING FREQUENCY
32 MAXREQ EQU 28000 ;MAXIMUM SAMPLING FREQUENCY
33 MINVOL EQU 0 ;MINIMUM VOLUME
34 MAXVOL EQU 64 ;MAXIMUM VOLUME

```

* Driver error codes

```

37 ND_NotUsed EQU -1
38 ND_NoMem EQU -2
39 ND_NoAudLib EQU -3
40 ND_MakeBad EQU -4
41 ND_UnitErr EQU -5
42 ND_CantAlloc EQU -6
43 ND_Unimpl EQU -7
44 ND_NoWrite EQU -8
45 ND_Expunged EQU -9
46 ND_PhonErr EQU -20
47 ND_RateErr EQU -21
48 ND_PitchErr EQU -22
49 ND_SexErr EQU -23
50 ND_ModeErr EQU -24
51 ND_FreqErr EQU -25
52 ND_VolErr EQU -26
53
54
55
56

```

```

; Can't allocate memory
; Can't open audio device
; Error in MakeLibrary call
; Unit other than 0
; Can't allocate the audio channel
; Unimplemented command
; Read for mouth shape without write
; Can't open, deferred expunges bit set
; Phoneme code spelling error
; Rate out of bounds
; Pitch out of bounds
; Sex not valid
; Mode not valid
; Sampling freq out of bounds
; Volume out of bounds

```

```

57 * :----- Write IOrequest block
58 STRUCTURE NDI_IOCSTD_SIZE
59 UMORD NDI_RATE
60 UMORD NDI_PITCH
61 UMORD NDI_MODE
62 UMORD NDI_SEX
63 APTR NDI_CHMASKS
64 UMORD NDI_NUMMASKS
65 UMORD NDI_VOLUME
66 UMORD NDI_SAMPFREQ
67 UBYTE NDI_MOUTHS
68 UBYTE NDI_CHANMASK
69 UBYTE NDI_NUMCHAN
70 UBYTE NDI_PAD
71 LABEL NDI_SIZE
72
73
74 * :----- Mouth read IOCB
75 STRUCTURE MRB_NDI_SIZE
76 UBYTE MRB_WIDTH
77 UBYTE MRB_HEIGHT
78 UBYTE MRB_SHAPE
79 UBYTE MRB_PAD
80 LABEL MRB_SIZE
81
82
83
84

```

```

:Speaking rate in words/minute
:Baseline pitch in Hertz
:F0 mode
:Speaker sex
:Pointer to audio channel masks
:Size of channel masks array
:Channel volume
:Sampling frequency
:Generate mouths? (Boolean value)
:Actual channel mask used (internal use)
:Number of channels used (internal use)
:For alignment
:Size of Narrator IOrequest block

```

```

:Mouth width
:Mouth height
:Compressed shape (height/width)
:Alignment

```

ENDC

```

1 *****
2 * Commodore-Amiga, Inc.
3 * parallel.i
4 *****
5 *****
6 *
7 * external declarations for Parallel Port Driver
8 *
9 * SOURCE CONTROL
10 * -----
11 * $Header: parallel.i,v 25.0 85/03/27 19:14:15 temp Exp $
12 *
13 * $Locker: $
14 *
15 *****
16 IFND DEVICES_PARALLEL_I
17 DEVICES_PARALLEL_I SET 1
18
19 IFND EXEC_STRINGS_I
20 include 'exec/strings.i'
21 ENDC
22 EXEC_STRINGS_I
23
24 IFND EXEC_IO_I
25 include 'exec/io.i'
26 ENDC
27 EXEC_IO_I
28
29 *
30 * Driver error definitions
31 *
32 *
33
34 ParErr_DevBusy EQU 1
35 ParErr_BufTooBig EQU 2
36 ParErr_InvParam EQU 3
37 ParErr_LineErr EQU 4
38 ParErr_NotOpen EQU 5
39 ParErr_PortReset EQU 6
40 ParErr_InitErr EQU 7
41
42 *
43 *
44 * Useful constants
45 *
46 *
47 *
48 PDCMD_QUERY EQU CMD_MONSTD
49 PDCMD_SETPARAMS EQU CMD_MONSTD+1
50 Par_DEVFISH EQU 10 ; number of device commands
51 *
52 *
53 *
54 * Driver Specific Commands
55 *
56 *

```

```

57 *--- PARALLELNAME is a generic macro to get the name of the driver. This
58 *--- way if the name is ever changed you will pick up the change. This
59 *--- automatically.
60 *---
61 *--- Normal usage would be:
62 *---
63 *--- internalName: PARALLELNAME
64 *---
65 *---
66 PARALLELNAME: MACRO
67 STRING 'parallel.device'
68 ENDM
69
70
71 BITDEF PAR_SHARED,5 ; PARFLACS non-exclusive access
72 BITDEF PAR_RAD_BOOGIE,3 ; " (not yet implemented)
73 BITDEF PAR_EOFMODE,1 ; " EOF mode enabled bit
74 BITDEF IOEPAR_QUEUED,6 ; IO_FLACS rqst-queued bit
75 BITDEF IOEPAR_ABORT,5 ; " rqst-aborted bit
76 BITDEF IOEPAR_ACTIVE,4 ; " rqst-qed-or-current bit
77 BITDEF IOET_RNDIR,3 ; IO_STATUS read=0,write=1
78 BITDEF IOET_BUSY,2 ; " printer in busy toggle
79 BITDEF IOET_PAPEROUT,1 ; " paper out
80 BITDEF IOET_FSEL,0 ; " printer selected
81 *
82 *
83 *****
84
85 STRUCTURE PTERMARRAY,0
86 ULONG PTERMARRAY_0
87 ULONG PTERMARRAY_1
88 LABEL PTERMARRAY_SIZE
89
90 *****
91 * CAUTION !!! IF YOU ACCESS the parallel.device, you MUST (!!!!) use an
92 * IOEXIPAR-sized structure or you may overlay innocent memory, okay ?!
93 *****
94
95 STRUCTURE IOEXIPAR,IOSTD_SIZE
96
97 * STRUCT MsgNode
98 * 0 APTR Succ
99 * 4 APTR Pred
100 * 8 UBYTE Type
101 * 9 UBYTE Pr1
102 * A APTR Name
103 * E APTR ReplyPort
104 * 12 UWORD MNLlength
105 * STRUCT IOExt
106 * 14 APTR IO_DEVICE
107 * 18 APTR IO_UNIT
108 * 1C UWORD IO_COMMAND
109 * 1E UBYTE IO_FLAGS
110 * 1F UBYTE IO_ERROR
111 * STRUCT IOStdExt
112 * 20 ULONG IO_ACTUAL

```

```

113 * 24 ULONG IO_LENGTH
114 * 28 APTR IO_DATA
115 * 2C ULONG IO_OFFSET
116 *
117 *
118 * 30 ULONG IO_PEXTFLACS ; (not used) flag extension area
119 UBYTE IO_PARSTATUS ; device status (see bit defs above)
120 UBYTE IO_PARFLACS ; see PARFLACS bit definitions above
121 STRUCT IO_PTERMARRAY,PTERMARRAY_SIZE ; termination char array
122 LABEL IOEXIPar_SIZE
123
124
125 ENDC IDEVICES_PARALLEL_1
126

```

```

1  IFND DEVICES_PRINTER_I
2  DEVICES_PRINTER_I EQU 1
3  *****
4  * Commodore-Amiga, Inc.
5  * printer.i
6  * *****
7  * *****
8  * printer device command definitions
9  *
10 *
11 * Source Control
12 * -----
13 * $Header: printer.i,v 1.2 85/10/09 16:16:27 kodlak Exp $
14 *
15 * $Locker: $
16 *
17 * *****
18 *
19 IFND EXEC_NODES_I
20 INCLUDE "exec/nodes.i"
21 ENDC
22
23 IFND EXEC_LISTS_I
24 INCLUDE "exec/lists.i"
25 ENDC
26
27 IFND EXEC_PORTS_I
28 INCLUDE "exec/ports.i"
29 ENDC
30
31 IFND EXEC_IO_I
32 INCLUDE "exec/io.i"
33 ENDC
34
35 DEVINIT
36
37 DEVCHD PRD_RAMWRITE
38 DEVCHD PRD_PROTOHAND
39 DEVCHD PRD_DUMPFORMAT
40
41 ;***** printer definitions
42 aRIS EQU 0 ; ESCc reset
43 aRIN EQU 1 ; ESC#1 initialize
44 aIND EQU 2 ; ESCD lf
45 aNEL EQU 3 ; ESCc return,lf
46 aRI EQU 4 ; ESCM reverse lf
47
48 aSCR0 EQU 5 ; ESC[0m normal char set
49 aSCR3 EQU 6 ; ESC[3m italics on
50 aSCR23 EQU 7 ; ESC[23m italics off
51 aSCR4 EQU 8 ; ESC[4m underline on
52 aSCR24 EQU 9 ; ESC[24m underline off
53 aSCR1 EQU 10 ; ESC[1m boldface on
54 aSCR22 EQU 11 ; ESC[22m boldface off
55 aSFC EQU 12 ; SCR30-39 set foreground color
56 aSBC EQU 13 ; SCR40-49 set background color

```

```

57 aSHORP0 EQU 14 ; ESC[0v normal pitch
58 aSHORP1 EQU 15 ; ESC[2v elite on
59 aSHORP2 EQU 16 ; ESC[2v elite off
60 aSHORP3 EQU 17 ; ESC[4v condensed line on
61 aSHORP4 EQU 18 ; ESC[4v condensed line off
62 aSHORP5 EQU 19 ; ESC[3v condensed off
63 aSHORP6 EQU 20 ; ESC[6v enlarged on
64 aSHORP7 EQU 21 ; ESC[6v enlarged off
65
66 aDENG EQU 21 ; ESC[6"z shadow print on
67 aDENS EQU 22 ; ESC[5"z shadow print off
68 aDENA EQU 23 ; ESC[4"z doublestrike on
69 aDEN3 EQU 24 ; ESC[3"z doublestrike off
70 aDEN2 EQU 25 ; ESC[2"z NLQ on
71 aDEN1 EQU 26 ; ESC[1"z NLQ off
72
73 aSUS2 EQU 27 ; ESC[2v superscript on
74 aSUS1 EQU 28 ; ESC[1v superscript off
75 aSUS4 EQU 29 ; ESC[4v subscript on
76 aSUS3 EQU 30 ; ESC[3v subscript off
77 aSUS0 EQU 31 ; ESC[0v normalize the line
78 aPLU EQU 32 ; ESCI partial line up
79 aPLD EQU 33 ; ESCC partial line down
80
81 aFNT0 EQU 34 ; ESC(B US char set
82 aFNT1 EQU 35 ; ESC(R French char set
83 aFNT2 EQU 36 ; ESC(K German char set
84 aFNT3 EQU 37 ; ESC(A UK char set
85 aFNT4 EQU 38 ; ESC(E Danish I char set
86 aFNT5 EQU 39 ; ESC(H Sweden char set
87 aFNT6 EQU 40 ; ESC(Y Italian char set
88 aFNT7 EQU 41 ; ESC(Z Spanish char set
89 aFNT8 EQU 42 ; ESC(J Japanese char set
90 aFNT9 EQU 43 ; ESC(6 Norwegian char set
91 aFNT10 EQU 44 ; ESC(C Danish II char set
92
93 aPROP2 EQU 45 ; ESC[2p proportional on
94 aPROP1 EQU 46 ; ESC[1p proportional off
95 aPROPO EQU 47 ; ESC[0p proportional clear
96 aTSS EQU 48 ; ESC[n E set proportional offset
97 aJFY5 EQU 49 ; ESC[5 F auto left justify
98 aJFY7 EQU 50 ; ESC[7 F auto right justify
99 aJFY6 EQU 51 ; ESC[6 F auto full justify
100 aJFY0 EQU 52 ; ESC[0 F auto justify off
101 aJFY2 EQU 53 ; ESC[2 F word space(auto center)
102 aJFY3 EQU 54 ; ESC[3 F letter space (justify)
103
104 aVERP0 EQU 55 ; ESC[0z 1/8" line spacing
105 aVERP1 EQU 56 ; ESC[1z 1/6" line spacing
106 aSLPP EQU 57 ; ESC[nt set form length n
107 aPERF EQU 58 ; ESC[nq perf skip n (n>0)
108 aPERF0 EQU 59 ; ESC[0q perf skip off
109
110 aLMS EQU 60 ; ESC#9 Left margin set
111 aRMS EQU 61 ; ESC#0 Right margin set
112 aTMS EQU 62 ; ESC#8 Top margin set

```

(special)
(special)

```

113 aBWS EQU 63 ; ESC#2 Bottom marg set
114 aSTEM EQU 64 ; ESC[Pn];Pn2r T&B margins
115 aSLRM EQU 65 ; ESC[Pn];PnZs L&R margin
116 aCAM EQU 66 ; ESC#3 Clear margins
117
118 aHTS EQU 67 ; ESCH Set horiz tab
119 aVTS EQU 68 ; ESCJ Set vertical tabs
120 aTBC0 EQU 69 ; ESC[0g Clr horiz tab
121 aTBC3 EQU 70 ; ESC[3g Clear all h tab
122 aTBC1 EQU 71 ; ESC[1g Clr vertical tabs
123 aTBC4 EQU 72 ; ESC[4g Clr all v tabs
124 aTECALL EQU 73 ; ESC#4 Clr all h & v tabs
125 aTESALL EQU 74 ; ESC#5 Set default tabs
126 aEXTEND EQU 75 ; ESC[Pn"x extended commands
127
128

```

```

169 PDERR_BUFFERMEMORY EQU 7 ; no memory for print buffer
170
171 ENDC

```

```

STRUCTURE IO_PrtCmdReq, IO_SIZE
129 UNORD io_PrtCommand ; printer command
130 UBYTE io_Parm0 ; first command parameter
131 UBYTE io_Parm1 ; second command parameter
132 UBYTE io_Parm2 ; thldr command parameter
133 UBYTE io_Parm3 ; fourth command parameter
134 LABEL ioper_SIZEOF
135
136

```

```

STRUCTURE IO_DRPReq, IO_SIZE
137 APTR io_RastPort ; raster port
138 APTR io_ColorMap ; color map
139 ULONG io_Modes ; graphics viewport modes
140 UNORD io_SrcX ; source x origin
141 UNORD io_SrcY ; source y origin
142 UNORD io_SrcWidth ; source x width
143 UNORD io_SrcHeight ; source x height
144 LONG io_DestCols ; destination x width
145 UNORD io_DestRows ; destination y height
146 UNORD io_Special ; option flags
147 LABEL iodrpr_SIZEOF
148
149

```

```

150 SPECIAL_MILCOLS EQU $01 ; DestCols specified in 1/1000"
151 SPECIAL_FULLCOLS EQU $02 ; DestRows specified in 1/1000"
152 SPECIAL_FRACCOLS EQU $04 ; make DestCols maximum possible
153 SPECIAL_FRACROWS EQU $08 ; make DestRows maximum possible
154 SPECIAL_FRACMASK EQU $10 ; DestCols is fraction of FULLCOLS
155 SPECIAL_DENSITY1 EQU $20 ; DestRows is fraction of FULLROWS
156 SPECIAL_DENSITY2 EQU $80 ; ensure correct aspect ratio
157 SPECIAL_DENSITY3 EQU $100 ; masks out density bits
158 SPECIAL_DENSITY4 EQU $200 ; lowest res
159 SPECIAL_DENSITY5 EQU $400 ; next res
160 SPECIAL_DENSITY6 EQU $800 ; next res
161 SPECIAL_DENSITY7 EQU $1600 ; highest res
162
163 PDERR_CANCEL EQU 1 ; user canceled a printer timeout
164 PDERR_NOTRAPHICS EQU 2 ; printer cannot output graphics
165 PDERR_INVERTHAM EQU 3 ; cannot invert hold & modify print
166 PDERR_BADDIMENSION EQU 4 ; print dimensions illegal
167 PDERR_DIMENSIONFLOW EQU 5 ; print dimensions too large
168 PDERR_INTERNALMEMORY EQU 6 ; no memory for internal variables

```

```

1 *****
2 * Commodore-Amiga, Inc.
3 * prtbase.1
4 *****
5 *
6 * printer device data definition
7 *
8 *
9 *****
10 IFND DEVICES_PRTBASE_I
11 DEVICES_PRTBASE_I EQU 1
12
13 IFND EXEC_NODES_I
14 INCLUDE "exec/nodes.i"
15 ENDC
16 IFND EXEC_LISTS_I
17 INCLUDE "exec/lists.i"
18 ENDC
19 IFND EXEC_PORTS_I
20 INCLUDE "exec/ports.i"
21 ENDC
22 IFND EXEC_LIBRARIES_I
23 INCLUDE "exec/libraries.i"
24 ENDC
25 IFND EXEC_TASKS_I
26 INCLUDE "exec/tasks.i"
27 ENDC
28
29 IFND DEVICES_PARALLEL_I
30 INCLUDE "devices/parallel.i"
31 ENDC
32 IFND DEVICES_SERIAL_I
33 INCLUDE "devices/serial.i"
34 ENDC
35 IFND DEVICES_TIMER_I
36 INCLUDE "devices/timer.i"
37 ENDC
38 IFND LIBRARIES_DOSEXTENS_I
39 INCLUDE "libraries/dosexpens.i"
40 ENDC
41 IFND INTUITION_INTUITION_I
42 INCLUDE "intuition/intuition.i"
43 ENDC
44
45 STRUCTURE DeviceData,LIB_SIZE
46 APTR dd_Segment ; A0 when initialized
47 APTR dd_ExecBase ; A6 for exec
48 APTR dd_EndVectors ; command table for device commands
49 APTR dd_OndBytes ; bytes describing which command queue
50 UMORDD dd_NumCommands ; the number of commands supported
51 LABEL dd_SIZEOF
52
53
54
55
56 *-----

```

```

57 *----- device driver private variables -----
58 *-----
59 du_Flags EQU LN_PRI ; various unit flags
60
61 *----- IO_FLAGS
62 BITDEF IO_QUEUED,4 ; command is queued to be performed
63 BITDEF IO_CURRENT,5 ; command is being performed
64 BITDEF IO_SERVICING,6 ; command is being actively performed
65 BITDEF IO_DONE,7 ; command is done
66
67 *----- du_Flags
68 BITDEF DU_STOPPED,0 ; commands are not to be performed
69
70
71 *----- Constants -----
72 P_PRIORITY EQU 0
73 P_STKSIZE EQU $800
74
75 *----- pd_Flags -----
76 BITDEF P_IOR0,0 ; IOR0 is in use
77 BITDEF P_IOR1,1 ; IOR1 is in use
78 BITDEF P_EXPUNCHED,7 ; device to be expunged when all closed
79
80 STRUCTURE PrinterData,dd_SIZEOF
81 STRUCT pd_Unit,MP_SIZE ; the one and only unit
82 EPTR pd_PrinterSegment ; the printer specific segment
83 UMORDD pd_PrINTERType ; the segment printer type
84 APTR pd_SegmentData ; the segment data structure
85 APTR pd_PrintBuf ; the raster print buffer
86 APTR pd_PWrite ; the parallel write function
87 APTR pd_PBothReady ; the parallel write function's done
88
89 IFGT IOEXTPar_SIZE-IOEXTPar_SIZE
90 STRUCT pd_IOR0,IOEXTPar_SIZE ; port I/O request 0
91 STRUCT pd_IOR1,IOEXTPar_SIZE ; and 1 for double buffering
92 ENDC
93
94 IFLE IOEXTPar_SIZE-IOEXTPar_SIZE
95 STRUCT pd_IOR0,IOEXTPar_SIZE ; port I/O request 0
96 STRUCT pd_IOR1,IOEXTPar_SIZE ; and 1 for double buffering
97 ENDC
98
99 STRUCT pd_TIMER,IOIV_SIZE ; timer I/O request
100 STRUCT pd_IORPort,MP_SIZE ; and message reply port
101 STRUCT pd_TC,TC_SIZE ; write task
102 STRUCT pd_Sk,P_STKSIZE ; and stack space
103 UBYTE pd_Flags ; device flags
104 UBYTE pd_Pad ;
105 STRUCT pd_Preferences,pf_SIZEOF ; the latest preferences
106 UBYTE pd_PWaitEnabled ; wait function switch
107 LABEL pd_SIZEOF ; warning! this may be odd
108
109 BITDEF PPC_CFX,0
110 BITDEF PPC_COLOR,1
111
112 PPC_BWALPHA EQU 0

```

```

113 PPC_BMGEX EQU 1
114 PPC_COLARGEX EQU 3
115
116 FCC_BW EQU 1
117 FCC_YMC EQU 2
118 FCC_YMC_BW EQU 3
119 FCC_YMCB EQU 4
120
121 STRUCTURE PrinterExtendedData, 0
122 APTR ped_PrinterName ; printer name, null terminated
123 APTR ped_Init ; called after LoadSeg
124 APTR ped_Expunge ; called before UnLoadSeg
125 APTR ped_Open ; called at OpenDevice
126 APTR ped_Close ; called at CloseDevice
127 UBYTE ped_PrinterClass ; printer class
128 UBYTE ped_ColorClass ; color class
129 UBYTE ped_MaxColumns ; number of print columns available
130 UBYTE ped_NumRows ; number of character sets
131 UWORD ped_NumCharSets ; number of raster rows in a raster dump
132 ULONG ped_MaxXDots ; number of dots maximum in a raster dump
133 ULONG ped_MaxYDots ; number of dots maximum in a raster dump
134 UWORD ped_YDotsInch ; horizontal dot density
135 UWORD ped_YDotsInch ; vertical dot density
136 APTR ped_Commands ; printer text command table
137 APTR ped_DoSpecial ; special command handler
138 APTR ped_Render ; raster render function
139 LONG ped_TimeoutSecs ; good write timeout
140 LABEL ped_SIZEOF
141
142 STRUCTURE PrinterSegment, 0
143 ULONG ps_NextSegment ; (actually a BPTR)
144 ULONG ps_RunAlert ; MOVEQ #0,D0 ; RTS
145 UWORD ps_Version ; segment revision
146 UWORD ps_Revision ; segment revision
147 LABEL ps_PED ; printer extended data
148
149 ENDC

```

```

1 *****
2 * Commodore-Amiga, Inc.
3 * serial.i
4 *****
5 *****
6 *
7 * external declarations for Serial Port Driver
8 *
9 * SOURCE CONTROL
10 * -----
11 * $Header: serial.i,v 25.0 85/03/27 19:14:15 temp Exp $
12 *
13 * $Locker: $
14 *
15 *****
16 IFND DEVICES_SERIAL_I
17 DEVICES_SERIAL_I SET 1
18
19 IFND EXEC_STRINGS_I
20 include 'exec/strings.i'
21 ENDC !EXEC_STRINGS_I
22
23 IFND EXEC_IO_I
24 include 'exec/io.i'
25 ENDC !EXEC_IO_I
26
27 *
28 * -----
29 *
30 * Useful constants
31 *
32 * -----
33 *
34 SER_CTL EQU $11130000 ; default char's for xON,xoff,reserved,rsvd.
35 SER_DBAUD EQU 9600 ; default baud
36 *
37 * -----
38 *
39 *
40 * Driver Specific Commands
41 *
42 SDCMD_QUERY EQU CMD_NONSTD
43 SDCMD_BREAK EQU CMD_NONSTD+1
44 SDCMD_SETPARANS EQU CMD_NONSTD+2
45
46 SER_DEVFINISH EQU CMD_NONSTD+2 ; number of device commands
47 *
48 * -----
49 *
50 --- SERIALNAME is a generic macro to get the name of the driver. This
51 --- way if the name is ever changed you will pick up the change
52 --- automatically.
53 ---
54 --- Normal usage would be:
55 --- InternalName: SERIALNAME
56 ---

```



```

1 *****
2 * Commodore-Amiga, Inc.
3 * timer.i
4 *****
5 *
6 * SOURCE CONTROL
7 * -----
8 *
9 * $Header: timer.i,v 27.1 85/06/24 13:32:40 neall Exp $
10 *
11 * $Locker: $
12 *
13 *****
14
15 IEND DEVICES_TIMER_I
16 DEVICES_TIMER_I SET 1
17
18 IEND EXEC_IO_I
19 INCLUDE "exec/io.i"
20 ENDC EXEC_IO_I
21
22 * unit definitions
23 UNIT_MICROHZ EQU 0
24 UNIT_VBLANK EQU 1
25
26 TIMERNAME MACRO
27 DC.B 'timer.device',0
28 DS.W 0
29 ENDM
30
31 STRUCTURE TIMEVAL,0
32 ULONG TV_SECS
33 ULONG TV_MICRO
34 LABEL TV_SIZE
35
36 STRUCTURE TIMEREQUEST,IO_SIZE
37 STRICT IO_TV_TIME,TV_SIZE
38 LABEL IO_TV_SIZE
39
40 * IO COMMAND to use for adding a timer
41 DEVINIT
42 DEVCMD TR_ADDRESS
43 DEVCMD TR_GETSYSTEMTIME
44 DEVCMD TR_SETSYSTEMTIME
45
46 END

```

```

1 *****
2 * Commodore-Amiga, Inc.
3 * trackdisk.i
4 *****
5 *
6 * trackdisk.i
7 *
8 * Source Control
9 * -----
10 *
11 *
12 * $Header: trackdisk.i,v 27.2 85/07/12 23:16:27 neall Exp $
13 *
14 * $Locker: $
15 *
16 *****
17
18 IEND DEVICES_TRACKDISK_I
19 DEVICES_TRACKDISK_I SET 1
20
21 IEND EXEC_IO_I
22 INCLUDE "exec/io.i"
23 ENDC EXEC_IO_I
24
25 -----
26 * Physical drive constants
27 *
28 *
29 *
30
31 NUMCYLS EQU 80 ; normal # of cylinders
32 MAXCYLS EQU NUMCYLS+20 ; max # of cyls to look for
33
34 NUMSECS EQU 11 ; during a calibrate
35
36 NUMHEADS EQU 2
37 MAXRETRY EQU 10
38 NUMTRACKS EQU NUMCYLS*NUMHEADS
39 NUMUNITS EQU 4
40
41
42 * Useful constants
43 *
44 *
45 *
46
47
48 *-- sizes before mfm encoding
49 TD_SECTOR EQU 512 ; log TD_SECTOR
50
51 TD_SECSHIFT EQU 9 ;
52
53
54 *
55 * Driver Specific Commands
56 *

```

```

113 *
114 *
115 *
116 * Driver error defines
117 *
118 *
119
120 TDERR_NotSpecified EQU 20
121 TDERR_NoSecHdr EQU 21
122 TDERR_BadSecPreamble EQU 22
123 TDERR_BadSecID EQU 23
124 TDERR_BadHdrSum EQU 24
125 TDERR_BadSecSum EQU 25
126 TDERR_TooFewSecs EQU 26
127 TDERR_BadSecHdr EQU 27
128 TDERR_WriteProt EQU 28
129 TDERR_DiskChanged EQU 29
130 TDERR_SeekError EQU 30
131 TDERR_NoMan EQU 31
132 TDERR_BadUnitNum EQU 32
133 TDERR_BadDriveType EQU 33
134 TDERR_DriveInUse EQU 34
135
136 ENDC DEVICE_TRACKDISK_I

```

```

57 *
58 *
59 *
60 *-- TD_NAME is a generic macro to get the name of the driver. This
61 *-- way if the name is ever changed you will pick up the change
62 *-- automatically.
63 *
64 *-- Normal usage would be:
65 *--
66 *-- internalName: TD_NAME
67 *--
68
69 TD_NAME: MACRO
70 DC.B 'trackdisk.device',0
71 DS.W 0
72 ENDM
73
74 BITDEF TD,EXTCOM,15
75
76 DEVINIT
77 DEVCMD TD_MOTOR ; control the disk's motor
78 DEVCMD TD_SEEK ; explicit seek (for testing)
79 DEVCMD TD_FORMAT ; format disk
80 DEVCMD TD_REMOVE ; notify when disk changes
81 DEVCMD TD_CHANGENUM ; number of disk changes
82 DEVCMD TD_CHANGESTATE ; is there a disk in the drive?
83 DEVCMD TD_PROTSTATUS ; is the disk write protected?
84
85 TD_LASTCOM EQU TD_PROTSTATUS
86 *
87 *
88 * The disk driver has an "extended command" facility. These commands
89 * take a superset of the normal IO Request block.
90 *
91 *
92 ETD_WRITE EQU (CMD_WRITE|TDE_EXTCOM)
93 ETD_READ EQU (CMD_READ|TDE_EXTCOM)
94 ETD_MOTOR EQU (TD_MOTOR|TDE_EXTCOM)
95 ETD_SEEK EQU (TD_SEEK|TDE_EXTCOM)
96 ETD_FORMAT EQU (TD_FORMAT|TDE_EXTCOM)
97 ETD_UPDATE EQU (CMD_UPDATE|TDE_EXTCOM)
98 ETD_CLEAR EQU (CMD_CLEAR|TDE_EXTCOM)
99
100 *
101 *
102 * extended IO has a larger than normal io request block.
103 *
104
105 STRUCTURE IOEXTD,IOSTD_SIZE ; removal/insertion count
106 ULONG IOTD_COUNT ; sector label data region
107 ULONG IOTD_SECLABEL ; sector label data region
108 LABEL IOTD_SIZE
109
110 * labels are TD_LABELSIZE bytes per sector
111 TD_LABELSIZE EQU 16
112

```

```

1  IFND EXEC_ABLES_I
2  EXEC_ABLES_I SET 1
3  *****
4  *
5  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6  *
7  *****
8  *
9  * Source Control:
10 *
11 * $Header: ables.i.v 1.0 85/08/28 15:05:30 carl Exp $
12 *
13 * $Locker:  $
14 *
15 *****
16
17 IFND EXEC_TYPES_I
18 INCLUDE "exec/types.i"
19 ENDC |EXEC_TYPES_I
20
21 IFND EXEC_EXECLBASE_I
22 INCLUDE "exec/execlbase.i"
23 ENDC |EXEC_EXECLBASE_I
24
25
26 -----
27 *
28 * Interrupt Exclusion Macros
29 *
30 -----
31
32 INT_ABLES MACRO * [scratchReg]
33 XREF _intena
34 ENDM
35
36
37 DISABLE MACRO * [scratchReg]
38 IFC '\1', '[scratchReg]
39 MOVE.W #$04000, _intena
40 ADDQ.B #1, IDNestCnt (A6)
41 ENDC
42 IFNC '\1', ''
43 MOVE.L 4, \1
44 MOVE.W #$04000, _intena
45 ADDQ.B #1, IDNestCnt (\1)
46 ENDC
47 ENDM
48
49
50 ENABLE IFC '\1', '[scratchReg]
51 SUBQ.B #1, IDNestCnt (A6)
52 BGE.S ENABLE\@
53 MOVE.W #90C000, _intena
54 ENABLE\@:
55 *IF_SETCLR+IF_INTEN
56 ENDC

```

```

57 IFNC '\1', ''
58 MOVE.L 4, \1
59 SUBQ.B #1, IDNestCnt (\1)
60 BGE.S ENABLE\@
61 MOVE.W #90C000, _intena
62 ENABLE\@:
63 ENDC
64 ENDM
65
66 -----
67 *
68 * Tasking Exclusion Macros
69 *
70 *
71 -----
72
73 TASK_ABLES MACRO
74 * INCLUDE 'execbase.i' for TDNestCnt offset
75 XREF _JVOPermit
76 ENDM
77
78
79 FORBID MACRO
80 ADDQ.B #1, IDNestCnt (A6)
81 ENDM
82
83
84 PERMIT MACRO
85 JSR _JVOPermit (A6)
86 ENDM
87
88 ENDC |EXEC_ABLES_I

```

```

1 IFND EXEC_ALERTS_I
2 EXEC_ALERTS_I SET 1
3 *****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *
8 *
9 * Source Control:
10 *
11 * $Header: alerts.1.v 1.0 85/08/28 15:05:58 carl Exp $
12 *
13 * $Locker: $
14 *
15 *
16 *
17 BITDEF S.ALERTWACK,1 * in ExecBase.SysFlags
18 *
19 *
20 *
21 *
22 * Format of the alert error number:
23 *
24 * +-----+
25 * |DI| SubSysId | General Error | SubSystem Specific Error |
26 * +-----+
27 *
28 * D: DeadEnd alert
29 * SubSysId: indicates ROM subsystem number.
30 * General Error: roughly indicates what the error was
31 * Specific Error: indicates more detail
32 *
33 *
34 *
35 * Use this macro for causing an alert. THIS MACRO MAY CHANGE!
36 * It is very sensitive to memory corruption.... like stepping on
37 * location 4! But it should work for now.
38 *
39 ALERT macro (alertNumber, paramArray, scratch)
40   movem.l d7/a5/a6,-(sp)
41   move.l 4,a6
42   IFNC '\2,1'
43     lea \2,a5
44   ENDC
45   move.l 4,a6 ; (use proper name!!!)
46   jsr _JVORAlert(a6)
47   movem.l (sp)+,d7/a5/a6
48   endm
49 *
50 *
51 *
52 *
53 * General Dead-End Alerts
54 *
55 * For example: timer.device cannot open math.library:
56 *

```

```

57 * ALERT (AN_TimerDev|AG_OpenLib|AO_MathLib),(A0),A1
58 *
59 *
60 *
61 * ----- alert types
62 AT_DeadEnd equ $80000000
63 AT_Recovery equ $00000000
64 *
65 * ----- general purpose alert codes
66 AG_NoMemory equ $00010000
67 AG_MakeLib equ $00020000
68 AG_OpenLib equ $00030000
69 AG_OpenDev equ $00040000
70 AG_OpenRes equ $00050000
71 AG_IOError equ $00060000
72 *
73 * ----- alert objects:
74 AO_ExecLib equ $00080001
75 AO_GraphicsLib equ $00080002
76 AO_LayersLib equ $00080003
77 AO_Intuition equ $00080004
78 AO_MathLib equ $00080005
79 AO_CListLib equ $00080006
80 AO_DOSLib equ $00080007
81 AO_RAMLib equ $00080008
82 AO_IconLib equ $00080009
83 AO_AudioDev equ $00080010
84 AO_ConsoleDev equ $00080011
85 AO_GamePortDev equ $00080012
86 AO_KeyboardDev equ $00080013
87 AO_TrackDiskDev equ $00080014
88 AO_TimerDev equ $00080015
89 AO_CIARsrc equ $00080020
90 AO_DiskRsrc equ $00080021
91 AO_MiscRsrc equ $00080022
92 AO_BootStrap equ $00080030
93 AO_Workbench equ $00080031
94 *
95 *
96 *
97 * Specific Dead-End Alerts:
98 *
99 * For example: exec.library -- corrupted memory list
100 *
101 *
102 * ALERT AN_MemCorrupt,(A0),A1
103 *
104 *
105 *
106 * ----- exec.library
107 AN_ExecLib equ $01000000
108 AN_ExecVect equ $81000001 ; 68000 exception vector checksum
109 AN_BaseChkSum equ $81000002 ; execbase checksum
110 AN_LibChkSum equ $81000003 ; library checksum failure
111 AN_LibMem equ $81000004 ; no memory to make library
112 AN_MemCorrupt equ $81000005 ; corrupted memory list

```

```

113 AN_IntrMem equ $81000006 ; no memory for interrupt servers
114 AN_InitAPtr equ $81000007 ; InitStruct() of an APTR source
115
116 ;----- graphics.library
117 AN_GraphicsLib equ $02000000
118 AN_CopDisplay equ $02010001 ; copper display list, no memory
119 AN_CopInstr equ $02010002 ; copper instruction list, no memory
120 AN_CopListOver equ $02000003 ; copper list overload
121 AN_CopListOver equ $02000004 ; copper intermediate list overload
122 AN_CopListHead equ $02010005 ; copper list head, no memory
123 AN_LongFrame equ $02010006 ; long frame, no memory
124 AN_ShortFrame equ $02010007 ; short frame, no memory
125 AN_FloodFill equ $02010008 ; flood fill, no memory
126 AN_TextImpKas equ $02010009 ; text, no memory for ImpKas
127 AN_BitBitMap equ $0201000A ; BitBitMap, no memory
128
129 ;----- layers.library
130 AN_LayersLib equ $03000000
131
132 ;----- intuition.library
133 AN_Intuition equ $04000000
134 AN_GadgetType equ $04000001 ; unknown gadget type
135 AN_BadGadget equ $04000001 ; Recovery form of AN_GadgetType
136 AN_CreatedPort equ $04010002 ; create port, no memory
137 AN_ItemAlloc equ $04010003 ; item plane alloc, no memory
138 AN_SubAlloc equ $04010004 ; sub alloc, no memory
139 AN_PlaneAlloc equ $04010005 ; plane alloc, no memory
140 AN_ItemBoxTop equ $04000006 ; item box top < RelZero
141 AN_OpenScreen equ $04010007 ; open screen, no memory
142 AN_OpenScreenRast equ $04010008 ; open screen, raster alloc, no memory
143 AN_SysScreenType equ $04000009 ; open sys screen, unknown type
144 AN_AddSMGadget equ $0401000A ; add SW gadgets, no memory
145 AN_OpenWindow equ $0401000B ; open window, no memory
146 AN_BadState equ $0400000C ; Bad State Return entering Intuition
147 AN_BadMessage equ $0400000D ; Bad Message received by IDCMP
148 AN_WeirdEcho equ $0400000E ; Weird echo causing incomprehension
149 AN_NoConsole equ $0400000F ; couldn't open the Console Device
150
151 ;----- math.library
152 AN_MathLib equ $05000000
153
154 ;----- clist.library
155 AN_CListLib equ $06000000
156
157 ;----- dos.library
158 AN_DOSLib equ $07000000
159 AN_StartMem equ $07010001 ; no memory at startup
160 AN_EndTask equ $07000002 ; EndTask didn't
161 AN_QpktFail equ $07000003 ; Qpkt failure
162 AN_AsyncPkt equ $07000004 ; Unexpected packet received
163 AN_FreeVec equ $07000005 ; Freevec failed
164 AN_DiskBlkSeq equ $07000006 ; Disk block sequence error
165 AN_BitMap equ $07000007 ; Bitmap corrupt
166 AN_KeyFree equ $07000008 ; Key already free
167 AN_BadChkSum equ $07000009 ; Invalid checksum
168 AN_DiskError equ $0700000A ; Disk Error

```

```

169 AN_KeyRange equ $0700000B ; Key out of range
170 AN_BadOverlay equ $0700000C ; Bad overlay
171
172 ;----- ramlib.library
173 AN_RAMLib equ $08000000
174
175 ;----- icon.library
176 AN_IconLib equ $09000000
177
178 ;----- audio.device
179 AN_AudioDev equ $10000000
180
181 ;----- console.device
182 AN_ConsoleDev equ $11000000
183
184 ;----- gamesport.device
185 AN_GamePortDev equ $12000000
186
187 ;----- keyboard.device
188 AN_KeyboardDev equ $13000000
189
190 ;----- trackdisk.device
191 AN_TrackDiskDev equ $14000000
192 AN_TDCalibSeek equ $14000001 ; calibrate: seek error
193 AN_TDDelay equ $14000002 ; delay: error on timer wait
194
195 ;----- timer.device
196 AN_TimerDev equ $15000000
197 AN_TMBadReq equ $15000001 ; bad request
198
199 ;----- cia.resource
200 AN_CIArsrc equ $20000000
201
202 ;----- disk.resource
203 AN_DiskRsrc equ $21000000
204 AN_DRHasDisk equ $21000001 ; get unit: already has disk
205 AN_DRIntNoAct equ $21000002 ; interrupt: no active unit
206
207 ;----- misc.resource
208 AN_MiscRsrc equ $22000000
209
210 ;----- bootstrap
211 AN_BootStrap equ $30000000
212 AN_BootError equ $30000001 ; boot code returned an error
213
214 ;----- workbench
215 AN_Workbench equ $31000000
216
217 ;-----
218 ;-----
219 ;-----
220 ;-----
221 ;-----
222 ;-----
223 ;-----
224 ;-----
225 ;-----
226 ;-----
227 ;-----
228 ;-----
229 ;-----
230 ;-----
231 ;-----
232 ;-----
233 ;-----
234 ;-----
235 ;-----
236 ;-----
237 ;-----
238 ;-----
239 ;-----
240 ;-----
241 ;-----
242 ;-----
243 ;-----
244 ;-----
245 ;-----
246 ;-----
247 ;-----
248 ;-----
249 ;-----
250 ;-----
251 ;-----
252 ;-----
253 ;-----
254 ;-----
255 ;-----
256 ;-----
257 ;-----
258 ;-----
259 ;-----
260 ;-----
261 ;-----
262 ;-----
263 ;-----
264 ;-----
265 ;-----
266 ;-----
267 ;-----
268 ;-----
269 ;-----
270 ;-----
271 ;-----
272 ;-----
273 ;-----
274 ;-----
275 ;-----
276 ;-----
277 ;-----
278 ;-----
279 ;-----
280 ;-----
281 ;-----
282 ;-----
283 ;-----
284 ;-----
285 ;-----
286 ;-----
287 ;-----
288 ;-----
289 ;-----
290 ;-----
291 ;-----
292 ;-----
293 ;-----
294 ;-----
295 ;-----
296 ;-----
297 ;-----
298 ;-----
299 ;-----
300 ;-----
301 ;-----
302 ;-----
303 ;-----
304 ;-----
305 ;-----
306 ;-----
307 ;-----
308 ;-----
309 ;-----
310 ;-----
311 ;-----
312 ;-----
313 ;-----
314 ;-----
315 ;-----
316 ;-----
317 ;-----
318 ;-----
319 ;-----
320 ;-----
321 ;-----
322 ;-----
323 ;-----
324 ;-----
325 ;-----
326 ;-----
327 ;-----
328 ;-----
329 ;-----
330 ;-----
331 ;-----
332 ;-----
333 ;-----
334 ;-----
335 ;-----
336 ;-----
337 ;-----
338 ;-----
339 ;-----
340 ;-----
341 ;-----
342 ;-----
343 ;-----
344 ;-----
345 ;-----
346 ;-----
347 ;-----
348 ;-----
349 ;-----
350 ;-----
351 ;-----
352 ;-----
353 ;-----
354 ;-----
355 ;-----
356 ;-----
357 ;-----
358 ;-----
359 ;-----
360 ;-----
361 ;-----
362 ;-----
363 ;-----
364 ;-----
365 ;-----
366 ;-----
367 ;-----
368 ;-----
369 ;-----
370 ;-----
371 ;-----
372 ;-----
373 ;-----
374 ;-----
375 ;-----
376 ;-----
377 ;-----
378 ;-----
379 ;-----
380 ;-----
381 ;-----
382 ;-----
383 ;-----
384 ;-----
385 ;-----
386 ;-----
387 ;-----
388 ;-----
389 ;-----
390 ;-----
391 ;-----
392 ;-----
393 ;-----
394 ;-----
395 ;-----
396 ;-----
397 ;-----
398 ;-----
399 ;-----
400 ;-----
401 ;-----
402 ;-----
403 ;-----
404 ;-----
405 ;-----
406 ;-----
407 ;-----
408 ;-----
409 ;-----
410 ;-----
411 ;-----
412 ;-----
413 ;-----
414 ;-----
415 ;-----
416 ;-----
417 ;-----
418 ;-----
419 ;-----
420 ;-----
421 ;-----
422 ;-----
423 ;-----
424 ;-----
425 ;-----
426 ;-----
427 ;-----
428 ;-----
429 ;-----
430 ;-----
431 ;-----
432 ;-----
433 ;-----
434 ;-----
435 ;-----
436 ;-----
437 ;-----
438 ;-----
439 ;-----
440 ;-----
441 ;-----
442 ;-----
443 ;-----
444 ;-----
445 ;-----
446 ;-----
447 ;-----
448 ;-----
449 ;-----
450 ;-----
451 ;-----
452 ;-----
453 ;-----
454 ;-----
455 ;-----
456 ;-----
457 ;-----
458 ;-----
459 ;-----
460 ;-----
461 ;-----
462 ;-----
463 ;-----
464 ;-----
465 ;-----
466 ;-----
467 ;-----
468 ;-----
469 ;-----
470 ;-----
471 ;-----
472 ;-----
473 ;-----
474 ;-----
475 ;-----
476 ;-----
477 ;-----
478 ;-----
479 ;-----
480 ;-----
481 ;-----
482 ;-----
483 ;-----
484 ;-----
485 ;-----
486 ;-----
487 ;-----
488 ;-----
489 ;-----
490 ;-----
491 ;-----
492 ;-----
493 ;-----
494 ;-----
495 ;-----
496 ;-----
497 ;-----
498 ;-----
499 ;-----
500 ;-----
501 ;-----
502 ;-----
503 ;-----
504 ;-----
505 ;-----
506 ;-----
507 ;-----
508 ;-----
509 ;-----
510 ;-----
511 ;-----
512 ;-----
513 ;-----
514 ;-----
515 ;-----
516 ;-----
517 ;-----
518 ;-----
519 ;-----
520 ;-----
521 ;-----
522 ;-----
523 ;-----
524 ;-----
525 ;-----
526 ;-----
527 ;-----
528 ;-----
529 ;-----
530 ;-----
531 ;-----
532 ;-----
533 ;-----
534 ;-----
535 ;-----
536 ;-----
537 ;-----
538 ;-----
539 ;-----
540 ;-----
541 ;-----
542 ;-----
543 ;-----
544 ;-----
545 ;-----
546 ;-----
547 ;-----
548 ;-----
549 ;-----
550 ;-----
551 ;-----
552 ;-----
553 ;-----
554 ;-----
555 ;-----
556 ;-----
557 ;-----
558 ;-----
559 ;-----
560 ;-----
561 ;-----
562 ;-----
563 ;-----
564 ;-----
565 ;-----
566 ;-----
567 ;-----
568 ;-----
569 ;-----
570 ;-----
571 ;-----
572 ;-----
573 ;-----
574 ;-----
575 ;-----
576 ;-----
577 ;-----
578 ;-----
579 ;-----
580 ;-----
581 ;-----
582 ;-----
583 ;-----
584 ;-----
585 ;-----
586 ;-----
587 ;-----
588 ;-----
589 ;-----
590 ;-----
591 ;-----
592 ;-----
593 ;-----
594 ;-----
595 ;-----
596 ;-----
597 ;-----
598 ;-----
599 ;-----
600 ;-----
601 ;-----
602 ;-----
603 ;-----
604 ;-----
605 ;-----
606 ;-----
607 ;-----
608 ;-----
609 ;-----
610 ;-----
611 ;-----
612 ;-----
613 ;-----
614 ;-----
615 ;-----
616 ;-----
617 ;-----
618 ;-----
619 ;-----
620 ;-----
621 ;-----
622 ;-----
623 ;-----
624 ;-----
625 ;-----
626 ;-----
627 ;-----
628 ;-----
629 ;-----
630 ;-----
631 ;-----
632 ;-----
633 ;-----
634 ;-----
635 ;-----
636 ;-----
637 ;-----
638 ;-----
639 ;-----
640 ;-----
641 ;-----
642 ;-----
643 ;-----
644 ;-----
645 ;-----
646 ;-----
647 ;-----
648 ;-----
649 ;-----
650 ;-----
651 ;-----
652 ;-----
653 ;-----
654 ;-----
655 ;-----
656 ;-----
657 ;-----
658 ;-----
659 ;-----
660 ;-----
661 ;-----
662 ;-----
663 ;-----
664 ;-----
665 ;-----
666 ;-----
667 ;-----
668 ;-----
669 ;-----
670 ;-----
671 ;-----
672 ;-----
673 ;-----
674 ;-----
675 ;-----
676 ;-----
677 ;-----
678 ;-----
679 ;-----
680 ;-----
681 ;-----
682 ;-----
683 ;-----
684 ;-----
685 ;-----
686 ;-----
687 ;-----
688 ;-----
689 ;-----
690 ;-----
691 ;-----
692 ;-----
693 ;-----
694 ;-----
695 ;-----
696 ;-----
697 ;-----
698 ;-----
699 ;-----
700 ;-----
701 ;-----
702 ;-----
703 ;-----
704 ;-----
705 ;-----
706 ;-----
707 ;-----
708 ;-----
709 ;-----
710 ;-----
711 ;-----
712 ;-----
713 ;-----
714 ;-----
715 ;-----
716 ;-----
717 ;-----
718 ;-----
719 ;-----
720 ;-----
721 ;-----
722 ;-----
723 ;-----
724 ;-----
725 ;-----
726 ;-----
727 ;-----
728 ;-----
729 ;-----
730 ;-----
731 ;-----
732 ;-----
733 ;-----
734 ;-----
735 ;-----
736 ;-----
737 ;-----
738 ;-----
739 ;-----
740 ;-----
741 ;-----
742 ;-----
743 ;-----
744 ;-----
745 ;-----
746 ;-----
747 ;-----
748 ;-----
749 ;-----
750 ;-----
751 ;-----
752 ;-----
753 ;-----
754 ;-----
755 ;-----
756 ;-----
757 ;-----
758 ;-----
759 ;-----
760 ;-----
761 ;-----
762 ;-----
763 ;-----
764 ;-----
765 ;-----
766 ;-----
767 ;-----
768 ;-----
769 ;-----
770 ;-----
771 ;-----
772 ;-----
773 ;-----
774 ;-----
775 ;-----
776 ;-----
777 ;-----
778 ;-----
779 ;-----
780 ;-----
781 ;-----
782 ;-----
783 ;-----
784 ;-----
785 ;-----
786 ;-----
787 ;-----
788 ;-----
789 ;-----
790 ;-----
791 ;-----
792 ;-----
793 ;-----
794 ;-----
795 ;-----
796 ;-----
797 ;-----
798 ;-----
799 ;-----
800 ;-----
801 ;-----
802 ;-----
803 ;-----
804 ;-----
805 ;-----
806 ;-----
807 ;-----
808 ;-----
809 ;-----
810 ;-----
811 ;-----
812 ;-----
813 ;-----
814 ;-----
815 ;-----
816 ;-----
817 ;-----
818 ;-----
819 ;-----
820 ;-----
821 ;-----
822 ;-----
823 ;-----
824 ;-----
825 ;-----
826 ;-----
827 ;-----
828 ;-----
829 ;-----
830 ;-----
831 ;-----
832 ;-----
833 ;-----
834 ;-----
835 ;-----
836 ;-----
837 ;-----
838 ;-----
839 ;-----
840 ;-----
841 ;-----
842 ;-----
843 ;-----
844 ;-----
845 ;-----
846 ;-----
847 ;-----
848 ;-----
849 ;-----
850 ;-----
851 ;-----
852 ;-----
853 ;-----
854 ;-----
855 ;-----
856 ;-----
857 ;-----
858 ;-----
859 ;-----
860 ;-----
861 ;-----
862 ;-----
863 ;-----
864 ;-----
865 ;-----
866 ;-----
867 ;-----
868 ;-----
869 ;-----
870 ;-----
871 ;-----
872 ;-----
873 ;-----
874 ;-----
875 ;-----
876 ;-----
877 ;-----
878 ;-----
879 ;-----
880 ;-----
881 ;-----
882 ;-----
883 ;-----
884 ;-----
885 ;-----
886 ;-----
887 ;-----
888 ;-----
889 ;-----
890 ;-----
891 ;-----
892 ;-----
893 ;-----
894 ;-----
895 ;-----
896 ;-----
897 ;-----
898 ;-----
899 ;-----
900 ;-----
901 ;-----
902 ;-----
903 ;-----
904 ;-----
905 ;-----
906 ;-----
907 ;-----
908 ;-----
909 ;-----
910 ;-----
911 ;-----
912 ;-----
913 ;-----
914 ;-----
915 ;-----
916 ;-----
917 ;-----
918 ;-----
919 ;-----
920 ;-----
921 ;-----
922 ;-----
923 ;-----
924 ;-----
925 ;-----
926 ;-----
927 ;-----
928 ;-----
929 ;-----
930 ;-----
931 ;-----
932 ;-----
933 ;-----
934 ;-----
935 ;-----
936 ;-----
937 ;-----
938 ;-----
939 ;-----
940 ;-----
941 ;-----
942 ;-----
943 ;-----
944 ;-----
945 ;-----
946 ;-----
947 ;-----
948 ;-----
949 ;-----
950 ;-----
951 ;-----
952 ;-----
953 ;-----
954 ;-----
955 ;-----
956 ;-----
957 ;-----
958 ;-----
959 ;-----
960 ;-----
961 ;-----
962 ;-----
963 ;-----
964 ;-----
965 ;-----
966 ;-----
967 ;-----
968 ;-----
969 ;-----
970 ;-----
971 ;-----
972 ;-----
973 ;-----
974 ;-----
975 ;-----
976 ;-----
977 ;-----
978 ;-----
979 ;-----
980 ;-----
981 ;-----
982 ;-----
983 ;-----
984 ;-----
985 ;-----
986 ;-----
987 ;-----
988 ;-----
989 ;-----
990 ;-----
991 ;-----
992 ;-----
993 ;-----
994 ;-----
995 ;-----
996 ;-----
997 ;-----
998 ;-----
999 ;-----
1000 ;-----

```

ENDC !EXEC_ALERTS_I

```

1  IFND EXEC_DEVICES_I
2  EXEC_DEVICES_I SET 1
3  *****
4  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
5  *
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: devices.i,v 1.0 85/08/28 15:07:02 carl Exp $
12 *
13 * $Locker: $
14 *
15 *
16 *
17 IFND EXEC_LIBRARIES_I
18 INCLUDE "exec/libraries.i"
19 ENDC !EXEC_LIBRARIES_I
20
21 IFND EXEC_PORTS_I
22 INCLUDE "exec/ports.i"
23 ENDC !EXEC_PORTS_I
24
25
26 -----
27 *
28 * Device Data Structure
29 *
30 *
31
32 STRUCTURE DD_LIB_SIZE          * identical to library
33 LABEL DD_SIZE
34
35
36 -----
37 *
38 * Suggested Unit Structure
39 *
40 *
41
42 STRUCTURE UNIT_MP_SIZE        * queue for requests
43 UBYTE UNIT_FLAGS
44 UWORD UNIT_PAD
45 UWORD UNIT_OPENCNT
46 LABEL UNIT_SIZE
47
48
49 ----- UNIT_FLAG definitions:
50
51 BITDEF UNIT_ACTIVE_0          * driver is active
52 BITDEF UNIT_INTPASK,1        * running in driver's task
53
54 ENDC !EXEC_DEVICES_I

```

```

1  IFND EXEC_ERRORS_I
2  EXEC_ERRORS_I SET 1
3  *****
4  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
5  *
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: errors.i,v 1.0 85/08/28 15:07:26 carl Exp $
12 *
13 * $Locker: $
14 *
15 *
16 *
17 *----- Standard IO Errors:
18 *
19 IOERR_OPENFAIL EQU -1      * device/unit failed to open
20 IOERR_ABORTED EQU -2      * request aborted
21 IOERR_NOCMD EQU -3        * command not supported
22 IOERR_BADLENGTH EQU -4   * not a valid length
23
24
25 ERR_OPENDEVICE EQU IOERR_OPENFAIL * REMOVE !!!
26
27 ENDC !EXEC_ERRORS_I

```

```

1 INCLUDE "exec/nodes.1"
2 INCLUDE "exec/lists.1"
3 INCLUDE "exec/interrupts.1"
4 INCLUDE "exec/memory.1"
5 INCLUDE "exec/ports.1"
6 INCLUDE "exec/tasks.1"
7 INCLUDE "exec/libraries.1"
8 INCLUDE "exec/devices.1"
9 INCLUDE "exec/lo.1"
10
11

```

```

1 *** Commodore-Amiga, Inc.
2 *** This file generated on Tue Nov 12 16:50:46 1985
3 *** $Header: gen-lib.ml,v 2.5 85/10/08 18:09:47 car1 Exp $
4 *** DO NOT EDIT: FILE BUILT AUTOMATICALLY
5 FUNCDEF Supervisor
6 FUNCDEF ExitIntr
7 FUNCDEF Schedule
8 FUNCDEF Reschedule
9 FUNCDEF Switch
10 FUNCDEF Dispatch
11 FUNCDEF Exception
12 FUNCDEF InitCode
13 FUNCDEF InitStruct
14 FUNCDEF MakeLibrary
15 FUNCDEF MakeFunctions
16 FUNCDEF FindResident
17 FUNCDEF InitResident
18 FUNCDEF Alert
19 FUNCDEF Debug
20 FUNCDEF Disable
21 FUNCDEF Enable
22 FUNCDEF Forbid
23 FUNCDEF Permit
24 FUNCDEF SetSR
25 FUNCDEF SuperState
26 FUNCDEF UserState
27 FUNCDEF SetIntVector
28 FUNCDEF AddIntServer
29 FUNCDEF RemIntServer
30 FUNCDEF Cause
31 FUNCDEF Allocate
32 FUNCDEF Deallocate
33 FUNCDEF AllocMem
34 FUNCDEF AllocAbs
35 FUNCDEF FreeMem
36 FUNCDEF AvailMem
37 FUNCDEF AllocEntry
38 FUNCDEF FreeEntry
39 FUNCDEF Insert
40 FUNCDEF AddHead
41 FUNCDEF AddTail
42 FUNCDEF Remove
43 FUNCDEF RemHead
44 FUNCDEF RemTail
45 FUNCDEF Enqueue
46 FUNCDEF FindName
47 FUNCDEF AddTask
48 FUNCDEF RemTask
49 FUNCDEF FindTask
50 FUNCDEF SetTaskPri
51 FUNCDEF SetSignal
52 FUNCDEF SetExcept
53 FUNCDEF Wait
54 FUNCDEF Signal
55 FUNCDEF AllocSignal
56 FUNCDEF FreeSignal

```



```

57 FUNCDEF AllocTrap
58 FUNCDEF FreeTrap
59 FUNCDEF AddPort
60 FUNCDEF RemPort
61 FUNCDEF PutMsg
62 FUNCDEF GetMsg
63 FUNCDEF ReplyMsg
64 FUNCDEF WaitPort
65 FUNCDEF FIndPort
66 FUNCDEF AddLibrary
67 FUNCDEF RemLibrary
68 FUNCDEF OldOpenLibrary
69 FUNCDEF CloseLibrary
70 FUNCDEF SetFunction
71 FUNCDEF SumLibrary
72 FUNCDEF AddDevice
73 FUNCDEF RemDevice
74 FUNCDEF OpenDevice
75 FUNCDEF CloseDevice
76 FUNCDEF DoIO
77 FUNCDEF SendIO
78 FUNCDEF CheckIO
79 FUNCDEF WaitIO
80 FUNCDEF AbortIO
81 FUNCDEF AddrResource
82 FUNCDEF RemResource
83 FUNCDEF OpenResource
84 FUNCDEF RawIOInit
85 FUNCDEF RawMayGetChar
86 FUNCDEF RawPutChar
87 FUNCDEF RawDoFmt
88 FUNCDEF GetCC
89 FUNCDEF TypeOfMem
90 FUNCDEF Procure
91 FUNCDEF Vacate
92 FUNCDEF OpenLibrary

```

```

1 IFND EXEC_EXECBASE_I
2 EXEC_EXECBASE_I SET 1
3 *****
4 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
5 *
6 *
7 *
8 * Source Control:
9 *
10 * $Header: execbase.1,v 1.1 85/11/12 16:10:51 carl Exp $
11 *
12 *
13 * $Locker: carl $
14 *
15 *****
16 IFND EXEC_LISTS_I
17 INCLUDE "exec/lists.1"
18 ENDC EXEC_LISTS_I
19
20 IFND EXEC_INTERRUPTS_I
21 INCLUDE "exec/interrupts.1"
22 ENDC EXEC_INTERRUPTS_I
23
24 IFND EXEC_LIBRARIES_I
25 INCLUDE "exec/libraries.1"
26 ENDC EXEC_LIBRARIES_I
27
28
29 ***** Static System Variables *****
30
31 STRUCTURE ExecBase,LIB_SIZE ; Standard library node
32
33 UMORD SoftVer ; kickstart release number
34 WORD LowMemChkSum ; checksum of 68000 trap vectors
35 ULONG ChkBase ; system base pointer complement
36 APTR CoolCapture ; cold soft capture vector
37 APTR WarmCapture ; warm soft capture vector
38 APTR SysStkUpper ; system stack base (upper bound)
39 APTR SysStkLower ; top of system stack (lower bound)
40 ULONG MaxLockMem ; last calculated local memory max
41 APTR DebugIntrny ; global debugger entry point
42 APTR DebugData ; global debugger data segment
43 APTR AlertData ; alert data segment
44 APTR RsvdExt ; reserved
45
46 WORD ChkSum ; for all of the above
47
48 ***** Interrupt Related *****
49
50 LABEL IntVects
51 STRUCT IVTBE_IV_SIZE
52 STRUCT IVDSKBLK_IV_SIZE
53 STRUCT IVSOFTINT_IV_SIZE
54
55
56

```

```

57  STRUCT  IVPORTS, IV_SIZE
58  STRUCT  IVOPER, IV_SIZE
59  STRUCT  IWERTB, IV_SIZE
60  STRUCT  IVBLIT, IV_SIZE
61  STRUCT  IVAUD0, IV_SIZE
62  STRUCT  IVAUD1, IV_SIZE
63  STRUCT  IVAUD2, IV_SIZE
64  STRUCT  IVAUD3, IV_SIZE
65  STRUCT  IVRBE, IV_SIZE
66  STRUCT  IVDSKSYNC, IV_SIZE
67  STRUCT  IVEXTER, IV_SIZE
68  STRUCT  IVINTEN, IV_SIZE
69  STRUCT  IVNMI, IV_SIZE
70
71
72  ***** Dynamic System Variables *****
73
74  APTR  ThisTask      ; pointer to current task
75  ULONG  IdleCount   ; idle counter
76  ULONG  DispCount   ; dispatch counter
77  UWORD  Quantum     ; time slice quantum
78  UWORD  Elapsed     ; current quantum ticks
79  UWORD  SysFlags    ; misc system flags
80  BYTE  IDNestCnt   ; interrupt disable nesting count
81  BYTE  TDNestCnt   ; task disable nesting count
82
83  UWORD  AttnFlags   ; special attention flags
84  UWORD  AttnResched ; rescheduling attention
85  APTR  ResModules  ; pointer to resident module array
86
87  APTR  TaskTrapCode ; default task trap routine
88  APTR  TaskExcpCode ; default task exception code
89  APTR  TaskExitCode ; default task exit code
90  ULONG  TaskSigAlloc ; preallocated signal mask
91  UWORD  TaskTrapAlloc ; preallocated trap mask
92
93
94  ***** System List Headers *****
95
96  STRUCT  MemList, LH_SIZE
97  STRUCT  ResourceList, LH_SIZE
98  STRUCT  DeviceList, LH_SIZE
99  STRUCT  IntrList, LH_SIZE
100  STRUCT  LibList, LH_SIZE
101  STRUCT  PortList, LH_SIZE
102  STRUCT  TaskReady, LH_SIZE
103  STRUCT  TaskWait, LH_SIZE
104
105  STRUCT  SoftInts, SH_SIZE*5
106
107  STRUCT  LastAlert, 4*4
108
109  LONG  ExecBaseReserved, 4*8
110
111  LABEL  SYSBASESIZE
112

```

```

113  ***** AttnFlags
114  * Processors and Co-processors:
115  AFB_68010 EQU 0 (will remain set for 68020 as well)
116  AFB_68020 EQU 1
117  AFB_68881 EQU 4
118  AFB_PAL EQU 8 PAL/NTSC
119  AFB_50HZ EQU 9 Clock Rate
120  ENDC IEXEC_EXECBASE_I

```

```

1 IFND EXEC_EXECNAME_I
2 EXEC_EXECNAME_I SET 1
3 * Commodore-Amiga, Inc.
4 EXECNAME macro
5 dc.b 'exec.library',0
6 ds.w 0
7 endm
8
9 ENDC !EXEC_EXECNAME_I
10

```

```

1 IFND EXEC_INITIALIZERS_I
2 EXEC_INITIALIZERS_I SET 1
3 *****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *****
8 *
9 * Source Control:
10 *
11 * $Header: initializers.i.v 1.0 85/08/28 15:09:29 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****
16
17
18 INITBYTE MACRO * &offset,&value
19 DC.B $e0
20 DC.B 0
21 DC.W \1
22 DC.B \2
23 DC.B 0
24 ENDM
25
26 INITWORD MACRO * &offset,&value
27 DC.B $d0
28 DC.B 0
29 DC.W \1
30 DC.W \2
31 ENDM
32
33 INITLONG MACRO * &offset,&value
34 DC.B $c0
35 DC.B 0
36 DC.W \1
37 DC.L \2
38 ENDM
39
40 INITSTRUCT MACRO * &size,&offset,&value,&count
41 DS.W 0
42 IFC '4',''
43 SET 0
44 ENDC
45 IENC '4',''
46 SET \4
47 ENDC
48 CMDF\@ SET (((\1)<<4)!COUNT\@)
49 IFLE (\2)-255 (CMD\@) !$80
50 DC.B (\2)
51 DC.B \2
52 MEXIT
53 ENDC
54 DC.B CMD\@!$0C0
55 DC.B (((\2)>>16)&$0FF)
56 DC.W (((\2)&$0FFF)

```

57 ENDM
58
59 ENDC !EXEC_INITIALIZERS_I

```

1  IFND EXEC_INTERRUPTS_I
2  EXEC_INTERRUPTS_I SET 1
3  *****
4  *
5  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: interrupts.i.v 1.0 85/08/28 15:10:16 carl Exp $
12 *
13 * $Locker: $
14 *
15 *
16 *
17 IFND EXEC_NODES_I
18 INCLUDE "exec/nodes.i"
19 ENDC !EXEC_NODES_I
20
21 IFND EXEC_LISTS_I
22 INCLUDE "exec/lists.i"
23 ENDC !EXEC_LISTS_I
24
25
26 -----
27 * Interrupt Structure
28 *
29 *
30 *
31 *
32 STRUCTURE IS LN_SIZE
33 APTR IS_DATA
34 APTR IS_CODE
35 LABEL IS_SIZE
36
37 -----
38 *
39 * Exec Internal Interrupt Vectors
40 *
41 *
42 *
43 *
44 STRUCTURE IV_0
45 APTR IV_DATA
46 APTR IV_CODE
47 APTR IV_NODE
48 LABEL IV_SIZE
49
50 -----
51 *----- System Flag bits (in SysBase.SysFlags )
52
53 BITDEF S,SAR,15 * scheduling attention required
54 BITDEF S,TOE,14 * time quantum expended -- time to resched
55 BITDEF S,SINT,13
56

```

```

57 *-----
58 *
59 *
60 * Software Interrupt List Headers
61 *
62 *-----
63 *
64 STRUCTURE SH, LH_SIZE
65 UMORD SH_PAD
66 LABEL SH_SIZE
67
68 SIH_PRIMASK EQU $0F0
69 SIH_QUEUES EQU 5
70
71 ENDC IEXEC_INTERRUPTS_I

```

```

1 IFND EXEC_IO_I
2 EXEC_IO_I SET 1
3 *****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *-----
8 *
9 * Source Control:
10 *
11 * $Header: io.i.v 1.0 85/08/28 15:10:43 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****
16
17 IFND EXEC_PORTS_I
18 INCLUDE "exec/ports.i"
19 ENDC IEXEC_PORTS_I
20
21 IFND EXEC_LIBRARIES_I
22 INCLUDE "exec/libraries.i"
23 ENDC IEXEC_LIBRARIES_I
24
25
26 -----
27 *
28 * IO Request Structures
29 *
30 -----
31
32 *----- Required portion of IO request:
33
34 STRUCTURE IO_MN_SIZE
35 APTR IO_DEVICE
36 APTR IO_UNIT
37 UMORD IO_COMMAND
38 UBYTE IO_FLAGS
39 BYTE IO_ERROR
40 LABEL IO_SIZE
41
42
43 *----- Standard IO request extension:
44
45 ULONG IO_ACTUAL
46 ULONG IO_LENGTH
47 APTR IO_DATA
48 ULONG IO_OFFSET
49 * ULONG IO_RESERVED1
50 * ULONG IO_RESERVED2
51 LABEL IOSTD_SIZE
52
53
54 *----- IO_FLAGS bit definitions:
55
56 BITDEF IO_QUICK, 0 * complete IO quickly

```

* device node pointer
* unit (driver private)
* device command
* special flags
* error or warning code

* actual # of bytes transferred
* requested # of bytes transferred
* pointer to data area
* offset for seeking devices

```

57 *-----
58 *
59 *
60 *
61 * Standard Device Library Functions
62 *-----
63 *
64 *
65 LIBINIT
66
67 LIBDEF DEV_BEGINIO * process IO request
68 LIBDEF DEV_ABORTIO * abort IO request
69
70
71 *-----
72 *
73 * IO Function Macros
74 *-----
75 *
76
77 BEGINIO MACRO
78 LINKLIB DEV_BEGINIO, IO_DEVICE (A1)
79 ENDM
80
81 ABORTIO MACRO
82 LINKLIB DEV_ABORTIO, IO_DEVICE (A1)
83 ENDM
84
85
86 *-----
87 *
88 * Standard Device Command Definitions
89 *-----
90 *
91 *----- Command definition macro:
92 DEVINIT MACRO * [baseOffset]
93 IFC '\1', ''
94 CMD_COUNT SET CMD_NONSTD
95 ENDC
96 IFC '\1', ''
97 CMD_COUNT SET \1
98 ENDC
99 ENDM
100
101
102 DEVCMD MACRO * cmdname
103 EQU CMD_COUNT
104 CMD_COUNT SET CMD_COUNT+1
105 ENDM
106
107
108 *----- Standard device commands:
109 DEVINIT 0
110
111 DEVCMD CMD_INVALID * invalid command
112

```

```

113 DEVCMD CMD_RESET
114 DEVCMD CMD_READ
115 DEVCMD CMD_WRITE
116 DEVCMD CMD_UPDATE
117 DEVCMD CMD_CLEAR
118 DEVCMD CMD_STOP
119 DEVCMD CMD_START
120 DEVCMD CMD_FLUSH
121
122
123 *----- First non-standard device command value:
124
125 DEVCMD CMD_NONSTD
126
127 ENDC !EXEC_IO_I

```

```

* reset as if just initied
* standard read
* standard write
* write out all buffers
* clear all buffers
* hold current and queued
* restart after stop
* abort entire queue

```

```

1  IFND EXEC_LIBRARIES_I
2  EXEC_LIBRARIES_I SET 1
3  *****
4  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
5  *
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: libraries.i.v 1.0 85/08/28 15:11:09 carl Exp $
12 *
13 * $Locker: $
14 *
15 *
16 IFND EXEC_NODES_I
17 INCLUDE "exec/nodes.i"
18 ENDC IEXEC_NODES_I
19
20
21 *----- Special Constants -----
22
23 LIB_VECTSIZE EQU 6
24 LIB_RESERVED EQU 4
25 LIB_BASE EQU $FFFFFFA * (-LIB_VECTSIZE)
26 LIB_USERDEF EQU LIB_BASE - (LIB_RESERVED*LIB_VECTSIZE)
27 LIB_NONSTD EQU LIB_USERDEF
28
29
30 *-----
31 *
32 * Library Definition Macros
33 *
34 *-----
35
36 *----- LIBINIT sets base offset for library function definitions:
37
38 LIBINIT MACRO * [baseOffset]
39 COUNT_LIB SET LIB_USERDEF
40
41 IFNC
42 COUNT_LIB SET '\1', '\1
43 ENDC
44
45 ENDM
46
47
48 *----- LIBDEF is used to define each library function entry:
49
50 LIBDEF MACRO * libraryFunctionSymbol
51 EQU COUNT_LIB
52 SET COUNT_LIB-LIB_VECTSIZE
53 ENDM
54
55 *-----

```

```

57 * Standard Library Functions
58 *
59 *-----
60 *
61 LIBINIT LIB_BASE
62
63 LIBDEF LIB_OPEN
64 LIBDEF LIB_CLOSE
65 LIBDEF LIB_EXPUNCE
66 LIBDEF LIB_EXTFUNC
67 * reserved *
68
69
70 *-----
71 *
72 * Standard Library Data Structure
73 *
74 *-----
75
76 STRUCTURE LIB_LN_SIZE
77 UBYTE LIB_FLAGS
78 UBYTE LIB_pad
79 UMORD LIB_NECSIZE
80 UMORD LIB_POSSIZE
81 UMORD LIB_VERSION
82 UMORD LIB_REVISION
83 APTR LIB_IDSTRING
84 ULONG LIB_SUM
85 UMORD LIB_OPENCNT
86 LABEL LIB_SIZE
87
88 *----- LIB_FLAGS bit definitions:
89
90 BITDEF LIB_SUMMING, 0
91 BITDEF LIB_CHANGED, 1
92 BITDEF LIB_SUMUSED, 2
93 BITDEF LIB_DELEXP, 3
94
95 *-----
96 *
97 * Function Invocation Macros
98 *
99 *-----
100 *
101 *-----
102 *
103 *----- CALLLIB for calling functions where A6 is already correct:
104 CALLLIB MACRO * functionOffset
105 IFGT NARG-1
106 FAIL !!! CALLLIB MACRO - too many arguments !!!
107 ENDC
108 JSR \1(A6)
109 ENDM
110
111
112

```

```

113 *----- LINKLIB for calling functions where A6 is incorrect:
114 LINKLIB MACRO * functionOffset,libraryBase
115 IFGT MARG-2
116 FAIL !!! LINKLIB MACRO - too many arguments !!!
117 ENDC
118
119 MOVE.L A6,-(SP)
120 MOVE.L \2,A6
121 CALLLIB \1
122 MOVE.L (SP)+,A6
123 ENDM
124
125 ENDC IEXEC_LIBRARIES_I

```

```

1 IFND EXEC_LISTS_I
2 EXEC_LISTS_I SET 1
3 *****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *****
8 *
9 * Source Control:
10 *
11 * $Header: lists.i,v 1.1 85/09/06 15:49:56 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****
16
17 IFND EXEC_NODES_I
18 INCLUDE "exec/nodes.i"
19 ENDC IEXEC_NODES_I
20
21
22 -----
23 *
24 * List Structures
25 *
26 -----
27
28 STRUCTURE LH,0
29 APTR LH_HEAD
30 APTR LH_TAIL
31 APTR LH_TAILPRED
32 UBYTE LH_TYPE
33 UBYTE LH_pad
34 LABEL LH_SIZE
35
36
37 NEWLIST MACRO * list
38 MOVE.L \1,(N1)
39 ADDQ.L #LH_TAIL,(N1)
40 CLR.L LH_TAIL,(N1)
41 MOVE.L \1,(LH_TAIL+LN_PRED)(N1)
42 ENDM
43
44
45 TSTLIST MACRO * [list]
46 IFC '\1',, [list]
47 CMP.L LH_TAIL+LN_PRED(A0),A0
48 ENDC
49 IFNC '\1',,
50 CMP.L LH_TAIL+LN_PRED(N1),\1
51 ENDC
52 ENDM
53 $SUCC MACRO * node,succ
54 MOVE.L (N1),\2
55 ENDM
56

```



```

57 FRED          MACRO * node,pred
58   MOVE.L LN_PRED(\1).\2
59   ENDM
60
61 IFEMPTY      MACRO * list,label
62   CMP.L LH_TAIL+LN_PRED(\1).\1
63   BEQ     \2
64   ENDM
65
66 IFNOTEMPTY  MACRO * list,label
67   CMP.L LH_TAIL+LN_PRED(\1).\1
68   BNE     \2
69   ENDM
70
71 TSTNODE     MACRO * node,next
72   MOVE.L (\1).\2
73   TST.L  (\2)
74   ENDM
75
76 NEXTNODE   MACRO * next,current,exit_label (DX,AX,DISP16)
77   MOVE.L (\1).\2
78   MOVE.L (\2).\1
79   BEQ     \3
80   ENDM
81
82 ADDRHEAD   MACRO
83   MOVE.L (A0),D0
84   MOVE.L A1,(A0)
85   MOVEM.L D0/A0,(A1)
86   MOVE.L D0,A0
87   MOVE.L A1,LN_PRED(A0)
88   ENDM
89
90 ADDTAIL    MACRO
91   LEA LH_TAIL(A0),A0
92   MOVE.L LN_PRED(A0),D0
93   MOVE.L A1,LN_PRED(A0)
94   MOVE.L A0,(A1)
95   MOVE.L D0,LN_PRED(A1)
96   MOVE.L D0,A0
97   MOVE.L A1,(A0)
98   ENDM
99
100 REMOVE    MACRO
101   MOVE.L (A1),A0
102   MOVE.L LN_PRED(A1),A1
103   MOVE.L A0,(A1)
104   MOVE.L A1,LN_PRED(A0)
105   ENDM
106
107 REMHEAD   MACRO
108   MOVE.L (A0),A1
109   MOVE.L (A1),D0
110   BEQ.S REMHEAD@
111   MOVE.L D0,(A0)
112   EXG.L D0,A1

```

```

113 REMHEAD@
114   MOVE.L A0,LN_PRED(A1)
115   ENDM
116
117 -----
118 * REMHEADQ -- remove-head quickly
119 *
120 * Useful when a scratch register is available, and
121 * list is known to contain at least one node.
122 *
123 -----
124
125 REMHEADQ   MACRO * head,node,scratchReg
126   MOVE.L (\1).\2
127   MOVE.L (\2).\3
128   MOVE.L \3(\1)
129   MOVE.L \1,LN_PRED(\3)
130   ENDM
131
132 REMTAIL    MACRO
133   MOVE.L LH_TAIL+LN_PRED(A0),A1
134   MOVE.L LN_PRED(A1),D0
135   BEQ.S REMTAIL@
136   MOVE.L D0,LH_TAIL+LN_PRED(A0)
137   EXG.L D0,A1
138   MOVE.L A0,(A1)
139   ADDQ.L #4,(A1)
140   ENDM
141 REMTAIL@
142   ENDM
143
144 ENDC !EXEC_LISTS_1

```

```

1  IFND EXEC_MEMORY_I
2  EXEC_MEMORY_I SET 1
3  *****
4  *
5  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6  *****
7  *
8  * Source Control:
9  *
10 *
11 * $Header: memory.1.v 1.0 85/08/28 15:12:02 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****
16
17 IFND EXEC_NODES_I
18 INCLUDE "exec/nodes.1"
19 ENDC IEXEC_NODES_I
20
21
22 -----
23 *
24 * Memory List Structures
25 *
26 -----
27 *
28 * A memory list appears in two forms: One is a requirements list*
29 * the other is a list of already allocated memory. The format is
30 * the same, with the requirements/address field occupying the same
31 * position.
32 *
33 * The format is a linked list of ML structures each of which has
34 * an array of ME entries.
35 *
36 -----
37
38 STRUCTURE ML, LN_SIZE
39 UMORD ML_NUMENTRIES
40 LABEL ML_ME
41 LABEL ML_SIZE
42
43
44 STRUCTURE ME, 0
45 LABEL ME_REQS
46 APTR ME_ADDR
47 *
48 ULONG ME_LENGTH
49 LABEL ME_SIZE
50
51 *----- memory options:
52
53 BITDEF MEM_PUBLIC, 0
54 BITDEF MEM_CHIP, 1
55 BITDEF MEM_FAST, 2
56

```

```

57 BITDEF MEM_CLEAR, 16
58 BITDEF MEM_LARGEST, 17
59
60
61 *----- alignment rules for a memory block:
62
63 MEM_BLOCKSIZE EQU 8
64 MEM_BLOCKMASK EQU (MEM_BLOCKSIZE-1)
65
66 -----
67 *
68 *
69 * Memory Region Header
70 *
71 -----
72
73 STRUCTURE MH, LN_SIZE
74 UMORD MH_ATTRIBUTES
75 APTR MH_FIRST
76 APTR MH_LOWER
77 APTR MH_UPPER
78 ULONG MH_FREE
79 LABEL MH_SIZE
80
81
82 -----
83 *
84 * Memory Chunk
85 *
86 -----
87
88 STRUCTURE MC, 0
89 APTR MC_NEXT
90 ULONG MC_BYTES
91 APTR MC_SIZE
92
93 ENDC IEXEC_MEMORY_I

```

* characteristics of this region
* first free region
* lower memory bound
* upper memory bound+1
* number of free bytes

* ptr to next chunk
* chunk byte size

```

1  IFND EXEC_NODES_I
2  EXEC_NODES_I SET 1
3  *****
4  *
5  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: nodes.i,v 1.1 85/11/12 18:23:08 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****
16
17
18 -----
19 * List Node Structure
20 *
21 *
22 *
23 *
24 STRUCTURE LN 0
25 APTX LN_SUCC EQU 0
26 APTX LN_PRED EQU 1
27 UBYTE LN_TYPE EQU 2
28 BYTE LN_PRI EQU 3
29 APTX LN_NAME EQU 4
30 LABEL LN_SIZE EQU 5
31
32
33 *----- Node Types:
34 NT_UNKNOWN EQU 6
35 NT_TASK EQU 7
36 NT_INTERRUPT EQU 8
37 NT_DEVICE EQU 9
38 NT_MESSAGE EQU 10
39 NT_RESOURCE EQU 11
40 NT_REPLMSG EQU 12
41 NT_LIBRARY EQU 13
42 NT_MEMORY EQU 14
43 NT_SOFTINT EQU 15
44 NT_FONT EQU 16
45 NT_PROCESS EQU 17
46 NT_SEMAPHORE EQU 18
47
48
49
50
51 END EXEC_NODES_I

```

```

1  IFND EXEC_PORTS_I
2  EXEC_PORTS_I SET 1
3  *****
4  *
5  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: ports.i,v 1.1 85/11/12 18:12:24 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****
16
17
18 -----
19 * Message Port Structure
20 *
21 *
22 *
23 *
24 STRUCTURE MP, LN_SIZE EQU MP_SICTASK
25 UBYTE MP_FLAGS EQU MP_SICTASK
26 APTX MP_SICBIT EQU MP_SICTASK
27 STRUCT MP_MSGLIST, LH_SIZE EQU MP_SICTASK
28 LABEL MP_SIZE EQU MP_SICTASK
29
30
31 *----- unions:
32
33 * signal bit number
34 * task to be signalled
35 * message linked list
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```

```

57 *-----
58 *
59 * Message Structure
60 *
61 *-----
62 *
63 STRUCTURE MN_LEN_SIZE
64 APTR MN_REPLYPORT * message reply port
65 UMWORD MN_LENGTH * message len in bytes
66 LABEL MN_SIZE
67
68 *-----
69 *
70 * Semaphore Message Port
71 *
72 *-----
73 *
74
75 STRUCTURE SM_MP_SIZE
76 WORD SM_BIDS * number of bids for lock
77 LABEL SM_SIZE
78
79 *----- unions:
80
81 SM_LOCKMSC EQU MP_SIGTASK
82
83
84 ENDC !EXEC_PORTS_1

```

```

1 IFND EXEC_RESIDENT_I
2 EXEC_RESIDENT_I SET 1
3 *****
4 *
5 * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6 *
7 *****
8 *
9 * Source Control:
10 *
11 * $Header: resident.i.v 1.0 85/08/28 15:13:41 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****
16 *
17 *-----
18 *
19 * Resident Module Tag
20 *
21 *-----
22 *
23
24 STRUCTURE RT_0
25 UMWORD RT_MATCHWORD * word to match
26 APTR RT_MATCHTAG * pointer to structure base
27 APTR RT_SKIP * address to continue scan
28 UBYTE RT_FLAGS * various tag flags
29 UBYTE RT_VERSION * release version number
30 UBYTE RT_TYPE * type of module
31 BYTE RT_PRI * initialization priority
32 APTR RT_NAME * pointer to node name
33 APTR RT_IDSTRING * pointer to id string
34 LABEL RT_INIT * pointer to init code
35
36
37 *----- Match word definition:
38
39 RTC_MATCHWORD EQU #4AFC * (ILLEGAL instruction)
40
41 *----- RT_FLAGS bit and field definitions:
42
43 BITDEF RT_COLDSTART,0 * RT_INIT points to data
44 BITDEF RT_AUTOINIT,7
45
46 * Compatibility:
47 RTM_WHEN EQU 1 * field position in RT_FLAGS
48 RTM_NEVER EQU 0 * never ever init
49 RTM_COLDSTART EQU 1 * init at coldstart time
50
51 ENDC !EXEC_RESIDENT_I

```

```

1  IFND EXEC_STRINGS_I
2  EXEC_STRINGS_I SET 1
3  *****
4  *
5  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: strings.i.v 1.0 85/08/28 15:14:06 carl Exp $
12 *
13 * $Locker: $
14 *
15 *****
16
17
18 ----- Terminal Control:
19
20 EQU 0
21 BELL EQU 7
22 LF EQU 10
23 CR EQU 13
24 BS EQU 8
25 DEL EQU $7E
26 NL EQU LF
27
28
29 -----
30 *
31 * String Support Macros
32 *
33 -----
34
35 STRING MACRO
36 DC.B \1
37 DC.B 0
38 CNOP 0,2
39 ENDM
40
41
42 STRING MACRO
43 DC.B 13,10
44 DC.B \1
45 DC.B 0
46 CNOP 0,2
47 ENDM
48
49
50 STRING MACRO
51 DC.B \1
52 DC.B 13,10,0
53 CNOP 0,2
54 ENDM
55
56

```

```

57 STRINGER MACRO
58 DC.B 13,10
59 DC.B \1
60 DC.B 13,10,0
61 CNOP 0,2
62 ENDM
63
64 ENDC !EXEC_STRINGS_I

```

```

1  IFEND EXEC_TASKS_I
2  EXEC_TASKS_I SET 1
3  *****
4  *
5  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: tasks.1.v 1.0 85/08/28 15:14:32 carl Exp $
12 *
13 * $Locker: $
14 *
15 *
16 *
17 IFEND EXEC_NODES_I
18 INCLUDE "exec/nodes.1"
19 ENDC IEXEC_NODES_I
20
21 IFEND EXEC_LISTS_I
22 INCLUDE "exec/lists.1"
23 ENDC IEXEC_LISTS_I
24
25
26 -----
27 *
28 * Task Control Structure
29 *
30 -----
31
32 STRUCTURE TC_LN_SIZE
33 UBYTE TC_FLAGS
34 UBYTE TC_STATE
35 BYTE TC_IDNESIGNT
36 BYTE TC_IDNESIGNT
37 ULONG TC_SIGNALLOC
38 ULONG TC_SIGWAIT
39 ULONG TC_SIGRECDV
40 ULONG TC_SIGEXCEPT
41 UNORD TC_TRAPALLO
42 UNORD TC_TRAPABLE
43 APTX TC_EXCEPTDATA
44 APTX TC_EXCEPTCODE
45 APTX TC_TRAPDATA
46 APTX TC_TRAPCODE
47 APTX TC_SPREG
48 APTX TC_SPLOWER
49 APTX TC_SPUPPER
50 APTX TC_SWITCH
51 APTX TC_LAUNCH
52 STRUCT TC_MENENTRY, LH_SIZE
53 APTX TC_Userdata
54 LABEL
55 TC_SIZE
56
* intr disabled nesting
* task disabled nesting
* sigs allocated
* sigs we are waiting for
* sigs we have received
* sigs we take as exceptions
* traps allocated
* traps enabled
* data for except proc
* exception procedure
* data for proc trap proc
* proc trap procedure
* stack pointer
* stack lower bound + 2
* stack upper bound
* task losing CPU
* task getting CPU
* allocated memory

```

```

57 *----- Flag Bits:
58
59 BITDEF T_PROCTIME, 0
60 BITDEF T_STACKCHK, 4
61 BITDEF T_EXCEPT, 5
62 BITDEF T_SWITCH, 6
63 BITDEF T_LAUNCH, 7
64
65
66 *----- Task States:
67 TS_INVALID EQU 0
68 TS_ADDED EQU TS_INVALID+1
69 TS_RUN EQU TS_ADDED+1
70 TS_READY EQU TS_RUN+1
71 TS_WAIT EQU TS_READY+1
72 TS_EXCEPT EQU TS_WAIT+1
73 TS_REMOVED EQU TS_EXCEPT+1
74
75
76 *----- System Task Signals:
77
78 SIGC_ABORT EQU $0001
79 SIGC_CHILD EQU $0002
80 SIGC_FLIT EQU $0010
81 SIGC_DOS EQU $0100
82
83 SIGB_ABORT EQU 0
84 SIGB_CHILD EQU 1
85 SIGB_FLIT EQU 4
86 SIGB_DOS EQU 8
87
88
89 SYS_SIGALLO EQU $0FFF
90 SYS_TRAPALLO EQU $0800
91
92 ENDC IEXEC_TASKS_I

```

; pre-allocated signals
; pre-allocated traps

```

1  IFND EXEC_TYPES_I
2  EXEC_TYPES_I SET 1
3  *****
4  *
5  * Commodore-Amiga, Inc. -- ROM Operating System Executive Include File
6  *
7  *
8  *
9  * Source Control:
10 *
11 * $Header: types.i,v 1.2 85/11/15 17:44:08 carl Exp $
12 *
13 * $Locker:  $
14 *
15 *****
16
17 EXTERN_LIB MACRO _JVO\1
18 XREF
19 ENDM
20
21 STRUCTURE MACRO
22 \1 SET 0 * for assembler's sake
23 SOFFSET SET \2
24 ENDM
25
26 BOOL MACRO
27 \1 SOFFSET SOFFSET+2
28 SOFFSET SET
29 ENDM
30
31 BYTE MACRO
32 \1 EQU SOFFSET
33 SOFFSET SET SOFFSET+1
34 ENDM
35
36 UBYTE MACRO
37 \1 EQU SOFFSET
38 SOFFSET SET SOFFSET+1
39 ENDM
40
41 WORD MACRO
42 \1 EQU SOFFSET
43 SOFFSET SET SOFFSET+2
44 ENDM
45
46 UWORD MACRO
47 \1 EQU SOFFSET
48 SOFFSET SET SOFFSET+2
49 ENDM
50
51 SHORT MACRO
52 \1 EQU SOFFSET
53 SOFFSET SET SOFFSET+2
54
55 USHORT MACRO

```

```

57 \1 SOFFSET
58 EQU SET SOFFSET+2
59 SET ENDM
60
61 LONG MACRO
62 \1 EQU SOFFSET
63 SET SOFFSET+4
64 SET ENDM
65
66 ULONG MACRO
67 \1 EQU SOFFSET
68 SET SOFFSET+4
69 SET ENDM
70
71 FLOAT MACRO
72 \1 EQU SOFFSET
73 SET SOFFSET+4
74 SET ENDM
75
76 APTR MACRO
77 \1 EQU SOFFSET
78 SET SOFFSET+4
79 SET ENDM
80
81 RPTR MACRO
82 \1 EQU SOFFSET
83 SET SOFFSET+2
84 SET ENDM
85
86 STRUCT MACRO
87 \1 EQU SOFFSET
88 SET SOFFSET+2
89 SET ENDM
90
91 LABEL MACRO
92 \1 EQU SOFFSET
93 ENDM
94
95 *----- bit definition macro -----
96 *
97 * Given:
98 *
99 * BITDEF MEM,CLEAR,16
100 *
101 * Yields:
102 *
103 * MEMB_CLEAR EQU 16
104 * MEME_CLEAR EQU (1.SL.MEMB_CLEAR)
105 *
106
107 BITDEF MACRO * prefix,&name,&bitnum
108 BITDEF0 \1,\2,B,\3
109 SET 1<<&3
110 BITDEF0 \1,\2,F,-,@BITDEF
111 ENDM
112

```

```
113 BITDEF0 MACRO * prefix, &name, &type, &value  
114 \1\3\2 EQU \4  
115 ENDM  
116  
117 LIBRARY_VERSION EQU 31  
118  
119 ENDC EXEC_TYPES_I
```



```

67 ISLESSY equ 2
68 ISCRTRX equ 4
69 ISCRTRY equ 8
70
71 ENDC

```

```

1 IFND GRAPHICS_CLIP_I
2 GRAPHICS_CLIP_I SET 1
3 *****
4 * Commodore-Amiga, Inc.
5 * clip.i
6 *****
7
8 IFND GRAPHICS_GEX_I
9 include 'graphics/gfx.i'
10 ENDC
11 IFND EXEC_PORTS_I
12 include 'exec/ports.i'
13 ENDC
14
15 STRUCTURE
16 Layer, 0
17 lr_Front
18 lr_Back
19 lr_ClipRect
20 lr_RastPort
21 WORD lr_MinX
22 WORD lr_MaxX
23 WORD lr_MaxY
24 BYTE lr_Lock
25 BYTE lr_LockCount
26 BYTE lr_LayerLockCount
27 BYTE lr_reserved1
28 WORD lr_reserved1
29 WORD lr_flags
30 LONG lr_SuperBitMap
31 LONG lr_SuperClipRect
32 LONG lr_Window
33 WORD lr_Scroll_X
34 WORD lr_Scroll_Y
35 STRUCT lr_LockPort, MP_SIZE
36 STRUCT lr_LockMessage, MN_SIZE
37 STRUCT lr_ReplyPort, MP_SIZE
38 STRUCT lr_l_LockMessage, MN_SIZE
39 APTR lr_DamagedList
40 APTR lr_cliprects
41 APTR lr_LayerInfo
42 APTR lr_LayerLocker
43 APTR lr_SuperSaverClipRects
44 APTR lr_cr
45 APTR lr_crnew
46 APTR lr_pl
47 APTR lr_pl
48 LABEL lr_SIZEOF
49
50 STRUCTURE ClipRect, 0
51 LONG cr_Next
52 LONG cr_Prev
53 LONG cr_Lobs
54 LONG cr_BitMap
55 WORD cr_MinX
56 WORD cr_MinY
57 WORD cr_MaxX
58 WORD cr_MaxY
59 APTR cr_p1
60 APTR cr_p2
61 LONG cr_reserved
62 LONG cr_flags
63 LABEL cr_SIZEOF
64
65 * defines for clipping
66 ISLESSY equ 1

```

```

1  IFND GRAPHICS_COPPER_I
2  COPPER_WAIT equ 1
3  GRAPHICS_COPPER_I SET 1
4  *****
5  * Commodore-Amiga, Inc.
6  *
7  *****
8
9  COPPER_MOVE equ 0 /* pseudo opcode for move #XXXX,dir */
10 COPPER_WAIT equ 1 /* pseudo opcode for wait y,x */
11 CPRNKTBUF equ 2 /* continue processing with next buffer */
12 CPR_NT_LOF equ $8000 /* copper instruction only for short frames */
13 CPR_NT_SHT equ $4000 /* copper instruction only for long frames */
14
15 STRUCTURE CopIns,0
16 WORD c1_OpCode * 0 = move, 1 = wait */
17 STRUCT c1_NxtList,0 * UNION
18 STRUCT c1_WaitPos,0
19 STRUCT c1_DestAddr,2
20
21 STRUCT c1_HWaitPos,0
22 STRUCT c1_DestData,2
23
24 LABEL c1_SIZEOF
25
26 * structure of cprlist that points to list that hardware actually executes */
27 STRUCTURE cprlist,0
28 APTR cpl_Next
29 APTR cpl_start
30 WORD cpl_max
31 LABEL cpl_SIZEOF
32
33 STRUCTURE CopList,0
34 APTR cpl_Next; /* next block for this copper list */
35 APTR cpl_CopList; /* system use */
36 APTR cpl_ViewPort *ViewPort; /* system use */
37 APTR cpl_CopIns; /* start of this block */
38 APTR cpl_CopPtr; /* intermediate ptr */
39 APTR cpl_CopLStart; /* mrgcop fills this in for Long Frames*/
40 APTR cpl_CopSStart; /* mrgcop fills this in for Short Frames*/
41 WORD cpl_Count; /* intermediate counter */
42 WORD cpl_MaxCount; /* max # of copins for this block */
43 WORD cpl_DyOffset; /* offset this copper list vertical waits */
44 LABEL c1_SIZEOF
45
46 STRUCTURE UCopList,0
47 APTR ucl_Next;
48 APTR ucl_FirstCopList; /* head node of this copper list */
49 APTR ucl_CopList; /* node in use */
50 LABEL ucl_SIZEOF
51
52 * private graphics data structure
53 STRUCTURE copinit,0
54 STRUCT copinit_diagstrt,8
55 STRUCT copinit_sprstrtp,2*(2*8*2)+2*(2*2)+2
56 STRUCT copinit_sprstop,4

```

```

57 LABEL copinit_SIZEOF
58
59 ENDC

```

```

1  IFND  GRAPHICS_DISPLAY_1
2  GRAPHICS_DISPLAY_1 SET 1
3  ***** display.1 *****
4  *
5  * Commodore-Amiga, Inc.
6  *
7  * Modification History
8  * date : author : Comments
9  * -----
10 * 8-24-84 Dale added this header file
11 *
12 *****
13 * include define file for display control registers */
14 * bplcon0 defines */
15 MODE_G40 equ $8000
16 PLCNWMSK equ $7
17
18 * how many bit planes? */
19 PLCNWMSHET equ 12
20 * bits to shift for bplcon0 */
21 PE2PRI equ $40
22 * bplcon2 bit */
23 COLORON equ $0200
24 * disable color burst */
25 DELPF equ $400
26 HOLDNMODIFY equ $800
27 INTERLACE equ 4
28
29 * bplcon1 defines */
30 PFA_FINE_SCROLL equ $F
31 PFB_FINE_SCROLL_SHIFT equ 4
32 PFC_FINE_SCROLL_MASK equ $F
33
34 * display window start and stop defines */
35 DIW_HORIZ_POS equ $7F * horizontal start/stop */
36 DIH_VERTCL_POS equ $1FF * vertical start/stop */
37 DIW_VERTCL_POS_SHIFT equ 7
38
39 * Data fetch start/stop horizontal position */
40 DFCTL_MASK equ $FF
41
42 * vposr bits */
43 VPOSRL0F equ $8000
44
45 ENDC

```

```

1  IFND  GRAPHICS_GELS_1
2  GRAPHICS_GELS_1 SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  * Graphics Library : Gels Definitions
6  *
7  *
8  *
9  *
10 * ----- VS_vSflags -----
11 *
12 * --- user-set vSprite flags ---
13 SUSERFLACS EQU $00FF ; mask of all user-settable vSprite-flags
14 BITDEF VS_VSPRITE,0 ; set if vSprite, clear if bob
15 BITDEF VS_SAVEBACK,1 ; set if background is to be saved/restored
16 BITDEF VS_OVERLAY,2 ; set to mask image of bob onto background
17 BITDEF VS_MUSTDRAW,3 ; set if vSprite absolutely must be drawn
18 * --- system-set vSprite flags ---
19 BITDEF VS_BACKSAVED,8 ; this bob's background has been saved
20 BITDEF VS_BOBUPDATE,9 ; temporary flag, useless to outside world
21 BITDEF VS_GELCONE,10 ; set if gel is completely clipped (offscreen)
22 BITDEF VS_VSOVERFLOW,11 ; vSprite overflow (if MUSTDRAW set we draw!)
23
24 *
25 * ----- R_flags -----
26 * --- these are the user flag bits ---
27 BUSERFLACS EQU $00FF ; mask of all user-settable bob-flags
28 BITDEF B_SAVERBOB,0 ; set to not erase bob
29 BITDEF B_BOBISCOMP,1 ; set to identify bob as animComp
30 * --- these are the system flag bits ---
31 BITDEF B_WAITING,8 ; set while bob is waiting on 'after'
32 BITDEF B_ERAMN,9 ; set when bob is drawn this DrawG pass
33 BITDEF B_BOBSAWAY,10 ; set to initiate removal of bob
34 BITDEF B_BOBNIX,11 ; set when bob is completely removed
35 BITDEF B_SAVEPRESERVE,12 ; for back-restore during double-buffer
36 BITDEF B_OUTSTEP,13 ; for double-clearing if double-buffer
37
38 * ----- defines for the animation procedures -----
39
40 ANERACSIZE EQU 6
41 ANIMHALF EQU $0020
42 RINGTRIGGER EQU $0001
43
44 * ----- macros -----
45 * these are GEL functions that are currently simple enough to exist as a
46 * definition. It should not be assumed that this will always be the case
47
48 InitAnimate MACRO * animKey
49 CLR.L \1
50 ENDM
51
52 RemBob OR.W #BF_BOBSAWAY,b_BobFlags*\1
53 ENDM
54
55
56

```

```

57 * ----- VS : vSprite -----
58 * STRUCTURE VS_0 ; vSprite
59 * -- SYSTEM VARIABLES --
60 *
61 * CEL linked list forward/backward pointers sorted by y,x value
62 * PTR vs_NextVSprite ; struct *vSprite
63 * PTR vs_PrevVSprite ; struct *vSprite
64 * CEL draw list constructed in the order the bobs are actually drawn, then
65 * list is copied to clear list
66 * must be here in vSprite for system boundary detection
67 * PTR vs_DrawPath ; struct *vSprite: pointer to overlay drawing
68 * PTR vs_ClearPath ; struct *vSprite: pointer for overlay clearing
69 * the vSprite positions are defined in (y,x) order to make sorting
70 * sorting easier, since (y,x) as a long integer
71 * WORD vs_OldY ;
72 * WORD vs_OldX ; previous position
73 * -- COMMON VARIABLES --
74 * WORD vs_VSFlags ; vSprite flags
75 * -- USER VARIABLES --
76 * the vSprite positions are defined in (y,x) order to make sorting
77 * easier, since (y,x) as a long integer
78 * WORD vs_Y ;
79 * WORD vs_X ; screen position
80 *
81 * WORD vs_Height ; number of words per row of image data
82 * WORD vs_Width ; number of planes of data
83 * WORD vs_Depth ; which types can collide with this vSprite
84 * WORD vs_Mask ; which types this vSprite can collide with
85 * PTR vs_ImageData ; *WORD pointer to vSprite image
86 * borderLine is the one-dimensional logical OR of all
87 * the vSprite bits, used for fast collision detection of edges
88 * PTR vs_BorderLine ; *WORD: logical OR of all vSprite bits
89 * PTR vs_CollMask ; *WORD: similar to above except this is a
90 * matrix pointer to this vSprite's color definitions (not used by bobs)
91 * PTR vs_SprColors ; *WORD
92 * PTR vs_VSBob ; struct *bob: points home if this vSprite is
93 * part of a bob
94 * planePick flag: set bit selects a plane from image, clear bit selects
95 * use of shadow mask for that plane
96 * OnOff flag: if using shadow mask to fill plane, this bit (corresponding
97 * to bit in planePick) describes whether to fill with 0's or 1's
98 * There are two uses for these flags:
99 * - if this is the vSprite of a bob, these flags describe how
100 * the bob is to be drawn into memory
101 * - if this is a simple vSprite and the user intends on setting
102 * the MUSTDRAW flag of the vSprite, these flags must be set
103 * too to describe which color registers the user wants for
104 * the image
105 * BYTE vs_PlanePick
106 * BYTE vs_PlaneOff
107 * LABEL vs_UserExt ; user definable
108 * LABEL vs_SIZEOF
109 *
110 *
111 *
112 * ----- BOB : bob -----

```

```

113 * STRUCTURE BOB_0 ; bob: blitter object
114 * -- COMMON VARIABLES --
115 * PTR bob_SavePlanes ; * *WORD for each plane in RastPort
116 * WORD bob_BobFlags ; general purpose flags (see definitions below
117 * -- USER VARIABLES --
118 * PTR bob_SaveBuffer ; *WORD pointer to the buffer for background
119 * save used by bobs for "cookie-cutting" and multi-plane masking
120 * PTR bob_ImageShadow ; *WORD
121 * pointer to BOBs for sequenced drawing of bobs
122 * for correct overlaying of multiple component animations
123 * PTR bob_Before ; struct *bob: draw this bob before bob pointed
124 * to by before
125 * PTR bob_After ; struct *bob: draw this bob after bob pointed
126 * to by after
127 * PTR bob_BobVSprite ; struct *vSprite: this bob's vSprite default
128 * PTR bob_BobComp ; struct *animComp: pointer to this bob's
129 * animComp def
130 * PTR bob_DBuffer ; struct dBufPacket: pointer to this bob's
131 * dBuf packet
132 * LABEL bob_UserExt ; bob user extension
133 * LABEL bob_SIZEOF
134 *
135 * ----- AC : animComp -----
136 *
137 * STRUCTURE AC_0 ; animComp
138 * -- COMMON VARIABLES --
139 * WORD ac_CompFlags ; animComp flags for system & user
140 * timer defines how long to keep this component active:
141 * if set non-zero, timer decrements to zero then switches to nextSeq
142 * if set to zero, animComp never switches
143 * WORD ac_Timer
144 * -- USER VARIABLES --
145 * initial value for timer when the animComp is activated by the system
146 * WORD ac_TimeSet
147 * pointer to next and previous components of animation object
148 * PTR ac_NextComp ; struct *animComp
149 * PTR ac_PrevComp ; struct *animComp
150 * pointer to component definition of next image in sequence
151 * PTR ac_NextSeq ; struct *animComp
152 * PTR ac_PrevSeq ; struct *animComp
153 * PTR ac_AnimCRoutine ; address of special animation procedure
154 * WORD ac_YTrans ; initial y translation (if this is a componen
155 * WORD ac_XTrans ; initial x translation (if this is a componen
156 * PTR ac_HeadOb ; struct *animOb
157 * PTR ac_AnimBob ; struct *animOb
158 * LABEL ac_SIZE
159 *
160 * ----- AO : animOb -----
161 *
162 * STRUCTURE AO_0 ; animOb
163 * -- SYSTEM VARIABLES --
164 * PTR ao_NextOb ; struct *animOb
165 * PTR ao_PrevOb ; struct *animOb
166 * number of calls to Animate this animOb has endured
167 * LONG ao_Clock
168 * WORD ao_ArOldY ; old y,x coordinates

```

```

169 *      WORD   ao_AnOldX      ;
170 *      -- COMMON VARIABLES --
171 WORD   ao_AnY
172 WORD   ao_AnX
173 *      -- USER VARIABLES --
174 WORD   ao_VVel
175 WORD   ao_XVel
176 WORD   ao_YAccel
177 WORD   ao_XAccel
178 WORD   ao_RingYTrans
179 WORD   ao_RingXTrans
180 APTR   ao_AnimCRoutine
181 APTR   ao_HeadComp
182 LABEL  ao_AUserExt
183 LABEL  ao_SIZEOF
184
185
186 *----- DEP : dBufPacket -----
187 * dBufPacket defines the values needed to be saved across buffer to buffer
188 * when in double-buffer mode
189
190 STRUCTURE DEP_0
191 WORD   dbp_BufY
192 WORD   dbp_BufX
193 APTR   dbp_BufPath
194
195 * these pointers must be filled in by the user
196 * pointer to other buffer's background save buffer
197 APTR   dbp_BufBuffer ; *WORD
198 * pointer to other buffer's background plane pointers
199 APTR   dbp_BufPlanes ; **WORD
200 LABEL  dbp_SIZEOF
201
202 ENDC

```

```

1 *****
2 *      Commodore-Amiga, Inc.
3 *      gfx.1
4 *
5 *
6 *****
7 IFND GRAPHICS_GFX_I
8 GRAPHICS_GFX_I SET 1
9
10 BITSET equ $8000
11 BITCLR equ 0
12 ACNUS equ 1
13 DENISE equ 1
14
15 STRUCTURE BitMap_0
16 WORD   bm_BytesPerRow
17 WORD   bm_Rows
18 BYTE   bm_Flags
19 BYTE   bm_Depth
20 WORD   bm_Pad
21 STRUCT bm_Planes_0*4
22 LABEL  bm_SIZEOF
23
24 STRUCTURE Rectangle_0
25 WORD   ra_MinX
26 WORD   ra_MinY
27 WORD   ra_MaxX
28 WORD   ra_MaxY
29 LABEL  ra_SIZEOF
30
31 ENDC

```

```

57
58 QBOWNER equ 1<<QBOWNERn
59
60 ENDC

```

```

1 ***** gfxbase.1 *****
2 *
3 * Commodore-Amiga, Inc.
4 *
5 *****
6 IFND GRAPHICS_GFXBASE_I
7 GRAPHICS_GFXBASE_I SET 1
8
9 IFND EXEC_LISTS_I
10 include 'exec/lists.1'
11 ENDC
12 IFND EXEC_LIBRARIES_I
13 include 'exec/libraries.1'
14 ENDC
15 IFND EXEC_INTERRUPTS_I
16 include 'exec/interrupts.1'
17 ENDC
18
19 STRUCTURE GfxBase, LIB_SIZE
20 APTR gb_ActivView ; struct 'View
21 APTR gb_copinit ; struct 'copinit ; ptr to copper start up list
22 APTR gb_cia ; for 6526 resource use
23 APTR gb_bltttr ; for blitter resource use
24 APTR gb_LOFlist ; current copper list being run
25 APTR gb_SHElist ; current copper list being run
26 APTR gb_bltbd ; struct 'bltnode
27 APTR gb_blttl
28 APTR gb_bsblltd
29 APTR gb_bsblltl
30 STRUCT gb_vbsrv, IS_SIZE
31 STRUCT gb_tmsrv, IS_SIZE
32 STRUCT gb_bltstrv, IS_SIZE
33 STRUCT gb_TextFonts, LH_SIZE
34 APTR gb_DefaultFont ; copy of bltcon0
35 UNWORD gb_Modes
36 BYTE gb_VBlank
37 BYTE gb_Debug
38 UNWORD gb_BeamSync
39 WORD gb_system_bplcon0
40 BYTE gb_SpriteReserved
41 BYTE gb_bytereserved
42
43 WORD gb_Flags
44 WORD gb_BlitLock
45 WORD gb_BlitWest
46 STRUCT gb_BlitWaitQ, LH_SIZE
47 APTR gb_BlitOwner
48 STRUCT gb_TOF_WaitQ, LH_SIZE
49 WORD gb_DisplayFlags
50
51 STRUCT gb_reserved, 8 ; 8 bytes reserved for future use
52 LABEL gb_SIZE
53
54 * bits for daletstuff, which may go away when blitter becomes a resource
55 ONBLITTERn equ 0 * blitter owned bit
56 QBOWNERn equ 1 * blitter owned by blit queuer

```

```

1 *
2 *
3 *
4 *
5 *
6 *
7 *
8 *
9 *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
24 *
25 *
26 *
27 *
28 *
29 *
30 *
31 *
32 *
33 *
34 *
35 *
36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *

```

Commodore-Amiga, Inc.
layers.1

```

IFND GRAPHICS_LAYERS_I
  GRAPHICS_LAYERS_I SET 1
IFND EXEC_PORTS_I
  include 'exec/ports.1'
ENDC
IFND EXEC_LISTS_I
  include 'exec/lists.1'
ENDC
STRUCTURE LayerInfo_extra, 0
STRUCT  lie_env, 13*4
STRUCT  lie_mem, LH_SIZE
APTR    lie_FreeClipRects
APTR    lie_bitbuff
LABEL   lie_SIZEOF
LH_REGION equ -1
STRUCTURE mem_node, 0
APTR    memnode_succ
APTR    memnode_pred
APTR    memnode_where
LONG    memnode_how_big
LABEL   memnode_SIZEOF
STRUCTURE Layer_Info, 0
APTR    li_top_layer
APTR    li_check_lp
APTR    li_obs
STRUCT  li_rp_replyPort, MP_SIZE
STRUCT  li_lockPort, MP_SIZE
BYTE    li_lock
BYTE    li_broadcast
BYTE    li_locknest
BYTE    li_pad
APTR    li_locker
STRUCT  li_bytereserved, 2
STRUCT  li_wordreserved, 4
STRUCT  li_longreserved, 4
APTR    li_LayerInfo_extra
LABEL   li_SIZEOF
NEW_LAYERINFO_CALLED equ 1
ENDC

```

```

1 ***** rastport.1 *****
2 *
3 *
4 *
5 *
6 *
7 *
8 *
9 *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
24 *
25 *
26 *
27 *
28 *
29 *
30 *
31 *
32 *
33 *
34 *
35 *
36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *

```

Commodore-Amiga, Inc.

```

IFND GRAPHICS_RASTPORT_I
  GRAPHICS_RASTPORT_I SET 1
IFND GRAPHICS_GFX_I
  include 'graphics/gfx.1'
ENDC
----- TR : TmpRas -----
STRUCTURE TmpRas, 0
APTR    tr_RasPtr
LONG    tr_Size
LABEL   tr_SIZEOF
----- CelsInfo
STRUCTURE CelsInfo, 0
BYTE    gi_sprRsvd
        * flag of which sprites to reserve from
        * vsprite system
        * reserved for system use
BYTE    gi_Flags
APTR    gi_gellHead
APTR    gi_gellTail
        * dummy vSprites for list management
        * pointer to array of 8 WORDS for sprite available lines
APTR    gi_nextLine;
        * pointer to array of 8 pointers for color-last-assigned to vSprites
APTR    gi_lastColor;
        * addresses of collision routines
SHORT   gi_leftmost
SHORT   gi_rightmost
SHORT   gi_topmost
SHORT   gi_bottommost
APTR    gi_firstBlissObj
APTR    gi_lastBlissObj
        * system use only
LABEL   gi_SIZEOF
----- RP_Flags -----
BITDEF RP_FIRST_DOT, 0
        ; draw the first dot of this line ?
BITDEF RP_ONE_DOT, 1
        ; use one dot mode for drawing lines
BITDEF RP_DBUFFER, 2
        ; flag set when RastPorts are double-buffered
        ; (only used for bobs)
BITDEF RP_AREAOUTLINE, 3
        ; used by areafiller
BITDEF RP_NOCROSSFILL, 5
        ; used by areafiller
----- RP_DrawMode -----
RP_JAMI EQU 0
RP_JAN2 EQU 1
RP_COMPLEMENT EQU 2
RP_INVERSVID EQU 4
----- RP_InvFlags -----
BITDEF RP_INVSCALE, 0
        ; inverse video for drawing modes

```

```

57 STRUCTURE RastPort, 0
58 LONG rp_Layer
59 LONG rp_BitMap
60 LONG rp_AreaPtrn
61 LONG rp_TmpRas
62 LONG rp_AreaInfo
63 LONG rp_CelsInfo
64 BYTE rp_Mask
65 BYTE rp_FgPen
66 BYTE rp_BgPen
67 BYTE rp_AOLPen
68 BYTE rp_DrawMode
69 BYTE rp_AreaPtSz
70 BYTE rp_Dummy
71 BYTE rp_LinPatcnt
72 WORD rp_Flags
73 WORD rp_LinePtrn
74 WORD rp_cp_x
75 WORD rp_cp_y
76 STRUCT rp_mainterm, 8
77 WORD rp_PenWidth
78 WORD rp_PenHeight
79 WORD rp_Font
80 BYTE rp_AlgoStyle
81 BYTE rp_TxFlags
82 WORD rp_TxHeight
83 WORD rp_TxWidth
84 WORD rp_TxBaseline
85 WORD rp_TxSpacing
86 APTR rp_RP_User
87 STRUCT rp_wordreserved, 14
88 STRUCT rp_longreserved, 8
89 STRUCT rp_reserved, 8
90 LABEL rp_SIZEOF
91
92 STRUCTURE AreaInfo, 0
93 LONG ai_VctrTbl
94 LONG ai_VctrPtr
95 LONG ai_FlagTbl
96 LONG ai_FlagPtr
97 WORD ai_Count
98 WORD ai_MaxCount
99 WORD ai_FirstX
100 WORD ai_FirstY
101 LABEL ai_SIZEOF
102
103 ONE_DOTn equ 1
104 ONE_DOT equ $2
105 FRST_DOTn equ 0
106 FRST_DOT equ 1
107
108
109 ENDC

```

```

1
2 IFND GRAPHICS_REGIONS_I
3 GRAPHICS_REGIONS_I SET 1
4 *****
5 * Commodore-Amiga, Inc.
6 * regions.i
7 *****
8
9 IFND GRAPHICS_CFX_I
10 include 'graphics/gfx.i'
11 ENDC
12
13 STRUCTURE Region, 0
14 STRUCT rg_bounds, ra_SIZEOF
15 APTR rg_RegionRectangle
16 LABEL rg_SIZEOF
17
18 STRUCTURE RegionRectangle, 0
19 APTR rr_Next
20 APTR rr_Prev
21 STRUCT rr_bounds, ra_SIZEOF
22 LABEL rr_SIZEOF
23
24 ENDC

```



```

1  IFND  GRAPHICS_SPRITE_I
2  GRAPHICS_SPRITE_I SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  * sprite.h
6  *****
7
8  STRUCTURE SimpleSprite,0
9  APTR      ss_posctldata
10 WORD     ss_height
11 WORD     ss_x
12 WORD     ss_y
13 WORD     ss_num
14 LABEL    ss_SIZEOF
15
16 ENDC

```

```

1  IFND  GRAPHICS_TEXT_I
2  GRAPHICS_TEXT_I SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  * text.i
6  *****
7
8  * graphics library text structures
9  *
10 *
11 *
12 *
13 IFND  EXEC_PORTS_I
14 INCLUDE "exec/ports.i"
15 ENDC
16
17 *----- Font Styles
18 FS_NORMAL EQU 0 ;normal text (no style attributes set)
19 BITDEF FS_EXTENDED,3 ;extended face (must be designed)
20 BITDEF FS_ITALIC,2 ;italic (slanted 1:2 right)
21 BITDEF FS_BOLD,1 ;bold face text (OKed w/ shifted right 1)
22 BITDEF FS_UNDERLINED,0 ;underlined (under baseline)
23
24 *----- Font Flags
25 BITDEF FP_ROMFONT,0 ;font is in rom
26 BITDEF FP_DISKFONT,1 ;font is from diskfont.library
27 BITDEF FP_REVPATH,2 ;designed path is reversed (e.g. left)
28 BITDEF FP_TALLOTT,3 ;designed for hires non-interlaced
29 BITDEF FP_WIDEDOT,4 ;designed for lores interlaced
30 BITDEF FP_PROPORTIONAL,5 ;character sizes can vary from nominal
31 BITDEF FP_DESIGNED,6 ;size is "designed", not constructed
32 BITDEF FP_REMOVED,7 ; the font has been removed
33
34
35 ***** TextAttr node *****
36 STRUCTURE TextAttr, 0
37 APTR ta_Name ;name of the desired font
38 UMWORD ta_YSize ;size of the desired font
39 UBYTE ta_Style ;desired font style
40 UBYTE ta_Flags ;font preferences
41 LABEL ta_SIZEOF
42
43
44 ***** TextFont node *****
45 STRUCTURE TextFont, MW_SIZE
46 *
47 UMWORD tf_YSize ;font name in LN \ used in this
48 UBYTE tf_Style ;font height | order to best
49 UBYTE tf_Flags ;font style | match a font
50 UMWORD tf_XSize ;preference attributes / request..
51 UMWORD tf_Baseline ;nomnal font width
52 UMWORD tf_BoldSmear ;distance from the top of char to baseline
53
54 UMWORD tf_Accessors ;smear to affect a bold enhancement
55
56 UBYTE tf_LoChar ;the first character described here

```

```

57 UBYTE tf_HiChar ;the last character described here
58 APTR tf_CharData ;the bit character data
59
60 UWORD tf_Modulo ;the row modulo for the strike font data
61 APTR tf_CharLoc ;ptr to location data for the strike font
62 * ; 2 words: bit offset then size
63 APTR tf_CharSpace ;ptr to words of proportional spacing data
64 APTR tf_CharKern ;ptr to words of kerning data
65 LABEL tf_SIZEOF
66
67 ENDC
    
```

```

1 IFND GRAPHICS_VIEW_I
2 GRAPHICS_VIEW_I SET 1
3 *****
4 * Commodore-Amiga, Inc.
5 * view.i
6 *****
7
8 IFND GRAPHICS_CFX_I
9 include 'graphics/gfx.1'
10 ENDC
11
12 V_PEBA EQU $40
13 V_DUALPF EQU $400
14 V_HIRES EQU $8000
15 V_LACE EQU 4
16 V_HAM EQU $800
17 V_SPRITES EQU $4000
18 GENLOCK_VIDEO EQU 2
19
20 STRUCTURE ColorMap,0
21 BYTE cm_Flags
22 BYTE cm_Type
23 WORD cm_Count
24 APTR cm_ColorTable
25 LABEL cm_SIZEOF
26
27
28 STRUCTURE ViewPort,0
29 LONG vp_Next
30 LONG vp_ColorMap
31 LONG vp_DepIns
32 LONG vp_SprIns
33 LONG vp_ClrIns
34 LONG vp_UCopIns
35 WORD vp_DWidth
36 WORD vp_DHeight
37 WORD vp_DxOffset
38 WORD vp_DyOffset
39 WORD vp_Modes
40 WORD vp_reserved
41 APTR vp_RasInfo
42 LABEL vp_SIZEOF
43
44
45 STRUCTURE View,0
46 LONG v_ViewPort
47 LONG v_LOFCprList
48 LONG v_SHFCprList
49 WORD v_DyOffset
50 WORD v_DxOffset
51 WORD v_Modes
52 LABEL v_SIZEOF
53
54
55 STRUCTURE collTable,0
56 LONG cp_collPtrs,16
    
```

```

57 LABEL cp_SIZEOF
58
59 STRUCTURE RasInfo,0
60 APTR ri_Next
61 LONG ri_BitMap
62 WORD ri_RxOffset
63 WORD ri_RyOffset
64 LABEL ri_SIZEOF
65
66 ENDC
67

```

```

1 *****
2 * adkbits.1 -- bit definitions for adkcon register
3 *
4 * Commodore-Amiga, Inc.
5 *
6 * $Header: adkbits.i.v 27.1 85/06/24 14:42:37 nell Exp $
7 *
8 * $Locker: $
9 *
10 *****
11 IFND HARDWARE_ADKBITS_I
12 HARDWARE_ADKBITS_I SET 1
13
14 ADKB_SETCLR EQU 15 ; standard set/clear bit
15 ADKB_PRECOMP1 EQU 14 ; two bits of precompensation
16 ADKB_PRECOMP0 EQU 13
17 ADKB_MEMPREC EQU 12 ; use mfm style precompensation
18 ADKB_UARTERK EQU 11 ; force uart output to zero
19 ADKB_WORDSYNC EQU 10 ; enable DSKSYNC register matching
20 ADKB_MSBSYNC EQU 9 ; (Apple GCR Only) sync on MSB for reading
21 ADKB_FAST EQU 8 ; 1 -> 2 us/bit (mfm), 2 -> 4 us/bit (gcr)
22 ADKB_USE3FN EQU 7 ; use aud chan 3 to modulate period of ??
23 ADKB_USE2P3 EQU 6 ; use aud chan 2 to modulate period of 3
24 ADKB_USE1P2 EQU 5 ; use aud chan 1 to modulate period of 2
25 ADKB_USE0P1 EQU 4 ; use aud chan 0 to modulate period of 1
26 ADKB_USE3VN EQU 3 ; use aud chan 3 to modulate volume of ??
27 ADKB_USE2V3 EQU 2 ; use aud chan 2 to modulate volume of 3
28 ADKB_USE1V2 EQU 1 ; use aud chan 1 to modulate volume of 2
29 ADKB_USE0V1 EQU 0 ; use aud chan 0 to modulate volume of 1
30
31 ADKF_SETCLR EQU (1<<15)
32 ADKF_PRECOMP1 EQU (1<<14)
33 ADKF_PRECOMP0 EQU (1<<13)
34 ADKF_MEMPREC EQU (1<<12)
35 ADKF_UARTERK EQU (1<<11)
36 ADKF_WORDSYNC EQU (1<<10)
37 ADKF_MSBSYNC EQU (1<<9)
38 ADKF_FAST EQU (1<<8)
39 ADKF_USE3FN EQU (1<<7)
40 ADKF_USE2P3 EQU (1<<6)
41 ADKF_USE1P2 EQU (1<<5)
42 ADKF_USE0P1 EQU (1<<4)
43 ADKF_USE3VN EQU (1<<3)
44 ADKF_USE2V3 EQU (1<<2)
45 ADKF_USE1V2 EQU (1<<1)
46 ADKF_USE0V1 EQU (1<<0)
47
48 ADKE_PRE00NS EQU 0 ; 000 ns of precomp
49 ADKE_PRE140NS EQU (ADKE_PRECOMP0) ; 140 ns of precomp
50 ADKE_PRE280NS EQU (ADKE_PRECOMP1) ; 280 ns of precomp
51 ADKE_PRE560NS EQU (ADKE_PRECOMP0|ADKE_PRECOMP1) ; 560 ns of precomp
52
53 ENDC
54 HARDWARE_ADKBITS_I

```

```

1 *****
2 * Commodore-Amiga, Inc.
3 * blit.1
4 *
5 * $Header: blit.i,v 27.1 85/06/24 14:42:42 neil Exp $
6 *
7 * $Locker:  $
8 *
9 *****
10
11 IFND HARDWARE_BLIT_I
12 HARDWARE_BLIT_I SET 1
13
14 STRUCTURE bitnode,0
15 LONG bn_n
16 LONG bn_function
17 BYTE bn_stat
18 BYTE bn_dummy
19 WORD bn_blitsize
20 WORD bn_beamsync
21 LONG bn_cleanup
22 LABEL bn_SIZEOF
23
24 * bit defines used by blit queuer
25 CLEANMEN equ 6
26 CLEANWE equ 1<<CLEANMEN
27
28 * include file for blitter */
29 HSIZEBITS equ 6
30 VSIZEBITS equ 16-HSIZEBITS
31 HSIZEMASK equ $3f /* 2^6 -- 1 */
32 VSIZEMASK equ $3ff /* 2^10 - 1 */
33
34 MAXBYTESPERROW EQU 128
35
36 * definitions for blitter control register 0 */
37
38 ABC equ $80
39 ABNC equ $40
40 ANBC equ $20
41 ANBNC equ $10
42 NABC equ $8
43 NABNC equ $4
44 NANBC equ $2
45 NANBNC equ $1
46
47 BC0B_DEST equ 8
48 BC0B_SRCC equ 9
49 BC0B_SRCB equ 10
50 BC0B_SRCA equ 11
51 BC0F_DEST equ $100
52 BC0F_SRCC equ $200
53 BC0F_SRCB equ $400
54 BC0F_SRCA equ $800
55
56 BC1F_DESC equ 2

```

```

57 DEST equ $100
58 SRCC equ $200
59 SRCB equ $400
60 SRCA equ $800
61
62
63 ASHIFTSHIFT equ 12 /* bits to right align ashift value */
64 BSHIFTSHIFT equ 12 /* bits to right align bshift value */
65
66 * definitions for blitter control register 1 */
67 LINEMODE equ $1
68 FILL_OR equ $8
69 FILL_XOR equ $10
70 FILL_CARRYIN equ $4
71 ONEDOT equ $2
72 OVFLAG equ $20
73 SIGNFLAG equ $40
74 BLITREVERSE equ $2
75
76 SUD equ $10
77 SUL equ $8
78 AUL equ $4
79
80 OCTANT8 equ 24
81 OCTANT7 equ 4
82 OCTANT6 equ 12
83 OCTANT5 equ 28
84 OCTANT4 equ 20
85 OCTANT3 equ 8
86 OCTANT2 equ 0
87 OCTANT1 equ 16
88
89 ENDC !HARDWARE_BLIT_I

```

```

1 *****
2 * Commodore-Amiga, Inc.
3 * cia.i
4 *****
5
6 CIANAME MACRO
7 DC.B 'ciaa.resource',0
8 ENDM
9
10 CIABNAME MACRO
11 DC.B 'ciab.resource',0
12 ENDM
13

```

```

1 *****
2 * Commodore-Amiga, Inc.
3 * custom.i
4 *
5 * $Header: custom.i,v 27.1 85/06/24 14:42:56 nell Exp $
6 *
7 * $Locker: $
8 *
9 *****
10
11 IFND HARDWARE_CUSTOM_I
12 HARDWARE_CUSTOM_I SET 1
13 *
14 *
15 * do this to get base of custom registers:
16 * XREF _custom;
17 *
18
19 bitddat EQU $000
20 dmaconr EQU $002
21 vposr EQU $004
22 vposr EQU $006
23 diskdtr EQU $008
24 joy0dat EQU $00A
25 joyldat EQU $00C
26 cixdat EQU $00E
27
28 adkconr EQU $010
29 pot0dat EQU $012
30 potldat EQU $014
31 potlup EQU $016
32 sardatr EQU $018
33 diskbytr EQU $01A
34 intsnar EQU $01C
35 intreqr EQU $01E
36
37 diskpt EQU $020
38 dsklen EQU $024
39 diskdat EQU $026
40 refptr EQU $028
41 vposw EQU $02A
42 vposw EQU $02C
43 copcon EQU $02E
44 sardat EQU $030
45 serper EQU $032
46 potgo EQU $034
47 joytest EQU $036
48 strequ EQU $038
49 strvbl EQU $03A
50 strhor EQU $03C
51 strlong EQU $03E
52
53 bltcon0 EQU $040
54 bltcon1 EQU $042
55 bltcfwm EQU $044
56 bltalwm EQU $046

```

```

57 bitcpt EQU $048
58 bitbpt EQU $04C
59 bitapt EQU $050
60 bitdpt EQU $054
61 bitsize EQU $058
62
63 bitcm0d EQU $060
64 bitcm0d EQU $062
65 bitcm0d EQU $064
66 bitcm0d EQU $066
67
68 bitcdat EQU $070
69 bitbdat EQU $072
70 bitadat EQU $074
71
72 dsksync EQU $07E
73
74 cop11c EQU $080
75 cop21c EQU $084
76 copjmp1 EQU $088
77 copjmp2 EQU $08A
78 coplins EQU $08C
79 diwstrt EQU $08E
80 diwstop EQU $090
81 ddfstrt EQU $092
82 ddfstop EQU $094
83 dmacon EQU $096
84 clxcon EQU $098
85 intena EQU $09A
86 intrcq EQU $09C
87 adkcon EQU $09E
88
89 aud EQU $0A0
90 aud0 EQU $0A0
91 aud1 EQU $0B0
92 aud2 EQU $0C0
93 aud3 EQU $0D0
94
95 * STRUCTURE AudChannel,0
96 ac_ptr EQU $00 ; ptr to start of waveform data
97 ac_len EQU $04 ; length of waveform in words
98 ac_per EQU $06 ; sample period
99 ac_vol EQU $08 ; volume
100 ac_dat EQU $0A ; sample pair
101 ac_sizeof EQU $10
102
103 bplpt EQU $0E0
104
105 bplcon0 EQU $100
106 bplcon1 EQU $102
107 bplcon2 EQU $104
108 bplmod EQU $108
109 bpl2mod EQU $10A
110
111 bpldat EQU $110
112

```

```

113 sprpt EQU $120
114
115 spr EQU $140
116 * STRUCTURE SpritesDef
117 sd_pos EQU $00
118 sd_ctl EQU $02
119 sd_dataa EQU $04
120 sd_datab EQU $08
121
122 color EQU $180
123
124 ENDC !HARDWARE_CUSTOM_I

```



```

1 *****
2 * Commodore-Amiga, Inc.
3 * dmabits.1
4 *
5 * $Header: dmabits.1.v 27.1 85/06/24 14:43:02 neil Exp $
6 *
7 * $Locker: $
8 *
9 *****
10
11 IFND HARDWARE_DMABITS_I
12 HARDWARE_DMABITS_I SET 1
13
14
15 * include file for defining dma control stuff */
16
17 * write definitions for dmaconv */
18 DMAF_SETCLR EQU $8000
19 DMAF_AUDIO EQU $000F /* 4 bit mask */
20 DMAF_AUD0 EQU $0001
21 DMAF_AUD1 EQU $0002
22 DMAF_AUD2 EQU $0004
23 DMAF_AUD3 EQU $0008
24 DMAF_DISK EQU $0010
25 DMAF_SPRITE EQU $0020
26 DMAF_BLITTER EQU $0040
27 DMAF_COOPER EQU $0080
28 DMAF_RASTER EQU $0100
29 DMAF_MASTER EQU $0200
30 DMAF_BLITHOG EQU $0400
31 DMAF_ALL EQU $01FE /* all dma channels */
32
33 * read definitions for dmaconv */
34 * bits 0-8 correspond to dmaconv definitions */
35 DMAF_BLITDONE EQU $4000
36 DMAF_BLITZERO EQU $2000
37
38 DMAB_SETCLR EQU 15
39 DMAB_AUD0 EQU 0
40 DMAB_AUD1 EQU 1
41 DMAB_AUD2 EQU 2
42 DMAB_AUD3 EQU 3
43 DMAB_DISK EQU 4
44 DMAB_SPRITE EQU 5
45 DMAB_BLITTER EQU 6
46 DMAB_COOPER EQU 7
47 DMAB_RASTER EQU 8
48 DMAB_MASTER EQU 9
49 DMAB_BLITHOG EQU 10
50 DMAB_BLITDONE EQU 14
51 DMAB_BLITZERO EQU 13
52
53 ENDC !HARDWARE_DMABITS_I

```

```

1 *****
2 * Commodore-Amiga, Inc.
3 * intnabits.1 -- definitions for the bits in the interrupt enable
4 * (and interrupt request) register
5 *
6 * $Header: intnabits.1.v 27.1 85/06/24 14:43:07 neil Exp $
7 *
8 * $Locker: $
9 *
10 *****
11
12 IFND HARDWARE_INTBITS_I
13 HARDWARE_INTBITS_I SET 1
14
15
16 INTB_SETCLR EQU (15) ;Set/Clear control bit. Determines if bits
17 ;written with a 1 get set or cleared. Bits
18 ;written with a zero are always unchanged.
19 INTB_INTEN EQU (14) ;Master interrupt (enable only)
20 INTB_EXTER EQU (13) ;External interrupt
21 INTB_DSksync EQU (12) ;Disk re-SYNChronized
22 INTB_REF EQU (11) ;serial port Receive Buffer Full
23 INTB_AUD3 EQU (10) ;Audio channel 3 block finished
24 INTB_AUD2 EQU (9) ;Audio channel 2 block finished
25 INTB_AUD1 EQU (8) ;Audio channel 1 block finished
26 INTB_AUD0 EQU (7) ;Audio channel 0 block finished
27 INTB_BLIT EQU (6) ;Blitter finished
28 INTB_VERTB EQU (5) ;start of Vertical Blank
29 INTB_COOPER EQU (4) ;Coprocessor
30 INTB_PORTS EQU (3) ;I/O Ports and timers
31 INTB_SOFTINT EQU (2) ;software interrupt request
32 INTB_DSKBLK EQU (1) ;Disk Block done
33 INTB_TBE EQU (0) ;serial port Transmit Buffer Empty
34
35
36
37 INTB_SETCLR EQU (1<<15)
38 INTB_INTEN EQU (1<<14)
39 INTB_EXTER EQU (1<<13)
40 INTB_DSksync EQU (1<<12)
41 INTB_REF EQU (1<<11)
42 INTB_AUD3 EQU (1<<10)
43 INTB_AUD2 EQU (1<<9)
44 INTB_AUD1 EQU (1<<8)
45 INTB_AUD0 EQU (1<<7)
46 INTB_BLIT EQU (1<<6)
47 INTB_VERTB EQU (1<<5)
48 INTB_COOPER EQU (1<<4)
49 INTB_PORTS EQU (1<<3)
50 INTB_SOFTINT EQU (1<<2)
51 INTB_DSKBLK EQU (1<<1)
52 INTB_TBE EQU (1<<0)
53
54 ENDC !HARDWARE_INTBITS_I

```



```

1  IFND INTUITION_INTUITION_I
2  INTUITION_INTUITION_I SET 1
3
4  ;** intuition.1 *****
5  ;** Commodore-Amiga, Inc.
6
7
8  ;** intuition.1 main include file for assembly-language programmers
9
10 ;**
11 ;** date : author : Comments
12 ;** -----
13 ;** 1-30-85 --RJ-- created this file!
14 ;** 6-12-85 Dale and Carl translated this from the c version
15 ;** 6-13-85 =VoodooDrJ= added back the comments
16 ;**
17 ;** *****
18
19 IFND GRAPHICS_GFX_I
20 Include 'graphics/gfx.1'
21 ENDC
22
23 IFND GRAPHICS_CLIP_I
24 Include 'graphics/clip.1'
25 ENDC
26
27 IFND GRAPHICS_VIEW_I
28 Include 'graphics/view.1'
29 ENDC
30
31 IFND GRAPHICS_RASTPORT_I
32 Include 'graphics/rastport.1'
33 ENDC
34
35 IFND GRAPHICS_LAYERS_I
36 Include 'graphics/layers.1'
37 ENDC
38
39 IFND GRAPHICS_TEXT_I
40 Include 'graphics/text.1'
41 ENDC
42
43 IFND EXEC_PORTS_I
44 Include 'exec/ports.1'
45 ENDC
46
47 IFND DEVICES_TIMER_I
48 Include 'devices/timer.1'
49 ENDC
50
51 IFND DEVICES_INPUTEVENT_I
52 Include 'devices/inputevent.1'
53 ENDC
54
55
56

```

```

57 ; == Menu
58 ;
59 ; STRUCTURE Menu, 0
60
61 APTR ml_NextMenu ; menu pointer, same level
62 WORD ml_LeftEdge ; dimensions of the select box;
63 WORD ml_TopEdge ; dimensions of the select box;
64 WORD ml_Width ; dimensions of the select box;
65 WORD ml_Height ; dimensions of the select box;
66 WORD ml_Flags ; see flag definitions below;
67 APTR ml_MenuName ; text for this Menu header
68 APTR ml_FirstItem ; pointer to first in chain;
69
70 ; these mysteriously-named variables are for internal use only
71 WORD ml_JazzX
72 WORD ml_JazzY
73 WORD ml_BeatX
74 WORD ml_BeatY
75
76 LABEL ml_SIZEOF
77
78 ; FLAGS SET BY BOTH THE APPLIPROG AND INTUITION
79 MENUENABLED equ $0001 ; whether or not this menu is enabled;
80
81 ; FLAGS SET BY INTUITION;
82 MIDRAWN equ $0100 ; this menu's items are currently drawn;
83
84
85
86
87
88 ; == MenuItem
89 ;
90 ; STRUCTURE MenuItem, 0
91
92 APTR ml_NextItem ; pointer to next in chained list
93 WORD ml_LeftEdge ; dimensions of the select box
94 WORD ml_TopEdge ; dimensions of the select box
95 WORD ml_Width ; dimensions of the select box
96 WORD ml_Height ; dimensions of the select box
97 WORD ml_Flags ; see the defines below
98
99 LONG ml_MutualExclude ; set bits mean this item excludes that item
100
101 APTR ml_ItemsFill ; points to Image, IntuiText, or NULL
102
103 ; when this item is pointed to by the cursor and the items highlight
104 ; mode HIGHIMAGE is selected, this alternate image will be displayed
105 APTR ml_SelectFill ; points to Image, IntuiText, or NULL
106
107 BYTE ml_Command ; only if appliprogram sets the CO#SEQ flag
108
109 ; The following variable is strictly from Kludge-City, where some people
110 ; still live. It is included solely because our types.1 macros aren't
111 ; smart enough to do the right thing, which would be the automatic
112

```

```

113 ; word-alignment to these references as it SHOULD be in order to duplicate
114 ; the way alignments are adjusted in the c-language. And instead of
115 ; correcting the problem, I am obliged to kludge up my include.i files.
116 ; So here it is!
117 BYTE ml_KludgeFill100 ; defined as a BYTE because this does
118
119 APTR ml_SubItem ; if non-zero, DrawMenu shows "->"
120
121 ; The NextSelect field represents the menu number of next selected
122 ; item (when user has drag-selected several items)
123 WORD ml_NextSelect
124
125 LABEL ml_SIZEOF
126
127 ; --- FLAGS SET BY THE APPLIPROC ---
128 CHECKIT equ $0001 ; whether to check this item if selected
129 ITEMTEXT equ $0002 ; set if textual, clear if graphical item
130 COMMSQ equ $0004 ; set if there's a command sequence
131 MENUTOGGLE equ $0008 ; set to toggle the check of a menu item
132 ITEMENABLED equ $0010 ; set if this item is enabled
133
134 ; these are the SPECIAL HIGHLIGHT FLAG state meanings
135 HIGHFLAGS equ $00C0 ; see definitions below for these bits
136 HIGHINACE equ $0000 ; use the user's "select image"
137 HIGHCOMP equ $0040 ; highlight by complementing the select box
138 HIGHBOX equ $0080 ; highlight by drawing a box around the image
139 HIGHNONE equ $00C0 ; don't highlight
140
141 ; --- FLAGS SET BY BOTH APPLIPROC AND INTUITION ---
142 CHECKED equ $0100 ; if CHECKIT, then set this when selected
143
144
145 ; --- FLAGS SET BY INTUITION ---
146 ISDRAWN equ $1000 ; this item's subs are currently drawn
147 HIGHLIGHT equ $2000 ; this item is currently highlighted
148 MENUTOGGLED equ $4000 ; this item was already toggled
149
150
151
152
153
154
155
156
157
158
159
160 ; the ClipRect and BitMap and used for rendering the requester
161 APTR rq_OlderRequest
162 WORD rq_LeftEdge ; dimensions of the entire box
163 WORD rq_TopEdge ; dimensions of the entire box
164 WORD rq_Width ; dimensions of the entire box
165 WORD rq_Height ; dimensions of the entire box
166
167 WORD rq_RelLeft ; get POINTREL Pointer relativity offsets
168 WORD rq_RelTop ; get POINTREL Pointer relativity offsets

```

```

169
170 APTR rq_ReqGadget ; pointer to the first of a list of gadgets
171 APTR rq_ReqBorder ; the box's border
172 APTR rq_ReqText ; the box's text
173
174 USHORT rq_Flags ; see definitions below
175
176 UBYTE rq_BackFill ; pen number for back-plane fill before draws
177
178 ; The following variable is strictly from Kludge-City, where some peopl
179 ; still live. It is included solely because our types.i macros aren't
180 ; smart enough to do the right thing, which would be the automatic
181 ; word-alignment to these references as it SHOULD be in order to duplic
182 ; the way alignments are adjusted in the c-language. And instead of
183 ; correcting the problem, I am obliged to kludge up my include.i files.
184 ; So here it is!
185 BYTE rq_KludgeFill100 ; defined as a BYTE because this does
186
187 APTR rq_ReqLayer ; layer in which requester rendered
188 STRUCT rq_ReqPad1,32 ; for backwards compatibility (reserved)
189
190 ; If the BitMap plane pointers are non-zero, this tells the system
191 ; that the image comes pre-drawn (if the appliprogram wants to define
192 ; it's own box, in any shape or size it wants!); this is OK by
193 ; intuition as long as there's a good correspondence between the image
194 ; and the specified Gadgets
195 APTR rq_ReqBMap ; points to the BitMap of FREDRAWN imagery
196
197 APTR rq_RWindow ; points back to requester's window
198 STRUCT rq_ReqPad2,36 ; for backwards compatibility (reserved)
199
200 LABEL rq_SIZEOF
201
202 ; FLAGS SET BY THE APPLIPROC
203 POINTREL equ $0001 ; if POINTREL set, TopLeft is relative to pointer
204 FREDRAWN equ $0002 ; if ReqBMap points to pre-drawn Requester imagery
205
206 ; FLAGS SET BY INTUITION;
207 REQOFWINDOW equ $1000 ; part of one of the Gadgets was offwindow
208 REQACTIVE equ $2000 ; this requester is active
209 SYSREQUEST equ $4000 ; this requester caused by system
210 DEFERREFRESH equ $8000 ; this Requester stops a Refresh broadcast
211
212
213
214
215
216
217
218
219
220
221
222
223
224

```

```

225 WORD gg_Width ; "hit box" of gadget
226 WORD gg_Height ; "hit box" of gadget
227
228 WORD gg_Flags ; see below for list of defines
229
230 WORD gg_Activation ; see below for list of defines
231
232 WORD gg_GadgetType ; see below for defines
233
234 ; appliprogram can specify that the Gadget be rendered as either as Border
235 ; or an Image. This variable points to which (or equals NULL if there's
236 ; nothing to be rendered about this Gadget)
237 APTR gg_GadgetRender
238
239 ; appliprogram can specify "highlighted" Imagery rather than algorithmic
240 ; this can point to either Border or Image data
241 APTR gg_SelectRender
242
243 APTR gg_GadgetText ; text for this gadget;
244
245 ; by using the MutualExclude word, the appliprogram can describe
246 ; which gadgets mutually-exclude which other ones. The bits in
247 ; MutualExclude correspond to the gadgets in object containing
248 ; the gadget list. If this gadget is selected and a bit is set
249 ; in this gadget's MutualExclude and the gadget corresponding to
250 ; that bit is currently selected (e.g. bit 2 set and gadget 2
251 ; is currently selected) that gadget must be unselected. Intuition
252 ; does the visual unselecting (with checkmarks) and leaves it up
253 ; to the program to unselect internally
254 LONG gg_MutualExclude ; set bits mean this gadget excludes that
255
256 ; pointer to a structure of special data required by Proportional, String
257 ; and Integer Gadgets
258 APTR gg_SpecialInfo
259
260 WORD gg_GadgetID ; user-definable ID field
261 APTR gg_UserData ; ptr to general purpose User data (Ignored by Intuit)
262
263 LABEL gg_SIZEOF
264
265 ; --- FLAGS SET BY THE APPLIPROG -----
266 ; combinations in these bits describe the highlight technique to be used
267 GADHIGBITS equ $0003
268 GADHCOMP equ $0000 ; Complement the select box
269 GADCHBOX equ $0001 ; Draw a box around the image
270 GADCHIMAGE equ $0002 ; Blast in this alternate image
271 GADCHNONE equ $0003 ; don't highlight
272
273 ; set this flag if the GadgetRender and SelectRender point to Image imagery,
274 ; clear if it's a Border
275 GADGIMAGE equ $0004
276
277 ; combinations in these next two bits specify to which corner the gadget's
278 ; Left & Top coordinates are relative. If relative to Top/Left,
279 ; these are "normal" coordinates (everything is relative to something in
280 ; this universe)

```

```

281 GRELBOTTOM equ $0008 ; set if rel to bottom, clear if rel top
282 GRELRIGHT equ $0010 ; set if rel to right, clear if to left
283 ; set the RELWIDTH bit to spec that Width is relative to width of screen
284 GRELWIDTH equ $0020
285 ; set the RELHEIGHT bit to spec that Height is rel to height of screen
286 GRELHEIGHT equ $0040
287
288 ; the SELECTED flag is initialized by you and set by Intuition. It
289 ; specifies whether or not this Gadget is currently selected/highlighted
290 SELECTED equ $0080
291
292 ; the GADDISABLED flag is initialized by you and later set by Intuition
293 ; according to your calls to On/OffGadget(). It specifies whether or not
294 ; this Gadget is currently disabled from being selected
295 GADDISABLED equ $0100
296
297
298
299 ; --- These are the Activation flag bits -----
300 ; RELVERIFY is set if you want to verify that the pointer was still over
301 ; the gadget when the select button was released
302 RELVERIFY equ $0001
303
304 ; the flag GADGIMMEDIATE, when set, informs the caller that the gadget
305 ; was activated when it was activated. this flag works in conjunction with
306 ; the RELVERIFY flag
307 GADGIMMEDIATE equ $0002
308
309 ; the flag ENDCADGET, when set, tells the system that this gadget, when
310 ; selected, causes the Requester or AbsMessage to be ended. Requesters or
311 ; AbsMessages that are ended are erased and unlinked from the system
312 ENDCADGET equ $0004
313
314 ; the FOLLOWMOUSE flag, when set, specifies that you want to receive
315 ; reports on mouse movements (ie, you want the REPORTMOUSE function for
316 ; your Window). When the Gadget is deselected (immediately if you have
317 ; no RELVERIFY) the previous state of the REPORTMOUSE flag is restored
318 ; You probably want to set the GADGIMMEDIATE flag when using FOLLOWMOUSE,
319 ; since that's the only reasonable way you have of learning why Intuition
320 ; is suddenly sending you a stream of mouse movement events. If you don't
321 ; set RELVERIFY, you'll get at least one Mouse Position event.
322 FOLLOWMOUSE equ $0008
323
324 ; if any of the BORDER flags are set in a Gadget that's included in the
325 ; Gadget list when a Window is opened, the corresponding Border will
326 ; be adjusted to make room for the Gadget
327 RIGTBORDER equ $0010
328 LEFTBORDER equ $0020
329 TOPBORDER equ $0040
330 BOTTOMBORDER equ $0080
331
332 TOGGLESELECT equ $0100 ; this bit for toggle-select mode
333
334 STRINCENTER equ $0200 ; center the String
335 STRINCRIGHT equ $0400 ; right-justify the String
336

```

```

337 LONGINT equ $0800 ; This String Gadget is a Long Integer
338
339 ALTKYMAP equ $1000 ; This String has an alternate keymapping
340
341
342
343 --- GADGET TYPES
344 ; These are the Gadget Type definitions for the variable GadgetType.
345 ; Gadget number type MUST start from one. NO TYPES OF ZERO ALLOWED.
346 ; first comes the mask for Gadget flags reserved for Gadget typing
347 CADGETTYPE equ $FC00 ; all Gadget Global Type flags (padded)
348 SYSGADGET equ $8000 ; 1 = SysGadget, 0 = AppliGadget
349 SCRGADGET equ $4000 ; 1 = ScreenGadget, 0 = WindowGadget
350 CZZGADGET equ $2000 ; 1 = Gadget for GIMMEZERO borders
351 REQCADGET equ $1000 ; 1 = this is a Requester Gadget
352 ; system gadgets
353 SIZING equ $0010
354 WRAPPING equ $0020
355 STRAGGING equ $0030
356 WUPFRONT equ $0040
357 SUPFRONT equ $0050
358 DOWNBACK equ $0060
359 SDOWNBACK equ $0070
360 CLOSE equ $0080
361 ; application gadgets
362 BOOLGADGET equ $0001
363 CADGET0002 equ $0002
364 PROPGADGET equ $0003
365 STRCADGET equ $0004
366
367
368
369
370
371
372
373
374
375 ; this is the special data required by the proportional Gadget
376 ; typically, this data will be pointed to by the Gadget variable SpecialInfo
377 STRUCTURE PropInfo,0
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392

```

```

393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448

```

; routines, to adjust the size of the AUTOKNOB according to how
; much of the data can be seen. This is also used to decide how
; far to advance the Pots when User hits the Container of the Gadget.
; For instance, if you were controlling the display of a 5-line
; Window of text with this Gadget, and there was a total of 15
; lines that could be displayed, you would set the VertBody value to
; (MAXBODY / (totalLines / DisplayLines)) = MAXBODY / 3.
; Therefore, the AUTOKNOB would fill 1/3 of the container, and if
; User hits the Container outside of the knob, the pot would advance
; 1/3 (plus or minus) If there's no body to show, or the total
; amount of displayable info is less than the display area, set the
; Body variables to the MAX. To adjust these after the Gadget is
; added to the System, use ModifyProp().
WORD pi_HorizBody ; horizontal Body
WORD pi_VertBody ; vertical Body
; these are the variables that Intuition sets and maintains
WORD pi_Width ; Container width (with any relativity absolved)
WORD pi_Height ; Container height (with any relativity absolved)
WORD pi_HPotRes ; pot increments
WORD pi_VPotRes ; pot increments
WORD pi_LeftBorder ; Container borders
WORD pi_TopBorder ; Container borders
LABEL pi_SIZEOF
; --- FLAG BITS
418 AUTOKNOB equ \$0001 ; this flag sez: gimme that old auto-knob
420 FREEHORIZ equ \$0002 ; if set, the knob can move horizontally
421 FREEVERT equ \$0004 ; if set, the knob can move vertically
422 PROPORDERLESS equ \$0008 ; if set, no border will be rendered
423 KNOBHIT equ \$0100 ; set when this Knob is hit
424
425
426 KNOBMIN equ 6 ; minimum horizontal size of the knob
427 KNOBMAX equ 4 ; minimum vertical size of the knob
428 MAXBODY equ \$FFFF ; maximum body value
429 MAXPOT equ \$FFFF ; maximum pot value
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448

```

449 ; Intuition initializes and maintains these variables for you
450 WORD si_UndoPos ; character position in the undo buffer
451 WORD si_NumChars ; number of characters currently in Buffer
452 WORD si_Dispcount ; number of whole characters visible in Container
453 WORD si_CLeft ; topleft offset of the container
454 WORD si_CTop ; topleft offset of the container
455 APTR si_LayerPtr ; the RastPort containing this Gadget
456
457 ; you can initialize this variable before the gadget is submitted to
458 ; Intuition, and then examine it later to discover what integer
459 ; the user has entered (if the user never plays with the gadget,
460 ; the value will be unchanged from your initial setting)
461 LONG si_Longint ; the LONG return value of a LONGINT String Gadget
462
463 ; If you want this Gadget to use your own Console keymapping, you
464 ; set the ALIKEVMAP bit in the Activation flags of the Gadget, and then
465 ; set this variable to point to your keymap. If you don't set the
466 ; ALIKEVMAP, you'll get the standard ASCII keymapping.
467 APTR si_AltKeyMap
468
469 LABEL si_L_SIZEOF
470
471
472
473
474
475
476
477
478 IntuiText ; a series of strings that start with a screen location
479 ; (always relative to the upper-left corner of something) and then the
480 ; text of the string. The text is null-terminated.
481
482 STRUCTURE IntuiText,0
483
484 UBYTE it_FrontPen ; the pens for rendering the text
485 UBYTE it_BackPen ; the pens for rendering the text
486
487 UBYTE it_DrawMode ; the mode for rendering the text
488
489 ; The following variable is strictly from Kludge-City, where some people
490 ; still live. It is included solely because our types.i macros aren't
491 ; smart enough to do the right thing, which would be the automatic
492 ; word-alignment to these references as it SHOULD be in order to duplicate
493 ; the way alignments are adjusted in the c-language. And instead of
494 ; correcting the problem, I am obliged to kludge up my include.i files.
495 ; So here it is!
496 BYTE it_KludgeFill100 ; defined as a BYTE because this does
497
498 WORD it_LeftEdge ; relative start location for the text
499 WORD it_TopEdge ; relative start location for the text
500
501 APTR it_ITextFont ; if NULL, you accept the defaults
502
503 APTR it_IText ; pointer to null-terminated text
504
505 APTR it_NextText ; continuation to TxFwrite another text
506
507
508
509
510
511
512
513
514
515 LABEL it_L_SIZEOF
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

561 ; all zeroes. Using these flags allows you to avoid wasting all that
562 ; memory in this way:
563 ; first, you specify which planes you want your data to appear
564 ; in using the PlanePick variable. For each bit set in the variable, the
565 ; next "plane" of your image data is blitted to the display. For each bit
566 ; clear in this variable, the corresponding bit in PlaneOnOff is examined.
567 ; If that bit is clear, a "plane" of zeroes will be used. If the bit is
568 ; set, ones will go out instead. So, for our example:
569 ; Gadget.PlanePick = 0x02;
570 ; Gadget.PlaneOnOff = 0x01;
571 ; Note that this also allows for generic Gadgets, like the System Gadgets,
572 ; which will work in any number of bit planes
573 ; Note also that if you want an image that is only a filled rectangle,
574 ; you can get this by setting PlanePick to zero (pick no planes of data)
575 ; and set PlaneOnOff to describe the pen color of the rectangle.
576 BYTE lg.PlanePick
577 BYTE lg.PlaneOnOff
578
579 ; If the NextImage variable is not NULL, Intuition presumes that
580 ; it points to another Image structure with another Image to be
581 ; rendered
582 APTR lg.NextImage
583
584 LABEL lg.SIZEOF
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616

```

```

617 ; the time values are copies of the current system clock time. Micros
618 ; are in units of microseconds, Seconds in seconds.
619 LONG im_Seconds
620 LONG im_Micros
621
622 ; the IDCMPWindow variable will always have the address of the Window o
623 ; this IDCMP
624 APTR im_IDCMPWindow
625
626 ; system-use variable
627 APTR im_SpecialLink
628
629 LABEL im_SIZEOF
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672

```

```

----- IDCP Classes -----
SIZEVERIFY equ $00000001 ; See the Programmer's Guide
MEMSIZE equ $00000002 ; See the Programmer's Guide
REFRESHWINDOW equ $00000004 ; See the Programmer's Guide
MOUSEBUTTONS equ $00000008 ; See the Programmer's Guide
MOUSEMOVE equ $00000010 ; See the Programmer's Guide
GADGETDOWN equ $00000020 ; See the Programmer's Guide
GADGETUP equ $00000040 ; See the Programmer's Guide
REQUEST equ $00000080 ; See the Programmer's Guide
MENUPIICK equ $00000100 ; See the Programmer's Guide
CLOSEWINDOW equ $00000200 ; See the Programmer's Guide
RAWKEY equ $00000400 ; See the Programmer's Guide
REQCLEAR equ $00001000 ; See the Programmer's Guide
MENUVERIFY equ $00002000 ; See the Programmer's Guide
NEWPREFS equ $00004000 ; See the Programmer's Guide
DISKINSERTED equ $00008000 ; See the Programmer's Guide
DISKREMOVED equ $00010000 ; See the Programmer's Guide
WBEINCHMESSAGE equ $00020000 ; See the Programmer's Guide
ACTIVEWINDOW equ $00040000 ; See the Programmer's Guide
INACTIVEWINDOW equ $00080000 ; See the Programmer's Guide
DELTAMOVE equ $00100000 ; See the Programmer's Guide
VANILLAKKEY equ $00200000 ; See the Programmer's Guide
INTUITICKS equ $00400000 ; See the Programmer's Guide
NOTEZ-BIEN: $80000000 is reserved for internal use by IDCP

; the IDCMP Flags do not use this special bit, which is cleared when
; Intuition sends its special message to the Task, and set when Intuition
; gets its Message back from the Task. Therefore, I can check here to
; find out fast whether or not this Message is available for me to send
LONELYMESSAGE equ $80000000

----- IDCP Codes -----
; This group of codes is for the MENUVERIFY function
MENUHOT equ $0001 ; IntuiWants verification or MENCUCANCEL
MENUHOT equ $0002 ; HOT Reply of this cancels Menu operation
MENUWAITING equ $0003 ; Intuition simply wants a ReplyMsg() ASAP

```

```

673 ; This group of codes is for the WRENCHMESSAGE messages
674 WRENCHOPEN equ $0001
675 WRENCHCLOSE equ $0002
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728

```

```

=====
=====
=====
STRUCTURE Window,0
APTR wd_NextWindow ; for the linked list of a Screen
WORD wd_LeftEdge ; screen dimensions
WORD wd_TopEdge ; screen dimensions
WORD wd_Width ; screen dimensions
WORD wd_Height ; screen dimensions
WORD wd_MouseY ; relative top top-left corner
WORD wd_MouseX ; relative top top-left corner
WORD wd_MinWidth ; minimum sizes
WORD wd_MinHeight ; minimum sizes
WORD wd_MaxWidth ; maximum sizes
WORD wd_MaxHeight ; maximum sizes
LONG wd_Flags ; see below for definitions
APTR wd_MenuStrip ; first in a list of menu headers
APTR wd_Title ; title text for the Window
APTR wd_FirstRequest ; first in linked list of active Requesters
APTR wd_DMRequest ; the double-menu Requester
WORD wd_ReqCount ; number of Requesters blocking this Window
APTR wd_WScreen ; this Window's Screen
APTR wd_RPort ; this Window's very own RastPort
; the border variables describe the window border. If you specify
; GIMMEZERO when you open the window, then the upper-left of the
; ClipRect for this window will be upper-left of the BitMap (with correct
; offsets when in SuperBitMap mode; you MUST select GIMMEZERO when
; using SuperBitMap). If you don't specify ZeroZero, then you save
; memory (no allocation of RastPort, Layer, ClipRect and associated
; Bitmaps), but you also must offset all your writes by BorderTop,
; BorderLeft and do your own mini-clipping to prevent writing over the
; system gadgets
BYTE wd_BorderLeft
BYTE wd_BorderTop
BYTE wd_BorderRight
BYTE wd_BorderBottom
APTR wd_BorderPort
; You supply a linked-list of gadget that you want for your Window.
; This list DOES NOT include system Gadgets. You get the standard

```

```

729 ; window system Gadgets by setting flag-bits in the variable Flags (see
730 ; the bit definitions below)
731 APTR wd_FirstGadget
732
733 ; these are for opening/closing the windows
734 APTR wd_Parent
735 APTR wd_Descendant
736
737 ; sprite data information for your own Pointer
738 ; set these AFTER you Open the Window by calling SetPointer ()
739 APTR wd_Pointer
740 BYTE wd_PtrHeight
741 BYTE wd_PtrWidth
742 BYTE wd_XOffset
743 BYTE wd_YOffset
744
745 ; the IDCMP Flags and User's and Intuition's Message Ports
746 ULONG wd_IDCMPFlags
747 APTR wd_UserPort
748 APTR wd_WindowPort
749 APTR wd_MessageKey
750
751 BYTE wd_DetailPen
752 BYTE wd_BlockPen
753
754 ; the CheckMark is a pointer to the imagery that will be used when
755 ; rendering MenuItems of this Window that want to be checkmarked
756 ; if this is equal to NULL, you'll get the default imagery
757 APTR wd_CheckMark
758
759 ; if non-null, Screen title when Window is active
760 APTR wd_ScreenTitle
761
762 ; These variables have the mouse coordinates relative to the
763 ; inner-Window of GIMMEZERO Windows. This is compared with the
764 ; MouseX and MouseY variables, which contain the mouse coordinates
765 ; relative to the upper-left corner of the Window, GIMMEZERO
766 ; notwithstanding
767 SHORT wd_GZZMouseX
768 SHORT wd_GZZMouseY
769 ; these variables contain the width and height of the inner-Window of
770 ; GIMMEZERO Windows
771 SHORT wd_GZZWidth
772 SHORT wd_GZZHeight
773
774 APTR wd_ExtData
775
776 ; general-purpose pointer to User data extension
777 APTR wd_UserData
778 APTR wd_MLayer ; stash of Window.RPort->Layer
779
780 LABEL wd_Size
781
782 ; --- FLAGS REQUESTED (NOT DIRECTLY SET THOUGH) BY THE APPLIPROC
783 WINDOXSIZING equ $0001 ; include sizing system-gadget?
784 WINDOWDRAG equ $0002 ; include dragging system-gadget?

```

```

785 WINDOWDEPTH equ $0004 ; Include depth arrangement gadget?
786 WINDOWCLOSE equ $0008 ; Include close-box system-gadget?
787
788 SIZEERIGHT equ $0010 ; size gadget uses right border
789 SIZEBOTTOM equ $0020 ; size gadget uses bottom border
790
791 ; --- refresh modes -----
792 ; combinations of the REFRESHBITS select the refresh type
793 REFRESHBITS equ $00C0
794 SMART_REFRESH equ $0000
795 SIMPLE_REFRESH equ $0040
796 SUPER_BITMAP equ $0080
797 OTHER_REFRESH equ $00C0
798
799 BACKDROP equ $0100 ; this is an ever-popular BACKDROP window
800
801 REPORTMOUSE equ $0200 ; set this to hear about every mouse move
802
803 GIMMEZERO equ $0400 ; make extra border stuff
804
805 BORDERLESS equ $0800 ; set this to get a Window sans border
806
807 ACTIVATE equ $1000 ; when Window opens, it's the Active one
808
809 ; FLAGS SET BY INTUITION
810 WINDOWACTIVE equ $2000 ; this window is the active one
811 INREQUEST equ $4000 ; this window is in request mode
812 MENUSTATE equ $8000 ; this Window is active with its Menus on
813
814 ; --- Other User Flags -----
815 RMBTRAP equ $0010000 ; Catch RMB events for your own
816 NOCAREREFRESH equ $0020000 ; not to be bothered with REFRESH
817
818
819 WINDOWREFRESH equ $01000000 ; Window is currently refreshing
820 WBEINCHWINDOW equ $02000000 ; WorkBench Window
821 WINDOWTICKED equ $04000000 ; only one timer tick at a time
822
823 SUPER_UNUSED equ $EFC00000 ;bits of Flag unused yet
824
825
826
827 ; --- see struct IntuiMessage for the IDCMP Flag definitions -----
828
829
830
831
832
833
834
835
836
837
838
839
840

```

```

841 STRUCTURE NewWindow,0
842
843 WORD nw_LeftEdge ; Initial Window dimensions
844 WORD nw_TopEdge ; Initial Window dimensions
845 WORD nw_Width ; Initial Window dimensions
846 WORD nw_Height ; Initial Window dimensions
847
848 BYTE nw_DetailPen ; for rendering the detail bits of the Window
849 BYTE nw_BlockPen ; for rendering the block-fill bits
850
851 ULONG nw_IDCMPFlags ; Initial IDCMP state
852
853 LONG nw_Flags ; see the Flag definition under Window
854
855 ; You supply a linked-list of Gadgets for your Window.
856 ; This list DOES NOT include system Gadgets. You get the standard
857 ; system Window Gadgets by setting flag-bits in the variable Flags (see
858 ; the bit definitions under the Window structure definition)
859 APTR nw_FirstGadget
860
861 ; the CheckMark is a pointer to the imagery that will be used when
862 ; rendering MenuItems of this Window that want to be checked
863 ; if this is equal to NULL, you'll get the default imagery
864 APTR nw_CheckMark
865
866 APTR nw_Title ; title text for the Window
867
868 ; the Screen pointer is used only if you've defined a CUSTOMSCREEN and
869 ; want this Window to open in it. If so, you pass the address of the
870 ; Custom Screen structure in this variable. Otherwise, this variable
871 ; is ignored and doesn't have to be initialized.
872 APTR nw_Screen
873
874 ; SUPER_BITMAP Window? If so, put the address of your BitMap structure.
875 ; in this variable. If not, this variable is ignored and doesn't have
876 ; to be initialized
877 APTR nw_BitMap
878
879 ; the values describe the minimum and maximum sizes of your Windows.
880 ; these matter only if you've chosen the WINDOWRESIZING Gadget option,
881 ; which means that you want to let the User to change the size of
882 ; this Window. You describe the minimum and maximum sizes that the
883 ; Window can grow by setting these variables. You can initialize
884 ; any one these to zero, which will mean that you want to duplicate
885 ; the setting for that dimension (if MinWidth = 0, MinWidth will be
886 ; set to the opening Width of the Window).
887 ; You can change these settings later using SetWindowLimits().
888 ; If you haven't asked for a SIZING Gadget, you don't have to
889 ; initialize any of these variables.
890 WORD nw_MinWidth
891 WORD nw_MinHeight
892 WORD nw_MaxWidth
893 WORD nw_MaxHeight
894
895 ; the type variable describes the Screen in which you want this Window
896 ; open. The type value can either be CUSTOMSCREEN or one of the

```


897 ; system standard Screen Types such as WBNCHSCREEN. See the
898 ; type definitions under the Screen structure
899 WORD nw_Type
900 LABEL nw_SIZE

901
902
903
904
905
906
907
908
909

STRUCTURE Screen, 0
APTR sc_NextScreen ; linked list of screens
APTR sc_FirstWindow ; linked list Screen's Windows

WORD sc_LeftEdge ; parameters of the screen
WORD sc_TopEdge ; parameters of the screen

WORD sc_Width ; null-terminated Title text
WORD sc_Height ; for Windows without ScreenTitle

WORD sc_MouseY ; position relative to upper-left
WORD sc_MouseX ; position relative to upper-left
WORD sc_Flags ; see definitions below

APTR sc_Title
APTR sc_DefaultTitle

; Bar sizes for this Screen and all Window's in this Screen
BYTE sc_BarHeight
BYTE sc_BarVBorder
BYTE sc_BarHBorder
BYTE sc_MenuVBorder
BYTE sc_MenuHBorder
BYTE sc_WBotTop
BYTE sc_WBotLeft
BYTE sc_WBotRight
BYTE sc_WBotBottom

938 ; The following variable is strictly from Kludge-City, where some people
939 ; still live. It is included solely because our types.1 macros aren't
940 ; smart enough to do the right thing, which would be the automatic
941 ; word-alignment to these references as it SHOULD be in order to duplicate
942 ; the way alignments are adjusted in the c-language. And instead of
943 ; correcting the problem, I am obliged to kludge up my include.1 files.
944 ; So here it is!
945 BYTE sc_KludgeFill00 ; defined as a BYTE because this does
946
947 ; the display data structures for this Screen
948 APTR sc_Font ; this screen's default font
949 STRUCT sc_ViewPort, vp_SIZEOF ; describing the Screen's display
950 STRUCT sc_RastPort, rp_SIZEOF ; describing Screen rendering
951 STRUCT sc_BitMap, bm_SIZEOF ; auxiliary graphexcess baggage
952

STRUCT sc_LayerInfo, li_SIZEOF ; each screen gets a LayerInfo
; You supply a linked-list of Gadgets for your Screen.
; This list DOES NOT include system Gadgets. You get the standard
; system Screen Gadgets by default
APTR sc_FirstGadget

BYTE sc_DetailPen ; for bar/border/gadget rendering
BYTE sc_BlockPen ; for bar/border/gadget rendering

; the following variable(s) are maintained by Intuition to support the
; DisplayBeep() color flashing technique
WORD sc_SaveColor0

; This layer is for the Screen and Menu bars
APTR BarLayer;

APTR sc_ExtData

APTR sc_UserData ; general-purpose pointer to User data

LABEL sc_SIZEOF

---- FLAGS SET BY INTUITION -----
; The SCREENTYPE bits are reserved for describing various Screen types
; available under Intuition.

SCREENTYPE equ \$000F ; all the screens types available

WBNCHSCREEN equ \$0001 ; Ta Da! The Workbench

CUSTOMSCREEN equ \$000E ; for that special look

SHOWTITLE equ \$0010 ; this gets set by a call to ShowTitle()

BEEPING equ \$0020 ; set when Screen is beeping

CUSTOMBITMAP equ \$0040 ; if you are supplying your own Bitmap

==== NewScreen =====

STRUCTURE NewScreen, 0

WORD ns_LeftEdge ; initial Screen dimensions

WORD ns_TopEdge ; initial Screen dimensions

WORD ns_Width ; initial Screen dimensions

WORD ns_Height ; initial Screen dimensions

WORD ns_Depth ; initial Screen dimensions

BYTE ns_DetailPen ; default rendering pens (for Windows too)

BYTE ns_BlockPen ; default rendering pens (for Windows too)

WORD ns_ViewModes ; display "modes" for this Screen
1008


```

1121 ; PrInterPort
1122 PARALLEL_PRINTER equ $00
1123 SERIAL_PRINTER equ $01
1124
1125 ; BaudRate
1126 BAUD_110 equ $00
1127 BAUD_300 equ $01
1128 BAUD_1200 equ $02
1129 BAUD_2400 equ $03
1130 BAUD_4800 equ $04
1131 BAUD_9600 equ $05
1132 BAUD_19200 equ $06
1133 BAUD_MIDI equ $07
1134
1135 ; PaperType
1136 FANFOLD equ $00
1137 SINGLE equ $80
1138
1139 ; PrintPitch
1140 PICA equ $000
1141 ELITE equ $400
1142 FINE equ $800
1143
1144 ; PrintQuality
1145 DRAFT equ $000
1146 LETTER equ $100
1147
1148 ; PrintSpacing
1149 SIX_LPI equ $000
1150 EIGHT_LPI equ $200
1151
1152 ; Print Image
1153 IMAGE_POSITIVE equ 0
1154 IMAGE_NEGATIVE equ 1
1155
1156 ; PrintAspect
1157 ASPECT_HORIZ equ 0
1158 ASPECT_VERT equ 1
1159
1160 ; PrintShade
1161 SHADE_LW equ $00
1162 SHADE_GREYSCALE equ $01
1163 SHADE_COLOR equ $02
1164
1165 ; PaperSize
1166 US_LETTER equ $00
1167 N_LEGAL equ $10
1168 N_TRACTOR equ $20
1169 W_TRACTOR equ $30
1170 CUSTOM equ $40
1171
1172 ; PrinterType
1173 CUSTOM_NAME equ $00
1174 ALPHA_P_101 equ $01
1175 BROTHER_15XL equ $02

```

```

1177 CEM_MFS1000 equ $03
1178 DIAB_630 equ $04
1179 DIAB_ADV_D25 equ $05
1180 DIAB_C_150 equ $06
1181 EPSON equ $07
1182 EPSON_JX_80 equ $08
1183 OKIMATE_20 equ $09
1184 QUME_IP_20 equ $0A
1185 ; new printer entries, 3 October 1985
1186 HP_LASERJET equ $0B
1187 HP_LASERJET_PLUS equ $0C
1188
1189
1190
1191
1192 ; Remember
1193
1194 ; this structure is used for remembering what memory has been allocated to
1195 ; date by a given routine, so that a premature abort or systematic exit
1196 ; can deallocate memory cleanly, easily, and completely
1197 ; STRUCTURE Remember, 0
1198
1199 APTR rm_NextRemember
1200 ULONG rm_RememberSize
1201 APTR rm_Memory
1202
1203 LABEL rm_SIZEOF
1204
1205
1206
1207
1208 ; Miscellaneous
1209
1210
1211 ; MACROS
1212 ;#define MENUNUM(n) (n & 0x1F)
1213 ;#define ITEMNUM(n) ((n >> 5) & 0x003F)
1214 ;#define SUBNUM(n) ((n >> 11) & 0x001F)
1215 ;#define SHIFITEM(n) (n & 0x1F)
1216 ;#define SHIFTSUB(n) ((n & 0x3F) << 5)
1217 ;#define SHIFTSUB(n) ((n & 0x1F) << 11)
1218
1219 ; = MENU STUFF
1220
1221 NOMENU equ $001F
1222 NOITEM equ $003F
1223 NOCSUB equ $001F
1224 MENUNULL equ $FFFF
1225
1226
1227
1228 ; = -RJ='s peculiarities
1229 ;#define FOREVER for(;;)
1230 ;#define SIGN(x) ((x) > 0) - ((x) < 0)
1231
1232
1233

```

```

1233 ; these defines are for the COMSEQ and CHECKIT menu stuff. If CHECKIT,
1234 ; I'll use a generic Width (for all resolutions) for the CheckMark.
1235 ; If COMSEQ, likewise I'll use this generic stuff
1236 CHECKWIDTH equ 19
1237 COMWIDTH equ 27
1238 LOWCHECKWIDTH equ 13
1239 LOWCOMWIDTH equ 16
1240
1241
1242 ; these are the AlertNumber defines. if you are calling DisplayAlert()
1243 ; the AlertNumber you supply must have the ALERT_TYPE bits set to one
1244 ; of these patterns
1245 ALERT_TYPE equ $80000000
1246 RECOVERY_ALERT equ $00000000 ; the system can recover from this
1247 DEADEND_ALERT equ $80000000 ; no recovery possible, this is it
1248
1249
1250 ; when you're defining IntuiText for the Positive and Negative Gadgets
1251 ; created by a call to AutoRequest(), these defines will get you
1252 ; reasonable-looking text. The only field without a define is the IText
1253 ; field; you decide what text goes with the Gadget
1254 AUTOFRONTIPEN equ 0
1255 AUTOBACKPEN equ 1
1256 AUTORAMMODE equ RP_JAM2
1257 AUTOLEFTEDGE equ 6
1258 AUTORIGHTEDGE equ 3
1259 AUTOITEXTFONT equ 0
1260 AUTONEXTTEXT equ 0
1261
1262
1263
1264
1265
1266
1267

```

```

IFND INTUITION_INTUITIONBASE_I
INTUITION_INTUITIONBASE_I SET 1
*** intuitionbase.1 *****
* Commodore-Amiga, Inc.
*
* the IntuitionBase structure and supporting structures
*
* Modification History
* date : author : Comments
* -----
* 3-1-85 -jmm
*
*****
IFND EXEC_LIBRARIES_I
INCLUDE "exec/libraries.i"
ENDC
IFND GRAPHICS_VIEW_I
INCLUDE "graphics/view.i"
ENDC
* Be sure to protect yourself against someone modifying these data as
* you look at them. This is done by calling:
* lock = LockIBase(0)..which returns a ULONG. When done call
* D0 :D0
* UnlockIBase(lock) where lock is what LockIBase() returned.
* NOTE: these library functions are simply stubs now, but should be called
* to be compatible with future releases.
*
* == IntuitionBase ==
* == IntuitionBase,0 ==
STRUCTURE IntuitionBase,0
STRUCT ib_LibNode,LIB_SIZE
STRUCT ib_ViewLord,SIZEOF_VIEW
APTR ib_ActiveWindow
APTR ib_ActiveScreen
* the FirstScreen variable points to the frontmost Screen. Screens are
* then maintained in a front to back order using Screen.NextScreen
APTR ib_FirstScreen
* there is not size here because...
*
* ENDC

```

```

1  IEND LIBRARIES_DISKFONT_I
2  LIBRARIES_DISKFONT_I SET 1
3  *****
4  * Commodore-Amiga, Inc.
5  *
6  * diskfont.i
7  *
8  *
9  * diskfont library definitions
10 *****
11 *****
12 *****
13 IEND EXEC_NODES_I
14 INCLUDE "exec/nodes.1"
15 ENDC
16 IEND EXEC_LISTS_I
17 INCLUDE "exec/lists.1"
18 ENDC
19 IEND GRAPHICS_TEXT_I
20 INCLUDE "graphics/text.1"
21 ENDC
22
23 MAXONTPATH EQU 256 ; including null terminator
24
25 STRUCTURE FC_0
26 STRUCT fc_FileName, MAXONTPATH
27 UNORD fc_ysize
28 UBYTE fc_Style
29 UBYTE fc_Flags
30 LABEL fc_SIZEOF
31
32 FCH_ID EQU $0f00
33
34 STRUCTURE FCH_0
35 UNORD fch_FileID ; FCH_ID
36 UNORD fch_NumEntries ; the number of FontContents elements
37 LABEL fch_FC ; the FontContents elements
38
39
40 DFH_ID EQU $0f80
41 MAXFONTNAME EQU 32 ; font name including ".font\0"
42
43 STRUCTURE DiskFontHeader_0
44 ; the following 8 bytes are not actually considered a part of the
45 ; DiskFontHeader, but immediately precede it. The NextSegment is supplied
46 ; by the linker/loader, and the ReturnCode is the code at the beginning
47 ; of the font in case someone runs it...
48 ; ULONG dfh_NextSegment ; actually a BEYR
49 ; ULONG dfh_ReturnCode ; MOVEQ #0,D0 ; RTS
50 ; here then is the official start of the DiskFontHeader...
51 STRUCT dfh_DF_LM_SIZE ; node to link disk fonts
52 UNORD dfh_FileID ; DFH_ID
53 UNORD dfh_Revision ; the font revision in this version
54 LONG dfh_Segment ; the segment address when loaded
55 STRUCT dfh_Name, MAXFONTNAME ; the font name (null terminated)
56 STRUCT dfh_TF, tf_SIZEOF ; loaded TextFont structure

```

```

57 LABEL dfh_SIZEOF
58
59 BITDEF AF_MEMORY, 0
60 BITDEF AF_DISK, 1
61
62 STRUCTURE AF_0
63 UNORD af_Type ; MEMORY or DISK
64 STRUCT af_Attr, ta_SIZEOF ; text attributes for font
65 LABEL af_SIZEOF
66
67 STRUCTURE AFH_0
68 UNORD afh_NumEntries ; number of AvailFonts elements
69 LABEL afh_AF ; the AvailFonts elements
70
71 ENDC
72

```

```

1  IFND LIBRARIES_DOS_I SET 1
2  LIBRARIES_DOS_I
3  *****
4  * Commodore-Amiga, Inc.
5  * dos.1
6  *
7  * Standard assembler header for Amiga DOS on the MC68000
8
9  * IFND EXEC_TYPES_I
10 * INCLUDE "exec/types.1"
11 * ENDC
12 *
13 *
14 *
15 * DOSNAME MACRO
16 * DC.B 'dos.library',0
17 * ENDM
18
19 * Predefined Amiga DOS global constants
20
21 * Mode parameter to Open()
22 MODE_OLDFILE EQU 1005 * Open existing file read/write
23 *
24 MODE_NEWFILE EQU 1006 * Open freshly created file (delete
25 * * old file) read/write
26
27 * Relative position to Seek()
28 OFFSET_BEGINNING EQU -1 * relative to Beginning Of File
29 OFFSET_CURRENT EQU 0 * relative to Current file position
30 OFFSET_END EQU 1 * relative to End Of File
31
32 OFFSET_BEGINNING EQU OFFSET_BEGINNING * Ancient compatibility
33
34 BITSPERBYTE EQU 8
35 BYTESPERLONG EQU 4
36 BITSPERLONG EQU 32
37 MAXINT EQU $7FFFFFFF
38 MININT EQU $80000000
39
40 * Passed as type to Lock()
41 SHARED_LOCK EQU -2 ; File is readable by others
42 ACCESS_READ EQU -2 ; Synonym
43 EXCLUSIVE_LOCK EQU -1 ; No other access allowed
44 ACCESS_WRITE EQU -1 ; Synonym
45
46
47 STRUCTURE DateStamp,0
48 LONG ds_Days ; Number of days since Jan. 1, 1978
49 LONG ds_Minute ; Number of minutes past midnight
50 LONG ds_Tick ; Number of ticks past minute
51 LABEL ds_SIZEOF DateStamp
52 TICKS_PER_SECOND EQU 50 ; Number of ticks in one second
53
54 * Returned by Examine() and ExInfo()
55 STRUCTURE FileInfoBlock,0
56 LONG fib_DiskKey

```

```

57 LONG fib_DirEntryType
58
59 STRUCT fib_FileNames,108
60 LONG fib_Protection
61 LONG fib_EntryType
62 LONG fib_Size
63 LONG fib_NumBlocks
64 STRUCT fib_DateStamp,ds_SIZEOF ; Date file last changed.
65 STRUCT fib_Comment,116 ; Null terminated.
66 LABEL fib_SIZEOF ; Comment associated with file
67 ; FileInfoBlock
68
69 * FIB stands for FileInfoBlock
70 * FIBB are bit definitions, FIBF are field definitions
71 BITDEF FIB_READ,3
72 BITDEF FIB_WRITE,2
73 BITDEF FIB_EXECUTE,1
74 BITDEF FIB_DELETE,0
75
76
77 * All BCPL data must be long word aligned. BCPL pointers are the long word
78 * address (i.e byte address divided by 4 (>>2))
79
80 * Macro to indicate BCPL pointers * Long word pointer
81 BPTR MACRO \1
82 LONG
83 ENDM
84 BSTR MACRO \1
85 LONG
86 ENDM
87
88 * #define BADDR ( bptr ) ( bptr << 2) * Convert BPTR to byte addressed pointer
89
90 * BCPL strings have a length in the first byte and then the characters.
91 * For example: s[0]=3 s[1]=S s[2]=Y s[3]=S
92
93 * returned by Info()
94 STRUCTURE InfoData,0
95 LONG id_NumSoftErrors
96 LONG id_UnitNumber
97 LONG id_DiskState
98 LONG id_NumBlocks
99 LONG id_BytesPerBlock
100 LONG id_DiskType
101 BPTR id_VolumeNode
102 LONG id_InUse
103 LABEL id_SIZEOF
104
105
106 * ID stands for InfoData
107 * Disk states
108 ID_WRITE_PROTECTED EQU 80 * Disk is write protected
109 ID_VALIDATING EQU 81 * Disk is currently being validated
110 ID_VALIDATED EQU 82 * Disk is consistent and writeable
111 * Disk types
112 ID_NO_DISK_PRESENT EQU -1

```

```

113 ID_UNREADABLE_DISK EQU ('B'<<24) | ('A'<<16) | ('D'<<8)
114 ID_NOT_REALY_DOS EQU ('N'<<24) | ('D'<<16) | ('O'<<8) | ('S')
115 ID_DOS_DISK EQU ('D'<<24) | ('O'<<16) | ('S'<<8)
116 ID_KICKSTART_DISK EQU ('K'<<24) | ('I'<<16) | ('C'<<8) | ('K')
117 * Errors from IoErr (), etc.
118 ERROR_NO_FREE_STORE EQU 103
119 ERROR_OBJECT_IN_USE EQU 202
120 ERROR_OBJECT_EXISTS EQU 203
121 ERROR_OBJECT_NOT_FOUND EQU 205
122 ERROR_ACTION_NOT_KNOWN EQU 209
123 ERROR_INVALID_COMPONENT_NAME EQU 210
124 ERROR_INVALID_LOCK EQU 211
125 ERROR_OBJECT_WRONG_TYPE EQU 212
126 ERROR_DISK_NOT_VALIDATED EQU 213
127 ERROR_DISK_WRITE_PROTECTED EQU 214
128 ERROR_RENAME_ACROSS_DEVICES EQU 215
129 ERROR_DIRECTORY_NOT_EMPTY EQU 216
130 ERROR_DEVICE_NOT_MOUNTED EQU 218
131 ERROR_SEEK_ERROR EQU 219
132 ERROR_COMMENT_TOO_BIG EQU 220
133 ERROR_DISK_FULL EQU 221
134 ERROR_DELETE_PROTECTED EQU 222
135 ERROR_WRITE_PROTECTED EQU 223
136 ERROR_READ_PROTECTED EQU 224
137 ERROR_NOT_A_DOS_DISK EQU 225
138 ERROR_NO_DISK EQU 226
139 ERROR_NO_MORE_ENTRIES EQU 232
140
141
142 * These are the return codes used by convention by AmigaDOS commands
143 * See FAILAT and IF for relevance to EXECUTE files
144 RETURN_OK EQU 0 * No problems, success
145 RETURN_WARN EQU 5 * A warning only
146 RETURN_ERROR EQU 10 * Something wrong
147 RETURN_FAIL EQU 20 * Complete or severe failure
148
149 * Bit numbers that signal you that a user has issued a break
150 BITDEF SIGBREAK_CTRL_C,12
151 BITDEF SIGBREAK_CTRL_D,13
152 BITDEF SIGBREAK_CTRL_E,14
153 BITDEF SIGBREAK_CTRL_F,15
154
155 ENDC LIBRARIES_DOS_1

```

```

*****
1 * Commodore-Amiga, Inc.
2 * dos.lib.1
3 *
4 *
5 *
6 * Library interface offsets for DOS library
7 *
8 reserve EQU 4
9 vsize EQU 6
10 count SET -vsize*(reserve+1)
11 LIBENT MACRO count
12 _LVO\1 EQU count-vsize
13 count SET count-vsize
14 ENDM
15 *
16 *
17 *
18 LIBENT Open
19 LIBENT Close
20 LIBENT Read
21 LIBENT Write
22 LIBENT Input
23 LIBENT Output
24 LIBENT Seek
25 LIBENT DeleteFile
26 LIBENT Rename
27 LIBENT Lock
28 LIBENT UnLock
29 LIBENT DupLock
30 LIBENT Examine
31 LIBENT ExNext
32 LIBENT Info
33 LIBENT Createdir
34 LIBENT CurrentDir
35 LIBENT IoErr
36 LIBENT CreateProc
37 LIBENT Exit
38 LIBENT LoadSeg
39 LIBENT UnLoadSeg
40 LIBENT GetPacket
41 LIBENT QueuePacket
42 LIBENT DeviceProc
43 LIBENT SetComment
44 LIBENT SetProtection
45 LIBENT DateStamp
46 LIBENT Delay
47 LIBENT WaitForChar
48 LIBENT ParentDir
49 LIBENT IsInteractive
50 LIBENT Execute

```

```

1 IFND LIBRARIES_DOSEXTENS_I
2
3 LIBRARIES_DOSEXTENS_I SET 1
4 *****
5 * Commodore-Amiga, Inc.
6 * dosextens.i
7 *****
8 * DOS structures not needed for the casual DOS user
9

```

```

10 IFND EXEC_TYPES_I
11 INCLUDE "exec/types.i"
12 ENDC
13 IFND EXEC_TASKS_I
14 INCLUDE "exec/tasks.i"
15 ENDC
16 IFND EXEC_PORTS_I
17 INCLUDE "exec/ports.i"
18 ENDC
19 IFND EXEC_LIBRARIES_I
20 INCLUDE "exec/libraries.i"
21 ENDC
22 IFND LIBRARIES_DOS_I
23 INCLUDE "libraries/dos.i"
24 ENDC

```

* All DOS processes have this STRUCTURE
 * Create and DeviceProc returns pointer to the MsgPort in this STRUCTURE
 * Process_addr = DeviceProc(..) - TC_SIZE

```

33 STRUCTURE Process, 0
34 STRUCT pr_Task, TC_SIZE
35 STRUCT pr_MsgPort, MP_SIZE
36 WORD pr_Pad
37 BPTR pr_SegList
38 LONG pr_StackSize
39 APTR pr_GlobVec
40 LONG pr_TaskNum
41 BPTR pr_StackBase
42 LONG pr_Result2
43 BPTR pr_CurrentDir
44 BPTR pr_CIS
45 BPTR pr_COS
46 APTR pr_ConsoleTask
47 APTR pr_FileSystemTask
48 BPTR pr_CLI
49 APTR pr_ReturnAddr
50 APTR pr_PktWait
51 APTR pr_WindowPtr
52 LABEL pr_SIZEOF
53

```

* The long word address (BPTR) of this STRUCTURE is returned by
 * Open() and other routines that return a file. You need only worry
 * about this STRUCT to do async io's via PutMsg() instead of

57 * standard file system calls

```

58 STRUCTURE FileHandle, 0
59 APTR fh_Link
60 APTR fh_Interactive
61 APTR fh_Type
62 LONG fh_Buf
63 LONG fh_Pos
64 LONG fh_End
65 LONG fh_Funcs
66 fh_Func1 EQU fh_Funcs
67 LONG fh_Func2
68 LONG fh_Func3
69 LONG fh_Args
70 fh_Arg1 EQU fh_Args
71 LONG fh_Arg2
72 LABEL fh_SIZEOF * FileHandle
73

```

* This is the extension to EXEC Messages used by DOS
 STRUCTURE DosPacket, 0

```

76 APTR dp_Link
77 APTR dp_Port
78 *
79 * LONG dp_Type
80 * LONG dp_Res1
81 *
82 * LONG dp_Res2
83 *
84 *
85 * LONG dp_Res2
86 *
87 * LONG dp_Arg1
88 *
89 * Device packets common equivalents
90 dp_Action EQU dp_Type
91 dp_Status EQU dp_Res1
92 dp_Status2 EQU dp_Res2
93 dp_BufAddr EQU dp_Arg1
94 LONG dp_Arg2
95 LONG dp_Arg3
96 LONG dp_Arg4
97 LONG dp_Arg5
98 LONG dp_Arg6
99 LONG dp_Arg7
100 LABEL dp_SIZEOF * DosPacket
101

```

* A Packet does not require the Message to before it in memory, but
 * for convenience it is useful to associate the two.
 * Also see the function init_std_pkt for initializing this STRUCTURE

```

106 STRUCTURE StandardPacket, 0
107 STRUCT sp_Msg, MW_SIZE
108 STRUCT sp_Pkt, dp_SIZEOF
109 LABEL sp_SIZEOF * StandardPacket
110
111
112 * Packet types

```



```

113 ACTION_NIL EQU 0
114 ACTION_GET_BLOCK EQU 2
115 ACTION_SET_MAP EQU 4
116 ACTION_DIE EQU 5
117 ACTION_EVENT EQU 6
118 ACTION_CURRENT_VOLUME EQU 7
119 ACTION_LOCATE_OBJECT EQU 8
120 ACTION_RENAME_DISK EQU 9
121 ACTION_WRITE EQU 'w'
122 ACTION_READ EQU 'r'
123 ACTION_FREE_LOCK EQU 15
124 ACTION_DELETE_OBJECT EQU 16
125 ACTION_RENAME_OBJECT EQU 17
126
127 ACTION_COPY_DIR EQU 19
128 ACTION_WAIT_CHAR EQU 20
129 ACTION_SET_PROTECT EQU 21
130 ACTION_CREATE_DIR EQU 22
131 ACTION_EXAMINE_OBJECT EQU 23
132 ACTION_EXAMINE_NEXT EQU 24
133 ACTION_DISK_INFO EQU 25
134 ACTION_DISK_INFO EQU 26
135
136 ACTION_SET_COMMENT EQU 28
137 ACTION_PARENT EQU 29
138 ACTION_TIMER EQU 30
139 ACTION_INHIBIT EQU 31
140 ACTION_DISK_TYPE EQU 32
141 ACTION_DISK_CHANGE EQU 33
142
143
144 * DOS library node structure.
145 * This is the data at positive offsets from the library node.
146 * Negative offsets from the node is the jump table to DOS functions
147 * node = (STRUCT DosLibary *) OpenLibrary( "dos.library" ... )
148
149 STRUCTURE DosLibary,0
150 STRUCT dl_lib,LIB_SIZE
151 APTR dl_Root
152 APTR dl_GV
153 LONG dl_A2
154 LONG dl_A5
155 LONG dl_A6
156 LABEL dl_SIZEOF * DosLibary
157 *
158
159
160 STRUCTURE RootNode,0
161 BPTR rn_TaskArray
162 *
163 * [0] is max number of CLI's
164 * [1] is APTR to process id of CLI 1
165 * [n] is APTR to process id of CLI n
166 * SegList for the CLI
167 BPTR rn_ConsoleSegment
168 STRUCT rn_Time,ds_SIZEOF
169 LONG rn_RestartSeg
170 BPTR rn_Info
171 LABEL rn_SIZEOF
172
173 * Pointer of the Info structure
174 * RootNode

```

```

169 STRUCTURE DosInfo,0
170 BPTR dl_McName
171 BPTR dl_DeVInfo
172 BPTR dl_Devices
173 BPTR dl_Handlers
174 APTR dl_NetHand
175 LABEL dl_SIZEOF
176
177 * Network name of this machine currently
178 * Device List
179 * Currently zero
180 * Network handler processid currently zero
181 * DosInfo
182
183 * DOS Processes started from the CLI via RUN or NEWCLI have this additional
184 * set to data associated with them
185
186 STRUCTURE CommandLineInterface,0
187 LONG cli_Result2
188 BSTR cli_SetName
189 BSTR cli_CommandDir
190 LONG cli_ReturnCode
191 BSTR cli_CommandName
192 LONG cli_FailLevel
193 BSTR cli_Prompt
194 BSTR cli_StandardInput
195 BSTR cli_CurrentInput
196 BSTR cli_CommandFile
197 LONG cli_Interactive
198 LONG cli_Background
199 BPTR cli_CurrentOutput
200 LONG cli_DefaultStack
201 BPTR cli_StandardOutput
202 BPTR cli_Module
203 LABEL cli_SIZEOF
204
205 * Value of IoErr from last command
206 * Name of current directory
207 * Lock associated with command directory
208 * Return code from last command
209 * Name of current command
210 * Fail level (set by FAILAT)
211 * Current prompt (set by PROMPT)
212 * Default (terminal) CLI input
213 * Current CLI input
214 * Name of EXECUTE command file
215 * Boolean True if prompts required
216 * Boolean True if CLI created by RUN
217 * Current CLI output
218 * Stack size to be obtained in long words
219 * Default (terminal) CLI output
220 * SegList of currently loaded command
221 * CommandLineInterface
222
223 * this structure needs some work. It should really be a union, because
224 * it can take on different valued depending on whether it is a device,
225 * an assigned directory, or a volume.
226 * For now, it reflects a volume.
227
228 STRUCTURE DevList,0
229 BPTR dl_Next
230 LONG dl_Type
231 APTR dl_Task
232 BPTR dl_Lock
233 STRUCT dl_VolumeDate,ds_SIZEOF
234 LONG dl_DiskType
235 LONG dl_Unused
236 BSTR dl_Name
237 LABEL DevList_SIZEOF
238
239 * bptr to next device list
240 * see DLT below
241 * ptr to handler task
242 * not for volumes
243 * creation date
244 * outstanding locks
245 * 'DOS', etc
246 * bptr to bcpl name
247
248 * definitions for dl_Type
249 DLT_DEVICE EQU 0
250 DLT_DIRECTORY EQU 1
251 DLT_VOLUME EQU 2

```

```

225 * a lock structure, as returned by Lock() or DupLock()
226 STRUCTURE FileLock, 0
227 BPTR fl_Link ; bcp1 pointer to next lock
228 LONG fl_Key ; disk block number
229 LONG fl_Access ; exclusive or shared
230 APTR fl_Task ; handler task's port
231 BPTR fl_Volume ; bptr to a DeviceList
232 LABEL fl_SIZEOF
233
234 ENDC LIBRARIES_DOSEXTENS_1

```

```

1 IFND LIBRARIES_TRANSLATOR_I
2 LIBRARIES_TRANSLATOR_I SET 1
3 *****
4 * Commodore-Amiga, Inc.
5 * translator.i
6 *****
7
8 * ;----- Translator error codes
9 TR_NotUsed EQU -1 ;This is an often used system rc
10 TR_NoMem EQU -2 ;Can't allocate memory
11 TR_MakeBad EQU -4 ;Error in MakeLibrary call
12
13
14
15 ENDC

```

* Commodore-Amiga, Inc.
* ciabase.i

* CIA Resource Data Definition

STRUCTURE CIAR, LIB_SIZE
APTR CR_HWADDR
UMWORD CR_IntMask
UBYTE CR_IEnable
UBYTE CR_IActive
STRUCT CR_INTNODE, IS_SIZE
STRUCT CR_IVTA, IV_SIZE
STRUCT CR_IVTB, IV_SIZE
STRUCT CR_IVALRM, IV_SIZE
STRUCT CR_IVSP, IV_SIZE
STRUCT CR_IVELG, IV_SIZE
LABEL CR_SIZE

1 IFND RESOURCES_DISK_I
2 RESOURCES_DISK_I SET I
3 *****
4 * Commodore-Amiga, Inc.
5 * disk.i
6 *****
7 *****

8 *****
9 * external declarations for disk resources
10 *
11 *
12 * SOURCE CONTROL
13 * -----
14 * \$Header: disk.i,v 27.3 85/07/12 23:17:43 neil Exp \$
15 *
16 * \$Locker: \$
17 *
18 *****

19 IFND EXEC_TYPES_I
20 INCLUDE "exec/types.i"
21 ENDC !EXEC_TYPES_I
22
23 IFND EXEC_LISTS_I
24 INCLUDE "exec/lists.i"
25 ENDC !EXEC_LISTS_I
26
27 IFND EXEC_PORTS_I
28 INCLUDE "exec/ports.i"
29 ENDC !EXEC_PORTS_I
30
31 IFND EXEC_INTERRUPTS_I
32 INCLUDE "exec/interrupts.i"
33 ENDC !EXEC_INTERRUPTS_I
34
35 IFND EXEC_LIBRARIES_I
36 INCLUDE "exec/libraries.i"
37 ENDC !EXEC_LIBRARIES_I
38
39
40
41
42
43 * Resource structures
44 *
45 *****

46
47 STRUCTURE DISRESOURCEUNIT, ML_SIZE
48 STRUCT DRU_DISCBLOCK, IS_SIZE
49 STRUCT DRU_DISCSYNC, IS_SIZE
50 STRUCT DRU_INDEX, IS_SIZE
51 LABEL DRU_SIZE
52
53
54
55 STRUCTURE DISRESOURCE, LIB_SIZE
56 APTR DR_CURRENT ; pointer to current unit structure

```

57 UBYTE DR_FLAGS
58 UBYTE DR_pad
59 APTR DR_SYSLIB
60 APTR DR_CTIRESOURCE
61 STRUCT DR_UNITID,4*4
62 STRUCT DR_WAITING,LR_SIZE
63 STRUCT DR_DISCKLOCK,IS_SIZE
64 STRUCT DR_DISCSYNC,IS_SIZE
65 STRUCT DR_INDEX,IS_SIZE
66 LABEL DR_SIZE
67
68 BITDEF DR_ALLOC,0 ; unit zero is allocated
69 BITDEF DR_ALLOC1,1 ; unit one is allocated
70 BITDEF DR_ALLOC2,2 ; unit two is allocated
71 BITDEF DR_ALLOC3,3 ; unit three is allocated
72 BITDEF DR_ACTIVE,7 ; is the disk currently busy?
73
74 *****
75 *
76 * Hardware Magic
77 *
78 *****
79 *****
80
81 DSKDMAOFF EQU $4000 ; idle command for dsklen register
82
83 *****
84 *****
85 *
86 * Resource specific commands
87 *
88 *****
89 *****
90
91 --- DR_NAME is a generic macro to get the name of the resource. This
92 --- way if the name is ever changed you will pick up the change
93 --- automatically.
94 ---
95 --- Normal usage would be:
96 ---
97 --- InternalName: DISKNAME
98 ---
99
100 DISKNAME: MACRO
101 DC.B 'disk.resource',0
102 DS.W 0
103 ENDM
104
105 LIBINIT LIB_BASE
106 LIBDEF DR_ALLOCUNIT
107 LIBDEF DR_FREEUNIT
108 LIBDEF DR_GETUNIT
109 LIBDEF DR_GIVEUNIT
110 LIBDEF DR_GETUNITID
111
112 DR_LASTCOMM EQU DR_GIVEUNIT

```

```

113 *****
114 *****
115 *****
116 *
117 * drive types
118 *
119 *****
120 *****
121 DRT_AMIGA EQU $00000000
122 DRT_37422D2S EQU $55555555
123 DRT_EMPTY EQU $FFFFFFF
124
125 ENDC

```

57 ENDC !RESOURCE_MISC_I

```

1  IFND RESOURCES_MISC_I
2  RESOURCES_MISC_I SET 1
3
4
5  IFND EXEC_TYPES_I
6  INCLUDE "exec/types.i"
7  ENDC !EXEC_TYPES_I
8
9  IFND EXEC_LIBRARIES_I
10 INCLUDE "exec/libraries.i"
11 ENDC !EXEC_LIBRARIES_I
12 *****
13 *****
14 ***** Commodore-Amiga, Inc. *****
15 ***** misc.i *****
16 *****
17 *****
18 *****
19 *****
20 ***** external declarations for misc system resources *****
21 *****
22 ***** SOURCE CONTROL *****
23 ***** ----- *****
24 ***** $Header: misc.1.v 27.3 85/07/12 16:29:36 mail Exp $ *****
25 *****
26 ***** $Locker: $ *****
27 *****
28 *****
29 *****
30 *****
31 *****
32 *****
33 ***** Resource structures *****
34 *****
35 *****
36 *****
37 MR_SERIALPORT EQU 0
38 MR_SERIALBITS EQU 1
39 MR_PARALLELPORT EQU 2
40 MR_PARALLELBITS EQU 3
41 *****
42 NUMBTYPES EQU 4
43 *****
44 ***** STRUCTURE MiscResource,LIB_SIZE *****
45 STRUCT mr_AllocArray,4*NUMBTYPES *****
46 LABEL mr_Sizeof *****
47 *****
48 LIBINIT LIB_BASE *****
49 LIBDEF MR_ALLOCMISRESOURCE *****
50 LIBDEF MR_FREEMISRESOURCE *****
51 *****
52 ***** MISCHNAME MACRO *****
53 DC.B 'misc.resource',0 *****
54 *****
55 ***** ENDM *****
56 *****

```

```

1  IFND  RESOURCES_POTGO_I
2  RESOURCES_POTGO_I EQU 1
3  *****
4  *      Commodore-Amiga, Inc.
5  *      potgo.i
6  *****
7  POTCONAME MACRO
8  DC.B 'potgo.resource'
9  DC.B 0
10 DS.W 0
11 ENDM
12 ENDC

```

```

1  IFND  WORKBENCH_ICON_I
2  WORKBENCH_ICON_I SET 1
3  *****
4  *      Commodore-Amiga, Inc.
5  *      icon.1
6  *****
7  *      Commodore-Amiga, Inc.
8  *      icon.1
9  *****
10 *****
11 *****
12 * icon.1 -- external declarations for workbench support library
13 *****
14 * SOURCE CONTROL
15 * -----
16 * $Header: icon.i.v 29.1 85/08/07 22:27:14 nail Exp $
17 *
18 * $Locker:  $
19 *
20 *****
21 *****
22 *****
23 *****
24 *
25 * Library structures
26 *
27 *****
28 *****
29 *****
30 ICONNAME MACRO
31 DC.B 'icon.library',0
32 ENDM
33 *****
34 ENDC !WORKBENCH_ICON_I

```

```

1 *** startup.i *****
2 *
3 *
4 * Workbench startup definitions
5 *
6 * Commodore-Amiga, Inc.
7 *
8 * $Header: startup.i.v 29.1 85/08/15 06:58:52 neil Exp $
9 *
10 * $Locker: $
11 *
12 *****
13
14 IFND EXEC_TYPES_I
15 INCLUDE "exec/types.i"
16 ENDC !EXEC_TYPES_I
17
18 IFND EXEC_PORTS_I
19 INCLUDE "exec/ports.i"
20 ENDC !EXEC_PORTS_I
21
22 IFND LIBRARIES_DOS_I
23 INCLUDE "libraries/dos.i"
24 ENDC !LIBRARIES_DOS_I
25
26 STRUCTURE WbStartup, 0
27 STRUCT sm_Message, MN_SIZE
28 APTR sm_Process
29 BPTR sm_Segment
30 LONG sm_NumArgs
31 APTR sm_ToolWindow
32 APTR sm_ArgList
33 LABEL sm_SIZEOF
34
35 STRUCTURE WbArg, 0
36 BPTR va_Lock
37 APTR va_Name
38 LABEL va_SIZEOF
39
; a standard message structure
; the process descriptor for you
; a descriptor for your code
; the number of elements in ArgList
; description of window
; the arguments themselves

; a lock descriptor
; a string relative to that lock

```

```

1 *****
2 *
3 *
4 * workbench.h
5 *
6 * Commodore-Amiga, Inc.
7 *
8 * $Header: workbench.i.v 31.2 85/09/02 21:32:18 neil Exp $
9 *
10 * $Locker: $
11 *
12 *****
13
14 IFND EXEC_TYPES_I
15 INCLUDE "exec/types.i"
16 ENDC !EXEC_TYPES_I
17
18 IFND EXEC_NODES_I
19 INCLUDE "exec/nodes.i"
20 ENDC !EXEC_NODES_I
21
22 IFND EXEC_LISTS_I
23 INCLUDE "exec/lists.i"
24 ENDC !EXEC_LISTS_I
25
26 IFND EXEC_TASKS_I
27 INCLUDE "exec/tasks.i"
28 ENDC !EXEC_TASKS_I
29
30 IFND INTUITION_INTUITION_I
31 INCLUDE "intuition/intuition.i"
32 ENDC !INTUITION_INTUITION_I
33
34 ; the Workbench object types
35 WBDISK EQU 1
36 WEDRAWER EQU 2
37 WETOOL EQU 3
38 WEPROJECT EQU 4
39 WECARBACE EQU 5
40 WEDevice EQU 6
41 WBEKICK EQU 7
42
43
44 ; the main workbench object structure
45 STRUCTURE DrawerData, 0
46 STRUCT dd_NewWindow, mn_SIZE ; args to open window
47 LONG dd_CurrentX ; current x coordinate of origin
48 LONG dd_CurrentY ; current y coordinate of origin
49 LONG dd_MinX ; smallest x coordinate in window
50 LONG dd_MinY ; smallest y coordinate in window
51 LONG dd_MaxX ; largest x coordinate in window
52 LONG dd_MaxY ; largest y coordinate in window
53 STRUCT dd_HorizScroll, 99_SIZEOF
54 STRUCT dd_VertScroll, 99_SIZEOF
55 STRUCT dd_UpMove, 99_SIZEOF
56

```

```

57 STRUCT dd_DownMove,gg_SIZEOF
58 STRUCT dd_LeftMove,gg_SIZEOF
59 STRUCT dd_RightMove,gg_SIZEOF
60 STRUCT dd_HorizImage,lg_SIZEOF
61 STRUCT dd_VertImage,lg_SIZEOF
62 STRUCT dd_HorizProp,pl_SIZEOF
63 STRUCT dd_VertProp,pl_SIZEOF
64 APTX dd_DrawerWin ; pointer to drawers window
65 APTX dd_Object ; back pointer to drawer object
66 STRUCT dd_Children,ln_SIZE ; where our children hang out
67 LONG dd_Lock
68 LABEL dd_SIZEOF
69
70 ; the amount of DrawerData actually written to disk
71 DRAWERDATAFILESIZE EQU (nw_SIZE+2*(4))
72
73
74 STRUCTURE DiskObject,0
75 UNORD do_Magic ; a magic num at the start of the file
76 UNORD do_Version ; a version number, so we can change it
77 STRUCT do_Gadget,gg_SIZEOF ; a copy of in core gadget
78 UNORD do_Type
79 APTX do_DefaultTool
80 APTX do_ToolTypes
81 LONG do_CurrentX
82 LONG do_CurrentY
83 APTX do_DrawerData ; only applies to tools
84 APTX do_ToolWindow ; only applies to tools
85 LONG do_StackSize
86 LABEL do_SIZEOF
87
88 MB_DISKMAGIC EQU 0x310 ; a magic number, not easily impersonated
89 MB_DISKVERSION EQU 1 ; our current version number
90
91 STRUCTURE FreeList,0
92 WORD fl_NumFree
93 STRUCT fl_MemList,ln_SIZE
94 ; weird name to avoid conflicts with FileLocks
95 LABEL FreeList_SIZEOF
96
97
98 STRUCTURE MRObject,0
99 STRUCT wo_MasterNode,ln_SIZE ; all objects are on this list
100 STRUCT wo_Siblings,ln_SIZE ; list of drawer members
101 STRUCT wo_SelectNode,ln_SIZE ; list of all selected objects
102 STRUCT wo_UtilityNode,ln_SIZE ; function specific linkages
103 APTX wo_Parent
104
105 ; object flags -- see below for definitions
106 UBYTE wo_Flags
107
108 UBYTE wo_Type ; what flavor object is this?
109 USHORT wo_UseCount ; number of references to this obj
110 APTX wo_Name ; this object's textual name
111 SHORT wo_NameOffset ; where to put the name
112 SHORT wo_NameYOffset

```

```

113 APTX wo_DefaultTool
114 APTX wo_DrawerData ; if this is a drawer or disk
115 APTX wo_IconWin ; each object's icon lives here
116 LONG wo_CurrentX ; virtual X in drawer
117 LONG wo_CurrentY ; virtual Y in drawer
118 APTX wo_ToolTypes ; the types for this tool
119 STRUCT wo_Gadget,gg_SIZEOF ; NOT a ptr, but an instance of it
120 STRUCT wo_FreeList,FreeList_SIZEOF ; this objects free list
121 APTX wo_ToolWindow ; character string for tool's window
122 LONG wo_StackSize ; how much stack to give to this
123 LONG wo_Lock ; if this tool is in the backdrop
124 LABEL wo_SIZEOF
125
126 ; workbench object flags
127 BITDEF MO,IconDisp,7 ; icon is currently in a window
128 BITDEF MO,DrawerOpen,6 ; we're a drawer, and it is open
129 BITDEF MO,Selected,5 ; our icon is selected
130 BITDEF MO,Background,4 ; set if icon is in background
131
132
133
134
135 * each message that comes into the WorkBenchPort must have a type field
136 * in the preceeding short. These are the defines for this type
137
138 MTYPE_PSTID EQU 1 ; a "standard Potion" message
139 MTYPE_TOOLEXIT EQU 2 ; exit message from our tools
140 MTYPE_DISKCHANGE EQU 3 ; dos telling us of a disk change
141 MTYPE_TIMER EQU 4 ; we got a timer tick
142 MTYPE_CLOSEDOWN EQU 5 ; <unimplemented>
143 MTYPE_IOPROC EQU 6 ; <unimplemented>
144
145 ; we use the gadget id field to encode some special information
146 GID_MROBJECT EQU 0 ; a normal workbench object
147 GID_HORIZSCROLL EQU 1 ; the horizontal scroll gadget for a drawer
148 GID_VERTSCROLL EQU 2 ; the vertical scroll gadget for a drawer
149 GID_LEFTSCROLL EQU 3 ; move one window left
150 GID_RIGHTSCROLL EQU 4 ; move one window right
151 GID_UPSCROLL EQU 5 ; move one window up
152 GID_DOWNSCROLL EQU 6 ; move one window down
153 GID_NAME EQU 7 ; the name field for an object
154
155
156 * workbench does different complement modes for its gadgets.
157 * It supports separate images, complement mode, and backfill mode.
158 * The first two are identical to intuitions GADIMAGE and GADIMAGE.
159 * backfill is similar to GADCOMP, but the region outside of the
160 * image (which normally would be color three when complemented)
161 * is flood-filled to color zero.
162
163 CADBACKFILL EQU 0001
164
165 * if an icon does not really live anywhere, set its current position
166 * to here
167 NO_ICON_POSITION EQU (00000000)
168

```




Appendix F

Exec Support Library

*
* Exec Support Functions -- NewList
*

INCLUDE "exec/types.i"
INCLUDE "exec/nodes.i"
INCLUDE "exec/lists.i"

section _NewList
xdef _NewList

_NewList:
 move.l 4(sp),a0
 NEWLIST a0
 rts

end

```
INCLUDE "exec/types.i"  
INCLUDE "exec/nodes.i"  
INCLUDE "exec/lists.i"  
INCLUDE "exec/libraries.i"  
INCLUDE "exec/ports.i"  
INCLUDE "exec/io.i"
```

```
section _BeginIO  
xdef _BeginIO
```

_BeginIO:

```
move.l 4(sp),a1  
BEGINIO  
rts
```

end

```
/*  
*  
*      Exec Support Functions -- Ports and Messages  
*  
*/
```

```
#include "exec/types.h"  
#include "exec/nodes.h"  
#include "exec/lists.h"  
#include "exec/memory.h"  
#include "exec/interrupts.h"  
#include "exec/ports.h"  
#include "exec/libraries.h"  
#include "exec/tasks.h"  
#include "exec/execbase.h"
```

```
extern APTR AllocMem();  
extern UBYTE AllocSignal();  
extern struct Task *FindTask();
```

```
struct MsgPort *CreatePort (name, pri)  
    char *name;  
    BYTE pri;  
{  
    UBYTE sigBit;  
    struct MsgPort *port;  
  
    if ((sigBit = AllocSignal (-1)) == -1)  
        return ((struct MsgPort *) 0);  
  
    port = AllocMem ((ULONG) sizeof (*port), MEMF_CLEAR | MEMF_PUBLIC);  
  
    if (port == 0) {  
        FreeSignal (sigBit);  
        return ((struct MsgPort *) (0));  
    }  
  
    port -> mp_Node.ln_Name = name;  
    port -> mp_Node.ln_Pri = pri;  
    port -> mp_Node.ln_Type = NT_MSGPORT;  
  
    port -> mp_Flags = PA_SIGNAL;  
    port -> mp_SigBit = sigBit;  
    port -> mp_SigTask = FindTask (0);  
  
    if (name != 0)  
        AddPort (port);  
    else  
        NewList (&(port -> mp_MsgList));  
  
    return (port);  
}
```

```
DeletePort(port)
    struct MsgPort *port;
{
    if ((port -> mp_Node.ln_Name) != 0)
        RemPort (port);

    port -> mp_Node.ln_Type = 0xff;
    port -> mp_MsgList.lh_Head = (struct Node *) - 1;

    FreeSignal (port -> mp_SigBit);

    FreeMem (port, (ULONG) sizeof (*port));
}
```

```
/*  
*  
*      Exec Support Functions -- Standard IO Requests  
*  
******/
```

```
#include "exec/types.h"  
#include "exec/nodes.h"  
#include "exec/lists.h"  
#include "exec/memory.h"  
#include "exec/interrupts.h"  
#include "exec/ports.h"  
#include "exec/libraries.h"  
#include "exec/io.h"  
#include "exec/tasks.h"  
#include "exec/execbase.h"
```

```
extern APTR AllocMem();
```

```
/*  
***** exec_support/CreateStdIO *****  
*  
* NAME  
* CreateStdIO() -- create a standard IO request  
*  
* SYNOPSIS  
* ioStdReq = CreateStdIO(ioReplyPort);  
*  
* FUNCTION  
* Allocates memory for and initializes a new IO request block.  
*  
* INPUTS  
* ioReplyPort - a pointer to an already initialized  
* message port to be used for this IO request's  
* reply port.  
*  
* RESULT  
* Returns a pointer to the new block. Pointer is of the type:  
* struct IOStdReq  
* 0 indicates inability to allocate enough memory for either  
* the request block.  
*  
* EXAMPLE  
* struct IOStdReq *myBlock;  
* if( (myBlock = CreateStdIO(myPort)) == NULL)  
* printf("Insufficient memory or not enough signals!");  
*  
* SEE ALSO  
* DeleteStdIO  
*  
******/
```

```
struct IOStdReq *CreateStdIO(ioReplyPort)  
struct MsgPort *ioReplyPort;  
{  
    struct IOStdReq *ioStdReq;
```



```
if (ioReplyPort == 0)
    return ((struct IOStdReq *) 0);

ioStdReq = AllocMem (sizeof (*ioStdReq), MEMF_CLEAR | MEMF_PUBLIC);

if (ioStdReq == 0)
    return ((struct IOStdReq *) 0);

ioStdReq -> io_Message.mn_Node.ln_Type = NT_MESSAGE;
ioStdReq -> io_Message.mn_Node.ln_Pri = 0;

ioStdReq -> io_Message.mn_ReplyPort = ioReplyPort;

return (ioStdReq);
}
```

```
/****** exec_support/DeleteStdIO *****/
*
* NAME
* DeleteStdIO(ioStdReq) - return memory allocated for IO request
*
* SYNOPSIS
* DeleteStdIO(ioStdReq);
*
* FUNCTION
* See summary line at NAME. Also frees the signal bit which
* had been allocated by the call to CreateStdIO.
*
* INPUTS
* A pointer to the IOStdReq block whose resources are to be freed.
*
* RESULT
* Frees the memory. Returns (no error conditions shown)
*
* EXAMPLE
* struct IOStdReq *myBlock;
* DeleteStdIO(myBlock);
*
* SEE ALSO
* CreateStdIO
*
*****/
```

```
DeleteStdIO(ioStdReq)
{
    struct IOStdReq *ioStdReq;

    ioStdReq -> io_Message.mn_Node.ln_Type = 0xff;
    ioStdReq -> io_Device = (struct Device *) -1;
    ioStdReq -> io_Unit = (struct Unit *) -1;

    FreeMem (ioStdReq, sizeof (*ioStdReq));
}
```

```

#include "exec/types.h"
#include "exec/nodes.h"
#include "exec/lists.h"
#include "exec/memory.h"
#include "exec/interrupts.h"
#include "exec/ports.h"
#include "exec/libraries.h"
#include "exec/tasks.h"
#include "exec/execbase.h"

extern APTR AllocMem();
extern struct Task *FindTask();

/*
 * Create a task with given name, priority, and stack size.
 * It will use the default exception and trap handlers for now.
 */
struct Task *CreateTask(name, pri, initPC, stackSize)
char *name;
UBYTE pri;
APTR initPC;
ULONG stackSize;
{
    struct Task *newTask;
    ULONG dataSize = (stackSize & 0xfffffc) + 1;

    /*
     * This should be broken into two allocations: task of PUBLIC
     * and stack of PRIVATE
     */
    newTask = AllocMem ((ULONG) sizeof (*newTask) + dataSize,
                       MEMF_CLEAR | MEMF_PUBLIC);

    if (!(ULONG) newTask) {
        return ((struct Task *) (0));
    }

    newTask -> tc_SPLower = (APTR)((long) newTask + (long) sizeof (*newTask));
    newTask -> tc_SPUpper = (APTR)((ULONG)(newTask -> tc_SPLower) + dataSize
                                   & 0xfffffe);
    newTask -> tc_SPReg = (APTR) ((long) (newTask -> tc_SPUpper));

    newTask -> tc_Node.ln_Type = NT_TASK;
    newTask -> tc_Node.ln_Pri = pri;
    newTask -> tc_Node.ln_Name = name;

    AddTask (newTask, initPC, 0);
    return (newTask);
}

DeleteTask(tc)
struct Task *tc;
{
    RemTask (tc); /* does not handle self deletion properly */
    FreeMem (tc, 1 + (ULONG) (tc -> tc_SPUpper) - (ULONG) tc);
}

```



```
/*
 *
 *      Exec Support Function -- Extended IO Request
 *
 */
```

```
#include "exec/types.h"
#include "exec/nodes.h"
#include "exec/lists.h"
#include "exec/memory.h"
#include "exec/interrupts.h"
#include "exec/ports.h"
#include "exec/libraries.h"
#include "exec/io.h"
#include "exec/tasks.h"
#include "exec/execbase.h"
```

```
extern APTR AllocMem();
```

```
/****** exec_support/CreateExtIO *****/
```

```
*
* NAME
* CreateExtIO() -- create an Extended IO request
```

```
* SYNOPSIS
* ioReq = CreateExtIO(ioReplyPort,size);
```

```
* FUNCTION
* Allocates memory for and initializes a new IO request block
* of a user-specified number of bytes.
```

```
* INPUTS
* ioReplyPort - a pointer to an already initialized
* message port to be used for this IO request's reply port.
```

```
* RESULT
* Returns a pointer to the new block. Pointer is of the type
* struct IORequest.
*
* 0 indicates inability to allocate enough memory for the request block
* or not enough signals available.
```

```
* EXAMPLE
* struct IORequest *myBlock;
* if( (myBlock = CreateExtIO(myPort,sizeof(struct IOExtID)) == NULL)
* exit(NO_MEM_OR_SIGNALS);
```

```
* example used to allocate space for IOExtID (trackdisk driver
* IO Request block for extended IO operations).
```

```
* SEE ALSO
* DeleteExtIO
```

```
*****/
```

```
struct IORequest *CreateExtIO(ioReplyPort,size)
```

```

struct MsgPort *ioReplyPort;
LONG size;
{
    struct IORequest *ioReq;

    if (ioReplyPort == 0)
        return ((struct IORequest *) 0);

    ioReq = (struct IORequest *)AllocMem (size, MEMF_CLEAR | MEMF_PUBLIC);

    if (ioReq == 0)
        return ((struct IORequest *) 0);

    ioReq -> io_Message.mn_Node.ln_Type = NT_MESSAGE;
    ioReq -> io_Message.mn_Node.ln_Pri = 0;

    ioReq -> io_Message.mn_ReplyPort = ioReplyPort;

    return (ioReq);
}

```

/****** exec_support/DeleteExtIO *****/

*
 * NAME
 * DeleteExtIO() - return memory allocated for extended IO request

* SYNOPSIS
 * DeleteExtIO(ioReq, size);

* FUNCTION
 * See summary line at NAME. Also frees the signal bit which
 * had been allocated by the call to CreateExtIO.

* INPUTS
 * A pointer to the IORequest block whose resources are to be freed.

* RESULT
 * Frees the memory. Returns (no error conditions shown)

* EXAMPLE
 * struct IORequest *myBlock;
 * DeleteExtIO(myBlock, (sizeof(struct IOExtTD)));
 *
 * example shows that CreateExtIO had been used to create a trackdisk
 * (extended) IO Request block.

* SEE ALSO
 * CreateExtIO

*****/

```

DeleteExtIO(ioExt, size)
    struct IORequest *ioExt;
    LONG size;
{
    ioExt -> io_Message.mn_Node.ln_Type = 0xff;
}

```

```
ioExt -> io_Device = (struct Device *) -1;  
ioExt -> io_Unit = (struct Unit *) -1;
```

```
FreeMem (ioExt, size);
```

```
}
```

TABLE OF CONTENTS

debug.lib/KDoFmt;
debug.lib/KGetChar;
debug.lib/KMayGetChar;
debug.lib/KPutChar;
debug.lib/KPutFmt;
debug.lib/KPutStr;

INTRODUCTION

This section outlines the routines available in the debug.lib.

The debug.lib is a linked-library rather than a shared library. You link to it at compile/link time rather than opening a library. This link-library becomes a physical part of your program code.

Routines that are described in the ROM Kernel manual as well as those listed here, can be called directly from tasks. Routines that are listed in the Lattice C manual, such as printf, and scanf for example, should only be called from a process, rather than from a task in that they require a process model in order to function. A process is started when you ask AmigaDOS to run your program. Any part of main() or any routines it calls become part of that process. If your program spawns any tasks on its own, to have those tasks execute any process-dependent code, the tasks should either send messages back to the main program, which in turn executes the appropriate code, or use these debug routines if the programmer has connected a 9600 baud device to the Amiga's serial port.

To start the debug mode correctly, you can set the serial port parameters by calling

```
romwack
```

from a CLI. Then, from the external terminal, type

```
go
```

The port is now set up and ready to go. The debug print and get-character routines will function as described below.

debug.lib/KDoFmt

debug.lib/KDoFmt

NAME

KDoFmt -- format data into a character stream.

SYNOPSIS

KDoFmt (FormatString, DataStream, PutChProc, PutChData);
A0 A1 A2 A3

FUNCTION

perform "C"-language-like formatting of a data stream,
outputting the result a character at a time

INPUTS

FormatString - a "C"-language-like null terminated format
string, with the following supported % types:
DataStream - a stream of data that is interpreted according to
the format string.
PutChProc - the procedure to call with each character to be
output, called as:
PutChProc (Char, PutChData);
D0-0:8 A3
the procedure is called with a null Char at the end of
the format string.
PutChData - an address register that passes thru to PutChProc.

debug.lib/KGetChar

debug.lib/KGetChar

NAME

KGetChar - get a character from the debug-console (a 9600 baud device attached to the serial port of the Amiga.)

SYNOPSIS

```
char = KGetChar();  
D0
```

FUNCTION

get the next character from the debug-console device.

debug.lib/KMayGetChar

debug.lib/KMayGetChar

NAME

KMayGetChar - return a char iff present, but don't block.
(device attached serial port).

SYNOPSIS

```
flagChar = KMayGetChar();  
D0
```

FUNCTION

return either a -1, saying that there is no char present, or
the char that was waiting

debug.lib/KPutChar

debug.lib/KPutChar

NAME

KPutChar - put a character to the debug-console
(device attached to Amiga serial port).

SYNOPSIS

```
char = KPutChar(char);  
D0          D0
```

FUNCTION

put a character to the debug-console device.

debug.lib/KPutFmt

debug.lib/KPutFmt

NAME

KPutFmt - print formatted data to the debug-console
(device attached to Amiga serial port).

SYNOPSIS

KPutFmt(format, values);
A0 A1

FUNCTION

print formatted data to the debug-console device

debug.lib/KPutStr

debug.lib/KPutStr

NAME

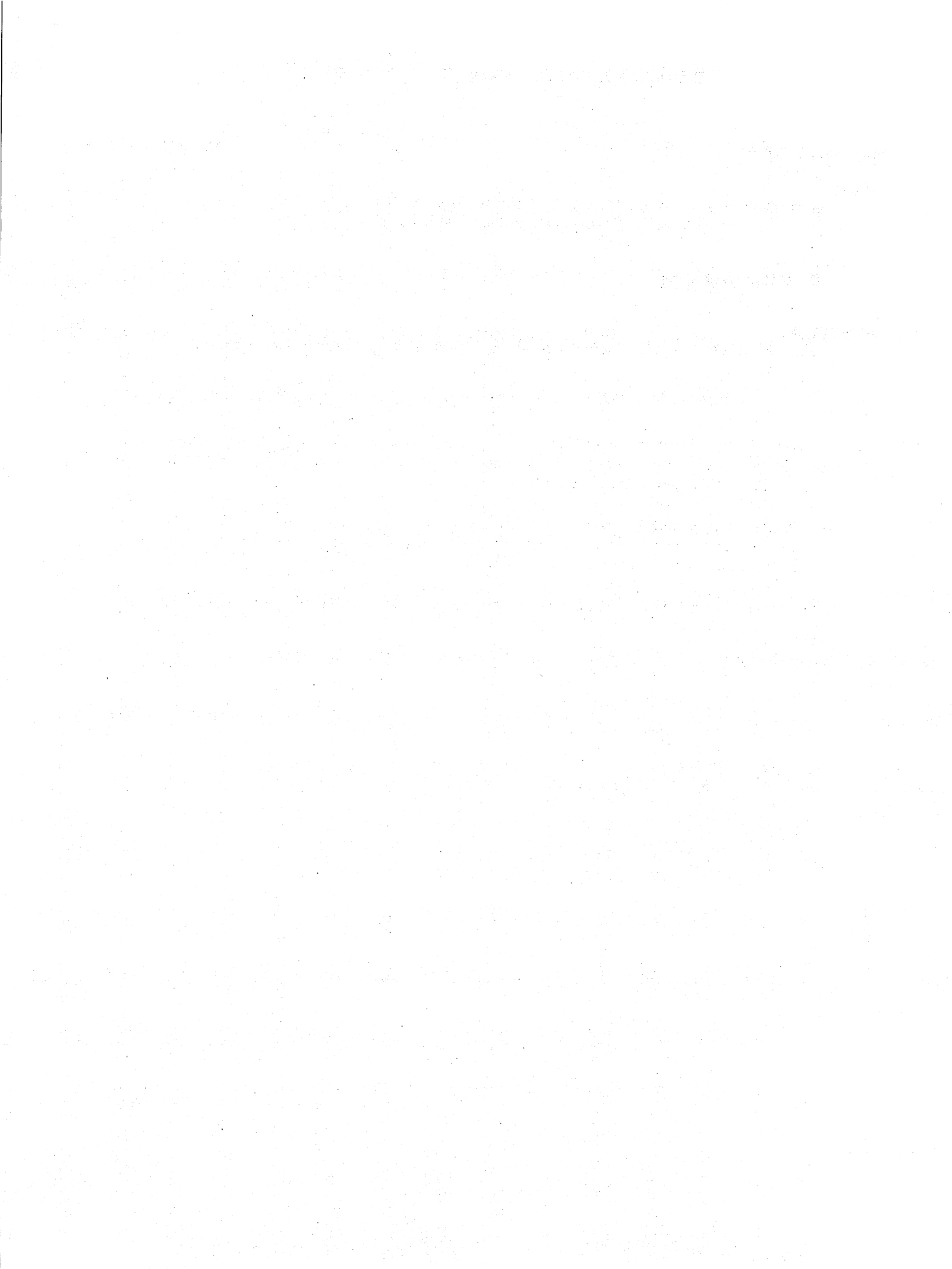
KPutStr - put a string to the debug-console
(attached to the Amiga serial port).

SYNOPSIS

KPutStr(string)
A0

FUNCTION

put a null terminated string to the debug-console device.



Appendix G

AmigaDOS Topics

The *Amiga ROM Kernel Manual* has become somewhat more than just a document covering the ROM Kernel. The name has been kept for historical reasons even though, as of this writing, the “ROM” software loads into kickstart RAM.

Certain topics were not available as of the most recent revision of the AmigaDOS manuals and are, therefore, included here to assure that Amiga developers will have the latest available information on these topics.

These DOS-related topics, which are additional information for the *AmigaDOS Technical Reference Manual*, are outlined on the following pages.

Chapter 4: AmigaDOS General Information

This chapter describes certain topics which are likely to be of interest to the advanced developer who may wish to create new devices to be added to the Amiga or who wish their code to run with Amiga computers which have been expanded beyond a 512k memory size.

The following topics are covered here:

Overlay Hunk Description
for developers putting together large programs

ATOM utility
works on a new binary file format to change allow developer to set the appropriate load bits.
Assures that program code and data that must be resident in CHIP memory (the lowest 512k of the system) for the program to function will indeed be placed there by AmigaDOS when it is loaded.
Otherwise the program code may not work on an extended memory machine.

Linking in a new DISK-device to AmigaDOS
lets a developer add a hard-disk or disk-like device as a name-addressable part of the filing system.

Linking in a new non-disk-device to AmigaDOS
lets a developer add such things as additional serial ports, parallel ports, graphics tablets, ram-disks or what-have-you to AmigaDOS (non filing system related).

Using AmigaDOS without using Intuition
for developers who may prefer to install and use their own screen handling in place of that provided by Intuition.

HUNK OVERLAY TABLE - OVERVIEW

When overlays are used, the linker basically produces one very large file containing all of the object modules as hunks of relocatable code. The hunk overlay table contains a data structure that describes the hunks and their relationship to each other.

When you are designing a program to use overlays, you must keep in mind how the overlay manager (also called the overlay supervisor) handles the interaction between the various segments of the file. What you must do, basically, is build a tree that reflects the relationships between the various code modules that are a part of the overall program and tell the linker how this tree should be constructed.

The hunk overlay table is generated as a sets of 8 longwords, each describing a particular overlay node that is part of the overall file. Each 8 longword entry is comprised of the following data:

HUNK OVERLAY SYMBOL TABLE-ENTRY DATA STRUCTURE:

```

long seekOffset; /* where in the file to find this node */
long dummy1; /* a value of 0 ... compatibility item */
long dummy2; /* a value of 0 ... compatibility item */
long level; /* level in the tree */
long ordinate; /* item number at that level */
long firstHunk; /* hunk number of the first hunk containing this node */
long symbolHunk; /* the hunk number in which this symbol is located */
long symbolOffsetX; /* (offset + 4), where offset is the offset within the symbol hunk at which this symbol's entry is located. */

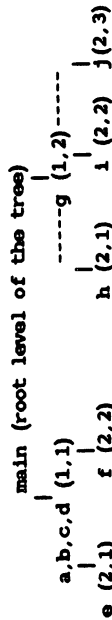
```

Each of these items is explained further in the sections that follow.

DESIGNING AN OVERLAY TREE

If you have, for example, the files: main, a, b, c, d, e, f, g, h, i and j. Lets say that main can call a,b,c, and d and that each of these files can call main. Additionally lets say that routine e can be called from a,b,c,d to be run, then a,b,c, and d need to be memory resident as well. Routine f is like e; that is, it needs nothing in e to be present, but can be called from a, b, c, or d. This means that the overlay manager can share the memory space between routines e and f, since neither need ever be memory coreisident with the other in order to run.

If you consider routine g to share the same space as the combination of a, b, c, and d and routines h, i and j sharing the same space, you have the basis for constructing the overlay tree for this program structure:



Not only have we drawn the tree, but we have labeled its branches to match the hunk overlay (level, ordinate) numbers that are found in the hunk overlay table that matches the nodes to which they are assigned.

From the description above, you can see that if main is to call any routine in program segment a-d, then all of those segments should be resident in memory at the same time. Thus they have all been assigned to a single node by the linker. While a-d are resident, if you call routines in e, the linker will automatically load routine e from disk, and reinitialize the module (each time it is again brought in), so that its subroutines will be available to be run. If any segment a-d calls a routine in f, the linker replaces e with the contents of f and initializes it. Thus a-d are at level 1 in the overlay tree, and routines e and f are at level 2, requiring that a-d be loaded before e or f can be accessed and loaded for execution.

NOTE: A routine can only perform calls to routines in other nodes which either are currently memory resident (the ancestors of the node in which the routine now in use is located), or a routine in a direct child node. That is, main cannot call e directly, but e can call routines in main since main is an ancestor.

Note also that within each branch of each subnode, for a given level, the ordinate numbers begin again with number 1.

DESCRIBING THE TREE

You create the tree by telling the overlay linker about its structure. The numerical values, similar to those noted in the figure above, are assigned sequentially by the linker itself and appear in the hunk node table. Here is the sequence of overlay link statements that cause the figure above to be built:

```

OVERLAY
a,b,c,d
*e
*f
g
*h
*i
  
```

This description tells the linker that a, b, c, d are part of a single node at a given level (in this case level 1), and the asterisk in front of e and f each say that these are one each on the next level down from a-d, and accessible only through a-d or anything closer towards the root of the tree. The name g has no asterisk, so it is considered on the same level as a-d, telling the linker that either a-d or g will be memory resident, but not both simultaneously. Names h and i are shown to be related to g, one level down.

The above paragraphs have explained the origin of the hunk node level and the hunk ordinate in the hunk overlay symbol-table.

SEEK OFFSET AMOUNT

The first value for each node in the overlay table is the seek offset. As specified earlier, the overlay linker builds a large single file containing all of the overlay nodes. The seek offset number is that value that can be given to the seek(file, byte_offset) routine to point to the first byte of the hunk header of a node.

INITIALHUNK

The initialhunk value in the overlay symbol-table is used by the overlay manager when unloading a node. It specifies the initial hunk that must have been loaded in order to have loaded the node that contains this symbol. When a routine is called at a different level and ordinate, (unless it is a direct, next level, child of the current node, it will become necessary to free the memory utilized by invalid hunks, so as to make room to overlay with the hunk(s) containing the desired symbol.

SYMBOLHUNK AND SYMBOLOFFSETX

These table entries for the symbols are used by the overlay manager to actually locate the entry point once it has either determined it is already loaded or has loaded it. The symbolhunk shows in which hunk to locate the symbol. SymbolOffsetX-4 shows the offset from the start of that hunk at which the entry point is actually located.

ATOM:

(Alink Temporary Object Modifier)

This document describes the ATOM utility, including its development history, the manner in which it has been implemented, and alternatives to its use.

The "problem":

Programmers need/want to be able to specify that parts of their program go into "chip" memory (the first 512k) so that the custom chips can access it. They also need/want to treat this data just like any other data in their program and therefore have it link and load normally.

Previous solutions:

The recommended way of dealing with this was to do an AllocMem with the chip memory bit set and copy data from where it was loaded ("fast" memory) to where it belonged (chip memory), then use pointers to get to it. This involved having two copies of your data in memory, the first loaded with your program, the second copied into the first 512k of memory.

The other "solution" is to have the program not run in machines with more than 512k. This should quickly become an unacceptable solution.

The ATOM solution:

- 1) Compile or assemble normally
- 2) Pass the object code through a post (or pre) processor called "ATOM". ATOM will interact with the user and the object file(s). It will flag the desired hunks (or all hunks) as "for chip memory" by changing the hunk type.
- 3) The linker will now take nine (9) hunk types instead of 3. The old types were hunk_code, hunk_data, and hunk_bss. The new ones will be:
 - hunk_code_chip = hunk_code + bit 30 set
 - hunk_code_fast = hunk_code + bit 31 set
 - hunk_data_chip = hunk_data + bit 30 set
 - hunk_data_fast = hunk_data + bit 31 set
 - hunk_bss_chip = hunk_bss + bit 30 set
 - hunk_bss_fast = hunk_bss + bit 31 set

The linker will pass all hunk types through to the LOADER, (coagulating if necessary). The LOADER uses the hunk

header information when loading.

You will recall from the information provided in the linker documentation that CODE hunks contain executable (68000) machine language, DATA hunks contain initialized data (constants,...) and BSS hunks contained uninitialized data (arrays, variable declarations, ...).

- 4) The LOADER will load according to information from step 3) above. Hunks will go into the designated memory type.
- 5) Old versions of the LOADER will interpret the new hunk types as VERY large hunk and not load (error 103, not enough memory).

Future Solutions:

The assembler and Lattice "C" may be changed to generate the new hunk types under programmer control.

How the bits work:

The hunk size is a word containing the number of words in the hunk. Therefore for the foreseeable future including 32 bit address space machines, the upper 2 bits are unused. The Bits have been redefined as follows:

| | | |
|-------|-------|-----------|
| ----- | Bit31 | MEME_FAST |
| | Bit30 | MEME_CHIP |
| | | |
| V | V | V |
| 0 | 0 | 0 |

If neither bit is set then get whatever memory is available, this is "backwards" compatible. Preference is given to "Fast" memory. Loader must get FAST memory, or fail. Loader must get CHIP memory, or fail. If Bit31 and Bit30 are both set then there is extra information available following this long word. This is reserved for future expansion, as needed. It is not currently used.

Perceived impact:

Old programs, programs that have not been compiled of assembled with the new options, and programs that have not been run through ATOM will run (or not) as well as ever. This includes crashing in extended memory, if poorly programmed. The "previous solutions" mentioned at the beginning of this chapter still hold.

Program development and test on a 512k machine could follow EXACTLY the same loop you have now... edit, compile, link, execute, test, edit,... UNTIL you are about to release. Then you edit, compile, ATOM, Alink, add external memory (>512k) and test. This works well for all three environments (Amiga, IBM and

Sun).

For native (Amiga) development on a >512k machine you may want to ATOM the few required object files so you can both run your linked program in an extended memory machine and take advantage of a large RAM: disk. The development cycle then becomes: edit, compile, optionally ATOM (if this code or data contains items needed by the blitter), link, execute, test, edit...

"New programs" will not load in a V1.0 Kickstart environment. The result will be error 103, not enough memory.

Old (V1.0 and before) versions of dumpobj and QMD will not work on files after ATOM has been run on them.

Working Environment:

To get all of this to work together you need Release 1.1 compatible copies of:

- ATOM (Version 1.0 or later)
- Alink (Version 3.30 or later)
- Kickstart (Release 1.1 or later) for DOS LOADER.
- DumpObj (Version 2.1) Needed if you wish to examine programs modified by ATOM.

ATOM Command Line Syntax:

The command line syntax is:

```
ATOM <infile> <outfile> [-I]
-- or --
ATOM <infile> <outfile> [-C[C|D|B]] [-F[C|D|B]] [-P[C|D|B]]
```

- Where:
- <infile> Represents an object file, just compiled, assembled or ATOMed (Yes you can re-ATOM an object file.
 - <outfile> The destination for the converted file.
 - C Change memory to CHIP
 - F Change memory to FAST
 - P Change memory to "Public". (Any type of memory available.)
 - C Change CODE hunks
 - D Change DATA hunks
 - B Change BSS hunks

Command Line Examples:

Example #1
In most cases there is no need to place CODE hunks in chip memory. Sometimes DATA and BSS hunks do need to

be placed in chip memory, therefore the following is a fairly common usage of ATOM. To cause all Code hunks go into Public RAM, Data and BSS hunks to go into chip RAM type:
ATOM Infile.obj outfile.obj -pc -cdb

Example #2
To cause all the hunks in object file to be loaded into chip memory type:
ATOM Infile.obj outfile.obj -c

Example #3
To set all data hunks to load into chip memory type:
atom myfile.o myfile.set.o -cd

Example #4
This is an interactive example. User input is in lower case, computer output is in upper case. In this example the code hunk is set to "Fast", the data hunk is set to "Chip". There were no BSS hunks. Note that help was requested in the beginning.

```
2> atom from.o from.set -1
AMIGA OBJECT MODIFIER V1.0
UNIT NAME FROM
HUNK NAME NONE
HUNK TYPE CODE
MEMORY ALLOCATION PUBLIC
DISPLAY SYMBOLS? [Y/N] Y
--base..
--xcovf..
--CMD22..
--printf..
--main....
```

{Note: code hunk}

MEMORY TYPE? [F|C|P] ?

- Please enter F for fast
C for Chip
P for Public
Q to quit
W to windup
N for Next hunk
- Memory type.
(cancels the operation, no output file is created)
(does not change the rest of the file, just passes it through)
(skip this hunk, show next)

{Note: request for help}

MEMORY TYPE? [F|C|P] f

```
UNIT NAME 0000
HUNKNAME NONE
HUNK TYPE DATA
MEMORY ALLOCATION PUBLIC
```

{Note: data hunk}

```

DISPLAY SYMBOLS? [Y/N] n
MEMORY TYPE? [F|C|P] c
UNIT NAME 0000
HUNKNAME NONE
HUNK TYPE BSS
MEMORY ALLOCATION PUBLIC
DISPLAY SYMBOLS? [Y/N] y
MEMORY TYPE? [F|C|P] p
2>

```

Error Messages:

Error Bad Args:

- a) An option does not start with a "-"
- b) wrong number of parameters
- c) "-" not followed by I, C, F or P.
- d) -x supplied in addition to -I etcetera.

Error Bad Infile:

File not found.

Error Bad Outfile:

File can not be created.

Error Bad Type ##: AUTOM has detected a hunk type that it does not recognize. The object file may be corrupt.

Error empty input: Input file does not contain any data.

Error ReadExternals: External reference or definition if of an undefined type. Object file may be corrupt.

Error premature end of file: An end of file condition (out of data) was detected while AUTOM was still expecting input. Object file may be corrupt.

Error This utility can only be used on files that have NOT been passed through ALINK:

The input file you specified has already been processed by the linker. External symbols have been removed and hunks coagulated. You need to run AUTOM on the object files produced by the C compiler or Macro Assembler BEFORE they are linked.

Document date: 20-Nov-1985

Making New Disk Devices

To create a new disk device, you must construct a new device node as described in section 3.3.1 of the AmigaDOS Technical manual. You must also write a device driver for the new disk device.

A device driver for a new disk device must mimic the calls that are performed by the trackdisk device (described in the Amiga ROM Kernel manual). It must include the ability to respond to commands such as read, write, seek, and return status information in the same way as described for the trackdisk driver.

For the following description, note that most pointers are of the type BPTR (as described in the AmigaDOS Technical Reference Manual), a machine pointer to some longword-aligned memory location (such as returned by AllocMem), shifted right by two.

Construct the new node with the following fields:

```

0      Next
0      dt_device
0      Task
0      Lock
0      Handler
210    Stacksize
10     Priority
BPTR to startup info
Seglist
0      Global vector
0      BSTR to name

```

The BSTR to a name is a BCPL pointer to the name of your new device (such as HD0:) represented as the length of the string in the first byte, and the characters following.

The Seglist must be the segment list of the filling system task. To obtain this, you must access a field in the process base of one of the filling system tasks.

The code as follows can be used for this purpose:

```

UBYTE *port;

port = DeviceProc("DF0:"); /* Returns msg port of filesystem task */
task = (struct Task *) (port-sizeof(struct Task));
list = ( task.pr_Seglist ) << 2; /* make machine ptr from SegArray */
segl = list[3]; /* Third element in SegArray is filesystem seglist */

```

Next, you must set up the startup info (again, remember to use BPTRs where needed). This info consists of a BPTR to three long words which contain:

- Unit number (do not use unit zero)
- Device driver name, stored as a BPTR to the device driver name.

Creating a New Device to Run under AmigaDOS

This section provides information about adding devices that are NOT part of the DOS filing system. The next section provides information about adding file-system-related devices (hard-disks, floppy disks), that is, devices that DOS can use to read and write files with their associated directories.

You would want to use this information to add a new device such as a new serial port or a new parallel port. In this case you may be creating a device named "SER2:" which is to act just like "SER:" as far as DOS is concerned.

There are two steps involved here. First, you must create a suitable device, a process that is not addressed here. Note: The code for creating a skeleton disk-resident device is contained in the Amiga ROM Kernel Manual.

Second, you must make this new device available as an AmigaDOS device. This process involves writing a suitable device handler (see ROM Kernel Manual) and installing it into the AmigaDOS structures.

This installation is handled by creating a suitable device node structure for your new device. This is similar to creating a Devinfo slot for a new disk device, except that the start-up argument can be anything you want, the Segment list slot is zero, and the file name of your disk-resident device handler is placed in the Filenames slot.

```

0 Next
0 dt_device
0 Task (or process id - see below)
0 Lock
0 BSTR Filenames of handler code
NNN Stacksize required
NN Priority required
XXX Startup information
0 Seglist (nonzero if you load the code)
0 Global vector required
0 BSTR Device Name
    
```

The device handler is the interface between your device and an application program. This is normally written in BCPL, and the AmigaDOS kernel will attempt to load the code of the handler and create a new process for it when it is first referenced. This is handled automatically when the kernel notices that the Task field in the Devinfo structure is zero. If the code is already loaded, the code segment pointer is placed in the Seglist field. If this field is zero, the kernel loads the code from the filename given in the Filenames field and updates the Seglist field.

If you want this automatic loading and process initialization to work, you must create a code module, which is written in BCPL or is written in assembler to look like a BCPL module. This ensures that the dynamic linking used by the kernel will work correctly.

If you are writing in assembler, the format of the code section must be as shown below. Note that you may use DATA and BSS sections, but

which must be terminated by a null byte which is included in the count (e.g., 4/H/'D'/0/0)

- EPTER to disk information

The disk size information contains the following longword fields:

- 11 Size of table
- 128 Disk block size in long words (assuming 512-byte blocksize)
- 0 Sector origin (i.e., first sector is sector zero)
- Number of surfaces (e.g., 2 for floppy disk)
- 1 Number of sectors per block (e.g., 11 for floppy disk)
- Number of blocks per track (e.g., 11 for floppy disk)
- 2 (or more, indicating number of blocks to be reserved at start)
- 0 Pre-allocation factor
- 0 Interleave factor (commonly 0)
- Lowest cylinder number
- Highest cylinder number (e.g., 79 for floppy disk)
- 5 (or more, indicating number of cache blocks)

Finally, the device node must be attached to the end of the list (note the Next fields are all EPTERs) of device nodes within the Info substructure. WARNING: the list to which this refers is NOT the same kind of list that is referenced in the Exec portion of the Amiga ROM Kernel manual, but is instead the kind of list described in the AmigaDOS Technical Reference Manual.

To partition a hard disk, you make two or more device nodes and set the lowest and highest cylinder numbers to partition the disk as desired.

each section must have same format as described here.

```

StartModule DC.L (EndModule-StartModule)//4 Size of module in lwords
EntryPoint ..... (your code)
              CNOP 0,4 Align to lword boundary
              DC.L 0 End marker
              DC.L 1 Define Global 1
              DC.L 1 EntryPoint-StartModule Offset of entry point
              DC.L 1 Highest global used
              END

```

In assembler, you will be started with register D1 holding a BCPL pointer to the initial packet passed from the kernel.

If you are writing in BCPL, a skeleton routine will appear as follows. The main job of the device handler is to convert Open, Read, Write, and Close requests into the device read and write requests. Other packet types are marked as an error.

// "Include files containing useful constants"

```

GET "LIBHDR"
GET "IOHDR"
GET "MANHR"
GET "EXECER"

```

// This is a handler for a skeleton Task.

// When the task is created, the parameter packet contains

// the following.

```

// param.pkt.ipkt.arg1 = BPTR to BCPL string of device name, i.e. ("SKEL:")
// param.pkt.ipkt.arg2 = extra info (if needed)
// param.pkt.ipkt.arg3 = BPTR to device info node

```

MANIFEST

```

$(
  IO.blocksizes = 30 // size of devices IO block
)

```

LET start(param.pkt) BE

```

$(
  LET extrainfo = param.pkt.ipkt.arg2
  LET read.pkt = 0
  LET write.pkt = 0

```

LET openstring = param.pkt.ipkt.arg1

```

LET inpkt = VEC pkt.res1
LET outpkt = VEC pkt.res1
LET IOB = VEC IO.blocksizes
LET IOBO = VEC IO.blocksizes

```

```

LET error = FALSE
LET devname = "serial.device*X00"

```

```

LET open = FALSE // flag to show whether device has been "opened"
                // with act.findinput or act.findoutput

```

```

LET node = param.pkt.ipkt.arg3

```

```

// Zero the block first so that we can see what goes
// into it when we call OpenDevice

```

```

FOR I=0 TO IO.blocksizes DO IOB[I] := 0

```

```

IF OpenDevice( IOB, devname, 0, 0 ) = 0 THEN error := TRUE

```

```

IF error THEN

```

```

$( returnpkt(param.pkt.FALSE,error,objectinuse)
  return
)

```

```

// Copy all the necessary info to the Output buffer too.

```

```

FOR I=0 TO IO.blocksizes DO IOBO[I] := IOB[I]

```

```

outpkt.ipkt.type := act.write

```

```

inpkt.ipkt.type := act.read

```

```

node|dev.task := taskid() // Insert process id into device node

```

```

// Finished with parameter packet...send back...

```

```

returnpkt( param.pkt, TRUE )

```

```

// This is the main repeat loop waiting for an event

```

```

$( LET p = taskwait()

```

```

  SWITCHON p.pkt.type INTO

```

```

  $(

```

```

    CASE act.findinput: // Open

```

```

    CASE act.findoutput:

```

```

    $( LET scb = p.pkt.arg1

```

```

      open := TRUE

```

```

      scb|scb.id := TRUE // Interactive

```

```

      returnpkt(p,TRUE)

```

```

    LOOP

```

```

  )
)

```

```

CASE act.end:

```

```

  node|dev.task := 0 // Close

```

```

  open := FALSE // Remove process id from device node

```

```

  returnpkt(p,TRUE)

```

```

  LOOP

```

```

CASE act.read:

```

```

  ipkt := p

```

```

  handle.return(IOBO,read.pkt)

```

```

  LOOP

```

the DevInfo structure. You must then send a message to the new process to get it started. This message might contain such things as the unit number of the device involved. The handler process should then wait for Open, Read, Write, and Close calls and handle them as described in the example above. C code does not need to insert the process id into the device node because this is done when code is loaded, as described above.

```

CASE act.writes:           // Write request returning
  outpkt := p
  handle.return(IOB0,write,pkt)
  LOOP

CASE 'r':                 // A read request
  read.pkt := p
  handle.request(IOB,IOC.read,p,inpkt)
  inpkt := 0
  LOOP

CASE 'w':                 // A write request
  write.pkt := p
  handle.request(IOB0,IOC.write,p,outpkt)
  outpkt := 0
  LOOP

DEFAULT:
UNLESS open DO node!dev.task := 0 //Remove process id unless open
$) REPEATWHILE open | outpkt = 0 | inpkt = 0
// Termination
CloseDevice( IOB )

$) // Handle an IO request. Passed command, transmission packet (tp)
// and request packet (rp). rp contains buffer and length in arg2/3.
AND handle.request(IOB, command rp, tp ) BE
$( LET buff = rp!pkt.arg2
LET len = rp!pkt.arg3
SetIO( IOB, command, ?, rp!pkt.arg3, 0 )
putlong ( IOB, IO.data, buff )
SendIO(IOB, tp )

$) // Handle a returning IO request. The user request packet is
// passed as p, and must be returned with success/failure message.
AND handle.return(IOB, p ) BE
$( LET errcode = IOB O.error
LET len = getlong( IOB, IO.actual )
TEST errcode = 0 THEN // No error
returnpkt(p, len)
ELSE
returnpkt(p, -1, errcode )

$)

```

If you wish to write your device handler in C, you cannot use the automatic load and process creation provided by the kernel. In this case, you must load the code yourself and use a call to CreateProc to create a process. The result from this call should be stored in the Task field of

Using AmigaDOS Without Workbench/Intuition

This information is provided to give developers some information about how AmigaDOS and Intuition interact with each other. As of this writing, it is not possible to fully close down Intuition or the input device. It is possible to install one's own input handler within the input stream (as is demonstrated in the Amiga ROM Kernel Manual, Input Device description) and thereby handle input events yourself, after your program has been loaded and started by AmigaDOS. If, after that point, you take over the machine in some manner, you can prevent AmigaDOS from trying to put up system requesters or otherwise interacting with the screen by modifying DOS as shown below. Basically, your own program must provide alternate ways to handle errors that would normally cause DOS to put up a requester.

Another alternative for taking over the machine is to ignore the AmigaDOS filing system altogether, and use the trackdisk.device to boot your code and data on your own. You will find details about the disk boot block and the track formatting in the Amiga ROM Kernel Manual, allowing this alternate means if you so choose.

Here are the details about AmigaDOS and Intuition:

AmigaDOS initializes itself and opens Intuition. It then attempts to open the configuration file (created by Preferences) and passes this to Intuition. It then opens the initial CLI window via Intuition and attempts to run the first CLI command. This is commonly a loadwb (load Workbench), followed by an endcli on the initial CLI.

An application program can be made to behave like Workbench, in that it spawns off a new process. The next CLI command is then endcli, which closes everything down, leaving only the new process running (along with the filesystem processes). This process would set the pr_WindowPtr field to -1, which indicates that the DOS should report errors quietly. Note that the application MUST handle all errors. There are further details on this in the AmigaDOS Technical manual, chapter 3. DOS will also have initialized the TrapHandler field of the user task to point to code that will display a requester after an error; this should be replaced by a user-provided routine. This will stop all uses of Intuition from the user task provided there are no serious memory corruption problems found, in which case DOS will call Exec Alert directly.

There is still the problem that the filesystem processes may ask for a requester, in the event of a disk error or if the filesystem task crashes due to memory corruption. To stop this, the pr_WindowPtr and tc_TrapHandler fields of the filesystem tasks must be set to -1 and a private Trap handler must be provided in the same way as was done for the user task. This is easily done as shown below.

Find the message port for each filesystem task by calling DeviceProc(), passing DF0, DF1, etc. An error indicates that the device is not present. From the message port you can find the task base for each filesystem task, and hence patch these two slots. This should be repeated for each disk unit.

The application program can now close Intuition. Workbench has, of

course, never been invoked. Note that as of this writing, it is not possible to stop DOS from opening Intuition.

Note that if the application want to use any other device such as SER, the handler process must be patched in exactly the same way as the filesystem processes. The application should obviously not attempt to open the CON: or RAW: once Intuition has become inactive.

Appendix H

IFF Interchange File Format

This appendix contains a document developed jointly by Commodore-Amiga and Electronic Arts.

The document contained here includes source code for routines that will both read and write this data format.

Commodore-Amiga has adopted this data interchange file format for our internal use and we recommend that our developers adopt it as well.

INFORMATION ABOUT COMPILING THESE ROUTINES:

Electronic arts apparently keeps a bunch of includes in a file called "graphics/system.h".

This is the contents of that include-file:

```
#include exec/types.h
#include exec/exec.h
#include graphics/gfx.h
#include graphics/gfxbase.h
#include graphics/copper.h
#include graphics/gels.h
#include graphics/regions.h
#include hardware/blit.h
#include devices/keymap.h
#include intuition/intuition.h
#include intuition/intuitionbase.h
#include libraries/dos.h
```

OFFSET_BEGINNING in iffr.c and iffw.c must be changed to OFFSET_BEGINING. It is spelled differently in libraries/dos.h.

In Ilbmr.c (p.3) there is a call to UnPackRow:

```
UnPackRow(&buf, *pDest, nFilled, srcRowBytes)
```

nFilled and srcRowBytes are defined as int and LONG.
The function expects WORD and WORD. To compile with
native Lattice required:

```
UnPackRow(&buf, *pDest, (WORD)nFilled, (WORD)srcRowBytes)
```

"EA IFF 85" Standard for Interchange Format Files

Document Date: November 15, 1985
From: Jerry Morrison, Electronic Arts
Status of Standard: Released and in use

1. Introduction

Standards are Good for Software Developers

As home computer hardware evolves to better and better media machines, the demand increases for higher quality, more detailed data. Data development gets more expensive, requires more expertise and better tools, and has to be shared across projects. Think about several ports of a product on one CD-ROM with 500M Bytes of common data!

Development tools need standard interchange file formats. Imagine scanning in images of "player" shapes, moving them to a paint program for editing, then incorporating them into a game. Or writing a theme song with a Macintosh score editor and incorporating it into an Amiga game. The data must at times be transformed, clipped, filled out, and moved across machine kinds. Media projects will depend on data transfer from graphic, music, sound effect, animation, and script tools.

Standards are Good for Software Users

Customers should be able to move their own data between independently developed software products. And they should be able to buy data libraries usable across many such products. The types of data objects to exchange are open-ended and include plain and formatted text, raster and structured graphics, fonts, music, sound effects, musical instrument descriptions, and animation.

The problem with expedient file formats—typically memory dumps—is that they're too provincial. By designing data for one particular use (e.g. a screen snapshot), they preclude future expansion (would you like a full page picture? a multi-page document?). In neglecting the possibility that other programs might read their data, they fail to save contextual information (how many bit planes? what resolution?). Ignoring that other programs might create such files, they're intolerant of extra data (texture palette for a picture editor), missing data (no color map), or minor variations (smaller image). In practice, a filed representation should rarely mirror an in-memory representation. The former should be designed for longevity; the latter to optimize the manipulations of a particular program. The same filed data will be read into different memory formats by different programs.

The IFF philosophy: "A little behind-the-scenes conversion when programs read and write files is far better than NxM explicit conversion utilities for highly specialized formats."

So we need some standardization for data interchange among development tools and products. The more developers that adopt a standard, the better for all of us and our customers.

Here is "EA IFF 1985"

Here is our offering: Electronic Arts' IFF standard for Interchange File Format. The full name is "EA IFF 1985". Alternatives and justifications are included for certain choices. Public domain subroutine packages and utility programs are available to make it easy to write and use IFF-compatible programs.

Part 1 introduces the standard. Part 2 presents its requirements and background. Parts 3, 4, and 5 define the primitive data types, FORMs, and LISTs, respectively, and how to define new high level types. Part 6 specifies the top level file structure. Appendix A is included for quick reference and Appendix B names the committee responsible for this standard.

References

American National Standard Additional Control Codes for Use with ASCII, ANSI standard 3.64-1979 for an 8-bit character set. See also ISO standard 2022 and ISO/DIS standard 6429.2.

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

Compiler Construction, An Advanced Course, edited by F. L. Bauer and J. Eickel (Springer-Verlag, 1976). This book is one of many sources for information on recursive descent parsing.

DIF Technical Specification © 1981 by Software Arts, Inc. DIF™ is the format for spreadsheet data interchange developed by Software Arts, Inc.
DIF™ is a trademark of Software Arts, Inc.

Electronic Arts™ is a trademark of Electronic Arts.

"FTXT" IFF Formatted Text, from Electronic Arts. IFF supplement document for a text format.

Inside Macintosh © 1982, 1983, 1984, 1985 Apple Computer, Inc., a programmer's reference manual.
Apple® is a trademark of Apple Computer, Inc.
Macintosh™ is a trademark licensed to Apple Computer, Inc.

"ILBM" IFF Interleaved Bitmap, from Electronic Arts. IFF supplement document for a raster image format.

M68000 16/32-Bit Microprocessor Programmer's Reference Manual © 1984, 1982, 1980, 1979 by Motorola, Inc.

PostScript Language Manual © 1984 Adobe Systems Incorporated.
PostScript™ is a trademark of Adobe Systems, Inc.
Times and Helvetica® are trademarks of Allied Corporation.

InterScript: A Proposal for a Standard for the Interchange of Editable Documents © 1984 Xerox Corporation.

Introduction to InterScript © 1985 Xerox Corporation.

2. Background for Designers

Part 2 is about the background, requirements, and goals for the standard. It's geared for people who want to design new types of IFF objects. People just interested in using the standard may wish to skip this part.

What Do We Need?

A standard should be long on prescription and short on overhead. It should give lots of rules for designing programs and data files for synergy. But neither the programs nor the files should cost too much more than the expedient variety. While we're looking to a future with CD-ROMs and perpendicular recording, the standard must work well on floppy disks.

For program portability, simplicity, and efficiency, formats should be designed with more than one implementation style in mind. (In practice, pure stream I/O is adequate although random access makes it easier to write files.) It ought to be possible to read one of many objects in a file without scanning all the preceding data. Some programs need to read and play out their data in real time, so we need good compromises between generality and efficiency.

As much as we need standards, they can't hold up product schedules. So we also need a kind of decentralized extensibility where any software developer can define and refine new object types without some "standards authority" in the loop. Developers must be able to extend existing formats in a forward- and backward-compatible way. A central repository for design information and example programs can help us take full advantage of the standard.

For convenience, data formats should heed the restrictions of various processors and environments. E.g. word-alignment greatly helps 68000 access at insignificant cost to 8088 programs.

Other goals include the ability to share common elements over a list of objects and the ability to construct composite objects containing other data objects with structural information like directories.

"Simple things should be simple and complex things should be possible."

Think Ahead

Let's think ahead and build programs that read and write files for each other and for programs yet to be designed. Build data formats to last for future computers so long as the overhead is acceptable. This extends the usefulness and life of today's programs and data.

To maximize interconnectivity, the standard file structure and the specific object formats must all be general and extensible. Think ahead when designing an object. It should serve many purposes and allow many programs to store and read back all the information they need; even squeeze in custom data. Then a programmer can store the available data and is encouraged to include fixed contextual details. Recipient programs can read the needed parts, skip unrecognized stuff, default missing data, and use the stored context to help transform the data as needed.

Scope

IFF addresses these needs by defining a standard file structure, some initial data object types, ways to define new types, and rules for accessing these files. We can accomplish a great deal by writing programs according to this standard, but don't expect direct compatibility with existing software. We'll need conversion programs to bridge the gap from the old world.

IFF is geared for computers that readily process information in 8-bit bytes. It assumes a "physical layer" of data storage and transmission that reliably maintains "files" as strings of 8-bit bytes. The standard treats a "file" as a container of data bytes and is independent of how to find a file and whether it has a byte count.

This standard does not by itself implement a clipboard for cutting and pasting data between programs. A clipboard needs software to mediate access, to maintain a "contents version number" so programs can detect updates, and to manage the data in "virtual memory".

Previous Work

Where our needs are similar, we borrow from existing standards.

Our basic need to move data between independently developed programs is similar to that addressed by the Apple Macintosh desk scrap or "clipboard" [Inside Macintosh chapter "Scrap Manager"]. The Scrap Manager works closely with the Resource Manager, a handy filer and swapper for data objects (text strings, dialog window templates, pictures, fonts...) including types yet to be designed [Inside Macintosh chapter "Resource Manager"]. The Resource Manager is a kin to Smalltalk's object swapper.

We will probably write a Macintosh desk accessory that converts IFF files to and from the Macintosh clipboard for quick and easy interchange with programs like MacPaint and Resource Mover.

Macintosh uses a simple and elegant scheme of 4-character "identifiers" to identify resource types, clipboard format types, file types, and file creator programs. Alternatives are unique ID numbers assigned by a central authority or by hierarchical authorities, unique ID numbers generated by algorithm, other fixed length character strings, and variable length strings. Character string identifiers double as readable signposts in data files and programs. The choice of 4 characters is a good tradeoff between storage space, fetch/compare/store time, and name space size. We'll honor Apple's designers by adopting this scheme.

"PICT" is a good example of a standard structured graphics format (including raster images) and its many uses [Inside Macintosh chapter "QuickDraw"]. Macintosh provides QuickDraw routines in ROM to create, manipulate, and display PICTs. Any application can create a PICT by simply asking QuickDraw to record a sequence of drawing commands. Since it's just as easy to ask QuickDraw to render a PICT to a screen or a printer, it's very effective to pass them between programs, say from an illustrator to a word processor. An important feature is the ability to store "comments" in a PICT which QuickDraw will ignore. Actually, it passes them to your optional custom "comment handler".

PostScript, Adobe's print file standard, is a more general way to represent any print image (which is a specification for putting marks on paper) [PostScript Language Manual]. In fact, PostScript is a full-fledged programming language. To interpret a PostScript program is to render a document on a raster output device. The language is defined in layers: a lexical layer of identifiers, constants, and operators; a layer of reverse polish semantics including scope rules and a way to define new subroutines; and a printing-specific layer of built-in identifiers and operators for rendering graphic images. It is clearly a powerful (Turing equivalent) image definition language. PICT and a subset of PostScript are candidates for structured graphics standards.

A PostScript document can be printed on any raster output device (including a display) but cannot generally be edited. That's because the original flexibility and constraints have been discarded. Besides, a PostScript program may use arbitrary computation to supply parameters like placement and size to each operator. A QuickDraw PICT, in comparison, is a more restricted format of graphic primitives parameterized by constants. So a PICT can be edited at the level of the primitives, e.g. move or thicken a line. It cannot be edited at the higher level of, say, the bar chart data which generated the picture.

PostScript has another limitation: Not all kinds of data amount to marks on paper. A musical instrument description is one example. PostScript is just not geared for such uses.

"DIF" is another example of data being stored in a general format usable by future programs [DIF Technical Specification]. DIF is a format for spreadsheet data interchange. DIF and PostScript are both expressed in plain ASCII text files. This is very handy for printing, debugging, experimenting, and transmitting across modems. It can have substantial cost in compaction and read/write work, depending on use. We won't store IFF files this way but we could define an ASCII alternate representation with a converter program.

InterScript is Xerox' standard for interchange of editable documents [Introduction to InterScript]. It approaches a harder problem: How to represent editable word processor documents that may contain formatted text, pictures, cross-references like figure numbers, and even highly specialized objects like mathematical equations? InterScript aims to define one standard representation for each kind of information. Each InterScript-compatible editor is supposed to preserve the objects it doesn't understand and even maintain nested cross-references. So a simple word processor would let you edit the text of a fancy document without discarding the equations or disrupting the equation numbers.

Our task is similarly to store high level information and preserve as much content as practical while moving it between programs. But we need to span a larger universe of data types and cannot expect to centrally define them all. Fortunately, we don't need to make programs preserve information that they don't understand. And for better or worse, we don't have to tackle cross-references yet.

3. Primitive Data Types

Atomic components such as integers and characters that are interpretable directly by the CPU are specified in a format most convenient for the Motorola MC68000 processor [M68000 16/32-Bit Microprocessor Programmer's Reference Manual].

N.B.: Part 3 dictates the format for "primitive" data types where—and only where—used in the overall file structure and standard kinds of chunks (Cf. Chunks). The number of such occurrences will be small enough that the costs of conversion, storage, and management of processor-specific files would far exceed the costs of conversion on-the-fly by "foreign" programs. A particular data chunk may be specified with a different format for its internal primitive types or with processor- or environment-specific variants if necessary to optimize local usage. Since that hurts data interchange, it's not recommended. (Cf. Designing New Data Sections, in Part 4.)

Alignment

All data objects larger than a byte are aligned on even byte addresses relative to the start of the file. This may require padding. Pad bytes are to be written as zeros, but don't count on that when reading.

This means that every odd-length "chunk" (see below) must be padded so that the next one will fall on an even boundary. Also, designers of structures to be stored in chunks should include pad fields where needed to align every field larger than a byte. Zeros should be stored in all the pad bytes.

Justification: Even-alignment causes a little extra work for files that are used only on certain processors but allows 68000 programs to construct and scan the data in memory and do block I/O. You just add an occasional pad field to data structures that you're going to block read/write or else stream read/write an extra byte. And the same source code works on all processors. Unspecified alignment, on the other hand, would force 68000 programs to (dis)assemble word and long-word data one byte at a time. Pretty cumbersome in a high level language. And if you don't conditionally compile that out for other processors, you won't gain anything.

Numbers

Numeric types supported are two's complement binary integers in the format used by the MC68000 processor—high byte first, high word first—the reverse of 8088 and 6502 format. They could potentially include signed and unsigned 8, 16, and 32 bit integers but the standard only uses the following (in C):

```
typedef unsigned char UBYTE;      /* 8 bits unsigned */
typedef short WORD;              /* 16 bits signed */
typedef unsigned short UWORD;    /* 16 bits unsigned */
typedef long LONG;              /* 32 bits signed */
```

Characters

The following character set is assumed wherever characters are used, e.g. in text strings, IDs, and TEXT chunks (see below).

Characters are encoded in 8-bit ASCII. Characters in the range NUL (hex 0) through DEL (hex 7F) are well defined by the 7-bit ASCII standard. IFF uses the graphic group " " (SP, hex 20) through "~" (hex 7E).

Most of the control character group hex 01 through hex 1F have no standard meaning in IFF. The control character LF (hex 0A) is defined as a "newline" character. It denotes an intentional line break, that is, a

paragraph or line terminator. (There is no way to store an automatic line break. That is strictly a function of the margins in the environment the text is placed.) The control character ESC (hex 1B) is a reserved escape character under the rules of ANSI standard 3.64-1979 American National Standard Additional Control Codes for Use with ASCII, ISO standard 2022, and ISO/DIS standard 6429.2.

Characters in the range hex 7F through hex FF are not globally defined in IFF. They are best left reserved for future standardization. But note that the FORM type FTXT (formatted text) defines the meaning of these characters within FTXT forms. In particular, character values hex 7F through hex 9F are control codes while characters hex A0 through hex FF are extended graphic characters like Å, as per the ISO and ANSI standards cited above. [See the supplementary document "FTXT" IFF Formatted Text.]

Dates

A "creation date" is defined as the date and time a stream of data bytes was created. (Some systems call this a "last modified date".) Editing some data changes its creation date. Moving the data between volumes or machines does not.

The IFF standard date format will be one of those used in MS-DOS, Macintosh, or Amiga DOS (probably a 32-bit unsigned number of seconds since a reference point). Issue: Investigate these three.

Type IDs

A "type ID", "property name", "FORM type", or any other IFF identifier is a 32-bit value: the concatenation of four ASCII characters in the range " " (SP, hex 20) through "~" (hex 7E). Spaces (hex 20) should not precede printing characters; trailing spaces are ok. Control characters are forbidden.

```
typedef CHAR ID[4];
```

IDs are compared using a simple 32-bit case-dependent equality test.

Data section type IDs (aka FORM types) are restricted IDs. (Cf. Data Sections.) Since they may be stored in filename extensions (Cf. Single Purpose Files) lower case letters and punctuation marks are forbidden. Trailing spaces are ok.

Carefully choose those four characters when you pick a new ID. Make them mnemonic so programmers can look at an interchange format file and figure out what kind of data it contains. The name space makes it possible for developers scattered around the globe to generate ID values with minimal collisions so long as they choose specific names like "MUS4" instead of general ones like "TYPE" and "FILE". EA will "register" new FORM type IDs and format descriptions as they're devised, but collisions will be improbable so there will be no pressure on this "clearinghouse" process. Appendix A has a list of currently defined IDs.

Sometimes it's necessary to make data format changes that aren't backward compatible. Since IDs are used to denote data formats in IFF, new IDs are chosen to denote revised formats. Since programs won't read chunks whose IDs they don't recognize (see Chunks, below), the new IDs keep old programs from stumbling over new data. The conventional way to choose a "revision" ID is to increment the last character if it's a digit or else change the last character to a digit. E.g. first and second revisions of the ID "XY" would be "XY1" and "XY2". Revisions of "CMAP" would be "CMA1" and "CMA2".

Chunks

Chunks are the building blocks in the IFF structure. The form (in C) is

```
typedef struct {
    ID      ckID;
    LONG    ckSize;          /* sizeof(ckData) */
    UBYTE   ckData[/* ckSize */];
} Chunk;
```

The fixed header part means "Here's another ckID type chunk, ckSize bytes long."

The ckID identifies the format and purpose of the chunk. As a rule, a program must recognize ckID to interpret ckData. It should skip over all unrecognized chunks. The ckID also serves as a format version number as long as we pick new IDs to identify new formats of ckData (see above).

The following ckIDs are universally reserved to identify chunks with particular IFF meanings: "LIST", "FORM", "PROP", "CAT", and " ". The special ID " " (4 spaces) is a ckID for "filler" chunks, that is, chunks that fill space but have no meaningful contents. The IDs "LIS1" through "LIS9", "FOR1" through "FOR9", and "CAT1" through "CAT9" are reserved for future "version number" variations. All IFF-compatible software must account for these 23 chunk IDs. Appendix A has a list of predefined IDs.

The ckSize is a logical block size—how many data bytes are in ckData. If ckData is an odd number of bytes long, a 0 pad byte follows which is not included in ckSize. (Cf. Alignment.) A chunk's total physical size is ckSize rounded up to an even number plus the size of the header. So the smallest chunk is 8 bytes long with ckSize = 0. For the sake of following chunks, programs must respect every chunk's ckSize as a virtual end-of-file for reading its ckData even if that data is malformed, e.g. if nested contents are truncated.

We can describe the syntax of a chunk as a regular expression with "#" representing the ckSize, i.e. the length of the following (braced) bytes. The "[0]" represents a sometimes needed pad byte. (The regular expressions in this document are collected in Appendix A along with an explanation of notation.)

```
Chunk      ::= ID #{ UBYTE* } [0]
```

One chunk output technique is to stream write a chunk header, stream write the chunk contents, then random access back to the header to fill in the size. Another technique is to make a preliminary pass over the data to compute the size, then write it out all at once.

Strings, String Chunks, and String Properties

In a string of ASCII text, LF denotes a forced line break (paragraph or line terminator). Other control characters are not used. (Cf. Characters.)

The ckID for a chunk that contains a string of plain, unformatted text is "TEXT". As a practical matter, a text string should probably not be longer than 32767 bytes. The standard allows up to $2^{31} - 1$ bytes.

When used as a data property (see below), a text string chunk may be 0 to 255 characters long. Such a string is readily converted to a C string or a Pascal STRING[255]. The ckID of a property must be the property name, not "TEXT".

When used as a part of a chunk or data property, restricted C string format is normally used. That means 0 to 255 characters followed by a NUL byte (ASCII value 0).

Data Properties

Data properties specify attributes for following (non-property) chunks. A data property essentially says

"Identifier = value", for example "XY = (10, 200)", telling something about following chunks. Properties may only appear inside data sections ("FORM" chunks, cf. Data Sections) and property sections ("PROP" chunks, cf. Group PROP).

The form of a data property is a special case of Chunk. The `ckID` is a property name as well as a property type. The `ckSize` should be small since data properties are intended to be accumulated in RAM when reading a file. (256 bytes is a reasonable upper bound.) Syntactically:

```
Property ::= Chunk
```

When designing a data object, use properties to describe context information like the size of an image, even if they don't vary in your program. Other programs will need this information.

Think of property settings as assignments to variables in a programming language. Multiple assignments are redundant and local assignments temporarily override global assignments. The order of assignments doesn't matter as long as they precede the affected chunks. (Cf. LISTs, CATs, and Shared Properties.)

Each object type (FORM type) is a local name space for property IDs. Think of a "CMAP" property in a "FORM ILBM" as the qualified ID "ILBM.CMAP". Property IDs specified when an object type is designed (and therefore known to all clients) are called "standard" while specialized ones added later are "nonstandard".

Links

Issue: A standard mechanism for "links" or "cross references" is very desirable for things like combining images and sounds into animations. Perhaps we'll define "link" chunks within FORMs that refer to other FORMs or to specific chunks within the same and other FORMs. This needs further work. EA IFF 1985 has no standard link mechanism.

For now, it may suffice to read a list of, say, musical instruments, and then just refer to them within a musical score by index number.

File References

Issue: We may need a standard form for references to other files. A "file ref" would name a directory and a file in the same type of operating system as the ref's originator. Following the reference would expect the file to be on some mounted volume. In a network environment, a file ref could name a server, too.

Issue: What about a means to reference a portion of another file? Would this be a "file ref" plus a reference to a "link" within the target file?

4. Data Sections

The first thing we need of a file is to tell if it contains IFF data or not and, if so, does it contain the kind of data we're looking for? So we come to the notion of a "data section".

A "data section" or "data object" or IFF "FORM" is the main point of an IFF file. It is one high level object such as a picture or sound effect; self-contained and self-identifying. It could be a composite object like a musical score with nested musical instrument descriptions.

Group FORM

A data section is a chunk with ckID "FORM" and this arrangement:

```
FORM          ::= "FORM" #{ FormType (LocalChunk | FORM | LIST | CAT)* }
FormType     ::= ID
LocalChunk   ::= Property | Chunk
```

The ID "FORM" is a syntactic keyword like "struct" in C. Think of a "struct ILBM" containing a field "CMAP". If you see "FORM" you'll know to expect a FORM type ID (the structure name, "ILBM" in this example) and a particular contents arrangement or "syntax" (local chunks, FORMs, LISTs, and CATs). (LISTs and CATs are discussed in part 5, below.) A "FORM ILBM", in particular, might contain a local chunk "CMAP", an "ILBM.CMAP" (to use a qualified name).

So the chunk ID "FORM" indicates a data section. It implies that the chunk contains an ID and some number of nested chunks. In reading a FORM, like any other chunk, programs must respect its ckSize as a virtual end-of-file for reading its contents, even if they're truncated.

The FormType (or FORM type) is a restricted ID that may not contain lower case letters or punctuation characters. (Cf. Type IDs. Cf. Single Purpose Files.)

The type-specific information in a FORM is composed of its "local chunks": data properties and other chunks. Each FORM type is a local name space for local chunk IDs. So "CMAP" local chunks in other FORM types may be unrelated to "ILBM.CMAP". More than that, each FORM type defines semantic scope. If you know what a FORM ILBM is, you'll know what an ILBM.CMAP is.

Local chunks defined when the FORM type is designed (and therefore known to all clients of this type) are called "standard" while specialized ones added later are "nonstandard".

Among the local chunks, property chunks give settings for various details like text font while the other chunks supply the essential information. This distinction is not clear cut. A property setting cancelled by a later setting of the same property has effect only on data chunks in between. E.g. in the sequence:

```
prop1 = x (propN = value)* prop1 = y
```

where the propNs are not prop1, the setting prop1 = x has no effect.

The following universal chunk IDs are reserved inside any FORM: "LIST", "FORM", "PROP", "CAT", " ", "LIS1" through "LIS9", "FOR1" through "FOR9", and "CAT1" through "CAT9". (Cf. Chunks. Cf. Group LIST. Cf. Group PROP.) For clarity, these universal chunk names may not be FORM type IDs, either.

Part 5, below, talks about grouping FORMs into LISTs and CATs. They let you group a bunch of FORMs but don't impose any particular meaning or constraints on the grouping. Read on.

Composite FORMs

A FORM chunk inside a FORM is a full-fledged data section. This means you can build a composite object like a multi-frame animation sequence from available picture FORMs and sound effect FORMs. You can insert additional chunks with information like frame rate and frame count.

Using composite FORMs, you leverage on existing programs that create and edit the component FORMs. Those editors may even look into your composite object to copy out its type of component, although it'll be the rare program that's fancy enough to do that. Such editors are not allowed to replace their component objects within your composite object. That's because the IFF standard lets you specify consistency requirements for the composite FORM such as maintaining a count or a directory of the components. Only programs that are written to uphold the rules of your FORM type should create or modify such FORMs.

Therefore, in designing a program that creates composite objects, you are strongly requested to provide a facility for your users to import and export the nested FORMs. Import and export could move the data through a clipboard or a file.

Here are several existing FORM types and rules for defining new ones.

FTXT

An FTXT data section contains text with character formatting information like fonts and faces. It has no paragraph or document formatting information like margins and page headers. FORM FTXT is well matched to the text representation in Amiga's Intuition environment. See the supplemental document "FTXT" IFF Formatted Text.

ILBM

"ILBM" is an InterLeaved BitMap image with color map; a machine-independent format for raster images. FORM ILBM is the standard image file format for the Commodore-Amiga computer and is useful in other environments, too. See the supplemental document "ILBM" IFF Interleaved Bitmap.

PICS

The data chunk inside a "PICS" data section has ID "PICT" and holds a QuickDraw picture. Issue: Allow more than one PICT in a PICS? See Inside Macintosh chapter "QuickDraw" for details on PICTs and how to create and display them on the Macintosh computer.

The only standard property for PICS is "XY", an optional property that indicates the position of the PICT relative to "the big picture". The contents of an XY is a QuickDraw Point.

Note: PICT may be limited to Macintosh use, in which case there'll be another format for structured graphics in other environments.

Other Macintosh Resource Types

Some other Macintosh resource types could be adopted for use within IFF files; perhaps MWRT, ICN, ICN#, and STR#.

Issue: Consider the candidates and reserve some more IDs.

Designing New Data Sections

Supplemental documents will define additional object types. A supplement needs to specify the object's purpose, its FORM type ID, the IDs and formats of standard local chunks, and rules for generating and interpreting the data. It's a good idea to supply typedefs and an example source program that accesses the new object. See "ILBM" IFF Interleaved Bitmap for a good example.

Anyone can pick a new FORM type ID but should reserve it with Electronic Arts at their earliest convenience. [Issue: EA contact person? Hand this off to another organization?] While decentralized format definitions and extensions are possible in IFF, our preference is to get design consensus by committee, implement a program to read and write it, perhaps tune the format, and then publish the format with example code. Some organization should remain in charge of answering questions and coordinating extensions to the format.

If it becomes necessary to revise the design of some data section, its FORM type ID will serve as a version number (Cf. Type IDs). E.g. a revised "VDEO" data section could be called "VDE1". But try to get by with compatible revisions within the existing FORM type.

In a new FORM type, the rules for primitive data types and word-alignment (Cf. Primitive Data Types) may be overridden for the contents of its local chunks—but not for the chunk structure itself—if your documentation spells out the deviations. If machine-specific type variants are needed, e.g. to store vast numbers of integers in reverse bit order, then outline the conversion algorithm and indicate the variant inside each file, perhaps via different FORM types. Needless to say, variations should be minimized.

In designing a FORM type, encapsulate all the data that other programs will need to interpret your files. E.g. a raster graphics image should specify the image size even if your program always uses 320 x 200 pixels x 3 bitplanes. Receiving programs are then empowered to append or clip the image rectangle, to add or drop bitplanes, etc. This enables a lot more compatibility.

Separate the central data (like musical notes) from more specialized information (like note beams) so simpler programs can extract the central parts during read-in. Leave room for expansion so other programs can squeeze in new kinds of information (like lyrics). And remember to keep the property chunks manageably short—let's say ≤ 256 bytes.

When designing a data object, try to strike a good tradeoff between a super-general format and a highly-specialized one. Fit the details to at least one particular need, for example a raster image might as well store pixels in the current machine's scan order. But add the kind of generality that makes it usable with foreseeable hardware and software. E.g. use a whole byte for each red, green, and blue color value even if this year's computer has only 4-bit video DACs. Think ahead and help other programs so long as the overhead is acceptable. E.g. run compress a raster by scan line rather than as a unit so future programs can swap images by scan line to and from secondary storage.

Try to design a general purpose "least common multiple" format that encompasses the needs of many programs without getting too complicated. Let's coalesce our uses around a few such formats widely separated in the vast design space. Two factors make this flexibility and simplicity practical. First, file storage space is getting very plentiful, so compaction is not a priority. Second, nearly any locally-performed data conversion work during file reading and writing will be cheap compared to the I/O time.

It must be ok to copy a LIST or FORM or CAT intact, e.g. to incorporate it into a composite FORM. So any kind of internal references within a FORM must be relative references. They could be relative to the start of the containing FORM, relative from the referencing chunk, or a sequence number into a collection.

With composite FORMs, you leverage on existing programs that create and edit the components. If you write a program that creates composite objects, please provide a facility for your users to import and export the nested FORMs. The import and export functions may move data through a separate file or a clipboard.

Finally, don't forget to specify all implied rules in detail.

5. LISTS, CATs, and Shared Properties

Data often needs to be grouped together like a list of icons. Sometimes a trick like arranging little images into a big raster works, but generally they'll need to be structured as a first class group. The objects "LIST" and "CAT" are IFF-universal mechanisms for this purpose.

Property settings sometimes need to be shared over a list of similar objects. E.g. a list of icons may share one color map. LIST provides a means called "PROP" to do this. One purpose of a LIST is to define the scope of a PROP. A "CAT", on the other hand, is simply a concatenation of objects.

Simpler programs may skip LISTS and PROPs altogether and just handle FORMs and CATs. All "fully-conforming" IFF programs also know about "CAT", "LIST", and "PROP". Any program that reads a FORM inside a LIST must process shared PROPs to correctly interpret that FORM.

Group CAT

A CAT is just an untyped group of data objects.

Structurally, a CAT is a chunk with chunk ID "CAT" containing a "contents type" ID followed by the nested objects. The `ckSize` of each contained chunk is essentially a relative pointer to the next one.

```
CAT          ::= "CAT" #{ ContentsType (FORM | LIST | CAT)* }
ContentsType ::= ID
```

In reading a CAT, like any other chunk, programs must respect its `ckSize` as a virtual end-of-file for reading the nested objects even if they're malformed or truncated.

The "contents type" following the CAT's `ckSize` indicates what kind of FORMs are inside. So a CAT of ILBM's would store "ILBM" there. It's just a hint. A CAT should have blank contents ID (" ") if it contains more than one kind of FORM.

CAT defines only the format of the group. The group's meaning is open to interpretation. This is like a list in LISP: the structure of cells is predefined but the meaning of the contents as, say, an association list depends on use. If you need a group with an enforced meaning (an "abstract data type" or Smalltalk "subclass"), some consistency constraints, or additional data chunks, use a composite FORM instead (Cf. Composite FORMs).

Since a CAT just means a concatenation of objects, CATs are rarely nested. Programs should really merge CATs rather than nest them.

Group LIST

A LIST defines a group very much like CAT but it also gives a scope for PROPs (see below). And unlike CATs, LISTS should not be merged without understanding their contents.

Structurally, a LIST is a chunk with `ckID` "LIST" containing a "contents type" ID, optional shared properties, and the nested contents (FORMs, LISTS, and CATs), in that order. The `ckSize` of each contained chunk is a relative pointer to the next one. A LIST is not an arbitrary linked list—the cells are simply concatenated.

```
LIST          ::= "LIST" #{ ContentsType PROP* (FORM | LIST | CAT)* }
ContentsType ::= ID
```

Group PROP

PROP chunks may appear in LISTS (not in FORMs or CATs). They supply shared properties for the FORMs in that LIST. This ability to elevate some property settings to shared status for a list of forms is useful for both indirection and compaction. E.g. a list of images with the same size and colors can share one "size" property and one "color map" property. Individual FORMs can override the shared settings.

The contents of a PROP is like a FORM with no data chunks:

```
PROP      ::= "PROP" #{ FormType Property* }
```

It means, "Here are the shared properties for FORM type <FormType>."

A LIST may have at most one PROP of a FORM type, and all the PROPs must appear before any of the FORMs or nested LISTS and CATs. You can have subsequences of FORMs sharing properties by making each subsequence a LIST.

Scoping: Think of property settings as variable bindings in nested blocks of a programming language. Where in C you could write:

```
TEXT_FONT text_font = Courier;          /* program's global default */

File(); {
    TEXT_FONT text_font = TimesRoman;   /* shared setting */

    {
        TEXT_FONT text_font = Helvetica; /* local setting */
        Print("Hello ");                 /* uses font Helvetica */
    }

    {
        Print("there.");                  /* uses font TimesRoman */
    }
}
```

An IFF file could contain:

```
LIST {
    PROP TEXT {
        FONT {TimesRoman}                /* shared setting */
    }

    FORM TEXT {
        FONT {Helvetica}                 /* local setting */
        CHRS {Hello }                    /* uses font Helvetica */
    }

    FORM TEXT {
        CHRS {there.}                     /* uses font TimesRoman */
    }
}
```

The shared property assignments selectively override the reader's global defaults, but only for FORMs within the group. A FORM's own property assignments selectively override the global and group-supplied values. So when reading an IFF file, keep property settings on a stack. They're designed to be small

enough to hold in main memory.

Shared properties are semantically equivalent to copying those properties into each of the nested FORMs right after their FORM type IDs.

Properties for LIST

Optional "properties for LIST" store the origin of the list's contents in a PROP chunk for the fake FORM type "LIST". They are the properties originating program "OPGM", processor family "OCPU", computer type "OCMP", computer serial number or network address "OSN", and user name "UNAM". In our imperfect world, these could be called upon to distinguish between unintended variations of a data format or to work around bugs in particular originating/receiving program pairs. Issue: Specify the format of these properties.

A creation date could also be stored in a property but let's ask that file creating, editing, and transporting programs maintain the correct date in the local file system. Programs that move files between machine types are expected to copy across the creation dates.

6. Standard File Structure

File Structure Overview

An IFF file is just a single chunk of type FORM, LIST, or CAT. Therefore an IFF file can be recognized by its first 4 bytes: "FORM", "LIST", or "CAT". Any file contents after the chunk's end are to be ignored.

Since an IFF file can be a group of objects, programs that read/write single objects can communicate to an extent with programs that read/write groups. You're encouraged to write programs that handle all the objects in a LIST or CAT. A graphics editor, for example, could process a list of pictures as a multiple page document, one page at a time.

Programs should enforce IFF's syntactic rules when reading and writing files. This ensures reliable data transfer. The public domain IFF reader/writer subroutine package does this for you. A utility program "IFFCheck" is available that scans an IFF file and checks it for conformance to IFF's syntactic rules. IFFCheck also prints an outline of the chunks in the file, showing the `ckID` and `ckSize` of each. This is quite handy when building IFF programs. Example programs are also available to show details of reading and writing IFF files.

A merge program "IFFJoin" will be available that logically appends IFF files into a single CAT group. It "unwraps" each input file that is a CAT so that the combined file isn't nested CATs.

If we need to revise the IFF standard, the three anchoring IDs will be used as "version numbers". That's why IDs "FOR1" through "FOR9", "LIS1" through "LIS9", and "CAT1" through "CAT9" are reserved.

IFF formats are designed for reasonable performance with floppy disks. We achieve considerable simplicity in the formats and programs by relying on the host file system rather than defining universal grouping structures like directories for LIST contents. On huge storage systems, IFF files could be leaf nodes in a file structure like a B-tree. Let's hope the host file system implements that for us!

There are two kinds of IFF files: single purpose files and scrap files. They differ in the interpretation of multiple data objects and in the file's external type.

Single Purpose Files

A single purpose IFF file is for normal "document" and "archive" storage. This is in contrast with "scrap files" (see below) and temporary backing storage (non-interchange files).

The external file type (or filename extension, depending on the host file system) indicates the file's contents. It's generally the FORM type of the data contained, hence the restrictions on FORM type IDs.

Programmers and users may pick an "intended use" type as the filename extension to make it easy to filter for the relevant files in a filename requestor. This is actually a "subclass" or "subtype" that conveniently separates files of the same FORM type that have different uses. Programs cannot demand conformity to its expected subtypes without overly restricting data interchange since they cannot know about the subtypes to be used by future programs that users will want to exchange data with.

Issue: How to generate 3-letter MS-DOS extensions from 4-letter FORM type IDs?

Most single purpose files will be a single FORM (perhaps a composite FORM like a musical score containing nested FORMs like musical instrument descriptions). If it's a LIST or a CAT, programs should skip over unrecognized objects to read the recognized ones or the first recognized one. Then a program that can read a single purpose file can read something out of a "scrap file", too.

Scrap Files

A "scrap file" is for maximum interconnectivity in getting data between programs; the core of a clipboard function. Scrap files may have type "IFF " or filename extension ".IFF".

A scrap file is typically a CAT containing alternate representations of the same basic information. Include as many alternatives as you can readily generate. This redundancy improves interconnectivity in situations where we can't make all programs read and write super-general formats. [Inside Macintosh chapter "Scrap Manager".] E.g. a graphically-annotated musical score might be supplemented by a stripped down 4-voice melody and by a text (the lyrics).

The originating program should write the alternate representations in order of "preference": most preferred (most comprehensive) type to least preferred (least comprehensive) type. A receiving program should either use the first appearing type that it understands or search for its own "preferred" type.

A scrap file should have at most one alternative of any type. (A LIST of same type objects is ok as one of the alternatives.) But don't count on this when reading; ignore extra sections of a type. Then a program that reads scrap files can read something out of single purpose files.

Rules for Reader Programs

Here are some notes on building programs that read IFF files. If you use the standard IFF reader module "IFFR.C", many of these rules and details will be automatically handled. (See "Support Software" in Appendix A.) We recommend that you start from the example program "ShowLBM.C". You should also read up on recursive descent parsers. [See, for example, Compiler Construction, An Advanced Course.]

- The standard is very flexible so many programs can exchange data. This implies a program has to scan the file and react to what's actually there in whatever order it appears. An IFF reader program is a parser.
- For interchange to really work, programs must be willing to do some conversion during read-in. If the data isn't exactly what you expect, say, the raster is smaller than those created by your program, then adjust it. Similarly, your program could crop a large picture, add or drop bitplanes, and create/discard a mask plane. The program should give up gracefully on data that it can't convert.
- If it doesn't start with "FORM", "LIST", or "CAT ", it's not an IFF-85 file.
- For any chunk you encounter, you must recognize its type ID to understand its contents.
- For any FORM chunk you encounter, you must recognize its FORM type ID to understand the contained "local chunks". Even if you don't recognize the FORM type, you can still scan it for nested FORMs, LISTs, and CATs of interest.
- Don't forget to skip the pad byte after every odd-length chunk.
- Chunk types LIST, FORM, PROP, and CAT are generic groups. They always contain a subtype ID followed by chunks.
- Readers ought to handle a CAT of FORMs in a file. You may treat the FORMs like document pages to sequence through or just use the first FORM.
- Simpler IFF readers completely skip LISTs. "Fully IFF-conforming" readers are those that handle LISTs, even if just to read the first FORM from a file. If you do look into a LIST, you must process shared

properties (in PROP chunks) properly. The idea is to get the correct data or none at all.

- The nicest readers are willing to look into unrecognized FORMs for nested FORM types that they do recognize. For example, a musical score may contain nested instrument descriptions and an animation file may contain still pictures.

Note to programmers: Processing PROP chunks is not simple! You'll need some background in interpreters with stack frames. If this is foreign to you, build programs that read/write only one FORM per file. For the more intrepid programmers, the next paragraph summarizes how to process LISTs and PROPs. See the general IFF reader module "IFFR.C" and the example program "ShowLBM.C" for details.

Allocate a stack frame for every LIST and FORM you encounter and initialize it by copying the stack frame of the parent LIST or FORM. At the top level, you'll need a stack frame initialized to your program's global defaults. While reading each LIST or FORM, store all encountered properties into the current stack frame. In the example ShowLBM, each stack frame has a place for a bitmap header property ILBM.BMHD and a color map property ILBM.CMAP. When you finally get to the ILBM's BODY chunk, use the property settings accumulated in the current stack frame.

Rules for Writer Programs

Here are some notes on building programs that write IFF files, which is much easier than reading them. If you use the standard IFF writer module "IFFW.C" (see "Support Software" in Appendix A), many of these rules and details will automatically be enforced. See the example program "Raw2ILBM.C".

- An IFF file is a single FORM, LIST, or CAT chunk.
- Any IFF-85 file must start with the 4 characters "FORM", "LIST", or "CAT ", followed by a LONG ckSize. There should be no data after the chunk end.
- Chunk types LIST, FORM, PROP, and CAT are generic. They always contain a subtype ID followed by chunks. These three IDs are universally reserved, as are "LIS1" through "LIS9", "FOR1" through "FOR9", "CAT1" through "CAT9", and " ".
- Don't forget to write a 0 pad byte after each odd-length chunk.
- Four techniques for writing an IFF group: (1) build the data in a file mapped into virtual memory, (2) build the data in memory blocks and use block I/O, (3) stream write the data piecemeal and (don't forget!) random access back to set the group length count, and (4) make a preliminary pass to compute the length count then stream write the data.
- Do not try to edit a file that you don't know how to create. Programs may look into a file and copy out nested FORMs of types that they recognize, but don't edit and replace the nested FORMs and don't add or remove them. That could make the containing structure inconsistent. You may write a new file containing items you copied (or copied and modified) from another IFF file, but don't copy structural parts you don't understand.
- You must adhere to the syntax descriptions in Appendix A. E.g. PROP's may only appear inside LIST's.

Appendix A. Reference

Type Definitions

The following C typedefs describe standard IFF structures. Declarations to use in practice will vary with the compiler. For example, 68000 Lattice C produces efficient comparison code if we define ID as an "unsigned long". A macro "MakeID" builds these IDs at compile time.

```

/* Standard IFF types. */

typedef unsigned char UBYTE;      /* 8 bits unsigned */
typedef short WORD;              /* 16 bits signed */
typedef unsigned short UWORD;    /* 16 bits unsigned */
typedef long LONG;               /* 32 bits signed */

typedef char ID[4];

typedef struct {
    ID ckID;
    LONG ckSize;                  /* sizeof(ckData) */
    UBYTE ckData[/* ckSize */];
} Chunk;

/* ID typedef and builder for 68000 Lattice C. */
typedef LONG ID;
#define MakeID(a,b,c,d) ( (a)<<24 | (b)<<16 | (c)<<8 | (d) )

/* Globally reserved IDs. */
#define FORM MakeID('F','O','R','M')
#define LIST MakeID('L','I','S','T')
#define PROP MakeID('P','R','O','P')
#define CAT MakeID('C','A','T',' ')
#define FILLER MakeID(' ',' ',' ',' ')

```

Syntax Definitions

Here's a collection of the syntax definitions in this document.

```

Chunk      ::= ID #{ UBYTE* } [0]

Property   ::= Chunk

FORM       ::= "FORM" #{ FormType (LocalChunk | FORM | LIST | CAT)* }
FormType   ::= ID
LocalChunk ::= Property | Chunk

CAT        ::= "CAT " #{ ContentsType (FORM | LIST | CAT)* }
ContentsType ::= ID

LIST       ::= "LIST" #{ ContentsType PROP* (FORM | LIST | CAT)* }
PROP       ::= "PROP" #{ FormType Property* }

```

In this extended regular expression notation, the token "#" represents a ckSize LONG count of the

following {braced} data bytes. Literal items are shown in "quotes", [square bracketed items] are optional, and "*" means 0 or more instances. A sometimes-needed pad byte is shown as "[0]".

Defined Chunk IDs

This is a table of currently defined chunk IDs. We'll also borrow some Macintosh IDs and data formats.

Group chunk IDs

FORM, LIST, PROP, CAT.

Future revision group chunk IDs

FOR1 ... FOR9, LIS1 ... LIS9, CAT1 ... CAT9.

FORM type IDs

(The above group chunk IDs may not be used for FORM type IDs.)

(Lower case letters and punctuation marks are forbidden in FORM type IDs.)

PLBM, ILBM, ANBM, FTXT, PICS, USCR, GSCR, UVOX, GVOX, SFX, FNTR, FNTV, VDEO, PDEF.

Data chunk IDs

" ", TEXT, PICT.

PROP LIST property IDs

OPGM, OCPU, OCOMP, OSN, UNAM.

Formats for sampled sound, musical instrument, and musical score are currently being developed.

Support Software

These public domain C source programs are available for use in building IFF-compatible programs:

IFF.H, IFFR.C, IFFW.C

IFF reader and writer package. These modules handle many of the details of reliably reading and writing IFF files.

IFFCheck.C

This handy utility program scans an IFF file, checks that the contents are well formed, and prints an outline of the chunks.

Packer.H, Packer.C, UnPacker.C

Run encoder and decoder used for ILBM files.

ILBM.H, ILBMR.C, ILBMW.C

Reader and writer support routines for raster image FORM ILBM. ILBMR calls IFFR and UnPacker. ILBMW calls IFFW and Packer.

ShowILBM.C

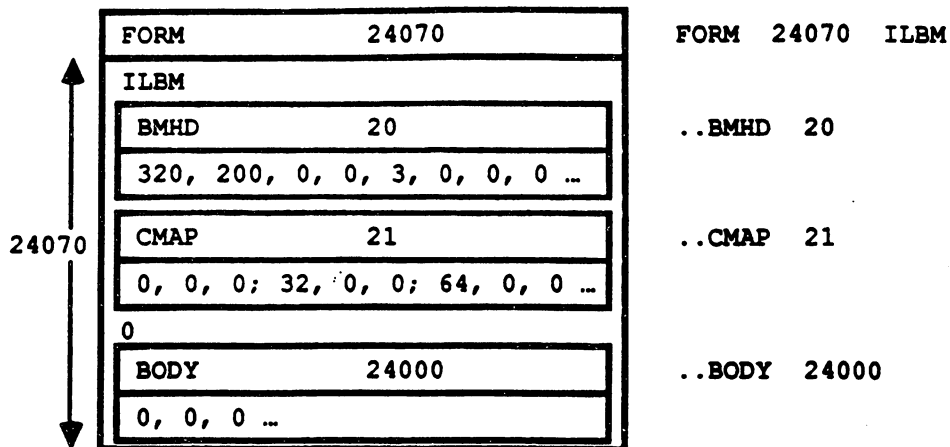
Example caller of IFFR and ILBMR modules. This

Raw2ILBM.C

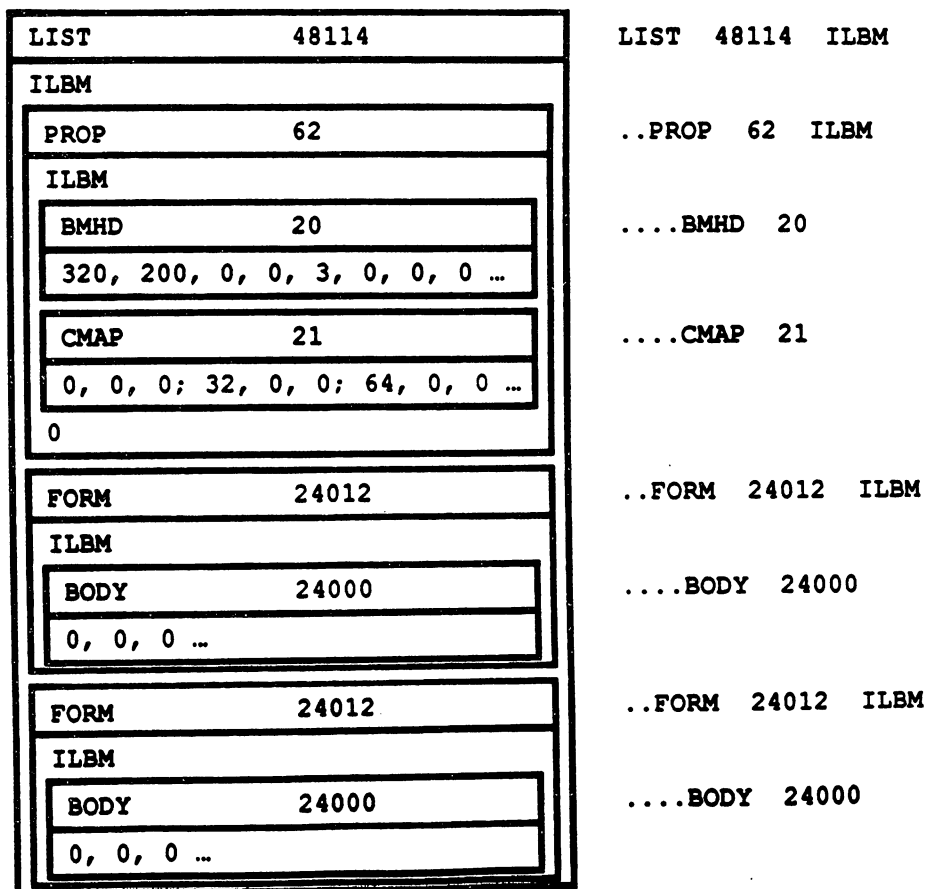
Commodore-Amiga program reads and displays a FORM ILBM. Example ILBM writer program. As a demonstration, it reads a raw raster image file and writes the image as a FORM ILBM file.

Example Diagrams

Here's a box diagram for an example IFF file, a raster image FORM ILBM. This FORM contains a bitmap header property chunk BMHD, a color map property chunk CMAP, and a raster data chunk BODY. This particular raster is 320 x 200 pixels x 3 bit planes uncompressed. The "0" after the CMAP chunk represents a zero pad byte; included since the CMAP chunk has an odd length. The text to the right of the diagram shows the outline that would be printed by the IFFCheck utility program for this particular file.



This second diagram shows a LIST of two FORMs ILBM sharing a common BMHD property and a common CMAP property. Again, the text on the right is an outline as la IFFCheck.



Appendix B. Standards Committee

The following people contributed to the design of this IFF standard:

Bob "Kodiak" Burns, Commodore-Amiga
R. J. Mical, Commodore-Amiga
Jerry Morrison, Electronic Arts
Greg Riker, Electronic Arts
Steve Shaw, Electronic Arts
Barry Walsh, Commodore-Amiga

"ILBM" IFF Interleaved Bitmap

Date: November 15, 1985
From: Jerry Morrison, Electronic Arts
Status: Released and in use

1. Introduction

"EA IFF 85" is Electronic Arts' standard for interchange format files. "ILBM" is the format for a 2 dimensional raster graphics image, specifically an InterLeaved bitplane BitMap image with color map. An ILBM is an IFF "data section" or "FORM type", which can be an IFF file or a part of one. (See the IFF reference.)

An ILBM is an archival representation designed for three uses. First, a standalone image that specifies exactly how to display itself (resolution, size, color map, etc.). Second, an image intended to be merged into a bigger picture which has its own depth, color map, and so on. And third, an empty image with a color map selection or "palette" for a paint program. ILBM is also intended as a building block for composite IFF FORMs like "animation sequence" and "structured graphics". Some uses of ILBM will be to preserve as much information as possible across disparate environments. Other uses will be to store data for a single program or highly cooperative programs while maintaining subtle details. So we're trying to accomplish a lot with this one format.

This memo is the IFF supplement for FORM ILBM. Section 2 defines the purpose and format of property chunks bitmap header "BMHD", color map "CMAP", hotspot "GRAB", destination merge data "DEST", sprite information "SPRT", and Commodore Amiga viewport mode "CAMG". Section 3 defines the standard data chunk "BODY". These are the "standard" chunks. Section 4 defines the nonstandard color range data chunk "CRNG". Additional specialized chunks like texture pattern can be added later. The ILBM syntax is summarized in Appendix A as a regular expression and in Appendix B as a box diagram. Appendix C explains the optional run encoding scheme. Appendix D names the committee responsible for this FORM ILBM standard.

Details of the raster layout are given in part 3, "Standard Data Chunk". Some elements are based on the Commodore Amiga hardware but generalized for use on other computers. An alternative to ILBM would be appropriate for computers with true color data in each pixel.

Reference:

"EA IFF 85" Standard for Interchange Format Files describes the underlying conventions for all IFF files.

Amiga™ is a trademark of Commodore-Amiga, Inc.
Electronic Arts™ is a trademark of Electronic Arts.
Macintosh™ is a trademark licensed to Apple Computer, Inc.
MacPaint™ is a trademark of Apple Computer, Inc.

2. Standard Properties

The required property "BMHD" and any optional properties must appear before any "BODY" chunk. (Since an ILBM has only one BODY chunk, any following properties are superfluous.) Any of these properties may be shared over a LIST of FORMs IBLM by putting them in a PROP ILBM. (See the "EA IFF 85" memo.)

BMHD

The required property "BMHD" holds a BitMapHeader as defined in these C declarations and following documentation. It describes the dimensions and encoding of the image, including data necessary to understand the BODY chunk to follow.

```

typedef UBYTE Masking;          /* Choice of masking technique. */
#define mskNone                0
#define mskHasMask             1
#define mskHasTransparentColor 2
#define mskLasso               3

typedef UBYTE Compression;     /* Choice of compression algorithm applied to
the rows of all source and mask planes. "cmpByteRun1" is the byte run
encoding described in Appendix C. Do not compress across rows! */
#define cmpNone                0
#define cmpByteRun1            1

typedef struct {
    UWORD w, h;                /* raster width & height in pixels */
    WORD x, y;                /* pixel position for this image */
    UBYTE nPlanes;            /* # source bitplanes */
    Masking masking;
    Compression compression;
    UBYTE pad1;                /* unused; for consistency, put 0 here */
    UWORD transparentColor;    /* transparent "color number" (sort of) */
    UBYTE xAspect, yAspect;    /* pixel aspect, a ratio width : height */
    WORD pageWidth, pageHeight; /* source "page" size in pixels */
} BitMapHeader;

```

Fields are filled in the order shown. The UBYTE fields are byte-packed.

The fields *w* and *h* indicate the size of the image rectangle in pixels. Each row of the image is stored in an integral number of 16 bit words. The number of words per row is $\text{Ceiling}(w/16)$. The fields *x* and *y* indicate the desired position of this image within the destination picture. Some reader programs may ignore *x* and *y*. A safe default for writing an ILBM is $(x, y) = (0, 0)$.

The number of source bitplanes in the BODY chunk (see below) is stored in *nPlanes*. An ILBM with a CMAP but no BODY and *nPlanes* = 0 is the recommended way to store a color map.

Note: Color numbers are color map index values formed by pixels in the destination bitmap, which may be deeper than *nPlanes* if a DEST chunk calls for merging the image into a deeper image.

The field *masking* indicates what kind of masking is to be used for this image. The value *mskNone* designates an opaque rectangular image. The value *mskHasMask* means that a mask plane is interleaved with the bitplanes in the BODY chunk (see below). The value *mskHasTransparentColor* indicates that pixels in the source planes matching *transparentColor* are to be considered "transparent". (Actually, *transparentColor* isn't a "color number" since it's matched with numbers formed by the source bitmap

rather than the possibly deeper destination bitmap. Note that having a transparent color implies ignoring one of the color registers. See CMAP, below.) The value `mskLasso` indicates the reader may construct a mask by lassoing the image as in MacPaint™. To do this, put a 1 pixel border of `transparentColor` around the image rectangle. Then do a seed fill from this border. Filled pixels are to be transparent.

Issue: Include in an appendix an algorithm for converting a transparent color to a mask plane, and maybe a lasso algorithm.

A code indicating the kind of data compression used is stored in `compression`. Beware that using data compression makes your data unreadable by programs that don't implement the matching decompression algorithm. So we'll employ as few compression encodings as possible. The run encoding `byteRun1` is documented in Appendix C, below.

The field `pad1` is a pad byte and must be set to 0 for consistency. This field could get used in the future.

The `transparentColor` specifies which bit pattern means "transparent". This only applies if masking is `mskHasTransparentColor` or `mskLasso` (see above). Otherwise, `transparentColor` should be 0.

The pixel aspect ratio is stored as a ratio in the two fields `xAspect` and `yAspect`. This may be used by programs to compensate for different aspects or to help interpret the fields `w`, `h`, `x`, `y`, `pageWidth`, and `pageHeight`, which are in units of pixels. The fraction `xAspect/yAspect` represents a pixel's width/height. It's recommended that your programs store proper fractions in `BitMapHeaders`, but aspect ratios can always be correctly compared with the test

$$xAspect \cdot yDesiredAspect = yAspect \cdot xDesiredAspect$$

Typical values for aspect ratio are width : height = 10 : 11 (Amiga 320 x 200 display) and 1 : 1 (Macintosh™).

The size in pixels of the source "page" (any raster device) is stored in `pageWidth` and `pageHeight`, e.g. (320, 200) for a low resolution Amiga display. This information might be used to scale an image or to automatically set the display format to suit the image. (The image can be larger than the page.)

CMAP

The optional (but encouraged) property "CMAP" stores color map data as triplets of red, green, and blue intensity values. The `n` color map entries ("color registers") are stored in the order 0 through `n-1`, totaling `3n` bytes. Thus `n` is the `ckSize/3`. Normally, `n` would equal `2nPlanes`.

A CMAP chunk contains a `ColorMap` array as defined below. (These typedefs assume a C compiler that implements packed arrays of 3-byte elements.)

```
typedef struct {
    UBYTE red, green, blue;          /* color intensities 0..255 */
} ColorRegister;                  /* size = 3 bytes */

typedef ColorRegister ColorMap[n]; /* size = 3n bytes */
```

The color components red, green, and blue represent fractional intensity values in the range 0 through 255/256ths. White is (255, 255, 255) and black is (0, 0, 0). If your machine has less color resolution, use the high order bits. Shift each field right on reading (or left on writing) and assign it to (from) a field in a local packed format like `Color4`, below. This achieves automatic conversion of images across environments with different color resolutions. On reading an ILBM, use defaults if the color map is absent or has fewer color registers than you need. Ignore any extra color registers.

The example type `Color4` represents the format of a color register in working memory of an Amiga computer, which has 4 bit video DACs. (The `:` tells the C compiler to pack the field into 4 bits.)

```
typedef struct {
    unsigned pad1 :4, red :4, green :4, blue :4;
} Color4; /* Amiga RAM format. Not filed. */
```

Remember that every chunk must be padded to an even length, so a color map with an odd number of entries would be followed by a 0 byte, not included in the `ckSize`.

GRAB

The optional property "GRAB" locates a "handle" or "hotspot" of the image relative to its upper left corner, e.g. when used as a mouse cursor or a "paint brush". A GRAB chunk contains a `Point2D`.

```
typedef struct {
    WORD x, y; /* relative coordinates (pixels) */
} Point2D;
```

DEST

The optional property "DEST" is a way to say how to scatter zero or more source bitplanes into a deeper destination image. Some readers may ignore DEST.

The contents of a DEST chunk is `DestMerge` structure:

```
typedef struct {
    UBYTE depth; /* # bitplanes in the original source */
    UBYTE pad1; /* unused; for consistency put 0 here */
    UWORD planePick; /* how to scatter source bitplanes into destination */
    UWORD planeOnOff; /* default bitplane data for planePick */
    UWORD planeMask; /* selects which bitplanes to store into */
} DestMerge;
```

The low order depth number of bits in `planePick`, `planeOnOff`, and `planeMask` correspond one-to-one with destination bitplanes. Bit 0 with bitplane 0, etc. (Any higher order bits should be ignored.) "1" bits in `planePick` mean "put the next source bitplane into this bitplane", so the number of "1" bits should equal `nPlanes`. "0" bits mean "put the corresponding bit from `planeOnOff` into this bitplane". Bits in `planeMask` gate writing to the destination bitplane: "1" bits mean "write to this bitplane" while "0" bits mean "leave this bitplane alone". The normal case (with no DEST property) is equivalent to `planePick = planeMask = 2nPlanes - 1`.

Remember that color numbers are formed by pixels in the destination bitmap (depth planes deep) not in the source bitmap (`nPlanes` planes deep).

SPRT

The presence of an "SPRT" chunk indicates that this image is intended as a sprite. It's up to the reader program to actually make it a sprite, if even possible, and to use or overrule the sprite precedence data inside the SPRT chunk:

```
typedef UWORD SpritePrecedence; /* relative precedence, 0 is the highest */
```

Precedence 0 is the highest, denoting a sprite that is foremost.

Creating a sprite may imply other setup. E.g. a 2 plane Amiga sprite would have `transparentColor = 0`. Color registers 1, 2, and 3 in the CMAP would be stored into the correct hardware color registers for the hardware sprite number used, while CMAP color register 0 would be ignored.

CAMG

A "CAMG" chunk is specifically for the Commodore Amiga computer. It stores a LONG "viewport mode". This lets you specify Amiga display modes like "dual playfield" and "hold and modify".

3. Standard Data Chunk

Raster Layout

Raster scan proceeds left-to-right (increasing X) across scan lines, then top-to-bottom (increasing Y) down columns of scan lines. The coordinate system is in units of pixels, where (0,0) is the upper left corner.

The raster is typically organized as bitplanes in memory. The corresponding bits from each plane, taken together, make up an index into the color map which gives a color value for that pixel. The first bitplane, plane 0, is the low order bit of these color indexes.

A scan line is made of one "row" from each bitplane. A row is one planes' bits for one scan line, but padded out to a word (2 byte) boundary (not necessarily the first word boundary). Within each row, successive bytes are displayed in order and the most significant bit of each byte is displayed first.

A "mask" is an optional "plane" of data the same size (w, h) as a bitplane. It tells how to "cut out" part of the image when painting it onto another image. "One" bits in the mask mean "copy the corresponding pixel to the destination" while "zero" mask bits mean "leave this destination pixel alone". In other words, "zero" bits designate transparent pixels.

The rows of the different bitplanes and mask are interleaved in the file (see below). This localizes all the information pertinent to each scan line. It makes it much easier to transform the data while reading it to adjust the image size or depth. It also makes it possible to scroll a big image by swapping rows directly from the file without random-accessing to all the bitplanes.

BODY

The source raster is stored in a "BODY" chunk. This one chunk holds all bitplanes and the optional mask, interleaved by row.

The BitMapHeader, in a BMHD property chunk, specifies the raster's dimensions w, h, and nPlanes. It also holds the `masking` field which indicates if there is a mask plane and the `compression` field which indicates the compression algorithm used. This information is needed to interpret the BODY chunk, so the BMHD chunk must appear first. While reading an ILBM's BODY, a program may convert the image to another size by filling (with `transparentColor`) or clipping.

The BODY's content is a concatenation of scan lines. Each scan line is a concatenation of one row of data from each plane in order 0 through `nPlanes-1` followed by one row from the mask (if `masking = hasMask`). If the BitMapHeader field `compression` is `cmpNone`, all h rows are exactly `Ceiling(w/16)` words wide. Otherwise, every row is compressed according to the specified algorithm and their stored widths depend on the data compression.

Reader programs that require fewer bitplanes than appear in a particular ILBM file can combine planes or drop the high-order (later) planes. Similarly, they may add bitplanes and/or discard the mask plane.

Do not compress across rows and don't forget to compress the mask just like the bitplanes. Remember to pad any BODY chunk that contains an odd number of bytes.

4. Nonstandard Data Chunk

The following data chunk was defined after various programs began using FORM ILBM so it's a "nonstandard" chunk. That means there's some slight chance of name collisions.

CRNG

A "CRNG" chunk contains "color register range" information. It's useful for identifying a contiguous range of color registers for color cycling. There can be zero or more CRNG chunks in an ILBM, but all should appear before the BODY chunk.

```
typedef struct {  
    WORD  pad1;           /* reserved for future use; store 0 here      */  
    WORD  rate;           /* color cycle rate                            */  
    WORD  active;         /* nonzero means cycle the colors              */  
    UBYTE low, high;     /* lower and upper color registers selected    */  
} CRange;
```

The fields `low` and `high` indicate the range of color registers (color numbers) selected by this CRange.

The field `active` indicates whether color cycling is on or off. Zero means off.

The field `rate` determines the speed at which the colors will step when color cycling is on. The units are such that a rate of 60 steps per second is represented as $2^{14} = 16384$. Slower rates can be obtained by linear scaling: for 30 steps/second, $rate = 8192$; for 1 step/second, $rate = 16384/60 = 273$.

Appendix A. ILBM Regular Expression

Here's a regular expression summary of the FORM ILBM syntax. This could be an IFF file or a part of one.

```

ILBM ::= "FORM" #{ "ILBM" BMHD [CMAP] [GRAB] [DEST] [SPRT] [CAMG]
                CRNG* [BODY]          }

BMHD  ::= "BMHD"  #{ BitMapHeader          }
CMAP  ::= "CMAP"  #{ (red green blue)*     } [0]
GRAB  ::= "GRAB"  #{ Point2D               }
DEST  ::= "DEST"  #{ DestMerge             }
SPRT  ::= "SPRT"  #{ SpritePrecedence     }
CAMG  ::= "CAMG"  #{ LONG                  }

CRNG  ::= "CRNG"  #{ CRange                 }
BODY  ::= "BODY"  #{ UBYTE*                 } [0]

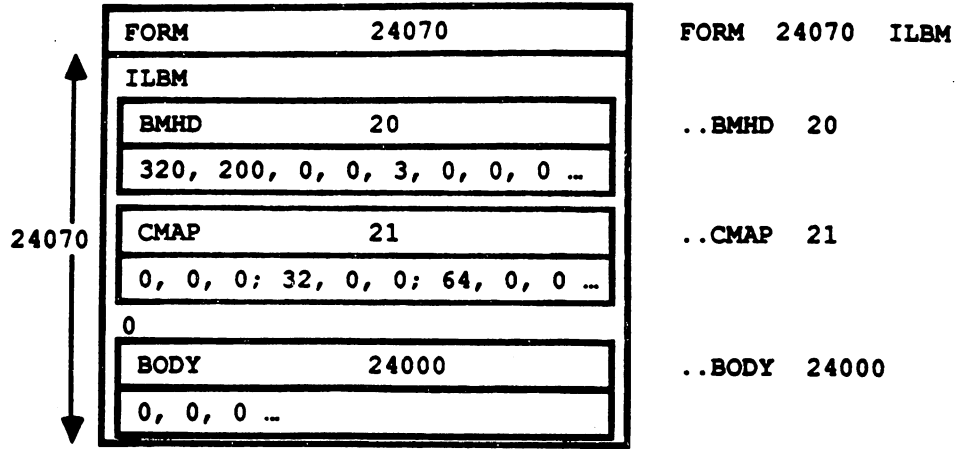
```

The token "#" represents a `ckSize` LONG count of the following (braced) data bytes. E.g. a BMHD's "#" should equal `sizeof(BitMapHeader)`. Literal strings are shown in "quotes", [square bracket items] are optional, and "*" means 0 or more repetitions. A sometimes-needed pad byte is shown as "[0]".

The property chunks (BMHD, CMAP, GRAB, DEST, SPRT, and CAMG) and any CRNG data chunks may actually be in any order but all must appear before the BODY chunk since ILBM readers usually stop as soon as they read the BODY. If any of the 6 property chunks are missing, default values are "inherited" from any shared properties (if the ILBM appears inside an IFF LIST with PROPs) or from the reader program's defaults. If any property appears more than once, the last occurrence before the BODY is the one that counts since that's the one that modifies the BODY.

Appendix B. ILBM Box Diagram

Here's a box diagram for a simple example: an uncompressed image 320 x 200 pixels x 3 bitplanes. The text to the right of the diagram shows the outline that would be printed by the IFFCheck utility program for this particular file.



The "0" after the CMAP chunk is a pad byte.

Appendix C. ByteRun1 Run Encoding

The run encoding scheme byteRun1 is best described by psuedo code for the decoder Unpacker (called UnPackBits in the Macintosh™ toolbox):

UnPacker:

```
LOOP until produced the desired number of bytes
  Read the next source byte into n
  SELECT n FROM
    [0..127]   => copy the next n+1 bytes literally
    [-1..-127] => replicate the next byte -n+1 times
    -128      => noop
  ENDCASE;
ENDLOOP;
```

In the inverse routine Packer, it's best to encode a 2 byte repeat run as a replicate run except when preceded and followed by a literal run, in which case it's best to merge the three into one literal run. Always encode 3 byte repeats as replicate runs.

Remember that each row of each scan line of a raster is separately packed.

Appendix D. Standards Committee

The following people contributed to the design of this FORM ILBM standard:

Bob "Kodlak" Burns, Commodore-Amiga
R. J. Mical, Commodore-Amiga
Jerry Morrison, Electronic Arts
Greg Riker, Electronic Arts
Steve Shaw, Electronic Arts
Dan Silva, Electronic Arts
Barry Walsh, Commodore-Amiga

"FTXT" IFF Formatted Text

Date: November 15, 1985
From: Steve Shaw and Jerry Morrison, Electronic Arts and Bob "Kodiak" Burns, Commodore-Amiga
Status: Draft 2.6

DRAFT DRAFT DRAFT DRAFT DRAFT

1. Introduction

This memo is the IFF supplement for FORM FTXT. An FTXT is an IFF "data section" or "FORM type"—which can be an IFF file or a part of one—containing a stream of text plus optional formatting information. "EA IFF 85" is Electronic Arts' standard for interchange format files. (See the IFF reference.)

An FTXT is an archival and interchange representation designed for three uses. The simplest use is for a "console device" or "glass teletype" (the minimal 2-D text layout means): a stream of "graphic" ("printable") characters plus positioning characters "space" ("SP") and line terminator ("LF"). This is not intended for cursor movements on a screen although it does not conflict with standard cursor-moving characters. The second use is text that has explicit formatting information (or "looks") such as font family and size, typeface, etc. The third use is as the lowest layer of a structured document that also has "inherited" styles to implicitly control character looks. For that use, FORMs FTXT would be embedded within a future document FORM type. The beauty of FTXT is that these three uses are interchangeable, that is, a program written for one purpose can read and write the others' files. So a word processor does not have to write a separate plain text file to communicate with other programs.

Text is stored in one or more "CHRS" chunks inside an FTXT. Each CHRS contains a stream of 8-bit text compatible with ISO and ANSI data interchange standards. FTXT uses just the central character set from the ISO/ANSI standards. (These two standards are henceforth called "ISO/ANSI" as in "see the ISO/ANSI reference".)

Since it's possible to extract just the text portions from future document FORM types, programs can exchange data without having to save both plain text and formatted text representations.

Character looks are stored as embedded control sequences within CHRS chunks. This document specifies which class of control sequences to use: the CSI group. This document does not yet specify their meanings, e.g. which one means "turn on italic face". Consult ISO/ANSI.

Section 2 defines the chunk types character stream "CHRS" and font specifier "FONS". These are the "standard" chunks. Specialized chunks for private or future needs can be added later. Section 3 outlines an FTXT reader program that strips a document down to plain unformatted text. Appendix A is a code table for the 8-bit ISO/ANSI character set used here. Appendix B is an example FTXT shown as a box diagram. Appendix C is a racetrack diagram of the syntax of ISO/ANSI control sequences.

Reference:

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

Electronic Arts™ is a trademark of Electronic Arts.

IFF: "EA IFF 85" Standard for Interchange Format Files describes the underlying conventions for all IFF files.

ISO/ANSI: ISO/DIS 6429.2 and ANSI X3.64-1979. International Organization for Standardization (ISO) and American National Standards Institute (ANSI) data-interchange standards. The relevant parts of these two standards documents are identical. ISO standard 2022 is also relevant.

2. Standard Data and Property Chunks

The main contents of a FORM FTXT is in its character stream "CHRS" chunks. Formatting property chunks may also appear. The only formatting property yet defined is "FONS", a font specifier. A FORM FTXT with no CHRS represents an empty text stream. A FORM FTXT may contain nested IFF FORMs, LISTs, or CATs, although a "stripping" reader (see section 3) will ignore them.

Character Set

FORM FTXT uses the core of the 8-bit character set defined by the ISO/ANSI standards cited at the start of this document. (See Appendix A for a character code table.) This character set is divided into two "graphic" groups plus two "control" groups. Eight of the control characters begin ISO/ANSI standard control sequences. (See "Control Sequences", below.) Most control sequences and control characters are reserved for future use and for compatibility with ISO/ANSI. Current reader programs should skip them.

- C0 is the group of control characters in the range NUL (hex 0) through hex 1F. Of these, only LF (hex 0A) and ESC (hex 1B) are significant. ESC begins a control sequence. LF is the line terminator, meaning "go to the first horizontal position of the next line". All other C0 characters are not used. In particular, CR (hex 0D) is not recognized as a line terminator.
- G0 is the group of graphic characters in the range hex 20 through hex 7F. SP (hex 20) is the space character. DEL (hex 7F) is the delete character which is not used. The rest are the standard ASCII printable characters "!" (hex 21) through "~" (hex 7E).
- C1 is the group of extended control characters in the range hex 80 through hex 9F. Some of these begin control sequences. The control sequence starting with CSI (hex 9B) is used for FTXT formatting. All other control sequences and C1 control characters are unused.
- G1 is the group of extended graphic characters in the range NBSP (hex A0) through "ÿ" (hex FF). It is one of the alternate graphic groups proposed for ISO/ANSI standardization.

Control Sequences

Eight of the control characters begin ISO/ANSI standard "control sequences" (or "escape sequences"). These sequences are described below and diagrammed in Appendix C.

```

G0      ::= (SP through DEL)
G1      ::= (NBSP through "ÿ")

ESC-Seq ::= ESC (SP through "/" ) * ("0" through "~")
ShiftToG2 ::= SS2 G0
ShiftToG3 ::= SS3 G0
CSI-Seq  ::= CSI (SP through "?") * ("@" through "~")
DCS-Seq  ::= (DCS | OSC | PM | APC) (SP through "~" | G1) * ST

```

"ESC-Seq" is the control sequence ESC (hex 1B), followed by zero or more characters in the range SP through "/" (hex 20 through hex 2F), followed by a character in the range "0" through "~" (hex 30 through hex 7E). These sequences are reserved for future use and should be skipped by current FTXT reader programs.

SS2 (hex 8E) and SS3 (hex 8F) shift the single following G0 character into yet-to-be-defined graphic sets G2 and G3, respectively. These sequences should not be used until the character sets G2 and G3 are standardized. A reader may simply skip the SS2 or SS3 (taking the following character as a corresponding G0 character) or replace the two-character sequence with a character like "?" to mean "absent".

FTXT uses "CSI-Seq" control sequences to store character formatting (font selection by number, type face, and text size) and perhaps layout information (position and rotation). "CSI-Seq" control sequences

start with CSI (the "control sequence introducer", hex 9B). Syntactically, the sequence includes zero or more characters in the range SP through "?" (hex 20 through hex 3F) and a concluding character in the range "@" through "~" (hex 40 through hex 7E). These sequences may be skipped by a minimal FTXT reader, i.e. one that ignores formatting information.

Note: A future FTXT standardization document will explain the uses of CSI-Seq sequences for setting character face (light weight vs. medium vs. bold, italic vs. upright, height, pitch, position, and rotation). For now, consult the ISO/ANSI references.

"DCS-Seq" is the control sequences starting with DCS (hex 90), OSC (hex 9D), PM (hex 9E), or APC (hex 9F), followed by zero or more characters each of which is in the range SP through "~" (hex 20 through hex 7E) or else a G1 character, and terminated by an ST (hex 9C). These sequences are reserved for future use and should be skipped by current FTXT reader programs.

Data Chunk CHRS

A CHRS chunk contains a sequence of 8-bit characters abiding by the ISO/ANSI standards cited at the start of this document. This includes the character set and control sequences as described above and summarized in Appendicies A and C.

A FORM FTXT may contain any number of CHRS chunks. Taken together, they represent a single stream of textual information. That is, the contents of CHRS chunks are effectively concatenated except that (1) each control sequence must be completely within a single CHRS chunk, and (2) any formatting property chunks appearing between two CHRS chunks affects the formatting of the latter chunk's text. Any formatting settings set by control sequences inside a CHRS carry over to the next CHRS in the same FORM FTXT. All formatting properties stop at the end of the FORM since IFF specifies that adjacent FORMs are independent of each other (although not independent of any properties inherited from an enclosing LIST or FORM).

Property Chunk FONS

The optional property "FONS" holds a FontSpecifier as defined in the C declaration below. It assigns a font to a numbered "font register" so it can be referenced by number within subsequent CHRS chunks. (This function is not provided within the ISO and ANSI standards.) The font specifier gives both a name and a description for the font so the recipient program can do font substitution.

By default, CHRS text uses font 1 until it selects another font. A minimal text reader always uses font 1. If font 1 hasn't been specified, the reader may use the local system font as font 1.

```
typedef struct {
    UBYTE id;          /* 0 through 9 is a font id number referenced by an SGR
                       control sequence selective parameter of 10 through 19.
                       Other values are reserved for future standardization. */
    UBYTE pad1;       /* reserved for future use; store 0 here */
    UBYTE proportional; /* proportional font? 0 = unknown, 1 = no, 2 = yes */
    UBYTE serif;      /* serif font? 0 = unknown, 1 = no, 2 = yes */
    char name[];     /* A NUL-terminated string naming the preferred font. */
} FontSpecifier;
```

Fields are filed in the order shown. The UBYTE fields are byte-packed (2 per 16-bit word). The field pad1 is reserved for future standardization. Programs should store 0 there for now.

The field `proportional` indicates if the desired font is proportional width as opposed to fixed width. The

field `serif` indicates if the desired font is serif as opposed to sans serif. [Issue: Discuss font substitution!]

Future Properties

New optional property chunks may be defined in the future to store additional formatting information. They will be used to represent formatting not encoded in standard ISO/ANSI control sequences and for "inherited" formatting in structured documents. Text orientation might be one example.

Positioning Units

Unless otherwise specified, position and size units used in FTXT formatting properties and control sequences are in decipoints (720 decipoints/inch). This is ANSI/ISO Positioning Unit Mode (PUM) 2. While a metric standard might be nice, decipoints allow the existing U.S.A. typographic units to be encoded easily, e.g. "12 points" is "120 decipoints".

3. FTXT Stripper

An FTXT reader program can read the text and ignore all formatting and structural information in a document FORM that uses FORMs FTXT for the leaf nodes. This amounts to stripping a document down to a stream of plain text. It would do this by skipping over all chunks except FTXT.CHRS (CHRS chunks found inside a FORM FTXT) and within the FTXT.CHRS chunks skipping all control characters and control sequences. (Appendix C diagrams this text scanner.) It may also read FTXT.FONS chunks to find a description for font 1.

Here's a Pascal-ish program for an FTXT stripper. Given a FORM (a document of some kind), it scans for all FTXT.CHRS chunks. This would likely be applied to the first FORM in an IFF file.

```

PROCEDURE ReadFORM4CHRS ();      {Read an IFF FORM for FTXT.CHRS chunks.}
  BEGIN
    IF the FORM's subtype = "FTXT"
      THEN ReadFTXT4CHRS ()
      ELSE WHILE something left to read in the FORM DO BEGIN
          read the next chunk header;
          CASE the chunk's ID OF
            "LIST", "CAT ": ReadCAT4CHRS ();
            "FORM": ReadFORM4CHRS ();
            OTHERWISE skip the chunk's body;
          END
        END
      END;

```

{Read a LIST or CAT for all FTXT.CHRS chunks.}

```

PROCEDURE ReadCAT4CHRS ();
  BEGIN
    WHILE something left to read in the LIST or CAT DO BEGIN
      read the next chunk header;
      CASE the chunk's ID OF
        "LIST", "CAT ": ReadCAT4CHRS ();
        "FORM": ReadFORM4CHRS ();
        "PROP": IF we're reading a LIST AND the PROP's subtype = "FTXT"
          THEN read the PROP for "FONS" chunks;
        OTHERWISE error--malformed IFF file;
      END
    END
  END;

```

```

PROCEDURE ReadFTXT4CHRS ();      {Read a FORM FTXT for CHRS chunks.}
  BEGIN
    WHILE something left to read in the FORM FTXT DO BEGIN
      read the next chunk header;
      CASE the chunk's ID OF
        "CHRS": ReadCHRS ();
        "FONS": BEGIN
          read the chunk's contents into a FontSpecifier variable;
          IF the font specifier's id = 1 THEN use this font;
          END;
        OTHERWISE skip the chunk's body;
      END
    END
  END;

```

```
{Read an FTXT.CHRS. Skip all control sequences and unused control chars.}
PROCEDURE ReadCHRS();
```

```
  BEGIN
    WHILE something left to read in the CHRS chunk DO
      CASE read the next character OF
        LF: start a new output line;
        ESC: SkipControl([' '..'/'], ['0'..'~']);
        IN [' '..'~'], IN [NBSP..'y']: output the character;
        SS2, SS3: ; {Just handle the following G0 character directly,
                     ignoring the shift to G2 or G3.}
        CSI: SkipControl([' '..'?'], ['@'..'~']);
        DCS, OSC, PM, APC: SkipControl([' '..'~'] + [NBSP..'y'], [ST]);
      END
    END;
```

```
{Skip a control sequence of the format (rSet)* (tSet), i.e. any number of
characters in the set rSet followed by a character in the set tSet.}
PROCEDURE SkipControl(rSet, tSet);
```

```
  VAR c: CHAR;
  BEGIN
    REPEAT c := read the next character
      UNTIL c NOT IN rSet;
    IF c NOT IN tSet
      THEN put character c back into the input stream;
    END
  END
```

The following program is an optimized version of the above routines ReadFORM4CHRS and ReadCAT4CHRS for the case where you're ignoring fonts as well as formatting. It takes advantage of certain facts of the IFF format to read a document FORM and its nested FORMs, LISTs, and CATs without a stack. In other words, it's a hack that ignores all fonts and faces to cheaply get to the plain text of the document.

```
{Cheap scan of an IFF FORM for FTXT.CHRS chunks.}
PROCEDURE ScanFORM4CHRS();
```

```
  BEGIN
    IF the document FORM's subtype = "FTXT"
      THEN ReadFTXT4CHRS()
    ELSE WHILE something left to read in the FORM DO BEGIN
      read the next chunk header;
      IF it's a group chunk (LIST, FORM, PROP, or CAT)
        THEN read its subtype ID;
      CASE the chunk's ID OF
        "LIST", "CAT ":; {NOTE: See explanation below.*}
        "FORM": IF this FORM's subtype = "FTXT" THEN ReadFTXT4CHRS()
          ELSE; {NOTE: See explanation below.*}
        OTHERWISE skip the chunk's body;
      END
    END
  END
END;
```

*Note: This implementation is subtle. After reading a group header other than FORM FTXT it just continues reading. This amounts to reading all the chunks inside that group as if they weren't nested in a group.

Appendix A: Character Code Table

This table corresponds to the ISO/DIS 6429.2 and ANSI X3.64-1979 8-bit character set standards. Only the core character set of those standards is used in FTXT.

Two G1 characters aren't defined in the standards and are shown as dark gray entries in this table. Light gray shading denotes control characters. (DEL is a control character although it belongs to the graphic group G0.) The following five rare G1 characters are left blank in the table below due to limitations of available fonts: hex A8, D0, DE, F0, and FE.

ISO/DIS 6429.2 and ANSI X3.64-1979 Character Code Table

| LSN ↓ | Most Significant Nibble (hex digit) | | | | | | | | | | | | | | | |
|----------|-------------------------------------|-----|----|---|---|---|---|-----|-----|-----|------|-----|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | NUL | | SP | 0 | @ | P | ` | p | | DCS | NBSP | · | À | Ñ | à | ñ |
| 1 | | | ! | 1 | A | Q | a | q | | | ¡ | ± | Á | Ò | á | õ |
| 2 | | | " | 2 | B | R | b | r | | | ¢ | ² | Â | Ó | â | ö |
| 3 | | | # | 3 | C | S | c | s | | | £ | ³ | Ã | Ô | ã | ó |
| 4 | | | \$ | 4 | D | T | d | t | | | ¤ | ´ | Ä | Õ | ä | ô |
| 5 | | | % | 5 | E | U | e | u | | | ¥ | µ | Å | Ö | å | ö |
| 6 | | | & | 6 | F | V | f | v | | | ¦ | ¶ | Æ | Ø | æ | ø |
| 7 | | | ' | 7 | G | W | g | w | | | § | · | Ç | | ç | |
| 8 | | | (| 8 | H | X | h | x | | | © | / | È | Ù | è | ù |
| 9 | | |) | 9 | I | Y | i | y | | | ® | í | É | Ú | é | ú |
| A | LF | | * | : | J | Z | j | z | | | ª | º | Ê | Û | ê | û |
| B | | ESC | + | ; | K | [| k | { | | | « | » | Ë | Ü | ë | ü |
| C | | | , | < | L | \ | l | | | | ¬ | ¼ | Ì | Ý | ì | ÿ |
| D | CR | | - | = | M |] | m | } | | | ST | 1/4 | Í | | í | |
| E | | | . | > | N | ^ | n | ~ | SS2 | OSC | SHY | 1/2 | Î | | î | |
| F | | | / | ? | O | o | o | DEL | SS3 | PM | ® | 3/4 | Ï | | ï | |
| | | | | | | | | | | APC | - | ¿ | ÿ | | ÿ | |

Control
group
C0

Graphic group
G0

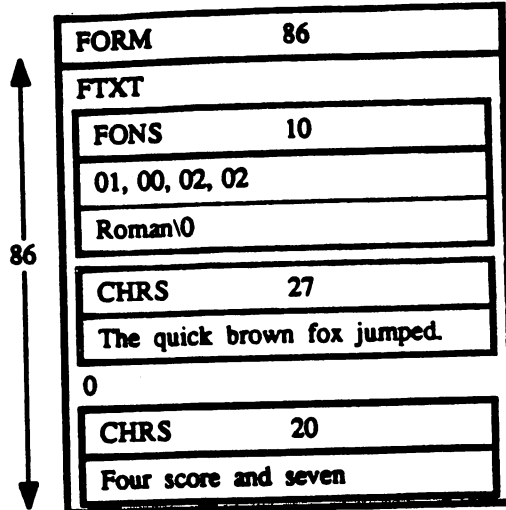
Control
group
C1

Graphic group
G1

"NBSP" is a "non-breaking space"
 "SHY" is a "soft hyphen"

Appendix B. FTXT Example

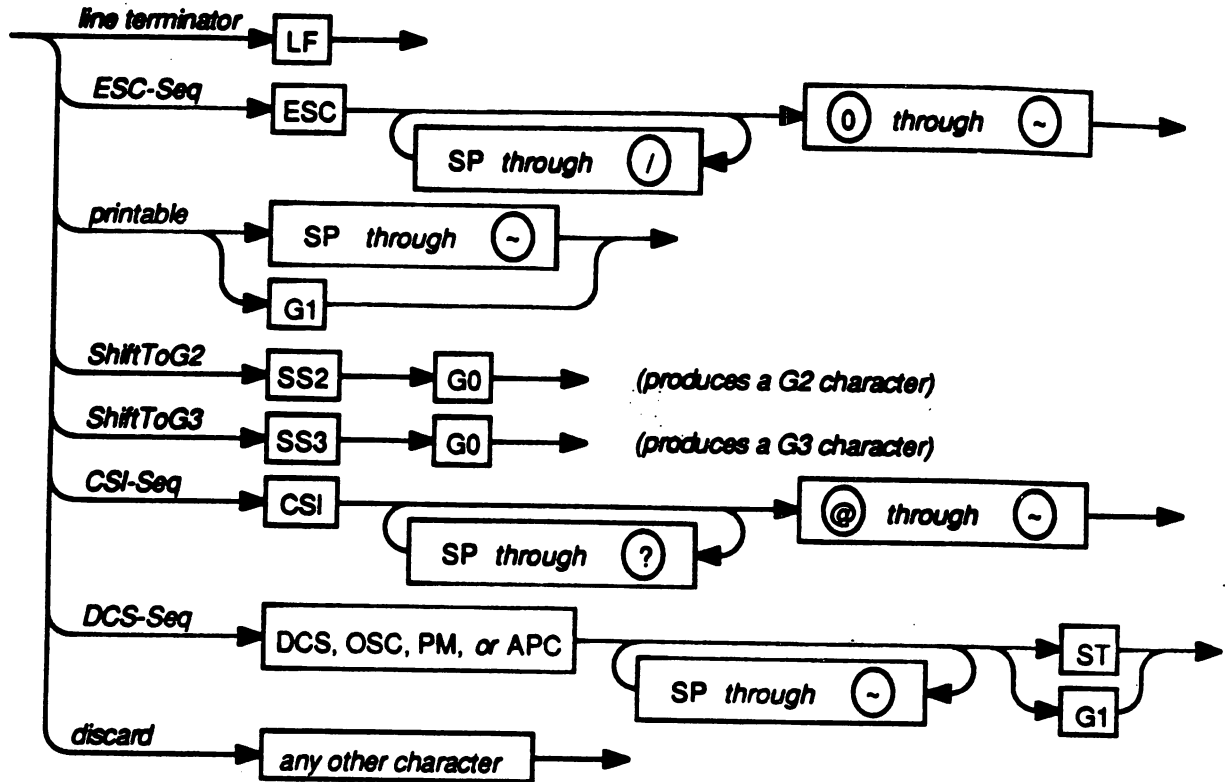
Here's a box diagram for a simple example: "The quick brown fox jumped.Four score and seven", written in a proportional serif font named "Roman".



The "0" after the first CHRS chunk is a pad byte.

Appendix C. ISO/ANSI Control Sequences

This is a racetrack diagram of the ISO/ANSI characters and control sequences as used in FTXT CHR8 chunks.



Of the various control sequences, only CSI-Seq is used for FTXT character formatting information. The others are reserved for future use and for compatibility with ISO/ANSI standards. Certain character sequences are syntactically malformed, e.g. CSI followed by a C0, C1, or G1 character. Writer programs should not generate reserved or malformed sequences and reader programs should skip them.

Consult the ISO/ANSI standards for the meaning of the CSI-Seq control sequences.

The two character set shifts SS2 and SS3 may be used when the graphic character groups G2 and G3 become standardized.


```

#endif IFF_H
#define IFF_H
/* IFF.H defs for IFF-85 Interchange Format Files. 10/10/85 */
/*
/* By Jerry Morrison and Steve Shaw, Electronic Arts.
/* This software is in the public domain.
*/

```

```

#include "libraries/dos.h"
#endif

typedef LONG IFFP; /* Status code result from an IFF procedure */
/* LONC, because must be type compatible with ID for CatChunkdir. */
/* Note that the error codes below are not legal IDs. */
#define IFF_OKAY 0 /* Keep going... */
#define END_MARK -1 /* As if there was a chunk at end of group. */
#define IFF_DONE -2 /* clientProc returns this when it has READ enough. */
/* It means return thru all levels. File is Okay. */

#define DOS_ERROR -3 /* not an IFF file. */
#define NOT_IFF -4 /* Tried to open file, DOS didn't find it. */
#define NO_FILE -5 /* Client made invalid request, for instance, asking
for more bytes than existed in chunk. */
#define CLIENT_ERROR -6 /* A client read proc complains about FORM semantics;
e.g. valid IFF, but missing a required chunk. */
#define BAD_FORM -7 /* Client asked to IFFReadBytes more bytes than left
in the chunk. Could be client bug or bad form. */
#define BAD_IFF -9 /* mal-formed IFF file. [TED] Expand this into a
range of error codes. */
#define LAST_ERROR BAD_IFF

```

```

/* This MACRO is used to RETURN immediately when a termination condition is
found. This is a pretty weird macro. It requires the caller to declare a
local "IFFP iff" and assign it. This wouldn't work as a subroutine since
it returns for it's caller. */
#define CheckIFFP() { if (iffp != IFF_OKAY) return(iffp); }

```

```

/* ----- ID ----- */
typedef LONG ID; /* An ID is four printable ASCII chars but
* stored as a LONG for efficient copy & compare. */

/* Four-character Identifier builder. */
#define MakeID(a,b,c,d) ((a)<<24 | (b)<<16 | (c)<<8 | (d))

/* Standard group IDs. A chunk with one of these IDs contains a
SubTypeID followed by zero or more chunks. */
#define FORM MakeID('F','O','R','M')
#define PROP MakeID('P','R','O','P')
#define LIST MakeID('L','I','S','T')
#define CAT MakeID('C','A','T','')
#define FILLER MakeID(' ',' ',' ',' ')
/* The IDs "FOR1"..."FOR9", "LIS1"..."LIS9", & "CAT1"..."CAT9" are reserved

```

```

/* for future standardization. */
/* Pseudo-ID used internally by chunk reader and writer. */
#define NULL_CHUNK 0L /* No current chunk. */

/* ----- Chunk ----- */
/* All chunks start with a type ID and a count of the data bytes that
follow--the chunk's "logical size" or "data size". If that number is odd,
a 0 pad byte is written, too. */
typedef struct {
ID ckID;
LONG ckSize;
} ChunkHeader;

typedef struct {
ID ckID;
LONG ckSize;
UBYTE ckData[ 1 /*REALLY: ckSize*/ ];
} Chunk;

/* Pass ckSize = szNotYetKnown to the writer to mean "compute the size". */
#define szNotYetKnown 0x80000001L

/* Need to know whether a value is odd so can word-align. */
#define IS_ODD(a) ((a) & 1)

/* This macro rounds up to an even number. */
#define WordAlign(size) ((size+1)&~1)

/* ALL CHUNKS MUST BE PADDED TO EVEN NUMBER OF BYTES.
ChunkPSize computes the total "physical size" of a padded chunk from
its "data size" or "logical size".
#define ChunkPSize(dataSize) (WordAlign(dataSize) + sizeof(ChunkHeader))

/* The Grouping chunks (LIST, FORM, PROP, & CAT) contain concatenations of
chunks after a subtype ID that identifies the content chunks.
* "FORM type XXXX" "LIST of FORM type XXXX", "PROPERTIES associated
* with FORM type XXXX", or "concatenation of XXXX". */
typedef struct {
ID ckID; /* this ckSize includes "grpSubID". */
LONG ckSize;
ID grpSubID;
} GroupHeader;

typedef struct {
ID ckID;
LONG ckSize;
ID grpSubID;
UBYTE grpData[ 1 /*REALLY: ckSize-sizeof(grpSubID) */ ];
} GroupChunk;

/* ----- IFF Reader ----- */

```

```

/***** Routines to support a stream-oriented IFF file reader *****/
*
* These routines handle lots of details like error checking and skipping
* over padding. They're also careful not to read past any containing context.
*
* These routines ASSUME they're the only ones reading from the file.
* Client should check IFFP error codes. Don't press on after an error!
* These routines try to have no side effects in the error case, except
* partial I/O is sometimes unavoidable.
*
* All of these routines may return DOS_ERROR. In that case, ask DOS for the
* specific error code.
*
* The overall scheme for the low level chunk reader is to open a "group read
* context" with OpenRIFF or OpenRGroup, read the chunks with GetChunkHdr
* (and its kin) and IFFReadBytes, and close the context with CloseRGroup.
*
* The overall scheme for reading an IFF file is to use ReadIFF, ReadList,
* and ReadCat to scan the file. See those procedures. ClientProc (below),
* and the skeleton IFF reader. */

/* Client passes ptrs to procedures of this type to ReadIFF which call them
* back to handle LISTS, FORMs, CATs, and PROPs.
*
* Use the GroupContext ptr when calling reader routines like GetChunkHdr.
* Look inside the GroupContext ptr for your ClientFrames ptr. You'll
* want to type cast it into a ptr to your containing struct to get your
* private contextual data (stacked property settings). See below. */
typedef IFFP ClientProc( /* struct_GroupContext * */);

/* Client's context for reading an IFF file or a group.
* Client should actually make this the first component of a larger struct
* (it's personal stack "frames") that has a field to store each "interesting"
* property encountered.
* Either initialize each such field to a global default or keep a boolean
* indicating if you've read a property chunk into that field.
* Your getList and getForm procs should allocate a new "frames" and copy the
* parent frames' contents. The getProp procedure should store into the frames
* allocated by getList for the containing LIST. */
typedef struct_ClientFrame {
    ClientProc *getList, *getProp, *getForm, *getCat;
    /* client's own data follows; place to stack property settings */
} ClientFrame;

/* Our context for reading a group chunk. */
typedef struct_GroupContext {
    struct_GroupContext *parent; /* Containing group; NULL => whole file. */
    ClientFrame *clientFrames; /* Reader data & client's context state. */
    IFFP file; /* Byte-stream file handle. */
    LONG position; /* The context's logical file position. */
    LONG bound; /* File-absolute context bound
    * or szNotYetKnown (writer only). */
    ChunkHeader ckHdr; /* Current chunk header. ckHdr.ckSize = szNotYetKnown
    * means we need to go back and set the size (writer only).
    * See also Pseudo-IDs, above. */
    ID subtype; /* Group's subtype ID when reading. */
}

```

```

LONG bytesSoFar; /* # bytes read/written of current chunk's data. */
} GroupContext;

/* Computes the number of bytes not yet read from the current chunk, given
* a group read context gc. */
#define ChunkMoreBytes(gc) ((gc)->ckHdr.ckSize - (gc)->bytesSoFar)

/***** Low Level IFF Chunk Reader *****/
*
* Given an open file, open a read context spanning the whole file.
* This is normally only called by ReadIFF.
* This sets new->clientFrames = clientFrames.
* ASSUME context allocated by caller but not initialized.
* ASSUME caller doesn't deallocate the context before calling CloseRGroup.
* NOT_IFF_ERROR if the file is too short for even a chunk header. */
extern IFFP OpenRIFF( /* IFFR, GroupContext *, ClientFrame * */;
    /* file, new, clientFrame */);

/* Open the remainder of the current chunk as a group read context.
* This will be called just after the group's subtype ID has been read
* (automatically by GetChunkHdr for LIST, FORM, PROP, and CAT) so the
* remainder is a sequence of chunks.
* This sets new->clientFrames = parent->clientFrames. The caller should re-point
* it at a new clientFrames if opening a LIST context so it'll have a "stack
* frames" to store PROPs for the LIST. (It's usually convenient to also
* allocate a new Frame when you encounter FORM of the right type.)
* ASSUME new context allocated by caller but not initialized.
* ASSUME caller doesn't deallocate the context or access the parent context
* before calling CloseRGroup.
* BAD_IFF_ERROR if context end is odd or extends past parent. */
extern IFFP OpenRGroup( /* GroupContext *, GroupContext * */;
    /* parent, new */);

/* Close a group read context, updating its parent context.
* After calling this, the old context may be deallocated and the parent
* context can be accessed again. It's okay to call this particular procedure
* after an error has occurred reading the group.
* This always returns IFF_OKAY. */
extern IFFP CloseRGroup( /* GroupContext * */;
    /* old */);

/* Skip any remaining bytes of the previous chunk and any padding, then
* read the next chunk header into context.ckHdr.
* If the ckID is LIST, FORM, CAT, or PROP, this automatically reads the
* subtype ID into context->subtype.
* Caller should dispatch on ckID (and subtype) to an appropriate handler.
* RETURNS context.ckHdr.ckID (the ID of the new chunk header); END_MARK
* if there are no more chunks in this context; or NOT_IFF if the top level
* file chunk isn't a FORM, LIST, or CAT; or BAD_IFF if malformed chunk, e.g.
* ckSize is negative or too big for containing context, ckID isn't positive,
* or we hit end-of-file.
* See also GetFChunkHdr, GetF1ChunkHdr, and GetPCChunkHdr, below. */

```

```

extern ID      GetChunkHdr (/ * GroupContext * */);
/* context.cldhdr.chkID context */

/* Read nBytes number of data bytes of current chunk. (Use OpenGroup, etc.
 * times to read the contents of a group chunk.) You can call this several
 * times to read the data piecemeal.
 * CLIENT_ERROR if nBytes < 0. SHORT_CHUNK if nBytes > ChunkMoreBytes (context)
 * which could be due to a client bug or a chunk that's shorter than it
 * ought to be (bad form). (on either CLIENT_ERROR or SHORT_CHUNK,
 * IFFReadBytes won't read any bytes.) */
extern IFFP IFFReadBytes (/ * GroupContext *, BYTE *, LONG */);
/* context.cldhdr.chkID context, buffer, nBytes */

***** IFF File Reader *****

/* This is a noop ClientProc that you can use for a getLIST, getFORM, getPROP,
 * or getCAT procedure that just skips the group. A simple reader might just
 * implement getFORM, store &readCat in the getCat field of clientFrame, and
 * use &SkipGroup for the getLIST and getProp procs.*/
extern IFFP SkipGroup (/ * GroupContext * */);

/* IFF file reader.
 * Given an open file, allocate a group context and use it to read the FORM,
 * LIST, or CAT and it's contents. The idea is to parse the file's contents,
 * and for each FORM, LIST, CAT, or PROP encountered, call the getFORM,
 * getLIST, getCat, or getProp procedure in clientFrame, passing the
 * GroupContext ptr.
 * This is achieved with the aid of ReadLIST (which your getLIST should
 * call) and ReadCat (which your getCat should call, if you don't just use
 * &readCat for your getCat). If you want to handle FORMs, LISTs, and CATs
 * nested within FORMs, the getFORM procedure must dispatch to getFORM,
 * getLIST, and getCat (it can use GetFChunkHdr to make this easy).
 * Normal return is IFF_OKAY (if whole file scanned) or IFF_DONE (if a client
 * proc said "done" first).
 * See the skeletal getLIST, getFORM, getCat, and getProp procedures. */
extern IFFP ReadIFF (/ * EPTR, ClientFrame * */);
/* file, clientFrame */

/* IFF LIST reader.
 * Your "getLIST" procedure should allocate a ClientFrame, copy the parent's
 * ClientFrame, and then call this procedure to do all the work.
 * Normal return is IFF_OKAY (if whole LIST scanned) or IFF_DONE (if a client
 * proc said "done" first).
 * BAD_IFF_ERROR if a PROP appears after a non-PROP. */
extern IFFP ReadILLIST (/ * GroupContext *, ClientFrame * */);
/* parent, clientFrame */

/* IFF CAT reader.
 * Most clients can simply use this to read their CATs. If you must do extra
 * setup work, put a ptr to your getCat procedure in the clientFrame, and
 * have that procedure call ReadCat to do the detail work.
 * Normal return is IFF_OKAY (if whole CAT scanned) or IFF_DONE (if a client

```

```

 * proc said "done" first).
 * BAD_IFF_ERROR if a PROP appears in the CAT. */
extern IFFP ReadICat (/ * GroupContext * */);
/* parent */

/* Call GetFChunkHdr instead of GetChunkHdr to read each chunk inside a FORM.
 * It just calls GetChunkHdr and returns BAD_IFF if it gets a PROP chunk. */
extern ID      GetFChunkHdr (/ * GroupContext * */);
/* context.cldhdr.chkID context */

/* GetFChunkHdr is like GetFChunkHdr, but it automatically dispatches to the
 * getFORM, getLIST, and getCat procedure (and returns the result) if it
 * encounters a FORM, LIST, or CAT. */
extern ID      GetFChunkHdr (/ * GroupContext * */);
/* context.cldhdr.chkID context */

/* Call GetPChunkHdr instead of GetChunkHdr to read each chunk inside a PROP.
 * It just calls GetChunkHdr and returns BAD_IFF if it gets a group chunk. */
extern ID      GetPChunkHdr (/ * GroupContext * */);
/* context.cldhdr.chkID context */

/* ----- IFF Writer -----

***** Routines to support a stream-oriented IFF file writer *****

 * These routines will random access back to set a chunk size value when the
 * caller doesn't know it ahead of time. They'll also do things automatically
 * like padding and error checking.
 * These routines ASSUME they're the only ones writing to the file.
 * Client should check IFFP error codes. Don't press on after an error!
 * These routines try to have no side effects in the error case, except that
 * partial I/O is sometimes unavoidable.
 * All of these routines may return DOS_ERROR. In that case, ask DOS for the
 * specific error code.
 * The overall scheme is to open an output GroupContext via OpenWIFF or
 * OpenWGroup, call either PutCk or {PutCkHdr {IFFWriteBytes} PutCkEnd} for
 * each chunk, then use CloseWGroup to close the GroupContext.
 * To write a group (LIST, FORM, PROP, or CAT), call StartWGroup, write out
 * its chunks, then call EndWGroup. StartWGroup automatically writes the
 * group header and opens a nested context for writing the contents.
 * EndWGroup closes the nested context and completes the group chunk. */

/* Given a file open for output, open a write context.
 * The "limit" arg imposes a fence or upper limit on the logical file
 * position for writing data in this context. Pass in szNotYetKnown to be
 * bounded only by disk capacity.
 * ASSUME new context structure allocated by caller but not initialized.
 * ASSUME caller doesn't deallocate the context before calling CloseWGroup.
 * The caller is only allowed to write out one FORM, LIST, or CAT in this top
 * level context (see StartWGroup and PutCkHdr).

```

```

* CLIENT_ERROR if limit is odd. */
extern IFFP OpenMIEFF( /* PTR, GroupContext *, LONG */;
/* file, new, limit {file position} */

/* Start writing a group (presumably LIST, FORM, PROP, or CAT), opening a
* nested context. The groupSize includes all nested chunks + the subtype ID.
*
* The subtype of a LIST or CAT is a hint at the contents' FORM type(s). Pass
* in FILLER if it's a mixture of different kinds.
*
* This writes the chunk header via PutCkHdr, writes the subtype ID via
* IFFWriteBytes, and calls OpenMGroup. The caller may then write the nested
* chunks and finish by calling EndMGroup.
*
* The OpenMGroup call sets new->clientFrame = parent->clientFrame.
*
* ASSUME new context structure allocated by caller but not initialized.
* ASSUME caller doesn't deallocate the context or access the parent context
* before calling CloseMGroup.
* ERROR conditions: See PutCkHdr, IFFWriteBytes, OpenMGroup.
extern IFFP StartMGroup( /* GroupContext *, ID, LONG, ID, GroupContext * */;
/* parent, groupType, groupSize, subtype, new */

/* End a group started by StartMGroup.
* This just calls CloseMGroup and PutCkEnd.
* ERROR conditions: See CloseMGroup and PutCkEnd.
extern IFFP EndMGroup( /* GroupContext * */;
/* old */

/* Open the remainder of the current chunk as a group write context.
* This is normally only called by StartMGroup.
*
* Any fixed limit to this group chunk or a containing context will impose
* a limit on the new context.
* This will be called just after the group's subtype ID has been written
* so the remaining contents will be a sequence of chunks.
* This sets new->clientFrame = parent->clientFrame.
* ASSUME new context structure allocated by caller but not initialized.
* ASSUME caller doesn't deallocate the context or access the parent context
* before calling CloseMGroup.
* CLIENT_ERROR if context end is odd or PutCkHdr wasn't called first.
extern IFFP OpenMGroup( /* GroupContext *, GroupContext * */;
/* parent, new */

/* Close a write context and update its parent context.
* This is normally only called by EndMGroup.
*
* If this is a top level context (created by OpenMIEFF) we'll set the file's
* EOF (end of file) but won't close the file.
* After calling this, the old context may be deallocated and the parent
* context can be accessed again.
*
* Amiga DOS Note: There's no call to set the EOF. We just position to the
* desired end and return. Caller must Close file at that position.
* CLIENT_ERROR if PutCkEnd wasn't called first.
extern IFFP CloseMGroup( /* GroupContext * */;
/* old */

```

```

/* Write a whole chunk to a GroupContext. This writes a chunk header, ckSize
* data bytes, and (if needed) a pad byte. It also updates the GroupContext.
* CLIENT_ERROR if ckSize = szNotYetKnown. See also PutCkHdr errors.
extern IFFP PutCk( /* GroupContext *, ID, LONG, BYTE */;
/* ckID, ckSize, *data */

/* Write just a chunk header. Follow this will any number of calls to
* IFFWriteBytes and finish with PutCkEnd.
* If you don't yet know how big the chunk is, pass in ckSize = szNotYetKnown,
* then PutCkEnd will set the ckSize for you later.
* Otherwise, IFFWriteBytes and PutCkEnd will ensure that the specified
* number of bytes get written.
* CLIENT_ERROR if the chunk would overflow the GroupContext's bound, if
* PutCkHdr was previously called without a matching PutCkEnd, if ckSize < 0
* (except szNotYetKnown), if you're trying to write something other
* than one FORM, LIST, or CAT in a top level (file level) context, or
* if ckID <= 0 (these illegal ID values are used for error codes).
extern IFFP PutCkHdr( /* GroupContext *, ID, LONG */;
/* context, ckID, ckSize */

/* Write nBytes number of data bytes for the current chunk and update
* GroupContext.
* CLIENT_ERROR if this would overflow the GroupContext's limit or the
* current chunk's ckSize, or if PutCkHdr wasn't called first, or if
* nBytes < 0.
extern IFFP IFFWriteBytes( /* GroupContext *, BYTE *, LONG */;
/* context, *data, nBytes */

/* Complete the current chunk, write a pad byte if needed, and update
* GroupContext.
* If current chunk's ckSize = szNotYetKnown, this goes back and sets the
* ckSize in the file.
* CLIENT_ERROR if PutCkHdr wasn't called first, or if client hasn't
* written 'ckSize' number of bytes with IFFWriteBytes.
extern IFFP PutCkEnd( /* GroupContext * */;
/* context */

```

#endif

```

#ifndef ILEM_H
#define ILEM_H
/* ILEM.H Definitions for InterLeaved BitMap raster image. 11/11/85
*
* By Jerry Morrison and Steve Shaw, Electronic Arts.
* This software is in the public domain.
*
* This version for the Commodore-Amiga computer.
*/
#ifndef EXEC_TYPES_H
#include "exec/types.h"
#endif
#ifndef GRAPHICS_GEX_H
#include "graphics/gfx.h"
#endif
#include "iff.h"
#define ID_ILEM MakeID('I', 'L', 'B', 'M')
#define ID_BMHD MakeID('B', 'M', 'H', 'D')
#define ID_CMAP MakeID('C', 'M', 'A', 'P')
#define ID_CRAB MakeID('C', 'R', 'A', 'B')
#define ID_DEST MakeID('D', 'E', 'S', 'T')
#define ID_SVRT MakeID('S', 'V', 'R', 'T')
#define ID_CAWG MakeID('C', 'A', 'W', 'G')
#define ID_BODY MakeID('B', 'O', 'D', 'Y')
/* ----- BitMapHeader ----- */
typedef UBYTE Masking; /* Choice of masking technique. */
#define maskNone 0
#define maskAsMask 1
#define maskAsTransparentColor 2
#define maskLasso 3
typedef UBYTE Compression; /* Choice of compression algorithm applied to
* each row of the source and mask planes. "cmpByteRun1" is the byte run
* encoding generated by Mac's PackBits. See Packer.h. */
#define cmpNone 0
#define cmpByteRun1 1
/* Aspect ratios: The proper fraction xAspect/yAspect represents the pixel
* aspect ratio pixel_width/pixel_height.
*
* For the 4 Amiga display modes:
* 320 x 200: 10/11 (these pixels are taller than they are wide)
* 320 x 400: 20/11
* 640 x 200: 5/11
* 640 x 400: 10/11
#define x320x200Aspect 10
#define y320x200Aspect 11
#define x320x400Aspect 20
#define y320x400Aspect 11
#define x640x200Aspect 5
#define y640x200Aspect 11

```

```

#define y640x200Aspect 11
#define x640x400Aspect 10
#define y640x400Aspect 11
/* A BitMapHeader is stored in a BMHD chunk. */
typedef struct {
  WORD w, h; /* raster width & height in pixels */
  WORD x, y; /* position for this image */
  UBYTE nPlanes; /* # source bitplanes */
  Masking masking; /* masking technique */
  Compression compression; /* compression algorithm */
  UBYTE pad1; /* UNUSED. For consistency, put 0 here. */
  WORD transparentColor; /* transparent "color number" */
  UBYTE xAspect, yAspect; /* aspect ratio, a rational number x/y */
  WORD pageWidth, pageHeight; /* source "page" size in pixels */
} BitMapHeader;
/* RowBytes computes the number of bytes in a row, from the width in pixels. */
#define RowBytes(w) (((w) + 15) >> 4 << 1)
/* ----- ColorRegister ----- */
/* A CMAP chunk is a packed array of ColorRegisters (3 bytes each). */
typedef struct {
  UBYTE red, green, blue; /* MUST be UBYTES so ">> 4" won't sign extend. */
} ColorRegister;
/* Use this constant instead of sizeof(ColorRegister). */
#define sizeofColorRegister 3
typedef WORD Color4; /* Amiga RAM version of a color-register
* with 4 bits each RGB in low 12 bits. */
/* Maximum number of bitplanes in RAM. Current Amiga max w/dual playfield. */
#define MaxAmDepth 6
/* ----- Point2D ----- */
/* A Point2D is stored in a GRAB chunk. */
typedef struct {
  WORD x, y; /* coordinates (pixels) */
} Point2D;
/* ----- DestMerge ----- */
/* A DestMerge is stored in a DEST chunk. */
typedef struct {
  UBYTE depth; /* # bitplanes in the original source */
  WORD planePick; /* UNUSED; for consistency store 0 here */
  WORD planeOff; /* how to scatter source bitplanes into destination */
  WORD planeMask; /* default bitplane data for planePick */
} DestMerge; /* selects which bitplanes to store into */
/* ----- SpritePrecedence ----- */
/* A SpritePrecedence is stored in a SPRT chunk. */
typedef WORD SpritePrecedence;

```

```
#####
# $Header: Makefile.v 1.0 85/04/02 19:29:23 kodiak Exp $
# $Locker: $
# $Log: Makefile.v $
#####
MAKEFILE= iffcheck.mk
MAKEMETA= /usr/commodore/amiga/V1/tools/makemeta
SRCDIRPATH= clipboard
SRCDIR= iff
SUBSYSNAME= iffcheck
DISKPATH= cll/c/iffcheck

STARTUP= ${LIBDIR}/startup.obj
PFLAGS= -Plab
MYLIBS= ${LIBDIR}/debug.lib
CFLAGS= '-DDEBUG'

CFILES= iffcheck.c iffr.c
OFILES= iffcheck.obj iffr.obj

all: ${SUBSYSNAME}.ld

.INCLUDE=${MAKEMETA}
SYMBOLOPT=
```

```
#####
# $Header: Makefile.v 1.0 85/04/02 19:29:23 kodiak Exp $
# $Locker: $
# $Log: Makefile.v $
#####
MAKEFILE= rav211bm.mk
MAKEMETA= /usr/commodore/amiga/V1/tools/makemeta
SRCDIRPATH= clipboard
SRCDIR= iff
SUBSYSNAME= rav211bm
DISKPATH= cll/c/rav211bm

STARTUP= ${LIBDIR}/startup.obj
PFLAGS= -Plab
MYLIBS= ${LIBDIR}/debug.lib
CFLAGS= '-DDEBUG'

CFILES= rav211bm.c iffv.c 11bmw.c packer.c
OFILES= rav211bm.obj iffv.obj 11bmw.obj packer.obj

all: ${SUBSYSNAME}.ld

.INCLUDE=${MAKEMETA}
SYMBOLOPT=
```

```

/*
 * IFFCheck.C Print out the structure of an IFF-85 file, 11/11/85
 * checking for structural errors.
 * DO NOT USE THIS AS A SKELETAL PROGRAM FOR AN IFF READER!
 * See SHOWILBM.C for a skeletal example.
 * This version for the Commodore-Amiga computer.
#include "exec/types.h"
#include "libraries/dos.h"
#include "iff.h"

/* ----- IFFCheck ----- */
/* [TBD] More extensive checking could be done on the IDs encountered in the
 * file. Check that the reserved IDs "FOR1", "FOR9", "LISI", "LIS9", and
 * "CAT1", "CAT9" aren't used. Check that reserved IDs aren't used as Form
 * types. Check that all IDs are made of 4 printable characters (trailing
 * spaces ok). */

typedef struct {
    ClientFrame clientFrame;
    int levels; /* # groups currently nested within. */
} Frame;

char MsgOkay[] = { "----- (IFF_OKAY) A good IFF file." };
char MsgEndMark[] = { "----- (END_MARK) How did you get this message??" };
char MsgDone[] = { "----- (IFF_DONE) How did you get this message??" };
char MsgDos[] = { "----- (DOS_ERROR) The DOS gave back an error." };
char MsgNot[] = { "----- (NOT_IFF) not an IFF file." };
char MsgNoFile[] = { "----- (NO_FILE) no such file found." };
char MsgClientError[] = { "----- (CLIENT_ERROR) IFF Checker bug." };
char MsgForm[] = { "----- (BAD_FORM) How did you get this message??" };
char MsgShort[] = { "----- (SHORT_CHUNK) How did you get this message??" };
char MsgBad[] = { "----- (BAD_IFF) a mangled IFF file." };

```

```

/* MUST GET THESE IN RIGHT ORDER!!! */
char *IFFMessages[-LAST_ERROR+1] = {
    /* IFF_OKAY */ MsgOkay,
    /* END_MARK */ MsgEndMark,
    /* IFF_DONE */ MsgDone,
    /* DOS_ERROR */ MsgDos,
    /* NOT_IFF */ MsgNot,
    /* NO_FILE */ MsgNoFile,
    /* CLIENT_ERROR */ MsgClientError,
    /* BAD_FORM */ MsgForm,
    /* SHORT_CHUNK */ MsgShort,
    /* BAD_IFF */ MsgBad
};

/* FORWARD REFERENCES */
extern IFFP GetList /* GroupContext */;
extern IFFP GetForm /* GroupContext */;
extern IFFP GetProp /* GroupContext */;
extern IFFP GetCat /* GroupContext */;

```

```

#####
# $Header: Makefile.v 1.0 85/04/02 19:29:23 kodlak Exp $
# $Locker: $
# $Log: Makefile.v $
#####
MAKEFILE= showilbm.mk
MAKEMETA= /usr/commodore/amiga/V1/tools/makemeta
SRCDIRPATH= clipboard
SRCDIR= iff
SUBSYSNAME= showilbm
DISKPATH= cli/c/showilbm
STARTUP= ${LIBDIR}/startup.obj
PFLAGS= -Plab
MYLIBS= ${LIBDIR}/debug.lib
CFLAGS= '-DDEBUG'
AFILES= movmam.asm
CFILES= showilbm.c iffr.c ilbm.c unpacker.c
OFILES= showilbm.obj iffr.obj ilbm.obj unpacker.obj movmam.obj
all: ${SUBSYSNAME}.ld
INCLUDE=${MAKEMETA}
SYMBOLOPT=

```

```

void iffCheck(name) char *name; {
    IFFP iffp;
    IFFR file = Open(name, MODE_OLDFILE);
    Frames frames;

    frames.levels = 0;
    frames.clientFrames.getList = GetList;
    frames.clientFrames.getForm = GetForm;
    frames.clientFrames.getProp = GetProp;
    frames.clientFrames.getCat = GetCat;

    printf("----- Checking file '%s' -----\n", name);
    if (file == 0)
        iffp = NO_FILE;
    else
        iffp = ReadIFF(file, (ClientFrame *)&frames);

    Close(file);
    printf("%s\n", IFFMessages[-iffp]);
}

main(argc, argv) int argc; char **argv; {
    if (argc != 1+1) {
        printf("Usage: iffcheck filename\n");
        exit(0);
    }
    iffCheck(argv[1]);
}

/* ----- Put... ----- */
PutLevels(count) int count; {
    for (; count > 0; --count) {
        printf(" ");
    }
}

PutID(id) ID id; {
    printf("%c%c%c", (id>>24)&0x7f, (id>>16)&0x7f, (id>>8)&0x7f, id&0x7f);
}

PutN(n) int n; {
    printf(" %d ", n);
}

/* Put something like "...BEID 14" or "...LIST 14 PLBN". */
PutHdr(context) GroupContext *context; {
    PutLevels( ((Frames *)context->clientFrames)->levels );
    PutID(context->ckid.ckid);
    PutN(context->ckid.ckSize);

    if (context->subtype != NULL_CHUNK)
        PutID(context->subtype);

    printf("\n");
}

```

```

}

/* ----- AtLeaf ----- */
/* At Leaf chunk. That is, a chunk which does NOT contain other chunks.
 * Print "ID size". */
IFFP AtLeaf(context) GroupContext *context; {
    PutHdr(context);
    /* A typical reader would read the chunk's contents, using the "Frames"
     * for local data, esp. shared property settings (PROP). */
    /* IFFReadBytes(context, ...buffer, context->ckid->ckSize); */
    return(IFF_OKAY);
}

/* ----- GetList ----- */
/* Handle a LIST chunk. Print "LIST size subTypeID".
 * Then dive into it. */
IFFP GetList(parent) GroupContext *parent; {
    Frames newFrames;

    newFrames = *(Frames *)parent->clientFrames; /* copy parent's frames */
    newFrames.levels++;
    PutHdr(parent);

    return( ReadList(parent, (ClientFrame *)&newFrames) );
}

/* ----- GetForm ----- */
/* Handle a FORM chunk. Print "FORM size subTypeID".
 * Then dive into it. */
IFFP GetForm(parent) GroupContext *parent; {
    /*CompilerBug register*/ IFFP iffp;
    GroupContext new;
    Frames newFrames;

    newFrames = *(Frames *)parent->clientFrames; /* copy parent's frames */
    newFrames.levels++;
    PutHdr(parent);

    iffp = OpenRGroup(parent, &new);
    CheckIFFP();
    new.clientFrames = (ClientFrame *)&newFrames;

    /* FORM reader for Checker. */
    /* LIST, FORM, PROP, CAT already handled by GetF1ChunkHdr. */
    do {if ( ( iffp = GetF1ChunkHdr(&new) ) > 0 )
        iffp = AtLeaf(&new);
        } while (iffp >= IFF_OKAY);

    CloseRGroup(&new);
    return(iffp == END_MARK ? IFF_OKAY : iffp);
}

```



```

/* ----- GetProp ----- */
/* Handle a PROP chunk. Print "PROP size subTypeID".
 * Then dive into it. */
IFFP GetProp(listContext) GroupContext *listContext; {
/*CompilerBug register*/ IFFP iffp;
GroupContext new;

PutHdr(listContext);

iffp = OpenRGoup(listContext, &new);
CheckIFFP();

/* PROP reader for Checker. */
((Frame *)listContext->clientFrame)->levels++;
do {if ((iffp = GetPChunkHdr(&new)) > 0)
    iffp = ALeaf(&new);
    } while (iffp >= IFF_OKAY);

((Frame *)listContext->clientFrame)->levels--;

CloseRGoup(&new);
return(iffp == END_MARK ? IFF_OKAY : iffp);
}

/* ----- GetCat ----- */
/* Handle a CAT chunk. Print "CAT size subTypeID".
 * Then dive into it. */
IFFP GetCat(parent) GroupContext *parent; {
IFFP iffp;

((Frame *)parent->clientFrame)->levels++;
PutHdr(parent);
iffp = ReadICat(parent);

((Frame *)parent->clientFrame)->levels--;
return(iffp);
}

/* ----- OpenRGoup ----- */
/* IFPP OpenRGoup(parent0, new0) GroupContext *parent0, *new0; {
register GroupContext *new = new0;
IFFP iffp = IFF_OKAY;

new->parent = NULL;
new->clientFrame = clientFrame;
new->file = file;
new->position = 0;
new->ckdir.ckID = new->subtype = NULL_CHUNK;
new->ckdir.ckSize = new->bytesSofar = 0;

/* Set new->bound. AmigaDOS specific code. */
if (file <= 0) return(NO_FILE);
Seek(file, 0, OFFSET_END); /* Seek to end of file. */
new->bound = Seek(file, 0, OFFSET_CURRENT); /* Pos'n == #bytes in file. */
if (new->bound < 0) return(DOS_ERROR); /* DOS being absurd. */
Seek(file, 0, OFFSET_BEGINNING); /* Go to file start. */
/* Would just do this if Amiga DOS maintained fh_End: */
/* new->bound = (FileHandle *)BADDR(file)->fh_End; */

if ( new->bound < sizeof(ChunkHeader) )
    iffp = NOT_IFF;
return(iffp);
}

/* ----- OpenRGoup ----- */
/* IFPP OpenRGoup(parent0, new0) GroupContext *parent0, *new0; {
register GroupContext *new = parent0;
IFFP iffp = IFF_OKAY;

new->parent = parent;
new->clientFrame = parent->clientFrame;
new->file = parent->file;
new->position = parent->position;
new->bound = parent->position + ChunkHeaderBytes(parent);
new->ckdir.ckID = new->subtype = NULL_CHUNK;
new->ckdir.ckSize = new->bytesSofar = 0;
}

```

```

/* ----- GetProp ----- */
/* Handle a PROP chunk. Print "PROP size subTypeID".
 * Then dive into it. */
IFFP GetProp(listContext) GroupContext *listContext; {
/*CompilerBug register*/ IFFP iffp;
GroupContext new;

PutHdr(listContext);

iffp = OpenRGoup(listContext, &new);
CheckIFFP();

/* PROP reader for Checker. */
((Frame *)listContext->clientFrame)->levels++;
do {if ((iffp = GetPChunkHdr(&new)) > 0)
    iffp = ALeaf(&new);
    } while (iffp >= IFF_OKAY);

((Frame *)listContext->clientFrame)->levels--;

CloseRGoup(&new);
return(iffp == END_MARK ? IFF_OKAY : iffp);
}

/* ----- GetCat ----- */
/* Handle a CAT chunk. Print "CAT size subTypeID".
 * Then dive into it. */
IFFP GetCat(parent) GroupContext *parent; {
IFFP iffp;

((Frame *)parent->clientFrame)->levels++;
PutHdr(parent);
iffp = ReadICat(parent);

((Frame *)parent->clientFrame)->levels--;
return(iffp);
}

/* ----- OpenRGoup ----- */
/* IFPP OpenRGoup(parent0, new0) GroupContext *parent0, *new0; {
register GroupContext *new = parent0;
IFFP iffp = IFF_OKAY;

new->parent = NULL;
new->clientFrame = clientFrame;
new->file = file;
new->position = 0;
new->ckdir.ckID = new->subtype = NULL_CHUNK;
new->ckdir.ckSize = new->bytesSofar = 0;

/* Set new->bound. AmigaDOS specific code. */
if (file <= 0) return(NO_FILE);
Seek(file, 0, OFFSET_END); /* Seek to end of file. */
new->bound = Seek(file, 0, OFFSET_CURRENT); /* Pos'n == #bytes in file. */
if (new->bound < 0) return(DOS_ERROR); /* DOS being absurd. */
Seek(file, 0, OFFSET_BEGINNING); /* Go to file start. */
/* Would just do this if Amiga DOS maintained fh_End: */
/* new->bound = (FileHandle *)BADDR(file)->fh_End; */

if ( new->bound < sizeof(ChunkHeader) )
    iffp = NOT_IFF;
return(iffp);
}

/* ----- OpenRGoup ----- */
/* IFPP OpenRGoup(parent0, new0) GroupContext *parent0, *new0; {
register GroupContext *new = parent0;
IFFP iffp = IFF_OKAY;

new->parent = parent;
new->clientFrame = parent->clientFrame;
new->file = parent->file;
new->position = parent->position;
new->bound = parent->position + ChunkHeaderBytes(parent);
new->ckdir.ckID = new->subtype = NULL_CHUNK;
new->ckdir.ckSize = new->bytesSofar = 0;
}

```

```

if ( new->bound > parent->bound || IS_ODD(new->bound) )
  iffp = BAD_IFF;
return(iffp);
}

/* ----- CloseGroup GroupContext *context; {
register LONG position;

if (context->parent == NULL) {
} /* Context for whole file.*/
else {
  position = context->position;
  context->parent->bytesSofar += position - context->parent->position;
  context->parent->position = position;
}
return(IFF_OKAY);
}

/* ----- SkipFwd ----- */
/* Skip over bytes in a context. Won't go backwards.*/
/* Updates context->position but not context->bytesSofar.*/
/* This implementation is AmigaDOS specific.*/
IFFP SkipFwd(context, bytes) GroupContext *context; LONG bytes; {
  IFFP iffp = IFF_OKAY;

  if (bytes > 0) {
    if (-1 == Seek(context->file, bytes, OFFSET_CURRENT))
      iffp = BAD_IFF; /* Ran out of bytes before chunk complete.*/
    else
      context->position += bytes;
  }
  return(iffp);
}

/* ----- GetChunkHdr ----- */
ID GetChunkHdr(context0) GroupContext *context0; {
  register GroupContext *context = context0;
  register IFFP iffp;
  LONG remaining;

  /* Skip remainder of previous chunk & padding. */
  iffp = SkipFwd(context,
    ChunkForeBytes(context) + IS_ODD(context->ckHdr.ckSize));
  CheckIFFP();

  /* Set up to read the new header. */
  context->ckHdr.ckID = BAD_IFF;
  context->subtype = NULL_CHUNK;
  context->bytesSofar = 0;

  /* Generate a pseudo-chunk if at end-of-context. */
  if (remaining == 0) {
    context->ckHdr.ckSize = 0;

```

```

context->ckHdr.ckID = END_MARK;
}

/* BAD_IFF if not enough bytes in the context for a ChunkHeader.*/
else if (sizeof(ChunkHeader) > remaining) {
  context->ckHdr.ckSize = remaining;
}

/* Read the chunk header (finally). */
else {
  switch (Read(context->file, &context->ckHdr, sizeof(ChunkHeader))) {
    case -1: return(context->ckHdr.ckID = DOS_ERROR);
    case 0: return(context->ckHdr.ckID = BAD_IFF);
  }

  /* Check: Top level chunk must be LIST or FORM or CAT. */
  if (context->parent == NULL)
    switch(context->ckHdr.ckID) {
      case LIST: case FORM: case CAT: break;
      default: return(context->ckHdr.ckID = NOT_IFF);
    }

  /* Update the context. */
  context->position += sizeof(ChunkHeader);
  remaining -= sizeof(ChunkHeader);

  /* Non-positive ID values are illegal and used for error codes.*/
  /* We could check for other illegal IDs...*/
  if (context->ckHdr.ckID <= 0)
    context->ckHdr.ckID = BAD_IFF;

  /* Check: ckSize negative or larger than # bytes left in context? */
  else if (context->ckHdr.ckSize < 0 ||
    context->ckHdr.ckSize > remaining) {
    context->ckHdr.ckSize = remaining;
    context->ckHdr.ckID = BAD_IFF;
  }

  /* Automatically read the LIST, FORM, PROP, or CAT subtype ID */
  else switch (context->ckHdr.ckID) {
    case LIST: case FORM: case PROP: case CAT: {
      iffp = IFFReadBytes(context,
        (BYTE *)&context->subtype,
        sizeof(ID));
      if (iffp != IFF_OKAY)
        context->ckHdr.ckID = iffp;
      break;
    }
  }

  return(context->ckHdr.ckID);
}

/* ----- IFFReadBytes ----- */
IFFP IFFReadBytes(context, buffer, nBytes)
GroupContext *context; BYTE *buffer; LONG nBytes; {

```

```

register IFFP iffp = IFF_OKAY;

if (nBytes < 0)
    iffp = CLIENT_ERROR;
else if (nBytes > ChunkMoreBytes (context))
    iffp = SHORT_CHUNK;
else if (nBytes > 0)
    switch ( Read (context->file, buffer, nBytes) ) {
        case -1: {iffp = DOS_ERROR; break;}
        case 0: {iffp = BAD_IFF; break;}
        default: {
            context->position += nBytes;
            context->bytesSofar += nBytes;
        }
    }
return (iffp);
}

/* ----- SkipGroup ----- */
IFFP SkipGroup (context) GroupContext *context; {
    /* Nothing to do, thanks to GetChunkHdr */
}

/* ----- ReadIFF ----- */
IFFP ReadIFF (file, clientFrame) BPTR file; ClientFrame *clientFrame; {
    /* CompilerBug register*/ IFFP iffp;
    GroupContext context;

    iffp = OpenRIFF (file, &context);
    context.clientFrame = clientFrame;

    if (iffp == IFF_OKAY)
        switch (iffp = GetChunkHdr (&context)) {
            case FORM: { iffp = (*clientFrame->getForm) (&context); break; }
            case LIST: { iffp = (*clientFrame->getList) (&context); break; }
            case CAT: { iffp = (*clientFrame->getCat) (&context); break; }
            /* default: Includes IFF_DONE, BAD_IFF, NOT_IFF... */
        }
    CloseGroup (&context);

    if (iffp > 0)
        iffp = NOT_IFF;
    return (iffp);
}

/* ----- ReadList ----- */
IFFP ReadList (parent, clientFrame)
GroupContext *parent; ClientFrame *clientFrame; {
    GroupContext listContext;
    IFFP iffp;
    BOOL propOk = TRUE;

    iffp = OpenGroup (parent, &listContext);
    CheckIFFP ();
}

```

```

/* One special case test lets us handle CATs as well as LISTs. */
if (parent->ckHdr.chkID == CAT)
    propOk = FALSE;
else
    listContext.clientFrame = clientFrame;

do {
    switch (iffp = GetChunkHdr (&listContext)) {
        case PROP: {
            if (propOk)
                iffp = (*clientFrame->getProp) (&listContext);
            else
                iffp = BAD_IFF;
            break;
        }
        case FORM: { iffp = (*clientFrame->getForm) (&listContext); break; }
        case LIST: { iffp = (*clientFrame->getList) (&listContext); break; }
        case CAT: { iffp = (*clientFrame->getCat) (&listContext); break; }
        /* default: Includes END_MARK, IFF_DONE, BAD_IFF, NOT_IFF... */
    }
    if (!listContext.ckHdr.chkID != PROP)
        propOk = FALSE; /* No PROPs allowed after this point. */
    } while (iffp == IFF_OKAY);

CloseGroup (&listContext);

if (iffp > 0) /* Only chunk types above are allowed in a LIST/CAT. */
    iffp = BAD_IFF;
return (iffp == END_MARK ? IFF_OKAY : iffp);
}

/* ----- ReadCat ----- */
/* By special arrangement with the ReadList implement'n, this is trivial. */
IFFP ReadCat (parent) GroupContext *parent; {
    return ( ReadList (parent, NULL) );
}

/* ----- GetChunkHdr ----- */
ID GetChunkHdr (context) GroupContext *context; {
    register ID id;

    id = GetChunkHdr (context);
    if (id == PROP)
        context->ckHdr.chkID = id = BAD_IFF;
    return (id);
}

/* ----- GetFChunkHdr ----- */
ID GetFChunkHdr (context) GroupContext *context; {
    register ID id;

    id = GetFChunkHdr (context);
    register ClientFrame *clientFrame = context->clientFrame;
    switch (id = GetChunkHdr (context)) {
        case PROP: { id = BAD_IFF; break; }
        case FORM: { id = (*clientFrame->getForm) (context); break; }
        case LIST: { id = (*clientFrame->getList) (context); break; }
    }
}

```

```

case CAT : { id = (*clientFrame->getCat) (context); break; }
/* Default: let the caller handle other chunks */
return (context->ckHdr.ckID = id);
}

/* ----- GetPChunkHdr ----- */
ID GetPChunkHdr (context) GroupContext *context; {
register ID id;

id = GetChunkHdr (context);
switch (id) {
case LIST: case FORM: case PROP: case CAT: {
id = context->ckHdr.ckID = BAD_IFF;
break; }
}
return (id);
}

```

```

/* ----- Support routines for writing IFF-85 files. ----- */
/* (IFF is Interchange Format File.) ----- 11/11/85 */
* By Jerry Morrison and Steve Shaw, Electronic Arts.
* This software is in the public domain.
* This version for the Commodore-Amiga computer.
#include "exec/types.h"
#include "libraries/dos.h"
#include "iff.h"
/* ----- IFF Writer ----- */
/* A macro to test if a chunk size is definite, i.e. not szNotYetKnown. */
#define Known(size) ((size) != szNotYetKnown)
/* Yet another weird macro to make the source code simpler... */
#define Ifffp(expr) {if (iffp == IFF_OKAY) iffp = (expr);}

/* ----- OpenMIEF ----- */
IEFP OpenMIEF (file, new0, limit) PTR file; GroupContext *new0; LONG limit; {
register GroupContext *new = new0;
register IFF iffp = IFF_OKAY;

new->parent = NULL;
new->clientFrame = NULL;
new->file = file;
new->position = 0;
new->bound = limit;
new->ckHdr.ckID = NULL_CHUNK; /* Indicates no current chunk */
new->ckHdr.ckSize = new->bytesSofar = 0;

if (0 > Seek (file, 0, OFFSET_BEGINNING)) /* Go to start of the file. */
iffp = DOS_ERROR;
else if (Known (limit) && IS_ODD (limit))
iffp = CLIENT_ERROR;
return (iffp);
}

/* ----- StartMGroup ----- */
IEFP StartMGroup (parent, groupType, groupSize, subtype, new)
GroupContext *parent, *new; ID groupType, subtype; LONG groupSize; {
register IFF iffp;

iffp = PutCkHdr (parent, groupType, groupSize);
Ifffp ( IFFWriteBytes (parent, (BYTE *) &subtype, sizeof (ID)) );
Ifffp ( OpenMGroup (parent, new) );
return (iffp);
}

/* ----- OpenMGroup ----- */
IEFP OpenMGroup (parent0, new0) GroupContext *parent0, *new0; {
register GroupContext *parent = parent0;

```

```

register GroupContext *new = new0;
register LONG ckEnd;
register IFFP iffp = IFF_OKAY;

new->parent = parent;
new->clientFrams = parent->clientFrams;
new->file = parent->file;
new->position = parent->position;
new->bound = parent->bound;
new->ckID = NULL_CHUNK;
new->ckHdr.ckSize = new->bytesSofar = 0;

if ( Known(parent->ckHdr.ckSize) ) {
    ckEnd = new->position + ChunkMoreBytes(parent);
    if ( new->bound == szNotYetKnown || new->bound > ckEnd )
        new->bound = ckEnd;
};

if ( parent->ckHdr.ckID == NULL_CHUNK || /* not currently writing a chunk */
    IS_ODD(new->position) ||
    (Known(new->bound) && IS_ODD(new->bound)) )
    iffp = CLIENT_ERROR;
return(iffp);
}

/* ----- CloseGroup ----- */
IFFP CloseGroup(oldd) GroupContext *oldd; {
register GroupContext *old = old0;

if ( old->ckHdr.ckID != NULL_CHUNK ) /* didn't close the last chunk */
    return(CLIENT_ERROR);
if ( old->parent == NULL ) {
    /* [TBD] set logical EOF */
}
else {
    old->parent->bytesSofar += old->position - old->parent->position;
    old->parent->position = old->position;
};
return( IFF_OKAY );
}

/* ----- EndGroup ----- */
IFFP EndGroup(oldd) GroupContext *oldd; {
register GroupContext *parent = old->parent;
register IFFP iffp;

iffp = CloseGroup(oldd);
Iffp( PutCkEnd(parent) );
return(iffp);
}

/* ----- PutCk ----- */
IFFP PutCk(context, ckID, ckSize, data)
GroupContext *context; ID ckID; LONG ckSize; BYTE *data; {
register IFFP iffp = IFF_OKAY;

```

```

if ( ckSize == szNotYetKnown )
    iffp = CLIENT_ERROR;
Iffp( PutCkHdr(context, ckID, ckSize) );
Iffp( IFFWriteBytes(context, data, ckSize) );
Iffp( PutCkEnd(context) );
return(iffp);
}

/* ----- PutCkHdr ----- */
IFFP PutCkHdr(context0, ckID, ckSize)
GroupContext *context0; ID ckID; LONG ckSize; {
register GroupContext *context = context0;
LONG minPSize = sizeof(ChunkHeader); /* physical chunk >= minPSize bytes */

/* CLIENT_ERROR if we're already inside a chunk or asked to write
 * other than one FORM, LIST, or CAT at the top level of a file */
/* Also, non-positive ID values are illegal and used for error codes. */
/* (We could check for other illegal IDs...) */
if ( context->ckHdr.ckID != NULL_CHUNK || ckID <= 0 )
    return(CLIENT_ERROR);
else if ( context->parent == NULL ) {
    switch (ckID) {
        case FORM: case LIST: case CAT: break;
        default: return(CLIENT_ERROR);
    }
    if ( context->position != 0 )
        return(CLIENT_ERROR);
}
if ( Known(ckSize) ) {
    if ( ckSize < 0 )
        return(CLIENT_ERROR);
    minPSize += ckSize;
};
if ( Known(context->bound) &&
    context->position + minPSize > context->bound )
    return(CLIENT_ERROR);

context->ckHdr.ckID = ckID;
context->ckHdr.ckSize = ckSize;
context->bytesSofar = 0;
if ( 0 > Write(context->file, &context->ckHdr, sizeof(ChunkHeader)) )
    return(DOS_ERROR);
context->position += sizeof(ChunkHeader);
return( IFF_OKAY );
}

/* ----- IFFWriteBytes ----- */
IFFP IFFWriteBytes(context0, data, nBytes)
GroupContext *context0; BYTE *data; LONG nBytes; {
register GroupContext *context = context0;

if ( context->ckHdr.ckID == NULL_CHUNK || /* not in a chunk */
    nBytes < 0 || /* negative nBytes */
    (Known(context->bound) &&
    context->position + nBytes > context->bound) ||

```

```

(Known(context->ckHdr.ckSize) && /* overflow chunk */
 context->bytesSoFar + nBytes > context->ckHdr.ckSize) )
return(CLIENT_ERROR);

if ( 0 > Write(context->file, data, nBytes) )
return(DOS_ERROR);

context->bytesSoFar += nBytes;
context->position += nBytes;
return(IEFF_OKAY);
}

/* ----- PutCrLf ----- */
IEFF PutCrLf(context0) GroupContext *context0; {
register GroupContext *context = context0;
WORD zero = 0; /* padding source */

if ( context->ckHdr.ckID == NULL_CHUNK ) /* not in a chunk */
return(CLIENT_ERROR);

if ( context->ckHdr.ckSize == szNotYetKnown ) {
/* go back and set the chunk size to bytesSoFar */
if ( 0 > Seek(context->file,
-(context->bytesSoFar + sizeof(LONG)),
OFFSET_CURRENT) ||
0 > Write(context->file, &context->bytesSoFar, sizeof(LONG)) ||
0 > Seek(context->file, context->bytesSoFar, OFFSET_CURRENT) )
return(DOS_ERROR);
}
else { /* make sure the client wrote as many bytes as planned */
if ( context->ckHdr.ckSize != context->bytesSoFar )
return(CLIENT_ERROR);
};

/* Write a pad byte if needed to bring us up to an even boundary.
* Since the context end must be even, and since we haven't
* overwritten the context, if we're on an odd position there must
* be room for a pad byte. */
if ( IS_ODD(context->bytesSoFar) ) {
if ( 0 > Write(context->file, &zero, 1) )
return(DOS_ERROR);
context->position += 1;
};

context->ckHdr.ckID = NULL_CHUNK;
context->ckHdr.ckSize = context->bytesSoFar = 0;
return(IEFF_OKAY);
}

```

```

/* ----- Support routines for reading ILEB files. ----- */
/* (IEFF is Interchange Format File.) ----- 11/11/85 ----- */
* By Jerry Morrison and Steve Shaw, Electronic Arts.
* This software is in the public domain.
* This version for the Commodore-Amiga computer.
#include "exec/types.h"
#include "packer.h"
#include "ilbm.h"

/* ----- GetCMAP ----- */
/* pNColorRegs is passed in as a pointer to the number of ColorRegisters
* caller has space to hold. GetCMAP sets to the number actually read. */
IEFF GetCMAP(ilbmContext, colorMap, pNColorRegs)
GroupContext *ilbmContext; WORD *colorMap; UBYTE *pNColorRegs;
{
register int nColorRegs;
register IFF iff;
ColorRegister colorReg;

nColorRegs = ilbmContext->ckHdr.ckSize / sizeof(ColorRegister);
if (*pNColorRegs < nColorRegs) nColorRegs = *pNColorRegs;
*pNColorRegs = nColorRegs; /* Set to the number actually there. */

for ( ; nColorRegs > 0; --nColorRegs) {
iff = IEFFReadBytes(ilbmContext, (BYTE *)&colorReg, sizeof(ColorRegister));
CheckIEFF(iff);
*colorMap++ = ( ( colorReg.red >> 4 ) << 8 ) |
( ( colorReg.green >> 4 ) << 4 ) |
( ( colorReg.blue >> 4 ) );
}
return(IEFF_OKAY);
}

/* ----- GetBODY ----- */
/* NOTE: This implementation could be a LOT faster if it used more of the
* supplied buffer. It would make far fewer calls to IEFFReadBytes (and
* therefore to DOS Read) and to movemem. */
IEFF GetBODY(context, bitMap, mask, bhHdr, buffer, bufferSize)
GroupContext *context; struct BitMap *bitMap; BYTE *mask;
BitMapHeader *bhHdr; BYTE *buffer; LONG bufferSize;
{
register IFF iff;
UBYTE srcPlaneCnt = bhHdr->nPlanes; /* Haven't counted for mask plane yet */
LONG srcRowBytes = RowBytes(bhHdr->w);
LONG bufRowBytes = MaxPackedSize(srcRowBytes);
int nRows = bhHdr->h;
Compression compression = bhHdr->compression;
register int iPlane, iRow, nEmpty, nFilled;
BYTE *buf, *nullDest, *nullBuf, *pDest;
BYTE *planes[MaxSrcPlanes]; /* array of ptrs to planes & mask */
}

```



```

/*-----
* ILEBMW.C Support routines for writing ILEM files. 11/11/85
* (IFF is Interchange Format File.)
* By Jerry Morrison and Steve Shaw, Electronic Arts.
* This software is in the public domain.
* This version for the Commodore-Amiga computer.
#include "exec/types.h"
#include "packer.h"
#include "ilbm.h"
/*----- InitBMHdr
IFFP InitBMHdr (bmHdr, bitmap, masking, compression, transparentColor,
pageWidth, pageHeight)
BitMapHeader *bmHdr; struct BitMap *bitmap;
int masking, compression, transparentColor, pageWidth, pageHeight;
/* Masking, Compression, UWORD, WORD */
{
register BitMapHeader *bmHdr = bmHdr;
register WORD rowBytes = bitmap->BytesPerRow;
bmHdr->w = rowBytes << 3;
bmHdr->h = bitmap->Rows;
bmHdr->x = bmHdr->y = 0; /* Default position is (0,0). */
bmHdr->nPlanes = bitmap->Depth;
bmHdr->masking = masking;
bmHdr->compression = compression;
bmHdr->pad1 = 0;
bmHdr->transparentColor = transparentColor;
bmHdr->xAspect = bmHdr->yAspect = 1;
bmHdr->pageWidth = pageWidth;
bmHdr->pageHeight = pageHeight;
if (pageWidth = 320)
switch (pageHeight) {
case 200: {bmHdr->xAspect = x320x200Aspect;
bmHdr->yAspect = y320x200Aspect; break;}
case 400: {bmHdr->xAspect = x320x400Aspect;
bmHdr->yAspect = y320x400Aspect; break;}
}
else if (pageWidth = 640)
switch (pageHeight) {
case 200: {bmHdr->xAspect = x640x200Aspect;
bmHdr->yAspect = y640x200Aspect; break;}
case 400: {bmHdr->xAspect = x640x400Aspect;
bmHdr->yAspect = y640x400Aspect; break;}
}
return( IS_ODD(rowBytes) ? CLIENT_ERROR : IFF_OKAY );
}
/*----- PutCMap
IFFP PutCMap (context, colorMap, depth)
GroupContext *context; WORD *colorMap; UBYTE depth;
*/

```

```

{
register LONG nColorRegs;
IFFP iffp;
ColorRegister colorReg;
if (depth > MaxAmDepth) depth = MaxAmDepth;
nColorRegs = 1 << depth;
ifp = PutCMap (context, ID_CMAP, nColorRegs * sizeofColorRegister);
CheckIFFP ();
for ( ; nColorRegs; --nColorRegs) {
colorReg.red = ( *colorMap >> 4 ) & 0xf0;
colorReg.green = ( *colorMap >> 4 ) & 0xf0;
colorReg.blue = ( *colorMap << 4 ) & 0xf0;
ifp = IFFWriteBytes (context, (BYTE *) & colorReg, sizeofColorRegister);
CheckIFFP ();
++colorMap;
}
ifp = PutCMapEnd (context);
return (iffp);
}
/*----- PutBODY
/* NOTE: This implementation could be a LOT faster if it used more of the
* supplied buffer. It would make far fewer calls to IFFWriteBytes (and
* therefore to DOS Write). */
IFFP PutBODY (context, bitmap, mask, bmHdr, buffer, bufferSize)
GroupContext *context; struct BitMap *bitmap; BYTE *mask;
BitMapHeader *bmHdr; BYTE *buffer; LONG bufferSize;
{
IFFP iffp;
LONG rowBytes = bitmap->BytesPerRow;
int dstDepth = bmHdr->nPlanes;
Compression compression = bmHdr->compression;
int planeCnt; /* number of bit planes including mask */
register int iPlane, iRow;
register LONG packedRowBytes;
BYTE *buf;
BYTE *planes [MaxAmDepth + 1]; /* array of ptrs to planes & mask */
if ( bufferSize < MaxPackedSize (rowBytes) || /* Must buffer a comprsd row */
compression > cmpByteRun1 || /* bad arg */
bitmap->Rows != bmHdr->h || /* inconsistent */
rowBytes != RowBytes (bmHdr->w) || /* inconsistent */
bitmap->Depth < dstDepth || /* inconsistent */
dstDepth > MaxAmDepth)
return (CLIENT_ERROR);
planeCnt = dstDepth * (mask == NULL ? 0 : 1);
/* Copy the ptrs to bit & mask planes into local array "planes" */
for (iPlane = 0; iPlane < dstDepth; iPlane++)
planes [iPlane] = (BYTE *) bitmap->Planes [iPlane];
if (mask != NULL)

```



```

planes[dstDepth] = mask;

/* Write out a BODY chunk header */
iffp = PutCkHdr(context, ID_BODY, szNotYetKnown);
CheckIFFP();

/* Write out the BODY contents */
for (iRow = hHdr->h; iRow > 0; iRow--) {
    for (iPlane = 0; iPlane < planesCnt; iPlane++) {
        /* Write next row.*/
        if (compression == cmpNone) {
            iffp = IFFWriteBytes(context, planes[iPlane], rowBytes);
            planes[iPlane] += rowBytes;
        }
        /* Compress and write next row.*/
        else {
            buf = buffer;
            packedRowBytes = PackRow(spplanes[iPlane], &buf, rowBytes);
            iffp = IFFWriteBytes(context, buffer, packedRowBytes);
        }
    }
    CheckIFFP();
}

/* Finish the chunk */
iffp = PutCkEnd(context);
return (iffp);
}

planes[dstDepth] = mask;

/* packer.c Convert data to "cmpByteRun1" run compression. 11/11/85
* By Jerry Morrison and Steve Shaw, Electronic Arts.
* This software is in the public domain.
*
* control bytes:
* [0..127] : followed by n+1 bytes of data.
* [-1..-127] : followed by byte to be repeated (-n)+1 times.
* -128 : NOOP.
*
* This version for the Commodore-Amiga computer.
*
*#include "exec/types.h"
*#include "packer.h"
*#define DUMP 0
*#define RUN 1
*#define MinRun 3
*#define MaxRun 128
*#define MaxDat 128
LONG putSize;
#define GetByte() (*source++)
#define PutByte(c) { *dest++ = (c); ++putSize; }
char buf[256]; /* [TRD] should be 128? on stack?*/
BYTE *PutDump(dest, nn) BYTE *dest; int nn; {
    int i;
    PutByte(nn-1);
    for (i = 0; i < nn; i++) PutByte(buf[i]);
    return(dest);
}
BYTE *PutRun(dest, nn, cc) BYTE *dest; int nn, cc; {
    PutByte(-(nn-1));
    PutByte(cc);
    return(dest);
}
#define OutDump(m) dest = PutDump(dest, nn)
#define OutRun(mn, cc) dest = PutRun(dest, nn, cc)
/*----- PackRow -----*/
/* Given POINTERS TO POINTERS, packs one row, updating the source and
destination pointers. RETURNS count of packed bytes.*/
LONG PackRow(pSource, pDest, rowSize)
BYTE *pSource, *pDest; LONG rowSize; {
    BYTE *source, *dest;
    char c, lastc = '\0';
    BOOL mode = DUMP;
    short rbuf = 0;
    short rstart = 0;
    /* number of chars in buffer */
    /* buffer index current run starts */

```

```

source = *pSource;
dest = *pDest;
putSize = 0;
buf[0] = lastc = c = GetByte(); /* so have valid lastc */
nbuf = 1; rowSize--; /* since one byte eaten.*/

for (; rowSize; --rowSize) {
    buf[nbuf++] = c = GetByte();
    switch (mode) {
        case DUMP:
            /* If the buffer is full, write the length byte,
            then the data */
            if (nbuf > MaxDat) {
                OutDump(nbuf-1);
                buf[0] = c;
                nbuf = 1; restart = 0;
                break;
            }
            if (c == lastc) {
                if (nbuf-restart >= MinRun) {
                    if (restart > 0) OutDump(rstart);
                    mode = RUN;
                }
                else if (restart == 0)
                    mode = RUN; /* no dump in progress,
                    so can't lose by making these 2 a run.*/
            }
            else restart = nbuf-1; /* first of run */
            break;
        case RUN: if ((c != lastc) || (nbuf-restart > MaxRun)) {
            /* output run */
            OutRun(nbuf-1-restart, lastc);
            buf[0] = c;
            nbuf = 1; restart = 0;
            mode = DUMP;
        }
        break;
    }
    lastc = c;
}

switch (mode) {
    case DUMP: OutDump(nbuf); break;
    case RUN: OutRun(nbuf-restart, lastc); break;
}

*pSource = source;
*pDest = dest;
return (putSize);
}

/** Raw2ILEM.c *****/
* Read an raw raster image file and write an IFF FORM ILEM file. 11/12/85.
* By Jerry Morrison and Steve Shaw, Electronic Arts.
* This software is in the public domain.
* USE THIS AS AN EXAMPLE PROGRAM FOR AN IFF WRITER.
* *****/
#include "exec/types.h"
#include "graphics/gfx.h"
#include "ilbm.h"

/* Size of the buffer for PutBODY. */
#define bufSize 512

/** PutAnILEM() *****/
* Write an entire BitMap as a FORM ILEM in an IFF file.
* This procedure assumes the image is in the Amiga's 320 x 200 display mode.
* Normal return result is IFF_OKAY.
* The utility program IFFCheck would print the following outline of the
* resulting file:
* FORM ILEM
*  BMAP
*  CMAP
*  BODY (compressed)
* *****/
#define CkErr(expression) {if (ifferr == IFF_OKAY) ifferr = (expression);}

IFF PutAnILEM(file, bitmap, mask, colorMap, depth, xy, buffer, bufSize)
LONG file; struct BitMap *bitmap; BYTE *mask; WORD *colorMap;
UBYTE depth; Point2D *xy; BYTE *buffer; LONG bufSize;
{
    BitMapHeader bmapHdr;
    GroupContext fileContext, formContext;
    IFF ifferr;

    ifferr = InitBMAP(&bmapHdr, bitmap, maskNone, cmpByteRun1, 0, 320, 200);
    /* You could write an uncompressed image by passing cmpNone instead
    * of cmpByteRun1 to InitBMAP. */
    bmapHdr.nPlanes = depth; /* This must be <= bitmap->Depth */
    if (mask != NULL) bmapHdr.masking = maskHasMask;
    bmapHdr.x = xy->x; bmapHdr.y = xy->y;

    CkErr ( OpenIFF (file, &fileContext, szNotYetKnown) );
    CkErr ( StartWGroup (&fileContext, FORM, szNotYetKnown, ID_ILEM, &formContext) );
    CkErr { PutBMAP (&formContext, &bmapHdr) };
    CkErr { PutCMAP (&formContext, colorMap, depth) };
}

```

```

CkErr( PutBODY(&formContext, bitmap, mask, &mdir, buffer, bufsize) );
CkErr( EndGroup(&formContext) );
CkErr( CloseGroup(&fileContext) );
return( ifferr );
}

/** PutPicture() *****/
* Put a picture into an IFF file.
* This procedure calls PutAnILBM, passing in an <x, y> location of <0, 0>.
* a NULL mask, and a locally-allocated buffer. It also assumes you want to
* write out all the bitplanes in the BitMap.
* *****/
Point2D nullPoint = {0, 0};
IFF PutPicture(file, bitmap, colorMap;
LONG file; struct BitMap *bitmap; WORD *colorMap;
{
BYTE buffer [bufSize];
return( PutAnILBM(file, bitmap, NULL,
colorMap, bitmap->Depth, &nullPoint,
buffer, bufSize) );
}

```

```

/** ShowILBM.c *****/
* Read an ILBM raster image file and display it. 11/12/85.
* By Jerry Morrison, Steve Shaw, and Steve Hayes, Electronic Arts.
* This software is in the public domain.
* USE THIS AS AN EXAMPLE PROGRAM FOR AN IFF READER.
* The IFF reader portion is essentially a recursive-descent parser.
* This program will look into a CAT or LIST to find a FORM ILBM, but it
* won't look inside another FORM type for a nested FORM ILBM.
* The display portion is specific to the Commodore Amiga computer.
* NOTE: This program displays an image, pauses, then exits. It doesn't
* bother to switch you back to the CLI or workbench "screen", so type
* Amiga-N when it's done.
* *****/
#include "exec/types.h"
#include "exec/memory.h"
#include "libraries/dos.h"
#include "graphics/gfxbase.h"
#include "graphics/rastport.h"
#include "graphics/gfx.h"
#include "graphics/view.h"
#include "ilbm.h"

/* This example's max number of planes in a bitmap. Could use MaxAmDepth. */
#define ENDdepth 5
#define maxColorReg (1<ENDdepth)
#define MIN(a,b) ((a)<(b)?(a):(b))

#define SafeFreeMem(p,q) (if(p)FreeMem(p,q) ;)

/* general usage pointers */
struct GfxBase *GfxBase;

/* Globals for displaying an image */
struct RastPort rP;
struct BitMap bitmap;
struct RasInfo rasInfo;
struct View v;
struct ViewPort vp;

/* Define the size of a temporary buffer used in unscrambling the ILBM rows.*/
#define bufSz 512

/* Message strings for IFF codes. */
char MsgOkay[] = { " (IFF_OKAY) No FORM ILBM in the file." };
char MsgEndMark[] = { " (END_MARK) How did you get this message?" };
char MsgDone[] = { " (IFF_DONE) All done." };
char MsgDos[] = { " (DOS_ERROR) The DOS returned an error." };
char MsgNot[] = { " (NOT_IFF) Not an IFF file." };

```

```

char MsgNoFile[] = { "NO_FILE No such file found." };
char MsgClientError[] = { "CLIENT_ERROR ShowILBM bug or insufficient RAM." };
char MsgForm[] = { "BAD_FORM A malformed FORM ILEM." };
char MsgShort[] = { "SHORT_CHUNK A malformed FORM ILEM." };
char MsgBad[] = { "BAD_IFF A mangled IFF file." };

/* THESE MUST APPEAR IN RIGHT ORDER!! */
char *IFFMessages[-LAST_ERROR+1] = {
    /* IFF OKAY*/ MsgOkay,
    /* END_MARK*/ MsgEndMark,
    /* IFF_DONE*/ MsgDone,
    /* DOS_ERROR*/ MsgDos,
    /* NOT_IFF*/ MsgNot,
    /* NO_FILE*/ MsgNoFile,
    /* CLIENT_ERROR*/ MsgClientError,
    /* BAD_FORM*/ MsgForm,
    /* SHORT_CHUNK*/ MsgShort,
    /* BAD_IFF*/ MsgBad
};

```

```

/*----- ILEM reader -----*/
/* ILEMFrames is our "client frames" for reading FORMs ILEM in an IFF file.
 * We allocate one of these on the stack for every LIST or FORM encountered
 * in the file and use it to hold BMAP & CMAP properties. We also allocate
 * an initial one for the whole file.
 * We allocate a new GroupContext (and initialize it by OpenRIFF or
 * OpenRCGroup) for every group (FORM, CAT, LIST, or PRCP) encountered. It's
 * just a context for reading (nested) chunks.
 *
 * If we were to scan the entire example file outlined below:
 *
 * reading proc(s) new new
 *
 * --whole file-- ReadPicture+ReadIFF GroupContext ILEMFrames
 * CAT ReadCat GroupContext
 * LIST GetLiILEM+ReadILList GroupContext ILEMFrames
 * PRCP ILEM GetPrILEM GroupContext
 * CMAP GetCMAP
 * BMAP GetBMAP
 * FORM ILEM GetFoILEM ILEMFrames
 * BODY GetBODY
 * FORM ILEM GetFoILEM ILEMFrames
 * BODY GetBODY
 * FORM ILEM GetFoILEM ILEMFrames
 *
typedef struct {
    ClientFrames clientFrame;
    UBYTE foundBMAP;
    UBYTE nColorRegs;
    BitMapHeader bmapHdr;
    Color4 colorMap[maxColorReg];
    /* If you want to read any other property chunks, e.g. GRAB or CAMC, add
     * fields to this record to store them. */
} ILEMFrames;

/* NOTE: For a simple version of this program, set Fancy to 0.

```

```

 * That'll compile a program that skips all LISTS and PROPs in the input
 * file. It will look in CATs for FORMs ILEM. That's suitable for most uses.
 *
 * For a fancy version that handles LISTS and PROPs, set Fancy to 1. */
#define Fancy 1

/* DisplayPic() ***** */
/* Interface to Amiga graphics ROM routines.
 * ***** */
DisplayPic(ptilbmFrame)
    ILEMFrames *ptilbmFrame;
{
    int i;

    InitView(&v);
    v.ViewPort = &vp;
    InitRastPort(&rp);
    rp.BitMap = &bitMap;
    rasInfo.BitMap = &bitMap;

    /* Always show the upper left-hand corner of this picture. */
    rasInfo.RxOffset = 0;
    rasInfo.RyOffset = 0;

    InitVPort(&vvp);
    vp.DWidth = ptilbmFrame->bmapHdr.pageWidth; /* Physical display WIDTH */
    vp.DHeight = ptilbmFrame->bmapHdr.pageHeight; /* Display height */

    #if 0
    /* Specify where on screen to put the ViewPort. */
    vp.DxOffset = ptilbmFrame->bmapHdr.x;
    vp.DyOffset = ptilbmFrame->bmapHdr.y;
    #else
    /* Always display it in upper left corner of screen. */
    #endif

    if (ptilbmFrame->bmapHdr.pageWidth <= 320)
        vp.Modes = 0;
    else vp.Modes = HIRES;
    if (ptilbmFrame->bmapHdr.pageHeight > 200)
        vp.Modes |= LACE; /* mod by rob peck 12/3, (missed!) */

    MakeVPort(&vvp, &rasInfo);
    MrGCop(&v);
    LoadView(&v);
    WaitBlit();
    WaitTOF();
    LoadRCBA(&vvp, ptilbmFrame->colorMap, ptilbmFrame->nColorRegs);

    for (i = 0; i < 5*60; ++i) WaitTOF(); /* Delay 5 seconds. */
    /* NOTE: We should switch back to the CLI screen here... */
}

```

```

/* GetFoilem() *****
 * Called via ReadPicture to handle every FORM encountered in an IFF file.
 * Reads FORMs ILEM and skips all others.
 * Inside a FORM ILEM, it stops once it reads a BODY. It complains if it
 * finds no BODY or if it has no BEMHD to decode the BODY.
 *
 * Once we find a BODY chunk, we'll allocate the BitMap and read the image.
 *
 * *****
 IFF GetFoilem(parent) GroupContext *parent; {
 /*compilerBug register*/ IFFP iffp;
 GroupContext formContext;
 ILEmGroup ilbmFrames; /* only used for non-clientFrame fields.*/
 register int i;
 int psize; /* Plane size in bytes. */
 int nPlanes; /* number of planes in our display image */
 BYTE buffer[bufSz];

 if (parent->subtype != ID_ILEM)
 return(IFE_OKAY); /* just continue scanning the file */

 ilbmFrames = *(ILEmGroup *)parent->clientFrame;
 iffp = OpenRGroup(parent, &formContext);
 CheckIFFP();

 do switch (iffp = GetFChunkHdr(&formContext)) {
 ilbmFrames.foundBEMHD = TRUE;
 iffp = GetBEMHD(&formContext, &ilbmFrames.bmHdr);
 break; }
 case ID_CHAP: {
 ilbmFrames.nColorRegs = maxColorReg; /* we have room for this many */
 iffp = GetCMAP(
 &formContext, (WORD *) ilbmFrames.colorMap, &ilbmFrames.nColorRegs);
 break; }
 case ID_BODY: {
 if (ilbmFrames.foundBEMHD) return(BAD_FORM); /* No BEMHD chunk! */

 nPlanes = MIN(ilbmFrames.bmHdr.nPlanes, EXDepth);
 InitBitMap(
 &bitmap,
 nPlanes,
 ilbmFrames.bmHdr.v,
 ilbmFrames.bmHdr.h);
 psize = RowBytes(ilbmFrames.bmHdr.v) * ilbmFrames.bmHdr.h;
 if (bitmap.Planes[0] =
 (PLANEPTR)Allocate(nPlanes * psize, MEME_CHIP))
 {
 for (i = 1; i < nPlanes; i++)
 bitmap.Planes[i] = (PLANEPTR) bitmap.Planes[0] + psize*i;
 iffp = GetBODY(
 &formContext,
 &bitmap,
 NULL,
 &ilbmFrames.bmHdr.

```

```

buffer,
 bufSz);
 if (iffp == IFE_OKAY) iffp = IFE_DONE; /* Eureka */
 else
 iffp = CLIENT_ERROR; /* not enough RAM for the bitmap */
 break; }
 case END_MARK: { iffp = BAD_FORM; break; } /* No BODY chunk! */
 } while (iffp >= IFE_OKAY); /* loop if valid ID of ignored chunk or a
 * subroutine returned IFE_OKAY (no errors). */

 if (iffp != IFE_DONE) return(iffp);

 /* If we get this far, there were no errors. */
 CloseRGroup(&formContext);
 DisplayPic(&ilbmFrames);
 return(iffp);
 }

 /* Notes on extending GetFoilem *****
 * To read more kinds of chunks, just add clauses to the switch statement.
 * To read more kinds of property chunks (CRAB, CAMG, etc.) add clauses to
 * the switch statement in GetPrilem, too.
 *
 * To read a FORM type that contains a variable number of data chunks--e.g.
 * a FORM FITX with any number of CHRS chunks--replace the ID_BODY case with
 * an ID_CHRS case that doesn't set iffp = IFE_DONE, and make the END_MARK
 * case do whatever cleanup you need.
 *
 * *****
 /* GetPrilem() *****
 * Called via ReadPicture to handle every PROP encountered in an IFF file.
 * Reads PROPs ILEM and skips all others.
 *
 * *****
 #if Fancy
 IFFP GetPrilem(parent) GroupContext *parent; {
 /*compilerBug register*/ IFFP iffp;
 GroupContext propContext;
 ILEmGroup *ilbmFrames = (ILEmGroup *)parent->clientFrame;

 if (parent->subtype != ID_ILEM)
 return(IFE_OKAY); /* just continue scanning the file */

 iffp = OpenRGroup(parent, &propContext);
 CheckIFFP();

 do switch (iffp = GetPChunkHdr(&propContext)) {
 case ID_BEMHD: {
 ilbmFrames->foundBEMHD = TRUE;
 iffp = GetBEMHD(&propContext, &ilbmFrames->bmHdr);
 break; }
 case ID_CMAPP: {

```

```

ilbmFrame->nColorRegs = maxColorReg; /* we have room for this many */
ifp = GetOWAP(
    &propContext, (WORD *) ilbmFrame->colorMap, &ilbmFrame->nColorRegs);
break; }
} while (iffp >= IFF_OKAY); /* loop if valid ID of ignored chunk or a
    * subroutine returned IFF_OKAY (no errors). */

CloseGroup(&propContext);
return(iffp == END_MARK ? IFF_OKAY : iffp);
}
#endif

/* GetLILBM() ***** */
/* Called via ReadPicture to handle every LIST encountered in an IFF file.
*****
# if Fancy
IFFP GetLILBM(parent) GroupContext *parent; {
    ILBMFrame newFrame; /* allocate a new Frame */

    newFrame = *(ILBMFrame *)parent->clientFrame; /* copy parent frame */
    return( ReadILList(parent, (ClientFrame *)&newFrame) );
}
#endif

/* ReadPicture() ***** */
/* Read a picture from an IFF file, given a file handle open for reading.
*****
IFFP ReadPicture(file)
    LONG file;
{
    ILBMFrame iFrame; /* Top level "client frame". */
    IFFP iffp = IFF_OKAY;

    #if Fancy
        iFrame.clientFrame.getList = GetLILBM;
        iFrame.clientFrame.getProp = GetPrILBM;
    #else
        iFrame.clientFrame.getList = SkipGroup;
        iFrame.clientFrame.getProp = SkipGroup;
    #endif
    iFrame.clientFrame.getForm = GetFoILBM;
    iFrame.clientFrame.getCat = ReadICat;

    /* Initialize the top-level client frame's property settings to the
    * program-wide defaults. This example just records that we haven't read
    * any BMHD property or OWAP color registers yet. For the color map, that
    * means the default is to leave the machine's color registers alone.
    * If you want to read a property like GRAB, init it here to (0, 0). */
    iFrame.foundBMHD = FALSE;
    iFrame.nColorRegs = 0;

```

```

iffp = ReadIFF(file, (ClientFrame *)&iFrame);
Close(file);
return(iffp);
}

/* main() ***** */
void main(load, filename)
    int load; char *filename;
{
    LONG file;
    IFFP iffp = NO_FILE;

    if (load) { /* load and display the picture */
        if (!GfxBase = (struct GfxBase *)OpenLibrary("graphics.library", 0))
            exit(0);
        file = Open(filename, MODE_OLDFILE);
        if (!file)
            iffp = ReadPicture(file);

        printf(" %s\n", IFFMessages[-iffp]);
    }
    else { /* cleanup */
        if (bitmap.Planes[0]) {
            FreeMem(bitmap.Planes[0],
                bitmap.BytesPerRow * bitmap.Rows * bitmap.Depth);
            FreeVPortCopLists(&vp);
            FreeCprList(v.LOFcprList);
        }
        CloseLibrary(GfxBase);
    }
}

/* main() ***** */
void main(argc, argv)
    int argc; char **argv;
{
    printf("Showing file '%s' ...", argv[1]);
    if (argc < 2)
        printf("\nUsage: 'ShowILBM filename'");
    else {
        main0(1, argv[1]);
        main0(0, NULL); /* free all resources */
    }
    printf("\n");
    exit(0);
}

```

```

/*----- Convert data from "cmpByteRun1" run compression. 11/11/85
 *
 * By Jerry Morrison and Steve Shaw, Electronic Arts.
 * This software is in the public domain.
 *
 * control bytes:
 * [0..127] : followed by n+1 bytes of data.
 * [-1..-127] : followed by byte to be repeated (-n)+1 times.
 * -128 : NOOP.
 *
 * This version for the Commodore-Amiga computer.
 *-----*/
#include "exec/types.h"
#include "packer.h"

/*----- UnPackRow -----*/
#define UCetByte() (*source++)
#define UPutByte(c) (*dest++ = (c))

/* Given POINTERS to POINTER variables, unpacks one row, updating the source
 * and destination pointers until it produces dstBytes bytes. */
BOOL UnPackRow(pSource, pDest, srcBytes0, dstBytes0)
BYTE *pSource, *pDest; WORD srcBytes0, dstBytes0; {
    register BYTE *source = *pSource;
    register WORD n;
    register BYTE c;
    register WORD srcBytes = srcBytes0, dstBytes = dstBytes0;
    BOOL error = TRUE; /* assumes error until we make it through the loop */
    WORD minus128 = -128; /* get the compiler to generate a CMP.W */

    while( dstBytes > 0 ) {
        if ( (srcBytes -- = 1) < 0 ) goto ErrorExit;
        n = UCetByte();

        if (n >= 0) {
            n += 1;
            if ( (srcBytes -- = n) < 0 ) goto ErrorExit;
            if ( (dstBytes -- = n) < 0 ) goto ErrorExit;
            do { UPutByte(UCetByte()); } while (--n > 0);
        }
        else if (n != minus128) {
            n = -n + 1;
            if ( (srcBytes -- = 1) < 0 ) goto ErrorExit;
            if ( (dstBytes -- = n) < 0 ) goto ErrorExit;
            c = UCetByte();
            do { UPutByte(c); } while (--n > 0);
        }
    }
    error = FALSE; /* success! */
ErrorExit:
}

```


Appendix I

Printer-Dependent Source Code

This appendix contains the printer-dependent source code for the following printers:

hpplus - Hewlett Packard LaserJet Plus

okimate20 - Okidata

epson - Epson X-80 series

diablo_c - Diablo C-150

In addition, this appendix includes *macros.i*, which is required in order to assemble any of the ".asm" files.

These files are intended to aid developers in creating their own custom printer drivers that can be added to the DEVS: directory on an AmigaDOS disk. The documentation that explains the contents of these files is in the "Printer Device" chapter of the *Amiga ROM Kernel manual*.

```
*****
*
* printer device macro definitions
*
*****
*----- external definition macros -----
XREF_EYE XREF MACRO _LVO\1
          ENDM
XREF_CFX XREF MACRO _LVO\1
          ENDM
*----- library dispatch macros -----
CALLXXE MACRO
        CALLIB _LVO\1
        ENDM
LINKXXE MACRO
        LINKLIB _LVO\1,_SysBase
        ENDM
LINKCEX MACRO
        LINKLIB _LVO\1,_CfxBase
        ENDM
```



```

/* diablo C-150 special printer functions */
/***** prInter_device/printers/Diablo_C-150_special_functions *****/
* NAME
* Diablo C-150 special functions implemented:
*****
#include "exec/types.h"
#include "devices/printer.h"
#include "devices/prtbase.h"
extern struct PrinterData *PD;

DoSpecial(command,outputBuffer,vline,currentVMI,criflflag,Params)
char outputBuffer[];
UNORD *command;

BYTE *vline;
UBYTE *currentVMI; /* used for color on this printer */
BYTE *criflflag;
UBYTE Params[];
{
    int x=0;
    int y=0;
    static BYTE ISOcolorTable[10] = {
        49,51,53,52,55,50,54,48,49,49 };
    static unsigned char InitMarg[] = "\033100\015\033r00\015";

    if (*command==aRIN) {
        *currentVMI=0x70; /* white background, black text */
        outputBuffer[x++] = '\015';
        outputBuffer[x++] = '\012';
        Params[0] = (PD->pd_Preferences.Pr-IntLeftMargin);
        Params[1] = (PD->pd_Preferences.Pr-IntRightMargin);
        *command=aSLRN;
    }
    if (*command==aSLRM) {
        Params[0] = Params[0] + 4;
        if (Params[0] < 5) Params[0] = 5;
        Params[1] = Params[1] + 5;
        if (Params[1] > 90) Params[1] = 90;
        InitMarg[2] = (char) (Params[0]/10 + '0');
        InitMarg[3] = (char) (Params[0] - (UBYTE) (Params[0]/10) * 10 + '0');
        InitMarg[7] = (char) (Params[1]/10 + '0');
        InitMarg[8] = (char) (Params[1] - (UBYTE) (Params[1]/10) * 10 + '0');
        while (y < 10) outputBuffer[x++] = InitMarg[y++];
        return(x);
    }
    if (*command==aSFC)
    {
        if (Params[0] == 39) Params[0] = 30; /* set defaults */
    }
}

```

```

    if (Params[0] == 49) Params[0] = 47;
    if (Params[0] < 40) *currentVMI = ((*currentVMI & 240) + (Params[0] - 30));
    else *currentVMI = ((*currentVMI & 15) + ((Params[0] - 40) * 16));
    outputBuffer[x++] = '\033';
    outputBuffer[x++] = '@';
    outputBuffer[x++] = ISOcolorTable[*currentVMI & 15];
    outputBuffer[x++] = ISOcolorTable[((( *currentVMI & 240) / 16)]];
    return(x);
}
    if (*command==aRIS) PD->pd_PWaitEnabled=253;
    return(0);
}

```

SECTION printer

----- Included Files -----

```

INCLUDE "exec/types.i"
INCLUDE "exec/nodes.i"
INCLUDE "exec/lists.i"
INCLUDE "exec/memory.i"
INCLUDE "exec/ports.i"
INCLUDE "exec/libraries.i"

INCLUDE "devices/macros.i"

```

----- Imported Functions -----

```

XREF_EXE ClosesLibrary
XREF_EXE OpenLibrary
XREF_AbsExecBase

```

XREF_PEDData

----- Exported Globals -----

```

XDEF _Init
XDEF _Expunge
XDEF _Open
XDEF _Close
XDEF _PD
XDEF _PED
XDEF _SysBase
XDEF _DOSBase
XDEF _GfxBase
XDEF _IntuitionBase

```

----- SECTION printer, DATA -----

```

_PD DC.L 0
_PED DC.L 0
_SysBase DC.L 0
_DOSBase DC.L 0
_GfxBase DC.L 0
_IntuitionBase DC.L 0

```

----- SECTION printer, CODE -----

```

_Init:
MOVE.L 4(A7),_PD
LEA _PEDData(PC),A0
MOVE.L A0,_PED
MOVE.L A6,-(A7)
MOVE.L _AbsExecBase,A6
MOVE.L A6,-_SysBase

```

* ;----- open the dos library

```

LEA DLName(PC),A1
MOVEQ #0,D0
CALLEXE OpenLibrary
MOVE.L D0,_DOSBase
BEQ InitDLerr

```

* ;----- open the graphics library

```

LEA GLName(PC),A1
MOVEQ #0,D0
CALLEXE OpenLibrary
MOVE.L D0,_GfxBase
BEQ InitGLErr

```

* ;----- open the intuition library

```

LEA ILName(PC),A1
MOVEQ #0,D0
CALLEXE OpenLibrary
MOVE.L D0,_IntuitionBase
BEQ InitILErr

```

pdirts: MOVEQ #0,D0

```

MOVE.L (A7)+,A6
RTS

```

InitPAErr:

```

MOVE.L _IntuitionBase,A1
LINKEE ClosesLibrary

```

InitLErr:

```

MOVE.L _GfxBase,A1
LINKEE ClosesLibrary

```

InitGLErr:

```

MOVE.L _DOSBase,A1
LINKEE ClosesLibrary

```

InitDLerr:

```

MOVEQ #-1,D0
ERRA.S pdirts

```

ILName: 'intuition.library'

```

DC.B 0
DC.B 0

```

DLName: 'dos.library'

```

DC.B 0
DC.B 0

```

GLName: 'graphics.library'

```

DC.B 0
DC.B 0
DS.W 0

```

```

-----
_Expunge:
MOVE L _IntuitionBase,A1
LINKEXE CloseLibrary

MOVE L _GfxBase,A1
LINKEXE CloseLibrary

MOVE L _DOSBase,A1
LINKEXE CloseLibrary

```

```

-----
_Open:
MOVEQ #0,D0
RTS

_Close:
MOVEQ #0,D0
RTS
END

```

```

*****printertag.asm for diablo_c*****
SECTION printer
----- Included Files -----
INCLUDE "exec/types.i"
INCLUDE "exec/nodes.i"
INCLUDE "exec/strings.i"
INCLUDE "devices/prtbase.i"

```

```

----- Imported Names -----
XREF _Init
XREF _Expunge
XREF _Open
XREF _Close
XREF _CommandTable
XREF _PrinterSegmentData
XREF _DoSpecial
XREF _Render

```

```

----- Exported Names -----
XDEF _PEDData

```

```

*****
MOVEQ #0,D0 ; show error for OpenLibrary()
RTS
DC.W VERSION
DC.W REVISION

_PEDData:
DC.L printerName
DC.L _Init
DC.L _Expunge
DC.L _Open
DC.L _Close
DC.B PPC_COLORCFX
DC.B PPC_YMCB
DC.B 80
DC.B 1
DC.W 4
DC.L 1024
DC.L 0
DC.W 120
DC.W 120
DC.L _CommandTable
DC.L _DoSpecial
DC.L _Render
DC.L 60
; PrinterClass
; ColorClass
; MaxColumns
; NumCharSets
; NumRows
; MaxXdots
; MaxYdots
; XDotsInch
; YDotsInch
; Commands
; twice normal: slow alpha

```

printerName:

```

STRING <'Diablo C-150'>
END

```

```

*****
#include <exec/types.h>
#include <exec/nodes.h>
#include <exec/lists.h>
#include <exec/memory.h>
#include " ../printer/prthbase.h"

extern struct PrinterData *PD;

/* for the DIABLO C-150 */

int
Render(ct, x, y, status) /* passed a color type */
  UBYTE ct; /* the color type to use (0, 1, 2 or 3) */
  UWORD x, y; /* the x & y co-ordinates */
  UBYTE status; /* or the pc & pr print values, or special */
                /* print status (0=init, 1=enter pixel, 2=dump) */
{
  static UWORD ROWSIZE;
  static UWORD COLORSIZE;
  static UWORD BUFSIZE;
  static UWORD colors[4]; /* color ptrs */
  static BYTE huns,tens,ones; /* used to program buffer size */
  static UWORD bufptr;

  UWORD i; /* used for double buffering; points to buffer 1 or 2 */
  BYTE err; /* mics. var */
                /* the error # */

  switch(status)
  {
  case 0 : /* alloc memory for printer buffer (uses double buffering) */
            ROWSIZE=(x*7)/8; /* pc/8 pixels per row on the DIABLO C-150 */
            huns=ROWSIZE/100;
            tens=(ROWSIZE-huns*100)/10;
            ones=(ROWSIZE-huns*100-tens*10);
            ROWSIZE += 7; /* plus 7 cmd bytes */
            COLORSIZE=(ROWSIZE*4); /* the size of each color buffer */
            BUFSIZE=(COLORSIZE*4*3); /* buffer size required for DIABLO C-150 */

            colors[0] = 7; /* black */
            colors[1] = COLORSIZE*2*7; /* yellow */
            colors[2] = COLORSIZE*7; /* magenta */
            colors[3] = COLORSIZE*3*7; /* cyan */
            PD->pd_PrintBuf = (UBYTE *)
            AllocMem(BUFSIZE*2, MEMF_PUBLIC); /* alloc public mem */
            if (err=(PD->pd_PrintBuf==0)) return(err);
            if (err=(*(PD->pd_PWrIte)("\033\rP",3)) return(err);
                /* reset printer to power-up */
            if (err=PWait(1,0)) return(err);
            if (err=(*(PD->pd_PWrIte)("\033I\r",4)) return(err);
                /* set 1 margin to .5 inch. */
            if (err=(*(PD->pd_PWrIte)("\033-90\r",5)) return(err);
                /* set r margin to 9 inch. */
            bufptr=0; /* init to first buffer */

```

```

return(0); /* flag all ok */
break;

case 1 : /* put pixel in buffer */
    i = bufptr+x/8*(y/63)*ROWSIZE+colors[ct];
    /* calc which byte to use */
    PD->pd_PrintBuf[i] = PD->pd_PrintBuf[i] | (1 << (7-(x&7)));
    /* fill print buffer */
return(0); /* flag all ok */
break;

case 2 : /* dump buffer to printer */
    if (err!=(PD->pd_PWrite)) (&PD->pd_PrintBuf[bufptr],BUFSIZE)
        return(err);
    bufptr=BUFSIZE-bufptr; /* switch to other buffer */
return(0); /* flag all ok */
break;

case 3 : /* clear and init buffer */
    for (i=bufptr; i<BUFSIZE+bufptr; i++)
        PD->pd_PrintBuf[i] = 0; /* clear buffer */
    for (i=0; i<16; i++) {
        PD->pd_PrintBuf[bufptr+i*ROWSIZE] = 27;
        PD->pd_PrintBuf[bufptr+i*ROWSIZE+1] = '9';
        PD->pd_PrintBuf[bufptr+i*ROWSIZE+2] = '1+0';
        PD->pd_PrintBuf[bufptr+i*ROWSIZE+3] = 'huns + 0';
        PD->pd_PrintBuf[bufptr+i*ROWSIZE+4] = 'tens + 0';
        PD->pd_PrintBuf[bufptr+i*ROWSIZE+5] = 'ones + 0';
        PD->pd_PrintBuf[bufptr+i*ROWSIZE+6] = ',';
        /* select # of bytes for each line */
    }
    PD->pd_PrintBuf[bufptr+BUFSIZE-3] = 27;
    PD->pd_PrintBuf[bufptr+BUFSIZE-2] = 'k';
    PD->pd_PrintBuf[bufptr+BUFSIZE-1] = '1';
return(0); /* flag all ok */
break;

case 4 : /* free the print buffer memory */
    err=(PD->pd_PBothReady)();
    /* wait for both buffers to be clear */
    FreeMem(PD->pd_PrintBuf,BUFSIZE*2);
    /* free the print buffers memory */
return(err); /* return status */
break;

default :
return(0); /* flag all ok */
}
}
}

```



```

/* Epson X80 special commands */
/***** printer.device/printers/Epson_functions *****/
*
* special functions implemented:
*
* aRIM, aSUS0, aSUS1, aSUS2, aSUS3, aSUS4,
* aPLU, aPLD, aVERP0, aVERP1, aSLRM, aIND, aCAM
*
*****/
#include "exec/types.h"
#include "devices/printer.h"
#include "devices/prtbase.h"

extern struct PrinterData *PD;

DoSpecial(command, outputBuffer, vline, currentVMI, crlFlag, Params)
char outputBuffer[];
UNORD *command;
BYTE *vline;
BYTE *currentVMI;
BYTE *crlFlag;
UBYTE Params[];
{
    int x=0;
    int y=0;
    static char InitMarg[]="0331L\033Q\033Q\120\033Q\210\033Q\240";
    static char
    InitThisPrInter []="033x\001\0332\022\0335\033P\033-\376\033F\n\033M";

    if(*command==aRIM) {
        while(x<18) {
            outputBuffer[x]=InitThisPrInter[x];
            x++;
        }
        outputBuffer[x++]='\000';
        outputBuffer[12]='\000';
    }
    if((PD->pd_Preferences.PrintQuality)==DRAFT)
        outputBuffer[2]='\000';

    *currentVMI=36; /* assumes 1/6 line spacing */
    if((PD->pd_Preferences.PrIntSpacing)==EIGHT_LPI)
        { /* wrong again */
            outputBuffer[4]='0';
            *currentVMI=27;
        }
    if((PD->pd_Preferences.PrIntPitch) != PICA)
        outputBuffer[x++]='\033';
    if((PD->pd_Preferences.PrIntPitch)==ELITE) outputBuffer[x++]='M';
    else if((PD->pd_Preferences.PrIntPitch)==FINE)
        outputBuffer[x++]='\017';

    Params[0]=(PD->pd_Preferences.PrIntLeftMargin);

    if(*command==aSLRM)
        {
            Parm[1]=(PD->pd_Preferences.PrIntRightMargin);
            *command=aSLRM;
        }
    if(*command==aSLRM) {
        InitMarg[2]=Params[0];
        InitMarg[5]=Params[1]+1;
        while(y<6) outputBuffer[x++]=InitMarg[y++];
        return(x);
    }
    if(*command==aCAM) {
        InitMarg[2]=1;
        InitMarg[5]=80;
        while(y<15) outputBuffer[x++]=InitMarg[y++];
        return(x);
    }
    if(*command==aPLU) {
        if(*vline==0) {
            (*vline)=1;
            *command=aSUS2;
            return(0);
        }
        if(*vline)<0 {
            (*vline)=0;
            *command=aSUS3;
            return(0);
        }
        return(-1);
    }
    if(*command==aPLD) {
        if(*vline==0) {
            (*vline)=-1;
            *command=aSUS4;
            return(0);
        }
        if(*vline)>0 {
            (*vline)=0;
            *command=aSUS1;
            return(0);
        }
        return(-1);
    }
    if(*command==aSUS0) *vline=0;
    if(*command==aSUS1) *vline=0;
    if(*command==aSUS2) *vline=1;
    if(*command==aSUS3) *vline=0;
    if(*command==aSUS4) *vline=(-1);
    if(*command==aVERP0) *currentVMI=27;
    if(*command==aVERP1) *currentVMI=36;
}

```

```

if (*command==aIND) {
    outputBuffer[x++]='\033';
    outputBuffer[x++]='J';
    outputBuffer[x++] = *currentVMI;
    return(x);
}
return(0);
}

```

```

*****init.asm for epson X-80 series*****
SECTION printer
----- Included Files -----
INCLUDE "exec/types.i"
INCLUDE "exec/nodes.i"
INCLUDE "exec/lists.i"
INCLUDE "exec/memory.i"
INCLUDE "exec/ports.i"
INCLUDE "exec/libraries.i"
INCLUDE "devices/macros.i"
----- Imported Functions -----
XREF_EXE CloseLibrary
XREF_EXE OpenLibrary
XREF_ABS AbsExecBase
XREF _PEDData
----- Exported Globals -----
XDEF _Init
XDEF _Expunge
XDEF _Open
XDEF _Close
XDEF _PD
XDEF _PED
XDEF _SysBase
XDEF _GfxBase
*****
SECTION printer,DATA
_PD DC.L 0
_PED DC.L 0
_SysBase DC.L 0
_GfxBase DC.L 0
*****
SECTION printer, CODE
_Init:
MOVE.L 4(A7),_PD
LEA _PEDData(PC),A0
MOVE.L A0,_PED
MOVE.L A6,-(A7)
MOVE.L _AbsExecBase,A6
MOVE.L A6,_SysBase
;----- open the graphics library

```

```

LEA   CLNames(PC),A1
MOVEQ #0,D0
CALLX OpenLibrary
MOVE.L D0,_GfxBase
BEQ   InitGLErr

MOVEQ #0,D0
MOVE.L (A7)+,A6
RTS

MOVEQ #-1,D0
BRA.S pdIRts

CLNames:
DC.B 'graphics.library'
DC.B 0
DS.W 0

```

```

-----
|  _Expunge:
|  LINKX CloseLibrary
|  -----
|

```

```

_Open:
_Close:
MOVEQ #0,D0
RTS
END

```

```

*****printertag.asm for epson X-80 series *****
*
* printer device dependent code tag
*
*****

```

```

SECTION printer
----- Included Files -----
INCLUDE "exec/types.i"
INCLUDE "exec/nodes.i"
INCLUDE "exec/strings.i"
INCLUDE "devices/prtbase.i"

```

```

----- Imported Names -----
XREF _Init
XREF _Expunge
XREF _Open
XREF _Close
XREF _CommandTable
XREF _PrinterSegmentData
XREF _DoSpecial
XREF _Render

```

```

----- Exported Names -----
XDEF _PEDData

```

```

*****
MOVEQ #0,D0
RTS
DC.W 1
DC.W 1
-----
_PEDData:
DC.L printerName
DC.L _Init
DC.L _Expunge
DC.L _Open
DC.L _Close
DC.B PPC_BACEX
DC.B FCC_BW
DC.B 80
DC.B 10
DC.W 8
DC.L 960
DC.L 0
DC.W 120
DC.W 82
DC.L _CommandTable
DC.L _DoSpecial
-----
MOVEQ #0,D0 ; show error for OpenLibrary()
RTS
DC.W 1 ; VERSION
DC.W 1 ; REVISION
-----
; PrinterClass
; ColorClass
; MaxColumns
; NumCharSets
; NumRows
; MaxXDots
; MaxYDots
; XDotsInch
; YDotsInch
; Commands

```

```

prPrinterName:
    DC.L    _Render
    DC.L    30
    STRING <'Epson'>
    END

```

```

/***** render.c for epson X-80 series *****/
#include "exec/types.h"
#include "exec/nodes.h"
#include "exec/lists.h"
#include "exec/memory.h"
#include "devices/prtbase.h"

extern struct PrinterData *PD;

/* for the EPSON */
int Render(ct, x, Y, status)
UBYTE ct; /* null for b/w printers */
UMWORD x, y; /* the x & y co-ordinates */
/* or the pc & pr print values, or special */
UBYTE status; /* print status (0-init, 1-enter pixel, 2-dump) */
{
    static UMWORD ROWSIZE;
    static UMWORD BUFSIZE;
    static UMWORD bufptr;
    UMWORD i;
    BYTE err;

    /* mics. var */
    /* the error # */

    switch(status)
    {
    case 0 : /* alloc memory for printer buffer */
        ROWSIZE=x; /* row size required for EPSON */
        BUFSIZE=(6*ROWSIZE); /* buffer size required for EPSON */
        PD->pd_PrintBuf = (UBYTE *)
            AllocMem(BUFSIZE*2, MEMG_PUBLIC); /* alloc public mem */
        if (err=(PD->pd_PrintBuf == 0)) return(err);
        /* reset printer to power-up state */
        if (err=(PD->pd_PWrite)("\033", 2)) return(err);
        if (err=Wait(1, 0)) return(err);
        if (err=(PD->pd_PWrite)("\033", 2)) return(err);
        bufptr=0;
        return(0); /* flag all ok */
        break;

    case 1 : /* put pixel in buffer */
        i = bufptr+x*4; /* calc which byte to use */
        PD->pd_PrintBuf[i] = PD->pd_PrintBuf[i] | (1 << (7-(y&7)));
        return(0); /* flag all ok */
        break;

    case 2 : /* dump buffer to printer */
        if (err=(PD->pd_PWrite) (&(PD->pd_PrintBuf[bufptr]), BUFSIZE))
            return(err);
        bufptr=BUFSIZE-bufptr;
        return(0); /* flag all ok */
        break;

    case 3 : /* clear and init buffer */
        for (i=bufptr; i<bufptr+BUFSIZE; i++)
            PD->pd_PrintBuf[i] = 0; /* clear buffer */
    }
}

```

```
PD->pd_PrintBuf[bufptr] = 27;
PD->pd_PrintBuf[bufptr+1] = 'L';
PD->pd_PrintBuf[bufptr+2] = ROMSIZE & 0xff;
PD->pd_PrintBuf[bufptr+3] = ROMSIZE >> 8;
PD->pd_PrintBuf[bufptr+BUFSIZE-2] = 10;
PD->pd_PrintBuf[bufptr+BUFSIZE-1] = 13;
return(0); /* flag all ok */
break;

case 4 : /* free the print buffer memory */
err= (* (PD->pd_Write)) ("\033", 2);
if (!err) err= (* (PD->pd_PbothReady)) ();
FreeMem(PD->pd_PrintBuf, BUFSIZE*2);
return(err); /* return status */
break;

default:
return(0);
}
}
```

```

/* HP command table */
/***** printer.device/HP_LaserJet_Plus_functions *****/
*
* NAME
* HP LaserJet 2686A functions implemented:
*
* ARIS, aIND, aNEL,
* aSCR0, aSCR3, aSCR23, aSCR4, aSCR24, aSCR1, aSCR22,
* aSHORP0, aSHORP1, aSHORP2, aSHORP3, aSHORP4
* aDEN3, aDEN4, aPLU, aPLD,
* aFNT0, aFNT3, aFNT8,
* aPROP0, aPROP1, aPROP2,
* aVERP0, aVERP1, aPERF, aPERF0, aCAM
*
*****
char *CommandTable[]={
  "\375\033E\375" /*reset*/
  "\377" /*initialize*/
  "\012" /* lf IND ESCD */
  "\015\012" /* return,lf NEL ESCZ */
  "\036a-IR" /* reverse lf RI ESON */
  "\036a-g\033(sB)", /*normal char set SCR 0 */
  "\033(sIS)", /*italics on*/
  "\033(sS)", /*italics off*/
  "\036d", /*underline on*/
  "\036d", /*underline off */
  "\036g", /*boldface on*/
  "\033(sB)", /*boldface off*/
  "\033(sB)", /* set foreground color */
  "\377", /* set background color */
  "\033(s10hIT)", /* normal pitch */
  "\033(s12h2IT)", /* elite on*/
  "\033(s10hIT)", /* elite off*/
  "\033(s15H)", /* condensed on*/
  "\033(s10H)", /* condensed off*/
  "\377", /* enlarged on*/
  "\377", /* enlarged off*/
  "\033(s7B)", /*shadow print on*/
  "\033(sB)", /*shadow print off*/
  "\033(s3B)", /*doublestrike on*/
  "\033(sB)", /*doublestrike off*/
  "\377", /* NLQ on*/
  "\377", /* NLQ off*/
  "\377", /*superscript on*/
  "\377", /*superscript off*/
  "\377", /*subscript on*/
  "\377", /*subscript off*/
  "\377", /* normalize */
  "\0336a-5R", /* partial line up PLU ESCL */
}

```

```

"\033=", /* partial line down PLD ESCX */
"\033(U)", /*US char set */
"\033(F)", /*French char set*/
"\033(G)", /*German char set*/
"\033(I)", /*UK char set*/
"\033(D)", /*Danish I char set*/
"\033(S)", /*Sweden char set*/
"\033(I)", /*Italian char set*/
"\033(1S)", /*Spanish char set*/
"\033(8K)", /*Japanese char set*/
"\033(D)", /*Norwegian char set*/
"\033(D)", /*Danish II char set*/
"\033(sIP)", /*proportional on*/
"\033(sp)", /*proportional off*/
"\033(sp)", /*proportional clear*/
"\377", /*set prop offset*/
"\377", /*auto left justify on*/
"\377", /*auto right justify on*/
"\377", /*auto full justify on*/
"\377", /*auto justify/center off*/
"\377", /*place holder */
"\377", /*auto center on*/
"\036l8D", /* 1/8" line space*/
"\036l6D", /* 1/6" line spacing*/
"\377", /* set form length n */
"\036l1L", /* perf skip n */
"\036l1L", /* Perf skip off */
"\377", /* Left margin set */
"\377", /* Right margin set */
"\377", /* Top margin set */
"\377", /* Bottom marg set */
"\377", /* T&B margin set STBM ESC [Pn1;Pn2r */
"\377", /* L&R margin set SLRM ESC [Pn1;Pn2s */
"\0339", /* Clear margins */
"\377", /* Set horiz tab */
"\377", /* Set vertical tab */
"\377", /* Clr horiz tab */
"\377", /* Clear all h tabs */
"\377", /* Clear vertical tab */
"\377", /* Clr all v tabs TBC 4 */
"\377", /* Clr all h & v tabs */
"\377", /* set default tabs */
"\377", /* extended commands */
}

```

```

/*
**** density.c ****
*/
#include <exec/types.h>
#include "devices/prtbase.h"
#include "devices/printr.h"

extern struct PrinterExtendedData *PED;
extern char density[];

SetDensity(level)
UMWORD level;
{
    switch (level) {
        case SPECIAL_DENSITY1:
            PED->ped_MaxXDots = 600;
            PED->ped_MaxYDots = 795;
            PED->ped_XDotsInch = PED->ped_YDotsInch = 75;
            density[3] = '0';
            density[4] = '7';
            density[5] = '5';
            break;

        case SPECIAL_DENSITY2:
            PED->ped_MaxXDots = 800;
            PED->ped_MaxYDots = 1060;
            PED->ped_XDotsInch = PED->ped_YDotsInch = 100;
            density[3] = '1';
            density[4] = '0';
            density[5] = '0';
            break;

        case SPECIAL_DENSITY3:
            PED->ped_MaxXDots = 1200;
            PED->ped_MaxYDots = 1590;
            PED->ped_XDotsInch = PED->ped_YDotsInch = 150;
            density[3] = '1';
            density[4] = '5';
            density[5] = '0';
            break;

        case SPECIAL_DENSITY4:
            PED->ped_MaxXDots = 2400;
            PED->ped_MaxYDots = 3180;
            PED->ped_XDotsInch = PED->ped_YDotsInch = 300;
            density[3] = '3';
            density[4] = '0';
            density[5] = '0';
            break;

        default:
            break;
    }
}
/*****

```

```

/* hp special printer functions */
/***** printer.device/printers/HP_LaserJet_Plus_special_functions *****/
*
* NAME
* HP LaserJet 2686A special functions implemented:
*
* aRIN,
* aSUS0, aSUS1, aSUS2, aSUS3, aSUS4
* aPLU, aPLD, aVERPO, aVERP1,
* aSLFP, aSLRM, aSTEM
*
*****
#include "exec/types.h"
#include "devices/printr.h"
#include "devices/prtbase.h"

extern struct PrinterData *PD;
UMWORD textlength, topmargin;

DoSpecial(command, outputBuffer, vline, currentVMI, crlfFlag, Params)
char outputBuffer[];
UMWORD *command;
BYTE *vline;
BYTE *currentVMI;
BYTE *crlfFlag;
UBYTE Params[];
{
    int x=0;
    int y=0;
    int j=0;
    static char InitThisPrinter [] = "\033&g@\033&l6D\033 (sb10tpsitu12V";
    static char InitMarg [] = "\033&a0001000M";
    static char InitMarg [] = "\033&l000e000F";
    static char InitForm [] = "\033&l002a000F";

    if (*command==aRIN) {
        while (x<24) {
            outputBuffer[x]=InitThisPrinter[x];
            x++;
        }
        if ((PD->pd_Preferences.PrintSpacing)==EIGHT_LPI) {
            /* wrong again */
            outputBuffer[7]='8';
        }
        if ((PD->pd_Preferences.PrintPitch)==ELITE) {
            outputBuffer[14]='2';
            outputBuffer[16]='2';
        }
        if ((PD->pd_Preferences.PrintPitch)==FINE) {
            outputBuffer[14]='5';
        }
    }
    j=x; /* set the formlength=textlength, top margin of 2 */
}
/*****

```



```

textlength=PD->pd_Preferences.PaperLength;
topmargin=2;
while (y<11) outputBuffer [x++] = initForm [y++];
numberString (textlength, j+7, outputBuffer);
y=0;
Params [0] = (PD->pd_Preferences.PrintLeftMargin);
Params [1] = (PD->pd_Preferences.PrintRightMargin);
*command=aSLRM;
}
if (*command==aSLRM) {
    j=x;
    while (y<11) outputBuffer [x++] = initMarg [y++];
    numberString (Params [0] - 1, j+3, outputBuffer);
    numberString (Params [1] - 1, j+7, outputBuffer);
    return (x);
}
if ((*command==aSUS2) && (*vline==0)) {
    *command=aPLU;
    *vline=1;
    return (0);
}
if ((*command==aSUS2) && (*vline<0)) {
    *command=aRI;
    *vline=1;
    return (0);
}
if ((*command==aSUS1) && (*vline>0)) {
    *command=aPLD;
    *vline=0;
    return (0);
}
if ((*command==aSUS4) && (*vline==0)) {
    *command=aPLD;
    *vline=(-1);
    return (0);
}
if ((*command==aSUS4) && (*vline>0)) {
    *command=aIND;
    *vline=(-1);
    return (0);
}
if ((*command==aSUS3) && (*vline<0)) {
    *command=aPLU;
    *vline=0;
    return (0);
}
if (*command==aSUS0)
{
    if (*vline>0) *command=aPLD;
    if (*vline<0) *command=aPLU;
}

```

```

    *vline=0;
    return (0);
}
if (*command==aPLU) {
    (*vline)++;
    return (0);
}
if (*command==aPLD) {
    (*vline)--;
    return (0);
}
if (*command==aSIRM) {
    if (Params [0] == 0) Params [0] = topmargin;
    else topmargin = --Params [0];
    if (Params [1] == 0) Params [1] = textlength;
    else textlength = Params [1];
    while (x<11) {
        outputBuffer [x] = initMarg [x];
        x++;
    }
    numberString (Params [0], 3, outputBuffer);
    numberString (Params [1] - Params [0], 7, outputBuffer);
    return (x);
}
if (*command==aSLPP) {
    while (x<11) {
        outputBuffer [x] = initForm [x];
        x++;
    }
    numberString (topmargin, 3, outputBuffer);
    /* restore textlength, margin */
    numberString (textlength, 7, outputBuffer);
    return (x);
}
if (*command==aRIS) PD->pd_PWaitEnabled=253;
return (0);
}
VOID
numberString (Param, x, outputBuffer)
BYTE Param;
int x;
char outputBuffer [];
{
    if (Param > 199) {
        outputBuffer [x++] = '2';
    }
}

```

```

    Param--200;
}
else if (Param>99) {
    outputBuffer[x++]='1';
    Param--100;
}
else outputBuffer[x++]='0'; /* always return 3 digits */
if (Param>9) outputBuffer[x++]=(BYTE) (Param/10)+'0';
else outputBuffer[x++]='0';
outputBuffer[x++] = Param%10+'0';
}
Close()
{
    /* (* (PD->pd_PWrite) ("\032", 2); */
    /* (* (PD->pd_PWrite) ("\014", 1);
    return(0);
}

```

*****init.asm for hpplus *****

```

SECTION printer
----- Included Files -----

```

```

INCLUDE "exec/types.i"
INCLUDE "exec/nodes.i"
INCLUDE "exec/lists.i"
INCLUDE "exec/memory.i"
INCLUDE "exec/ports.i"
INCLUDE "exec/libraries.i"

INCLUDE "devices/macros.i"

```

```

----- Imported Functions -----

```

```

XREF_EXE ClosesLibrary
XREF_EXE OpenLibrary
XREF AbsExecBase

```

```

XREF _PEDData

```

```

----- Exported Globals -----

```

```

XDEF _Init
XDEF _Expunge
XDEF _Open
XDEF _PD
XDEF _PED
XDEF _SysBase
XDEF _DOSBase
XDEF _GfxBase
XDEF _IntuitionBase

```

```

SECTION pr:inter.DATA
_PD DC.L 0
_PED DC.L 0
_SysBase DC.L 0
_DOSBase DC.L 0
_GfxBase DC.L 0
_IntuitionBase DC.L 0

```

```

SECTION printer.CODE
_Init:
    MOVE.L 4(A7),_PD
    LEA _PEDData(PC),A0
    MOVE.L A0,_PED
    MOVE.L A6,-(A7)
    MOVE.L _AbsExecBase,A6

```


**** printertag.asm for hppplus ****

SECTION printer

----- Included Files -----

INCLUDE "exec/types.i"
INCLUDE "exec/nodes.i"
INCLUDE "exec/strings.i"
INCLUDE "devices/prtbases.i"

----- Imported Names -----

XREF _Init
XREF _Expunge
XREF _Open
XREF _Close
XREF _CommandTable
XREF _PrinterSegmentData
XREF _DoSpecial
XREF _Render

----- Exported Names -----

XDEF _PEDData

MOVEQ #0,D0 ; show error for OpenLibrary()
RIS
DC.W VERSION
DC.W REVISION

_PEDData:
DC.L printerName
DC.L _Init
DC.L _Expunge
DC.L _Open
DC.L _Close
DC.B PPC_BWCFX ; PrinterClass
DC.B FCC_BW ; ColorClass
DC.B 0 ; MaxColumns
DC.B 0 ; NumCharSets
DC.W 1 ; NumRows
DC.L 800 ; MaxXDots
DC.L 1020 ; MaxYDots
DC.W 100 ; XDotsInch
DC.W 100 ; YDotsInch
DC.L _CommandTable ; Commands
DC.L _DoSpecial
DC.L _Render
DC.L 30

prInterNames:


```

/* okimate 20 commands */
/***** printer.device/printers/Okimate_20_functions *****/
*
* NAME
* Okimate 20 functions implemented:
*
* aRI, aIND, aNEL, aSCR0, aSCR3, aSCR23, aSCR4, aSCR24
* aSHORP0, aSHORP1, aSHORP2, aSHORP3, aSHORP4, aSHORP5, aSHORP6
* aDEN1, aDEN2, aSUS0, aSUS1, aSUS2, aSUS3, aSUS4,
* aVERP0, aVERP1, aSLPP, aPERF, aPERF0
*
* special functions implemented:
* aRI, aRI, aSUS0, aSUS1, aSUS2, aSUS3, aSUS4, aPLU, aPLD
*
*****
char *CommandTable[] = {
  "\033M\376\022\033A\014\033Z\033I\001\033ZH\033-\376\033T\033C\376\011",
  /*reset aRI ESCC */
  "\377", /*initialize aRI IND */
  "\015\012", /* lf IND ESCD */
  "\377", /* return,lf NEL ESCC */
  /* reverse lf RI ESCM */
  "\033ZH\033-\376",
  /*normal char set SCR 0 ESC[0m */
  "\033XC", /*italics on SCR 3 ESC[3m */
  "\033XH", /*italics off SCR 23 ESC[23m */
  "\033XU", /*underline on SCR 4 ESC[4m */
  "\033-\001", /*underline off SCR 24 ESC[24m */
  "\377", /*boldface on SCR 1 ESC[1m */
  "\377", /*boldface off SCR 22 ESC[22m */
  "\377", /* set foreground color */
  "\377", /* set background color */
  /*normal spacing DECSHOP ESC[0w */
  "\033:", /*elite on DECSHOP ESC[2w */
  "\022", /*elite off DECSHOP ESC[1w */
  "\017", /* fine on GSH (special) */
  "\022", /* fine off GSH (special) */
  "\033M\001", /*enlarged on GSH (special) */
  "\033M\376", /*enlarged off GSH (special) */
  "\377", /*shadow print on*/
  "\377", /*shadow print off*/
  "\377", /*doublestrike on*/
  "\377", /*doublestrike off*/
  "\033I\002", /* NLQ on*/
  "\033I\001", /* NLQ off*/
  "\033S\376", /*superscript on PLU ESCI */
  "\033T", /*superscript off PLD (special) */
  "\033S\001", /*subscript on PLD ESCX */
  "\033T", /*subscript off PLU (special) */
}

```

```

"\033T", /* normalize */
"\377", /* partial line up PLU ESCI */
"\377", /* partial line down PLD ESCX */
"\377", /*US char set ESC(B */
"\377", /*French char set ESC(R */
"\377", /*German char set ESC(K */
"\377", /*UK char set ESC(A */
"\377", /*Danish I char set ESC(E */
"\377", /*Sweden char set ESC(H */
"\377", /*Italian char set FNT 6 */
"\377", /*Spanish char set FNT 7 */
"\377", /*Japanese char set FNT 8 */
"\377", /*Norwegian char set FNT 9 */
"\377", /*Danish II char set*/
"\377", /*proportional on */
"\377", /*proportional off*/
"\377", /*proportional clear*/
"\377", /*set prop offset TSS */
"\377", /*auto left justify JFY 5 */
"\377", /*auto right justify JFY 7 */
"\377", /*auto full justify JFY 3.6 */
"\377", /*auto justify off JFY 0 */
"\377", /*place holder */
"\377", /*auto center on JFY 2.6 */
"\0330", /* 1/8" line space DECVERP ESC[0z */
"\033A\014\0332", /* 1/6" line spacing DECVERP ESC[lz */
"\033C", /* set form length DECSLPP ESC[Pnt */
"\033M\001", /* perf skip n */
"\0330", /* perf skip off */
"\377", /* Left margin set DECSLRM ESC[Pn1;Pn2s */
"\377", /* Right margin set DECSLRM ESC[Pn1;Pn2r */
"\377", /* Top margin set DECSLRM ESC[Pn1;Pn2t */
"\377", /* Bottom margin set DECSLRM ESC[Pn1;Pn2b */
"\377", /* T&B margin set STEM ESC[Pn1;Pn2r */
"\377", /* L&R margin set SLRM ESC[Pn1;Pn2s */
"\377", /* Clear margins */
"\377", /* Set horiz tab HTS ESCH */
"\377", /* Set vertical tab VTS ESCJ */
"\377", /* Clr horiz tab TBC 0 ESCOg */
"\033D\376", /* Clear all h tabs TBC 3 ESC3g */
"\033D\376", /* Clr vertical tab TBC 1 ESC1g */
"\377", /* Clr all v tabs TBC 4 ESC4g */
"\033D\010\020\030\040\050\060\070\100\110\120\376", /* Clr all h & v tabs */
/* set default tabs */
"\377", /* extended command */

```

```

/* okimate 20 special commands */
/***** printer.devices/printers/Okimate_20_special_functions *****/
* NAME
* Okimate 20 special functions
*
#include "exec/types.h"
#include "devices/printer.h"
#include "devices/prtbase.h"

extern struct PrinterData *PD;
extern struct PrinterExtendedData *PED;

DoSpecial(command,outputBuffer,vline,currentVMI,crLfFlag,Parms)
char outputBuffer[];
UNORD *command;
BYTE *vline;
BYTE *currentVMI;
BYTE *crLfFlag;
UBYTE Parms[];

{
    int x=0;
    static char initThisPrinter[] =
        "\033I\001\022\0330\033ZH\033-\0376r\033M";
    if (*command==aRIN)
    {
        while(x<15){
            outputBuffer[x]=initThisPrinter[x];
            x++;
        }
        outputBuffer[11]='\000';
        outputBuffer[x++]='\000';
        if ((PD->pd_Preferences.Pr.IntQuality)==LETTER)
            outputBuffer[2]='\002';
        if ((PD->pd_Preferences.Pr.IntPitch)==ELITE) {
            outputBuffer[x++]='\033';
            outputBuffer[x++]=':';
        }
        else if ((PD->pd_Preferences.Pr.IntPitch)==FINE)
            outputBuffer[x++]='\017';
        *currentVMI=27; /* assume 1/8 line spacing */
        if ((PD->pd_Preferences.PrintSpacing)==SIX_JPI) {
            /* wrong again */
            outputBuffer[x++]='\033';
            outputBuffer[x++]='A';
            outputBuffer[x++]='\014';
            outputBuffer[x++]='\033';
            outputBuffer[x++]='2';
            *currentVMI=36;
        }
    }
}

}

return(x);
}

if(*command==aPLU) {
    if((*vline)==0){
        (*vline)=1;
        *command=aSUS2;
        return(0);
    }
    if((*vline)<0) {
        (*vline)=0;
        *command=aSUS3;
        return(0);
    }
    return(-1);
}

if(*command==aPLD) {
    if((*vline)==0){
        (*vline)=(-1);
        *command=aSUS4;
        return(0);
    }
    if((*vline)>0) {
        (*vline)=0;
        *command=aSUS1;
        return(0);
    }
    return(-1);
}

if(*command==aVERP0) *currentVMI=27;
if(*command==aVERP1) *currentVMI=36;
return(0);
}

```

*****Init.asm for okimate20*****

SECTION printer
----- Included Files

INCLUDE "exec/types.i"
INCLUDE "exec/nodes.i"
INCLUDE "exec/lists.i"
INCLUDE "exec/memory.i"
INCLUDE "exec/ports.i"
INCLUDE "exec/libraries.i"
INCLUDE "devices/macros.i"
INCLUDE "devices/prthbase.i"

----- Imported Functions

XREF_EXE ClosesLibrary
XREF_EXE OpenLibrary
XREF AbsExecBase

XREF _PEDData
XREF _RenderBW
XREF _RenderColor

----- Exported Globals

XDEF _Init
XDEF _Expunge
XDEF _Open
XDEF _Close
XDEF _PD
XDEF _PED
XDEF _SysBase
XDEF _DOSBase
XDEF _GfxBase
XDEF _IntuitionBase

----- SECTION printer, DATA

_PD DC.L 0
_PED DC.L 0
_SysBase DC.L 0
_DOSBase DC.L 0
_GfxBase DC.L 0
_IntuitionBase DC.L 0

----- SECTION printer, CODE

_Init: MOVE.L 4(A7),_PD

LEA _PEDData(PC),A0
MOVE.L A0,_PED
MOVE.L A6,-(A7)
MOVE.L _AbsExecBase,A6
MOVE.L A6,_SysBase

----- open the dos library

LEA DLName(PC),A1
MOVEQ #0,D0
CALLX OpenLibrary
MOVE.L D0,_DOSBase
BEQ initDLerr

----- open the graphics library

LEA GLName(PC),A1
MOVEQ #0,D0
CALLX OpenLibrary
MOVE.L D0,_GfxBase
BEQ initGLErr

----- open the intuition library

LEA ILName(PC),A1
MOVEQ #0,D0
CALLX OpenLibrary
MOVE.L D0,_IntuitionBase
BEQ initILErr

MOVEQ #0,D0

MOVE.L (A7)+,A6
RTS

MOVE.L _IntuitionBase,A1
LINKX ClosesLibrary

MOVE.L _GfxBase,A1
LINKX ClosesLibrary

MOVE.L _DOSBase,A1
LINKX ClosesLibrary

MOVEQ #-1,D0
BRA.S pdlrts

DC.B 'intuition.library'
DC.B 0

DC.B 'dos.library'
DC.B 0


```

*****prIntertag.asm for okimate20*****
SECTION          printer
----- Included Files -----

```

```

INCLUDE "exec/types.i"
INCLUDE "exec/nodes.i"
INCLUDE "exec/strings.i"
INCLUDE "devices/prtbase.i"

```

```

----- Imported Names -----

```

```

XREF      _Init
XREF      _Expunge
XREF      _Open
XREF      _Close
XREF      _CommandTable
XREF      _PrinterSegmentData
XREF      _DoSpecial

```

```

----- Exported Names -----

```

```

XDEF      _PEDData

```

```

*****
MOVEQ    #0,D0          ; show error for OpenLibrary()
RTS
DC.W     VERSION
DC.W     REVISION

```

```

_PEDData:

```

```

DC.L     printerName
DC.L     _Init
DC.L     _Expunge
DC.L     _Open
DC.L     _Close
DC.B     PFC_COLORCFX
DC.B     PCC_YMC_EW
DC.B     80
DC.B     1
DC.W     24
DC.W     960
DC.L     0
DC.W     120
DC.W     144
DC.L     _CommandTable
DC.L     _DoSpecial
DC.L     0
DC.L     30
; PrinterClass
; ColorClass
; MaxColumns
; NumCharSets
; NumRows
; MaxIDots
; MaxYDots
; XDotsInch
; YDotsInch
; Commands
; Render

```

```

prInterName:

```

```

STRING  <'OKIMATE 20'>

```

```

DC.B     'graphics.library'
DC.B     0
DS.W     0

```

```

MOVE.L   _IntuitionBase,A1
LINKEXE CloseLibrary

MOVE.L   _GfxBase,A1
LINKEXE CloseLibrary

MOVE.L   _DOSBase,A1
LINKEXE CloseLibrary

```

```

_Open:

```

```

MOVE.L   _PD,A0
CMPI.W   #SHADE_COLOR,pd_Preferences+pf_PrintShade(A0)
BEQ.S    colorRender
LEA      _RenderEM,A0
MOVE.L   A0,_PEDData+ped_Render
ERRA.S   openEnd

```

```

colorRender:

```

```

LEA      _RenderColor,A0
MOVE.L   A0,_PEDData+ped_Render

```

```

openEnd:

```

```

MOVEQ    #0,D0
RTS

```

```

_Close:

```

```

MOVEQ    #0,D0
RTS

```

```

END

```

```

END
/*****
#include <exec/types.h>
#include <exec/nodes.h>
#include <exec/lists.h>
#include <exec/memory.h>
#include "devices/prtbase.h"

extern struct PrinterData *PD;

static UMORD rowsize;
static UMORD bufsize;
static UMORD bufptr;
static UMORD colors[4]; /* color ptrs */

/* for the OKIMATE 20 (Color) */
int RenderColor(ct, x, Y, status) /* passed a color type */
    UBYTE ct; /* the color type to use (0, 1, or 2) */
    UMORD x, Y; /* the x & y co-ordinates */
/* or the pc & pr print values, or special */
    UBYTE status; /* print status (0-Init, 1-enter pixel, 2-dump, 3-end) */
{
    UMORD i; /* mics. var */
    BYTE err; /* the error # */

    switch(status)
    {
        case 0 : /*alloc memory for printer buffer */
            rowsize=(x*3);
            /* pc pixels per row x 3 colors on the OKIMATE 20 */
            bufsize=(rowsize*3*31);
            /* buffer size required for OKIMATE 20 */
            colors[0] = 0;
            colors[1] = 7;
            colors[2] = 9+rowsize*7;
            colors[3] = (9+rowsize)*2*7;
            PD->pd_PrintBuf = (UBYTE *)
                AllocMem(bufsize*2, MEME_PUBLIC); /* alloc public mem */
            if (err=(PD->pd_PrintBuf == 0)) return(err);
            bufptr=0;
            /* set line spacing to 24 printer lines
            (24/144 -> 36/216 inch) */
            return(("PD->pd_PWrite")("\033\044", 3));
            /* thats Esc3<36> */
            break;

        case 1 : /* put pixel in buffer */
            i = bufptr+(y % 24)/8 + x*3 + colors[ct];
            /* calc which byte to use */
            PD->pd_PrintBuf[i] = PD->pd_PrintBuf[i] | (1 << (7-(y&7)));
            /* fill print buffer */
            return(0); /* flag all ok */
            break;

        case 2 : /* dump buffer to printer */
            if (err=(PD->pd_PWrite))(&PD->pd_PrintBuf[bufptr], bufsize))
                return(err);
    }
}

```

```

bufptr=bufsize-bufptr;
return(0); /* flag all ok */
break;

case 3 : /* clear and init buffer */
for (i=bufptr; i<bufsize+bufptr; i++)
    PD->pd_PrintBuf[i] = 0; /* clear buffer */
PD->pd_PrintBuf[bufptr] = 27; /* align ribbon */
for (ct=0; ct<3; ct++) {
    PD->pd_PrintBuf[2+bufptr+ct*(rowsize+9)] = 27;
    PD->pd_PrintBuf[3+bufptr+ct*(rowsize+9)] = 'w';
    PD->pd_PrintBuf[4+bufptr+ct*(rowsize+9)] = '0';
    /* enter 24-dot mode */
    PD->pd_PrintBuf[5+bufptr+ct*(rowsize+9)] =
        (rowsize/3) & 0xff;
    PD->pd_PrintBuf[6+bufptr+ct*(rowsize+9)] =
        (rowsize/3) >> 8; /* set # of dots */
    PD->pd_PrintBuf[1+bufptr+(ct+1)*(rowsize+9)] = 13;
    /* advance color */
}
PD->pd_PrintBuf[bufptr+bufsize-2] = 10; /* lf */
PD->pd_PrintBuf[bufptr+bufsize-1] = 13; /* cr */
return(0); /* flag all ok */
break;

case 4 : /* free the print buffer memory */
err=(* (PD->pd_PBothReady)) ();
/* wait for both buffers to empty */
FreeMem(PD->pd_PrintBuf, bufsize*2); /* free printers memory */
return(err); /* return status */
break;
default:
}
}
/*****
/* for the OKIMATE 20 (b/w) */
int RenderBW(ct, x, y, status) /* passed a color type */
UBYTE ct; /* not used with b/w printers */
UNWORD x, y; /* the x & y co-ordinates */
/* or the pc & pr print values, or special */
UBYTE status; /* print status (0-init, 1-enter pixel, 2-dump, 3-end) */
{
    UNWORD i; /* mics. var */
    BYTE err; /* the error # */

    switch(status)
    {
        case 0 : /*alloc memory for printer buffer */
            rowsize=(x*3);
            /* pc pixels per row x 3 blocks on the OKIMATE 20 bw */
            bufsize=(rowsize*7);
            PD->pd_PrintBuf = (UBYTE *)
                AllocMem(bufsize*2, MEME_PUBLIC); /* alloc public mem */

```

```

if (err=(PD->pd_PrintBuf == 0)) return(err);
bufptr = 0; /* init to first buffer */
/* set line spacing to 24 printer lines
(24/144 -> 36/216 inch) */
return ((* (PD->pd_PWrite)) ("0333\044", 3));
/* thats Esc3<36> */
break;

case 1 : /* put pixel in buffer */
i = bufptr + (y % 24)/8 + x*3 + 5; /* calc which byte to use */
PD->pd_PrintBuf[i] = PD->pd_PrintBuf[i] | (1 << (7-(y&7)));
/* fill print buffer */
return(0); /* flag all ok */
break;

case 2 : /* dump buffer to printer */
if (err=(* (PD->pd_PWrite)) (&(PD->pd_PrintBuf[bufptr]), bufsize))
    return(err);
bufptr = bufsize - bufptr; /* swith to other buffer */
return(0); /* flag all ok */
break;

case 3 : /* clear and init buffer */
for (i=bufptr; i<bufptr+bufsize; i++)
    PD->pd_PrintBuf[i] = 0; /* clear buffer */
PD->pd_PrintBuf[bufptr] = 27;
PD->pd_PrintBuf[bufptr+1] = 'w';
PD->pd_PrintBuf[bufptr+2] = '0'; /* enter 24-dot mode */
PD->pd_PrintBuf[bufptr+3] = (rowsize/3) & 0xff;
PD->pd_PrintBuf[bufptr+4] = (rowsize/3) >> 8;
/* there is rowsize dots */
PD->pd_PrintBuf[bufptr+bufsize-2] = 13; /* cr */
PD->pd_PrintBuf[bufptr+bufsize-1] = 10; /* lf */
return(0); /* flag all ok */
break;

case 4 : /* free the print buffer memory */
err=(* (PD->pd_PBothReady)) ();
/* wait for both buffers to empty */
FreeMem(PD->pd_PrintBuf, bufsize*2);
/* free the print buffer mem */
return(err);
break;
default:
    return(0);
}
}

```

Types of Supported Printers

The available printers that are supported for the Amiga include both whole character (daisy wheel) and dot matrix (wire, ink jet, and laser) types. As with printer capabilities, printer prices range widely, from just over \$200 to over \$3500. In general, the dot matrix printers are capable of graphics output, while "whole character" printers are not.

Every attempt has been made to support a given feature on each printer that, itself, supports that feature. For example, the daisy wheel printers lack the capability to produce characters such as enlarged or italic print. Similarly, the dot matrix printers often lack such features as proportional spacing.

None of the supported printers currently supports all of the available features. (The Epson JX-80 and the HP LaserJet Plus come closest.) Whenever the system requests an unsupported feature, the PRT: handler simply ignores that request. (The "generic" printer driver currently ignores all feature requests.)

If two or more features are each available for a particular printer, they should be usable in combination. For example, Bold-Italic-Underscore is a possible style for many printers.

If your printer is not among those supported for the Amiga, you have two options. If your printer shares a number of common features with one of the supported printers, you can select that printer in Preferences. Keep in mind, however, that one or more of the chosen printer's features might not produce a similar effect on your printer.

Your second option is to select "Custom" from the list of supported printers in Preferences and "Generic" as the custom printer name. You can then construct a custom printer driver following the directions in the *Amiga ROM Kernel Manual*.

The following table lists the printers that are currently supported for the Amiga, grouped according to print technology.

Table 1: Printers Supported on the Amiga

Dot Matrix (Wire), Parallel

| Manufacturer | Model |
|--------------|-------------------------|
| Commodore | CBM MPS 1000 |
| Epson | Epson JX-80 |
| Epson | Epson MX-80, FX-80, ... |
| Okimate | Okimate 20 |

Daisy Wheel, Parallel

| Manufacturer | Model |
|--------------|------------------------------|
| Alphacom | Alphapro 101 |
| Brother | HR-15XL |
| Diablo | 630 (Some models are serial) |
| Diablo | Advantage D25 |
| Qume | LetterPro 20 |

Ink Jet, Parallel

| Manufacturer | Model |
|--------------|-------|
| Diablo | C-150 |

Laser, Serial

| Manufacturer | Model |
|-----------------|----------------|
| Hewlett Packard | Laser Jet |
| Hewlett Packard | Laser Jet Plus |

Other (Custom)

| Manufacturer | Model |
|--------------|-------|
|--------------|-------|

Limited support is offered for a "generic" printer.

Table 2: Printer Functions Supported on the Amiga

Legend:

ISO indicates that the sequence has been defined by the International Standards Organization. This is also very similar to ANSI x3.64.

DEC indicates a control sequence defined by Digital Equipment Corporation.

••• indicates a sequence unique to Amiga.

Printer name:

Alphacom
AlphaPro101

Brother

Diablo 630
HR-15XL

| Code* | Description | Defined | | | |
|-------|---------------------------------------|---------|---|---|---|
| c | Reset | ISO | x | x | x |
| #1 | Initialize | *** | x | x | x |
| D | Line feed | ISO | x | x | x |
| E | Line feed, CR | ISO | x | x | x |
| M | Reverse line feed | ISO | x | x | x |
| [0m | Normal char. set | ISO | x | x | x |
| [3m | Italics on | ISO | | | |
| [23m | Italics off | ISO | | | |
| [4m | Underline on | ISO | x | x | x |
| [24m | Underline off | ISO | x | x | x |
| [1m | Boldface on | ISO | x | x | x |
| [22m | Boldface off | ISO | x | x | x |
| [nm | Set foreground color (n = {30-39}) | ISO | | x | |
| [nm | Set background color (n = {40-49}) | ISO | | | |
| [0w | Normal pitch | DEC | x | x | x |
| [2w | Elite on | DEC | x | x | x |
| [1w | Elite off | DEC | x | x | x |
| [4w | Condensed fine on | DEC | x | x | x |
| [3w | Condensed off | DEC | x | x | x |
| [6w | Enlarged on | DEC | | | |
| [5w | Enlarged off | DEC | | | |
| [6"z | Shadow print on | DEC | x | x | x |
| [5"z | Shadow print off | DEC | x | x | x |
| [4"z | Doublestrike on | DEC | x | x | x |
| [3"z | Doublestrike off | DEC | x | x | x |
| [2"z | NLQ on ** | DEC | | | |
| [1"z | NLQ off ** | DEC | | | |

* Entire escape sequence consists of ESC (ASCII 27) plus indicated code.

** Near Letter Quality

*** Sequence unique to Amiga

Printer name:

Alphacom
AlphaPro101

Brother

Diablo 630
HR-15XL

| Code | Description | Defined | Alphacom AlphaPro101 | Brother | Diablo 630 HR-15XL |
|------|--------------------|---------|-------------------------|---------|-----------------------|
| [2v | Superscript on | *** | x | x | x |
| [1v | Superscript off | *** | x | x | x |
| [4v | Subscript on | *** | x | x | x |
| [3v | Subscript off | *** | x | x | x |
| [0v | Normalize the line | *** | x | x | x |
| L | Partial line up | ISO | x | x | x |
| K | Partial line down | ISO | x | x | x |

| | | |
|----|----------------|-----|
| (B | U.S. char. set | DEC |
| (R | French " " | DEC |
| (K | German " " | DEC |
| (A | UK " " | DEC |
| (E | Danish I " " | DEC |
| (H | Swedish " " | DEC |
| (Y | Italian " " | DEC |
| (Z | Spanish " " | DEC |
| (J | Japanese " " | *** |
| (6 | Norwegian " " | DEC |
| (C | Danish II " " | *** |

| | | | | | |
|------|-------------------------|-----|---|---|---|
| [2p | Proportional on | *** | x | x | x |
| [1p | Proportional off | *** | x | x | x |
| [0p | Proportional clear | *** | x | x | x |
| [n E | Set prop. offset (n) | ISO | | x | |
| [5 F | Auto left justify | ISO | x | | x |
| [7 F | Auto right justify | ISO | x | | |
| [6 F | Auto full justify | ISO | | | |
| [0 F | Justify off | ISO | x | | x |
| [3 F | Letter space (justify) | ISO | | | |
| [1 F | Word fill (auto center) | ISO | | | x |

| | | | | | |
|-----|---------------------|-----|---|---|---|
| [0z | 1/8" line spacing | *** | x | x | x |
| [1z | 1/6" line spacing | *** | x | x | x |
| [nt | Set form length (n) | DEC | x | x | x |
| [nq | Perf skip (n>0)* | *** | | | |
| [0q | Perf skip off | *** | | | |

*Paper perforation skip, n lines

Printer name:

Alphacom
AlphaPro101

Brother

Diablo 630
HR-15XL

| Code | Description | Defined | | | |
|---------|----------------------|---------|---|---|---|
| [0z | 1/8" line spacing | *** | x | x | x |
| [1z | 1/6" line spacing | *** | x | x | x |
| [nt | Set form length (n) | DEC | x | x | x |
| [nq | Perf skip (n>0)* | *** | | | |
| [0q | Perf skip off | *** | | | |
| #9 | Left margin set | *** | x | x | x |
| #0 | Right margin set | *** | x | x | |
| #8 | Top margin set | *** | x | x | x |
| #2 | Bottom margin set | *** | x | x | x |
| [n1;n2r | Top;Bottom margins | DEC | | | |
| [n1;n2s | Left;Right margins | DEC | | x | x |
| #3 | Clear margins | *** | | x | x |
| H | Set horiz. tab | ISO | | x | x |
| J | Set vert. tab | ISO | | x | x |
| [0g | Clear horiz. tab | ISO | | x | x |
| [3g | Clear all hor. tabs | ISO | | x | x |
| [1g | Clear vert. tab | ISO | | | x |
| [4g | Clear all vert. tabs | ISO | | x | |
| #4 | Clear all h & v tabs | *** | | x | x |
| #5 | Set default tabs | *** | | x | x |
| [n"x | (Extended commands) | *** | | | |

Printer name:

CBM Epson Epson
MPS-1000 JX-80 X-80

| Code* | Description | Defined | | | |
|-------|---------------------------------------|---------|---|---|---|
| c | Reset | ISO | x | x | x |
| #1 | Initialize | *** | x | x | x |
| D | Line feed | ISO | x | x | x |
| E | Line feed, CR | ISO | x | x | x |
| M | Reverse line feed | ISO | | x | |
| [0m | Normal char. set | ISO | x | x | x |
| [3m | Italics on | ISO | | x | x |
| [23m | Italics off | ISO | | x | x |
| [4m | Underline on | ISO | x | x | x |
| [24m | Underline off | ISO | x | x | x |
| [1m | Boldface on | ISO | x | x | x |
| [22m | Boldface off | ISO | x | x | x |
| [nm | Set foreground color (n = {30-39}) | ISO | | x | |
| [nm | Set background color (n = {40-49}) | ISO | | | |
| [0w | Normal pitch | DEC | x | x | x |
| [2w | Elite on | DEC | x | x | x |
| [1w | Elite off | DEC | x | x | x |
| [4w | Condensed fine on | DEC | x | x | x |
| [3w | Condensed off | DEC | x | x | x |
| [6w | Enlarged on | DEC | x | x | x |
| [5w | Enlarged off | DEC | x | x | x |
| [6"z | Shadow print on | DEC | | | |
| [5"z | Shadow print off | DEC | | | |
| [4"z | Doublestrike on | DEC | x | x | x |
| [3"z | Doublestrike off | DEC | x | x | x |
| [2"z | NLQ on ** | DEC | x | x | x |
| [1"z | NLQ off ** | DEC | x | x | x |

- * Entire escape sequence consists of ESC (ASCII 27) plus indicated code.
- ** Near Letter Quality
- *** Sequence unique to Amiga

| Printer name: | | CBM | Epson | Epson | |
|---------------|-------------------------|----------|-------|-------|---|
| | | MPS-1000 | JX-80 | X-80 | |
| Code | Description | Defined | | | |
| [2v | Superscript on | *** | x | x | x |
| [1v | Superscript off | *** | x | x | x |
| [4v | Subscript on | *** | x | x | x |
| [3v | Subscript off | *** | x | x | x |
| [0v | Normalize the line | *** | x | x | x |
| L | Partial line up | ISO | x | x | x |
| K | Partial line down | ISO | x | x | x |
| (B | U.S. char. set | DEC | x | x | x |
| (R | French " " | DEC | x | x | x |
| (K | German " " | DEC | x | x | x |
| (A | UK " " | DEC | x | x | x |
| (E | Danish I " " | DEC | x | x | x |
| (H | Swedish " " | DEC | x | x | x |
| (Y | Italian " " | DEC | x | x | x |
| (Z | Spanish " " | DEC | x | x | x |
| (J | Japanese " " | *** | x | x | x |
| (6 | Norwegian " " | DEC | x | x | x |
| (C | Danish II " " | *** | x | x | x |
| [2p | Proportional on | *** | x | x | x |
| [1p | Proportional off | *** | x | x | x |
| [0p | Proportional clear | *** | | | |
| [n E | Set prop. offset (n) | ISO | | | |
| [5 F | Auto left justify | ISO | | x | x |
| [7 F | Auto right justify | ISO | | x | x |
| [6 F | Auto full justify | ISO | x | x | x |
| [0 F | Justify off | ISO | x | x | x |
| [3 F | Letter space (justify) | ISO | | x | x |
| [1 F | Word fill (auto center) | ISO | | x | x |
| [0z | 1/8" line spacing | *** | x | x | x |
| [1z | 1/6" line spacing | *** | x | x | x |
| [nt | Set form length (n) | DEC | x | x | x |
| [nq | Perf skip (n>0)* | *** | x | x | x |
| [0q | Perf skip off | *** | x | x | x |

*Paper perforation skip, n lines

Printer name:

| Code | Description | Defined | CBM MPS-1000 | Epson JX-80 | Epson X-80 |
|---------|----------------------|---------|-----------------|----------------|---------------|
| #9 | Left margin set | *** | | | |
| #0 | Right margin set | *** | | | |
| #8 | Top margin set | *** | | | |
| #2 | Bottom margin set | *** | | | |
| [n1;n2r | Top;Bottom margins | DEC | | | |
| [n1;n2s | Left;Right margins | DEC | x | x | x |
| #3 | Clear margins | *** | x | x | x |
| H | Set horiz. tab | ISO | | | |
| J | Set vert. tab | ISO | | | |
| [0g | Clear horiz. tab | ISO | | | |
| [3g | Clear all hor. tabs | ISO | x | x | x |
| [1g | Clear vert. tab | ISO | | | |
| [4g | Clear all vert. tabs | ISO | x | x | x |
| #4 | Clear all h & v tabs | *** | x | x | x |
| #5 | Set default tabs | *** | x | x | x |
| [n"x | (Extended commands) | *** | | | |

Printer name:

Diablo Advantage25 Diablo C-150

Okimate 20

| Code* | Description | Defined | | | |
|-------|---------------------------------------|---------|---|---|---|
| c | Reset | ISO | x | x | x |
| #1 | Initialize | *** | x | x | x |
| D | Line feed | ISO | | | |
| E | Line feed, CR | ISO | x | x | x |
| M | Reverse line feed | ISO | x | | |
| [0m | Normal char. set | ISO | x | | x |
| [3m | Italics on | ISO | | | x |
| [23m | Italics off | ISO | | | x |
| [4m | Underline on | ISO | x | | x |
| [24m | Underline off | ISO | x | | x |
| [1m | Boldface on | ISO | x | | |
| [22m | Boldface off | ISO | x | | |
| [nm | Set foreground color (n = {30-39}) | ISO | x | x | |
| [nm | Set background color (n = {40-49}) | ISO | | x | |
| [0w | Normal pitch | DEC | x | | x |
| [2w | Elite on | DEC | x | | x |
| [1w | Elite off | DEC | x | | x |
| [4w | Condensed fine on | DEC | x | | x |
| [3w | Condensed off | DEC | x | | x |
| [6w | Enlarged on | DEC | | | x |
| [5w | Enlarged off | DEC | | | x |
| [6"z | Shadow print on | DEC | x | | |
| [5"z | Shadow print off | DEC | x | | |
| [4"z | Doublestrike on | DEC | x | | |
| [3"z | Doublestrike off | DEC | x | | |
| [2"z | NLQ on ** | DEC | | | x |
| [1"z | NLQ off ** | DEC | | | x |

* Entire escape sequence consists of ESC (ASCII 27) plus indicated code.

** Near Letter Quality

*** Sequence unique to Amiga

Printer name:

Diablo Advantage25 Diablo C-150

Okimate 20

| Code | Description | Defined | | | |
|------|-------------------------|---------|---|---|---|
| [2v | Superscript on | *** | X | X | X |
| [1v | Superscript off | *** | X | X | X |
| [4v | Subscript on | *** | X | X | X |
| [3v | Subscript off | *** | X | X | X |
| [0v | Normalize the line | *** | X | X | X |
| L | Partial line up | ISO | X | | X |
| K | Partial line down | ISO | X | | X |
| (B | U.S. char. set | DEC | | | |
| (R | French " " | DEC | | | |
| (K | German " " | DEC | | | |
| (A | UK " " | DEC | | | |
| (E | Danish I " " | DEC | | | |
| (H | Swedish " " | DEC | | | |
| (Y | Italian " " | DEC | | | |
| (Z | Spanish " " | DEC | | | |
| (J | Japanese " " | *** | | | |
| (6 | Norwegian " " | DEC | | | |
| (C | Danish II " " | *** | | | |
| [2p | Proportional on | *** | X | | X |
| [1p | Proportional off | *** | X | | X |
| [0p | Proportional clear | *** | X | | X |
| [n E | Set prop. offset (n) | ISO | X | | |
| [5 F | Auto left justify | ISO | X | | |
| [7 F | Auto right justify | ISO | | | |
| [6 F | Auto full justify | ISO | | | |
| [0 F | Justify off | ISO | X | | |
| [3 F | Letter space (justify) | ISO | | | |
| [1 F | Word fill (auto center) | ISO | | | |
| [0z | 1/8" line spacing | *** | X | | X |
| [1z | 1/6" line spacing | *** | X | | X |
| [nt | Set form length (n) | DEC | X | X | X |
| [nq | Perf skip (n>0)* | *** | X | | X |
| [0q | Perf skip off | *** | X | | X |

*Paper perforation skip, n lines

Printer name:

Diablo Advantage25 Diablo C-150

Okimate 20

| Code | Description | Defined | | |
|---------|----------------------|---------|---|---|
| #9 | Left margin set | *** | x | x |
| #0 | Right margin set | *** | x | x |
| #8 | Top margin set | *** | x | |
| #2 | Bottom margin set | *** | x | |
| [n1;n2t | Top;Bottom margins | DEC | | |
| [n1;n2s | Left;Right margins | DEC | x | x |
| #3 | Clear margins | *** | x | x |
| H | Set horiz. tab | ISO | x | x |
| J | Set vert. tab | ISO | x | |
| [0g | Clear horiz. tab | ISO | x | x |
| [3g | Clear all hor. tabs | ISO | x | x |
| [1g | Clear vert. tab | ISO | x | |
| [4g | Clear all vert. tabs | ISO | | |
| #4 | Clear all h & v tabs | *** | x | x |
| #5 | Set default tabs | *** | x | x |
| [n"x | (Extended commands) | *** | | |

Printer name:

HP LaserJet HP LaserJet Plus LetterPro 20 Qume

| Code* | Description | Defined | | | |
|-------|---------------------------------------|---------|---|---|---|
| c | Reset | ISO | x | x | x |
| #1 | Initialize | *** | x | x | x |
| D | Line feed | ISO | x | x | x |
| E | Line feed, CR | ISO | x | x | x |
| M | Reverse line feed | ISO | | | x |
| [0m | Normal char. set | ISO | x | x | x |
| [3m | Italics on | ISO | x | x | |
| [23m | Italics off | ISO | x | x | |
| [4m | Underline on | ISO | x | x | x |
| [24m | Underline off | ISO | x | x | x |
| [1m | Boldface on | ISO | x | x | x |
| [22m | Boldface off | ISO | x | x | x |
| [nm | Set foreground color (n = {30-39}) | ISO | | | x |
| [nm | Set background color (n = {40-49}) | ISO | | | |
| [0w | Normal pitch | DEC | x | x | x |
| [2w | Elite on | DEC | x | x | x |
| [1w | Elite off | DEC | x | x | x |
| [4w | Condensed fine on | DEC | x | x | x |
| [3w | Condensed off | DEC | x | x | x |
| [6w | Enlarged on | DEC | | | |
| [5w | Enlarged off | DEC | | | |
| [6"z | Shadow print on | DEC | | | x |
| [5"z | Shadow print off | DEC | | | x |
| [4"z | Doublestrike on | DEC | x | x | x |
| [3"z | Doublestrike off | DEC | x | x | x |
| [2"z | NLQ on ** | DEC | | | |
| [1"z | NLQ off ** | DEC | | | |

* Entire escape sequence consists of ESC (ASCII 27) plus indicated code.

** Near Letter Quality

*** Sequence unique to Amiga

Printer name:

HP
LaserJet

HP
LaserJet Plus LetterPro 20

Qume

| Code | Description | Defined | | | |
|------|-------------------------|---------|---|---|---|
| [2v | Superscript on | *** | X | X | X |
| [1v | Superscript off | *** | X | X | X |
| [4v | Subscript on | *** | X | X | X |
| [3v | Subscript off | *** | X | X | X |
| [0v | Normalize the line | *** | X | X | X |
| L | Partial line up | ISO | X | X | X |
| K | Partial line down | ISO | X | X | X |
| (B | U.S. char. set | DEC | X | X | |
| (R | French " " | DEC | | | |
| (K | German " " | DEC | | | |
| (A | UK " " | DEC | X | X | |
| (E | Danish I " " | DEC | | | |
| (H | Swedish " " | DEC | | | |
| (Y | Italian " " | DEC | | | |
| (Z | Spanish " " | DEC | | | |
| (J | Japanese " " | *** | X | X | |
| (6 | Norwegian " " | DEC | | | |
| (C | Danish II " " | *** | | | |
| [2p | Proportional on | *** | X | X | X |
| [1p | Proportional off | *** | X | X | X |
| [0p | Proportional clear | *** | X | X | X |
| [n E | Set prop. offset (n) | ISO | | | X |
| [5 F | Auto left justify | ISO | | | |
| [7 F | Auto right justify | ISO | | | |
| [6 F | Auto full justify | ISO | | | |
| [0 F | Justify off | ISO | X | | |
| [3 F | Letter space (justify) | ISO | | | |
| [1 F | Word fill (auto center) | ISO | | | |
| [0z | 1/8" line spacing | *** | X | X | X |
| [1z | 1/6" line spacing | *** | X | X | X |
| [nt | Set form length (n) | DEC | X | X | X |
| [nq | Perf skip (n>0)* | *** | X | X | |
| [0q | Perf skip off | *** | X | X | |

*Paper perforation skip, n lines

Printer name:

HP
LaserJet

HP
LaserJet Plus LetterPro 20

Qume

| Code | Description | Defined | | | |
|---------|----------------------|---------|---|---|---|
| #9 | Left margin set | *** | | | x |
| #0 | Right margin set | *** | | | x |
| #8 | Top margin set | *** | | | |
| #2 | Bottom margin set | *** | | | |
| [n1;n2r | Top;Bottom margins | DEC | x | x | x |
| [n1;n2s | Left;Right margins | DEC | x | x | |
| #3 | Clear margins | *** | x | x | x |
| H | Set horiz. tab | ISO | | | |
| J | Set vert. tab | ISO | | | |
| [0g | Clear horiz. tab | ISO | | | |
| [3g | Clear all hor. tabs | ISO | | | |
| [1g | Clear vert. tab | ISO | | | |
| [4g | Clear all vert. tabs | ISO | | | |
| #4 | Clear all h & v tabs | *** | | | |
| #5 | Set default tabs | *** | | | |
| [Pn"x | (Extended commands) | *** | | | |



Appendix J

Software Memory Map

This appendix is for the convenience of the software developer who may be directly accessing the system hardware registers. It is simply the hardware memory map appropriate to the ROM Kernel manual.

A true software memory map, showing system utilization of the various sections of RAM and free space is not provided. The system is dynamically allocated and linked such that it would not be possible to show precisely which parts of RAM are utilized by the ROM Kernel. User code, if written with the Amiga Assembler or Amiga C, and linked by the Amiga linker is relocatable and can load and execute wherever there is a large enough area of memory in which it can fit.

Therefore, aside from specifying that Exec manages the lowest parts of the 68000 memory space (exception and trap vectors), no actual software memory utilization map can be provided.

SYSTEM MEMORY MAP -

This system memory map for the ROM Kernel manual is a combination of the Appendicies D (mem.map) and F (8520 info) from the Amiga PC Hardware Manual. Actual bit assignments for the 8520's and current additional details can be found in the Amiga Hardware Manual.

Note: If you select to read or write an address that is not specifically decoded, you WILL generate an address error.

| ADDRESS RANGE | NOTES |
|--|--|
| | [256k RAM] |
| 000000-3FFFFFF | RAM space for 256k RAM |
| 040000-080000 | Not used if extra 256k board not installed. Do NOT access in this range - dangerous side effects. |
| | ----- |
| | [512k RAM] |
| 000000-07FFFF | RAM space for 512k RAM |
| 080000-1FFFFFF | Do NOT access in this range. |
| | ----- |
| 200000-9FFFFFF | Expansion space (8 megabytes) |
| | ----- |
| A00000-BFFFFFF | External decoder expansion space Reserved for future use. See note (1) below. |
| | ----- |
| <u>B</u> <u>F</u> D000- <u>B</u> <u>F</u> DE00 | 8520-B (accessed only at EVEN byte addresses) |

The underlined digit chooses which of the 16 internal registers of the 8520 is to be accessed. See also note (5)

| ADDRESS RANGE | NOTES |
|--------------------------------|--------------------------|
| <u>B</u> FE001- <u>B</u> FEF01 | 8520-A (accessed only at |
| = | = |
| | ODD byte addresses) |

The underlined digit chooses which of the 16 internal registers of the 8520 is to be accessed.

Register Names are given in note (2) below.

Other addresses in the range of:

| | |
|---------------|---|
| C00000-DEFFFF | Reserved for future use |
| DEF000-DEFFFF | Special purpose chips, where the last three digits specify the chip register WORD address. |
| | The chip addresses are specified in separate pages immediately following this overall memory map. |
| E00000-E7FFFF | Reserved for future use. |
| E80000-FFFFFF | Expansion Slot decoding, see note (1) below. |
| F00000-F7FFFF | Reserved for future use. |
| F80000-FFFFFF | SYSTEM ROM or kickstart RAM. |

=====

NOTES FOR THE SYSTEM MEMORY MAP:

(1) Expansion Slot decoding:

Boards designed to respond in this range must adhere to the auto-configuration guidelines to be published in Dec. 1985 by Commodore-Amiga Inc.

(2) The names of the registers within the 8520's are as follows. The address at which each are to be accessed (per note 1 above) is given here in this list.

Address for:

| 8520-A | 8520-B | NAME | EXPLANATION |
|--------|--------|--------|-----------------------------|
| | | | (write)/(read mode) |
| BFE001 | BED000 | PRA | Peripheral Data Register A |
| BFE101 | BED100 | PRB | Peripheral Data Register B |
| BFE201 | BED200 | DDRB | Data Direction Register "A" |
| BFE301 | BED300 | DDRA | Data Direction Register "B" |
| BFE401 | BED400 | TALO | TIMER A Low Register |
| BFE501 | BED500 | TAHI | TIMER A High Register |
| BFE601 | BED600 | TBLO | TIMER B Low Register |
| BFE701 | BED700 | TBHI | TIMER B High Register |
| BFE801 | BED800 | TODLO | Low TOD Clock |
| BFE901 | BED900 | TODMID | Mid TOD Clock |
| BFEA01 | BEDA00 | TODHI | High TOD Clock Register |
| BFEB01 | BEDB00 | ----- | Unused |
| BFEC01 | BEDC00 | SDR | Serial Data Register |
| BFED01 | BEDD00 | ICR | Interrupt Control Register |
| BFEE01 | BEDE00 | CRA | Control Register A |
| BFEF01 | BEDE00 | CRB | Control Register B |

SPECIAL PURPOSE CHIP ADDRESSES

The following are the "offset" addresses for the special purpose chips in the Amiga PC. Each address figure shown below must be added to the base address of hex DEF000. Again the addressing follows the convention shown in note 1 of the system memory map.

Each register is located at an even WORD (16-bit) boundary.

| NAME | OFFSET | R/W | EXPLANATION |
|---------|--------|-----|--|
| ADKCON | 09E | W | Audio, Disk, Control write |
| ADKCONR | 010 | R | Audio, disk, Control read |
| AUDODAT | 0AA | W | Audio channel 0 Data |
| AUDOLCH | 0A0 | W | Audio channel 0 location (High 3 bits) |
| AUDOLCL | 0A2 | W | Audio channel 0 location (Low 16 bits) |
| AUDOLEN | 0A4 | W | Audio Channel 0 length |
| AUDOPER | 0A6 | W | Audio channel 0 Period |
| AUDOVOL | 0A8 | W | Audio Channel 0 Volume |

| NAME | OFFSET | R/W | EXPLANATION |
|----------|--------|-----|--|
| AUD1DAT | 0BA | W | Audio channel 1 Data |
| AUD1LCH | 0B0 | W | Audio channel 1 location (High 3 bits) |
| AUD1LCL | 0B2 | W | Audio channel 1 location (Low 16 bits) |
| AUD1LEN | 0B4 | W | Audio Channel 1 length |
| AUD1PER | 0B6 | W | Audio channel 1 Period |
| AUD1VOL | 0B8 | W | Audio Channel 1 Volume |
| AUD2DAT | 0CA | W | Audio channel 2 Data |
| AUD2LCH | 0C0 | W | Audio channel 2 location (High 3 bits) |
| AUD2LCL | 0C2 | W | Audio channel 2 location (Low 16 bits) |
| AUD2LEN | 0C4 | W | Audio Channel 2 length |
| AUD2PER | 0C6 | W | Audio channel 2 Period |
| AUD2VOL | 0C8 | W | Audio Channel 2 Volume |
| AUD3DAT | 0DA | W | Audio channel 3 Data |
| AUD3LCH | 0D0 | W | Audio channel 3 location (High 3 bits) |
| AUD3LCL | 0D2 | W | Audio channel 3 location (Low 16 bits) |
| AUD3LEN | 0D4 | W | Audio Channel 3 length |
| AUD3PER | 0D6 | W | Audio channel 3 Period |
| AUD3VOL | 0D8 | W | Audio Channel 3 Volume |
| BLTAFWM | 044 | W | Blitter first word mask for source A |
| BLTALWM | 046 | W | Blitter last word mask for source A |
| BLTCON0 | 040 | W | Blitter control register 0 |
| BLTCON1 | 042 | W | Blitter control register 1 |
| BLTSIZE | 058 | W | Blitter start and size (window width, height) |
| BLTADAT | 074 | W | Blitter source A data reg |
| BLTAMOD | 064 | W | Blitter Modulo A |
| BLTAPTH | 050 | W | Blitter Pointer to src or dst.A (High 3 bits) |
| BLTAPTL | 052 | W | Blitter Pointer A (Low 16 bits) |
| BLTBDAT | 072 | W | Blitter source B data reg |
| BLTBMOD | 062 | W | Blitter Modulo B |
| BLTBPTH | 04C | W | Blitter Pointer to src or dst.B (High 3 bits) |
| BLTBPTL | 04E | W | Blitter Pointer B (Low 16 bits) |
| BLTCDAT | 070 | W | Blitter source C data reg |
| BLTCMOD | 060 | W | Blitter Modulo C |
| BLTCPPTH | 048 | W | Blitter Pointer to src or dst.C (High 3 bits) |
| BLTCPPTL | 04A | W | Blitter Pointer C (Low 16 bits) |
| BLTDMOD | 066 | W | Blitter Modulo D |
| BLTDPTH | 054 | W | Blitter Pointer to src or dst.D (High 3 bits) |
| BLTDPTL | 056 | W | Blitter Pointer D (Low 16 bits) |
| BPL1MOD | 108 | W | Bit plane modulo (odd planes) |
| BPL2MOD | 10A | W | Bit Plane modulo (even planes) |

| NAME | OFFSET | R/W | EXPLANATION |
|---------|--------|-----|---|
| BPLCON0 | 100 | W | Bit plane control reg. (misc control bits) |
| BPLCON1 | 102 | W | Bit plane control reg. (priority control) |
| BPLCON2 | 104 | W | Bit Plane control reg. (horiz scroll control) |
| BPL1DAT | 110 | W | Bit plane 1 data (Parallel to serial convert) |
| BPL2DAT | 112 | W | Bit plane 2 data (Parallel to serial convert) |
| BPL3DAT | 114 | W | Bit plane 3 data (Parallel to serial convert) |
| BPL4DAT | 116 | W | Bit plane 4 data (Parallel to serial convert) |
| BPL5DAT | 118 | W | Bit plane 5 data (Parallel to serial convert) |
| BPL6DAT | 11A | W | Bit plane 6 data (Parallel to serial convert) |
| BPL1PTH | 0E0 | W | Bit plane 1 pointer (High 3 bits) |
| BPL1PTL | 0E2 | W | Bit plane 1 pointer (Low 16 bits) |
| BPL2PTH | 0E4 | W | Bit plane 2 pointer (High 3 bits) |
| BPL2PTL | 0E6 | W | Bit plane 2 pointer (Low 16 bits) |
| BPL3PTH | 0E8 | W | Bit plane 3 pointer (High 3 bits) |
| BPL3PTL | 0EA | W | Bit plane 3 pointer (Low 16 bits) |
| BPL4PTH | 0EC | W | Bit plane 4 pointer (High 3 bits) |
| BPL4PTL | 0EE | W | Bit plane 4 pointer (Low 16 bits) |
| BPL5PTH | 0F0 | W | Bit plane 5 pointer (High 3 bits) |
| BPL5PTL | 0F2 | W | Bit plane 5 pointer (Low 16 bits) |
| BPL6PTH | 0F4 | W | Bit plane 6 pointer (High 3 bits) |
| BPL6PTL | 0F6 | W | Bit plane 6 pointer (Low 16 bits) |
| CLXCON | 098 | W | Collision control |
| CLXDAT | 00E | R | Collision data reg. (Read and clear) |
| COLORxx | 180 | W | Color table xx (32 WORD ENTRIES, START COLOR 00) |
| COP1LCH | 080 | W | Coprocessor first location reg. (High 3 bits) |
| COP1LCL | 082 | W | Coprocessor first location reg. (Low 16 bits) |
| COP2LCH | 084 | W | Coprocessor secnd location reg. (High 3 bits) |
| COP2LCL | 086 | W | Coprocessor second location reg. (Low 16 bits) |
| COPINS | 08C | W | Coprocessor inst. fetch identify |
| COPJMP1 | 088 | S | Coprocessor restart at first location |
| COPJMP2 | 08A | S | Coprocessor restart at second location |
| DDESTOP | 094 | W | Display bit plane data fetch stop (hor pos) |
| DDESTRT | 092 | W | Display bit plane data fetch start (hor pos) |

| NAME | OFFSET | R/W | EXPLANATION |
|----------|--------|-----|--|
| DIWSTOP | 090 | W | Disp Window Stop (lower right vert-hor pos) |
| DIWSTRT | 08E | W | Disp Window Start (upper left vert-hor pos) |
| DMACON | 096 | W | DMA control write (clear or set) |
| DMACONR | 002 | R | DMA control (and blitter status) read |
| DSKBYTR | 01A | R | Disk Data byte and status read |
| DSKDAT | 026 | W | Disk DMA Data write |
| DSKDATR | 008 | ER | Disk DMA Data read (early read dummy address) |
| DSKLEN | 024 | W | Disk length |
| DSKPTH | 020 | W | Disk pointer (High 3 bits) |
| DSKPTL | 022 | W | Disk pointer (Low 16 bits) |
| INTENA | 09A | W | Interrupt Enable bits (clear or set bits) |
| INTENAR | 01C | R | Interrupt Enable bits Read |
| INTREQ | 09C | W | Interrupt Request bits (clear or set) |
| INTREQR | 01E | R | Interrupt request bits (read) |
| JOY0DAT | 00A | R | Joystick-mouse 0 data (vert,horiz) |
| JOY1DAT | 00C | R | Joystick-mouse 1 data (vert,horiz) |
| POT0DAT | 012 | R | Pot counter data left pair (vert,horiz) |
| POT1DAT | 014 | R | Pot counter data right pair (vert,horiz) |
| POTGO | 034 | W | Pot Port (4 bit) Direction and Data, |
| POTINP | 016 | R | Pot pin data read |
| REFPTR | 028 | W | Refresh pointer |
| SERDAT | 030 | W | Serial Port Data and stop bits write |
| SERDATR | 018 | R | Serial Port Data and Status read |
| SERPER | 032 | W | Serial Port Period and control |
| SPR0POS | 140 | W | Sprite 0 Vert-Horiz start position data |
| SPR0CTL | 142 | W | Sprite 0 Vert stop position and control data |
| SPR0DATA | 144 | W | Sprite 0 image data register A |
| SPR0DATB | 146 | W | Sprite 0 image data register B |
| SPR1POS | 148 | W | Sprite 1 Vert-Horiz start position data |
| SPR1CTL | 14A | W | Sprite 1 Vert stop position and control data |
| SPR1DATA | 14C | W | Sprite 1 image data register A |
| SPR1DATB | 14E | W | Sprite 1 image data register B |

| NAME | OFFSET | R/W | EXPLANATION |
|----------|--------|-----|--|
| SPR2POS | 150 | W | Sprite 2 Vert-Horiz start position data |
| SPR2CTL | 152 | W | Sprite 2 Vert stop position and control data |
| SPR2DATA | 154 | W | Sprite 2 image data register A |
| SPR2DATB | 156 | W | Sprite 2 image data register B |
| SPR3POS | 158 | W | Sprite 3 Vert-Horiz start position data |
| SPR3CTL | 15A | W | Sprite 3 Vert stop position and control data |
| SPR3DATA | 15C | W | Sprite 3 image data register A |
| SPR3DATB | 15E | W | Sprite 3 image data register B |
| SPR4POS | 160 | W | Sprite 4 Vert-Horiz start position data |
| SPR4CTL | 162 | W | Sprite 4 Vert stop position and control data |
| SPR4DATA | 164 | W | Sprite 4 image data register A |
| SPR4DATB | 166 | W | Sprite R image data register B |
| SPR5POS | 168 | W | Sprite 5 Vert-Horiz start position data |
| SPR5CTL | 16A | W | Sprite 5 Vert stop position and control data |
| SPR5DATA | 16C | W | Sprite 5 image data register A |
| SPR5DATB | 16E | W | Sprite 5 image data register B |
| SPR6POS | 170 | W | Sprite 6 Vert-Horiz start position data |
| SPR6CTL | 172 | W | Sprite 6 Vert stop position and control data |
| SPR6DATA | 174 | W | Sprite 6 image data register A |
| SPR6DATB | 176 | W | Sprite 6 image data register B |
| SPR7POS | 178 | W | Sprite 7 Vert-Horiz start position data |
| SPR7CTL | 17A | W | Sprite 7 Vert stop position and control data |
| SPR7DATA | 17C | W | Sprite 7 image data register A |
| SPR7DATB | 17E | W | Sprite 7 image data register B |
| SPR0PTH | 120 | W | Sprite 0 pointer (High 3 bits) |
| SPR0PTL | 122 | W | Sprite 0 pointer (Low 16 bits) |
| SPR1PTH | 124 | W | Sprite 1 pointer (High 3 bits) |
| SPR1PTL | 126 | W | Sprite 1 pointer (Low 16 bits) |
| SPR2PTH | 128 | W | Sprite 2 pointer (High 3 bits) |
| SPR2PTL | 12A | W | Sprite 2 pointer (Low 16 bits) |
| SPR3PTH | 12C | W | Sprite 3 pointer (High 3 bits) |
| SPR3PTL | 12E | W | Sprite 3 pointer (Low 16 bits) |
| SPR4PTH | 130 | W | Sprite 4 pointer (High 3 bits) |
| SPR4PTL | 132 | W | Sprite 4 pointer (Low 16 bits) |
| SPR5PTH | 134 | W | Sprite 5 pointer (High 3 bits) |
| SPR5PTL | 136 | W | Sprite 5 pointer (Low 16 bits) |
| SPR6PTH | 138 | W | Sprite 6 pointer (High 3 bits) |
| SPR6PTL | 13A | W | Sprite 6 pointer (Low 16 bits) |
| SPR7PTH | 13C | W | Sprite 7 pointer (High 3 bits) |
| SPR7PTL | 13E | W | Sprite 7 pointer (Low 16 bits) |

| NAME | OFFSET | R/W | EXPLANATION |
|---------|--------|-----|--|
| STREQU | 038 | S | Strobe for horiz sync with VB and EQU |
| STRHOR | 03C | S | Strobe for horiz sync |
| STRLONG | 03E | S | Strobe for identification of long horizontal line. |
| STRVBL | 03A | S | Strobe for horiz sync with VB (Vert. Blank) |
| VHPOSR | 004 | R | Read Vert and horiz Position of beam |
| VHPOSW | 02A | W | Write Vert and horiz Position of beam |
| VPOSR | 006 | R | Read Vert most sig. bit (and frame flop) |
| VPOSW | 02C | W | Write Vert most sig. bit (and frame flop) |

The following listing of the chip special purpose addresses is provided in numerical order by chip address for the convenience of software developers who may prefer this ordering sequence.

| | | | |
|---------|-----|----|---|
| DMACONR | 002 | R | DMA control (and blitter status) read |
| VHPOSR | 004 | R | Read Vert and horiz Position of beam |
| VPOSR | 006 | R | Read Vert most sig. bit (and frame flop) |
| DSKDATR | 008 | ER | Disk DMA Data read (early read dummy address) |
| JOY0DAT | 00A | R | Joystick-mouse 0 data (vert,horiz) |
| JOY1DAT | 00C | R | Joystick-mouse 1 data (vert,horiz) |
| CLXDAT | 00E | R | Collision data reg. (Read and clear) |
| ADKCONR | 010 | R | Audio, disk, Control read |
| POT0DAT | 012 | R | Pot counter data left pair (vert,horiz) |
| POT1DAT | 014 | R | Pot counter data right pair (vert,horiz) |
| POTINP | 016 | R | Pot pin data read |
| SERDATR | 018 | R | Serial Port Data and Status read |
| DSKBYTR | 01A | R | Disk Data byte and status read |
| INTENAR | 01C | R | Interrupt Enable bits Read |
| INTREQR | 01E | R | Interrupt request bits (read) |
| DSKPTH | 020 | W | Disk pointer (High 3 bits) |
| DSKPTL | 022 | W | Disk pointer (Low 16 bits) |
| DSKLEN | 024 | W | Disk length |
| DSKDAT | 026 | W | Disk DMA Data write |
| REFPTR | 028 | W | Refresh pointer |
| VHPOSW | 02A | W | Write Vert and horiz Position of beam |
| VPOSW | 02C | W | Write Vert most sig. bit (and frame flop) |
| SERDAT | 030 | W | Serial Port Data and stop bits write |
| SERPER | 032 | W | Serial Port Period and control |
| POTGO | 034 | W | Pot Port (4 bit) Direction and Data, |
| STREQU | 038 | S | Strobe for horiz sync with VB and EQU |
| STRVBL | 03A | S | Strobe for horiz sync with VB (Vert. Blank) |
| STRHOR | 03C | S | Strobe for horiz sync |
| STRLONG | 03E | S | Strobe for ident. of long horizontal line. |
| BLTCON0 | 040 | W | Blitter control register 0 |
| BLTCON1 | 042 | W | Blitter control register 1 |
| BLTAFWM | 044 | W | Blitter first word mask for source A |
| BLTALWM | 046 | W | Blitter last word mask for source A |
| BLTCPTR | 048 | W | Blitter Pointer to src or dst.C (High 3 bits) |

| | | | |
|---------|-----|---|---|
| BLTCPTL | 04A | W | Blitter Pointer C (Low 16 bits) |
| BLTBPTH | 04C | W | Blitter Pointer to src or dst.B (High 3 bits) |
| BLTBPTL | 04E | W | Blitter Pointer B (Low 16 bits) |
| BLTAPTH | 050 | W | Blitter Pointer to src or dst.A (High 3 bits) |
| BLTAPTL | 052 | W | Blitter Pointer A (Low 16 bits) |
| BLTDPTH | 054 | W | Blitter Pointer to src or dst.D (High 3 bits) |
| BLTDPTL | 056 | W | Blitter Pointer D (Low 16 bits) |
| BLTSIZE | 058 | W | Blitter start and size (window width, height) |
| BLTCMOD | 060 | W | Blitter Modulo C |
| BLTBMOD | 062 | W | Blitter Modulo B |
| BLTAMOD | 064 | W | Blitter Modulo A |
| BLTDMOD | 066 | W | Blitter Modulo D |
| BLTCDAT | 070 | W | Blitter source C data reg |
| BLTBDAT | 072 | W | Blitter source B data reg |
| BLTADAT | 074 | W | Blitter source A data reg |
| COP1LCH | 080 | W | Coprocessor first location reg (High 3 bits) |
| COP1LCL | 082 | W | Coprocessor first location reg. (Low 16 bits) |
| COP2LCH | 084 | W | Coprocessor secnd location reg. (High 3 bits) |
| COP2LCL | 086 | W | Coprocessor second location reg (low 16 bits) |
| COPJMP1 | 088 | S | Coprocessor restart at first location |
| COPJMP2 | 08A | S | Coprocessor restart at second location |
| COPINS | 08C | W | Coprocessor inst. fetch identify |
| DIWSTRT | 08E | W | Disp Window Start (upper left vert-hor pos) |
| DIWSTOP | 090 | W | Disp Window Stop (lower right vert-hor pos) |
| DDFSTRT | 092 | W | Display bit plane data fetch start (hor pos) |
| DDFSTOP | 094 | W | Display bit plane data fetch stop (hor pos) |
| DMACON | 096 | W | DMA control write (clear or set) |
| CLXCON | 098 | W | Collision control |
| INTENA | 09A | W | Interrupt Enable bits (clear or set bits) |
| INTREQ | 09C | W | Interrupt Request bits (clear or set) |
| ADKCON | 09E | W | Audio, Disk, Control write |
| AUD0LCH | 0A0 | W | Audio channel 0 location (High 3 bits) |
| AUD0LCL | 0A2 | W | Audio channel 0 location (Low 16 bits) |
| AUD0LEN | 0A4 | W | Audio Channel 0 length |
| AUD0PER | 0A6 | W | Audio channel 0 Period |
| AUD0VOL | 0A8 | W | Audio Channel 0 Volume |
| AUD0DAT | 0AA | W | Audio channel 0 Data |
| AUD1LCH | 0B0 | W | Audio channel 1 location (High 3 bits) |
| AUD1LCL | 0B2 | W | Audio channel 1 location (Low 16 bits) |
| AUD1LEN | 0B4 | W | Audio Channel 1 length |
| AUD1PER | 0B6 | W | Audio channel 1 Period |
| AUD1VOL | 0B8 | W | Audio Channel 1 Volume |
| AUD1DAT | 0BA | W | Audio channel 1 Data |
| AUD2LCH | 0C0 | W | Audio channel 2 location (High 3 bits) |
| AUD2LCL | 0C2 | W | Audio channel 2 location (Low 16 bits) |
| AUD2LEN | 0C4 | W | Audio Channel 2 length |
| AUD2PER | 0C6 | W | Audio channel 2 Period |
| AUD2VOL | 0C8 | W | Audio Channel 2 Volume |
| AUD2DAT | 0CA | W | Audio channel 2 Data |
| AUD3LCH | 0D0 | W | Audio channel 3 location (High 3 bits) |
| AUD3LCL | 0D2 | W | Audio channel 3 location (Low 16 bits) |
| AUD3LEN | 0D4 | W | Audio Channel 3 length |
| AUD3PER | 0D6 | W | Audio channel 3 Period |
| AUD3VOL | 0D8 | W | Audio Channel 3 Volume |
| AUD3DAT | 0DA | W | Audio channel 3 Data |
| BPL1PTH | 0E0 | W | Bit plane 1 pointer (High 3 bits) |

| | | | |
|----------|-----|---|---|
| BPL1PTL | 0E2 | W | Bit plane 1 pointer (Low 16 bits) |
| BPL2PTH | 0E4 | W | Bit plane 2 pointer (High 3 bits) |
| BPL2PTL | 0E6 | W | Bit plane 2 pointer (Low 16 bits) |
| BPL3PTH | 0E8 | W | Bit plane 3 pointer (High 3 bits) |
| BPL3PTL | 0EA | W | Bit plane 3 pointer (Low 16 bits) |
| BPL4PTH | 0EC | W | Bit plane 4 pointer (High 3 bits) |
| BPL4PTL | 0EE | W | Bit plane 4 pointer (Low 16 bits) |
| BPL5PTH | 0F0 | W | Bit plane 5 pointer (High 3 bits) |
| BPL5PTL | 0F2 | W | Bit plane 5 pointer (Low 16 bits) |
| BPL6PTH | 0F4 | W | Bit plane 6 pointer (High 3 bits) |
| BPL6PTL | 0F6 | W | Bit plane 6 pointer (Low 16 bits) |
| BPLCON0 | 100 | W | Bit plane control reg. (misc control bits) |
| BPLCON1 | 102 | W | Bit plane control reg. (priority control) |
| BPLCON2 | 104 | W | Bit Plane control reg. (horiz scroll control) |
| BPL1MOD | 108 | W | Bit plane modulo (odd planes) |
| BPL2MOD | 10A | W | Bit Plane modulo (even planes) |
| BPL1DAT | 110 | W | Bit plane 1 data (Parallel to serial convert) |
| BPL2DAT | 112 | W | Bit plane 2 data (Parallel to serial convert) |
| BPL3DAT | 114 | W | Bit plane 3 data (Parallel to serial convert) |
| BPL4DAT | 116 | W | Bit plane 4 data (Parallel to serial convert) |
| BPL5DAT | 118 | W | Bit plane 5 data (Parallel to serial convert) |
| BPL6DAT | 11A | W | Bit plane 6 data (Parallel to serial convert) |
| SPR0PTH | 120 | W | Sprite 0 pointer (High 3 bits) |
| SPR0PTL | 122 | W | Sprite 0 pointer (Low 16 bits) |
| SPR1PTH | 124 | W | Sprite 1 pointer (High 3 bits) |
| SPR1PTL | 126 | W | Sprite 1 pointer (Low 16 bits) |
| SPR2PTH | 128 | W | Sprite 2 pointer (High 3 bits) |
| SPR2PTL | 12A | W | Sprite 2 pointer (Low 16 bits) |
| SPR3PTH | 12C | W | Sprite 3 pointer (High 3 bits) |
| SPR3PTL | 12E | W | Sprite 3 pointer (Low 16 bits) |
| SPR4PTH | 130 | W | Sprite 4 pointer (High 3 bits) |
| SPR4PTL | 132 | W | Sprite 4 pointer (Low 16 bits) |
| SPR5PTH | 134 | W | Sprite 5 pointer (High 3 bits) |
| SPR5PTL | 136 | W | Sprite 5 pointer (Low 16 bits) |
| SPR6PTH | 138 | W | Sprite 6 pointer (High 3 bits) |
| SPR6PTL | 13A | W | Sprite 6 pointer (Low 16 bits) |
| SPR7PTH | 13C | W | Sprite 7 pointer (High 3 bits) |
| SPR7PTL | 13E | W | Sprite 7 pointer (Low 16 bits) |
| SPR0POS | 140 | W | Sprite 0 Vert-Horiz start position data |
| SPR0CTL | 142 | W | Sprite 0 Vert stop position and control data |
| SPR0DATA | 144 | W | Sprite 0 image data register A |
| SPR0DATB | 146 | W | Sprite 0 image data register B |
| SPR1POS | 148 | W | Sprite 1 Vert-Horiz start position data |
| SPR1CTL | 14A | W | Sprite 1 Vert stop position and control data |
| SPR1DATA | 14C | W | Sprite 1 image data register A |
| SPR1DATB | 14E | W | Sprite 1 image data register B |
| SPR2POS | 150 | W | Sprite 2 Vert-Horiz start position data |
| SPR2CTL | 152 | W | Sprite 2 Vert stop position and control data |
| SPR2DATA | 154 | W | Sprite 2 image data register A |
| SPR2DATB | 156 | W | Sprite 2 image data register B |
| SPR3POS | 158 | W | Sprite 3 Vert-Horiz start position data |
| SPR3CTL | 15A | W | Sprite 3 Vert stop position and control data |
| SPR3DATA | 15C | W | Sprite 3 image data register A |
| SPR3DATB | 15E | W | Sprite 3 image data register B |
| SPR4POS | 160 | W | Sprite 4 Vert-Horiz start position data |
| SPR4CTL | 162 | W | Sprite 4 Vert stop position and control data |

| | | | |
|----------|-----|---|--|
| SPR4DATA | 164 | W | Sprite 4 image data register A |
| SPR4DATB | 166 | W | Sprite 4 image data register B |
| SPR5POS | 168 | W | Sprite 5 Vert-Horiz start position data |
| SPR5CTL | 16A | W | Sprite 5 Vert stop position and control data |
| SPR5DATA | 16C | W | Sprite 5 image data register A |
| SPR5DATB | 16E | W | Sprite 5 image data register B |
| SPR6POS | 170 | W | Sprite 6 Vert-Horiz start position data |
| SPR6CTL | 172 | W | Sprite 6 Vert stop position and control data |
| SPR6DATA | 174 | W | Sprite 6 image data register A |
| SPR6DATB | 176 | W | Sprite 6 image data register B |
| SPR7POS | 178 | W | Sprite 7 Vert-Horiz start position data |
| SPR7CTL | 17A | W | Sprite 7 Vert stop position and control data |
| SPR7DATA | 17C | W | Sprite 7 image data register A |
| SPR7DATB | 17E | W | Sprite 7 image data register B |
| COLORxx | 180 | W | Color table xx (32 WORD ENTRIES, START COLOR 00) |

Appendix K

Skeleton Device, Skeleton Library

This appendix contains source code for a skeleton device and a skeleton library. You can use this code to create your own custom devices and libraries to add to the Amiga.

```
*****
*
*      Copyright (C) 1985, Commodore Amiga Inc. All rights reserved.
*
*****
```

```
*****
*
* asmsupp.1 -- random low level assembly support routines
*
* Source Control
* -----
*
* $Header: asmsupp.1,v 31.1 85/10/13 23:12:33 neil Exp $
*
* $Locker: $
*
*****
```

```
CLEAR  MACRO          ; quick way to clear a D register on 68000
        MOVEQ    #0,\1
        ENDM

BHS    MACRO
        BCC.\0 \1
        ENDM

BLO    MACRO
        BCS.\0 \1
        ENDM

EVEN   MACRO          ; word align code stream
        DS.W    0
        ENDM

LINKSYS MACRO          ; link to a library without having to see a _LVO
        LINKLIB _LVO\1,\2
        ENDM

CALLSYS MACRO          ; call a library without having to see _LVO
        CALLLIB _LVO\1
        ENDM

XLIB   MACRO          ; define a library reference without the _LVO
        XREF    _LVO\1
        ENDM
```

section section

DEBUG equ 1

```
*****
*
* Copyright (C) 1985, Commodore Amiga Inc. All rights reserved. *
*
*****
```

```
*****
*
* mylib.i -- external declarations for skeleton library
*
* SOURCE CONTROL
* -----
* $Header: ramlib.i,v 31.1 85/10/13 23:12:51 neil Exp $
*
* $Locker: neil $
*
*****
```

```
-----
;
; library function definitions
;
;
-----
```

```
LIBINIT
LIBDEF      MLFUNC0
LIBDEF      MLFUNC1
```

```
-----
;
; library data structures
;
;
-----
```

```
STRUCTURE MyLib, LIB_SIZE
  ULONG   ml_SysLib
  ULONG   ml_DosLib
  ULONG   ml_SeqList
  UBYTE   ml_Flags
  UBYTE   ml_pad
  LABEL   MyLib_Sizeof
```

```
MYLIBNAME      MACRO
                DC.B   'mylib.library',0
                ENDM
```

```
*****  
*  
* Copyright (C) 1985, Commodore Amiga Inc. All rights reserved. *  
*  
*****/
```

```
*****  
*  
* mylib.asm -- skeleton library code  
*  
* Source Control  
* -----  
*  
* $Header: amain.asm,v 31.3 85/10/18 19:04:04 neil Exp $  
*  
* $Locker: neil $  
*  
* $Log: amain.asm,v $  
*  
*****/
```

```
SECTION section
```

```
NOLIST  
include "exec/types.i"  
include "exec/nodes.i"  
include "exec/lists.i"  
include "exec/libraries.i"  
include "exec/alerts.i"  
include "exec/initializers.i"  
include "exec/resident.i"  
include "libraries/dos.i"  
  
include "asmsupp.i"  
  
include "mylib.i"
```

```
LIST
```

```
;----- These don't have to be external, but it helps some  
;----- debuggers to have them globally visible  
XDEF Init  
XDEF Open  
XDEF Close  
XDEF Expunge  
XDEF Null  
XDEF myName  
XDEF MyFunc0  
XDEF MyFunc1  
  
XREF _AbsExecBase  
  
XLIB OpenLibrary  
XLIB CloseLibrary  
XLIB Alert
```

```

XLIB   FreeMem
XLIB   Remove

```

```

; The first executable location. This should return an error
; in case someone tried to run you as a program (instead of
; loading you as a library).

```

```

Start:  CLEAR   d0
        rts

```

```

-----
; A romtag structure. Both "exec" and "ramlib" look for
; this structure to discover magic constants about you
; (such as where to start running you from...).
-----

```

```

; Most people will not need a priority and should leave it at zero.
; the RT_PRI field is used for configuring the roms. Use "mods" from
; wack to look at the other romtags in the system
MYPRI  EQU     0

```

```

initDDescrip:

```

```

        DC.W   RTC_MATCHWORD      ; STRUCTURE RT, 0
        DC.L   initDDescrip      ; UWORD RT_MATCHWORD
        DC.L   EndCode            ; APTR RT_MATCHTAG
        DC.B   RTE_AUTOINIT       ; APTR RT_ENDSKIP
        DC.B   VERSION            ; UBYTE RT_FLAGS
        DC.B   NT_LIBRARY         ; UBYTE RT_VERSION
        DC.B   MYPRI              ; UBYTE RT_TYPE
        DC.L   myName             ; BYTE RT_PRI
        DC.L   idString           ; APTR RT_NAME
        DC.L   Init               ; APTR RT_IDSTRING
        DC.L   Init               ; APTR RT_INIT
        DC.L   Init               ; LABEL RT_SIZE

```

```

myName: ; this is the name that the library will have
        MYLIBNAME

```

```

VERSION: ; a major version number.
        EQU     1

```

```

REVISION: ; A particular revision. This should uniquely identify the bits in the
; library. I use a script that advances the revision number each time
; I recompile. That way there is never a question of which library
; that really is.
        EQU     17

```

```

idString: ; this is an identifier tag to help in supporting the library
; format is 'name version.revision (dd MON yyyy)', <cr>, <lf>, <null>
        dc.b   'mylib 1.0 (31 Oct 1985)', 13, 10, 0

```

```

dosName:  DOSNAME

```



```

; force word alignment
ds.w 0

```

```

; The romtag specified that we were "RTF_AUTOINIT". This means
; that the RT_INIT structure member points to one of these
; tables below. If the AUTOINIT bit was not set then RT_INIT
; would point to a routine to run.

```

```
Init:
```

```

DC.L MyLib_Sizeof ; data space size
DC.L funcTable ; pointer to function initializers
DC.L dataTable ; pointer to data initializers
DC.L initRoutine ; routine to run

```

```
funcTable:
```

```
;----- standard system routines
```

```
dc.l Open
dc.l Close
dc.l Expunge
dc.l Null

```

```
;----- my libraries definitions
```

```
dc.l MyFunc0
dc.l MyFunc1

```

```
;----- function table end marker
```

```
dc.l -1

```

```

; The data table initializes static data structures.
; The format is specified in exec/InitStruct routine's
; manual pages. The INITBYTE/INITWORD/INITLONG routines
; are in the file "exec/initializers.i". The first argument
; is the offset from the library base for this byte/word/long.
; The second argument is the value to put in that cell.
; The table is null terminated

```

```
dataTable:
```

```

INITBYTE LH_TYPE,NT_LIBRARY
INITLONG LN_NAME,myName
INITBYTE LIB_FLAGS,LIBF_SUMUSED!LIBF_CHANGED
INITWORD LIB_VERSION,VERSION
INITWORD LIB_REVISION,REVISION
INITLONG LIB_IDSTRING,idString
DC.L 0

```

```

; This routine gets called after the library has been allocated.
; The library pointer is in D0. The segment list is in A0.
; If it returns non-zero then the library will be linked into
; the library list.

```

```
initRoutine:
```

```
;----- get the library pointer into a convenient A register
```

```

    move.l  a5,-(sp)
    move.l  d0,a5

    ;----- save a pointer to exec
    move.l  a6,ml_SysLib(a5)

    ;----- save a pointer to our loaded code
    move.l  a0,ml_SegList(a5)

    ;----- open the dos library
    lea    dosName(pc),a1
    CLEAR  d0
    CALLSYS  OpenLibrary

    move.l  d0,ml_DosLib(a5)
    bne.s  1$

    ;----- can't open the dos!  what gives
    ALERT  AG_OpenLib!AO_DOSLib

1$:
    ;----- now build the static data that we need

    ;
    ; put your initialization here...
    ;

    move.l  a5,d0
    move.l  (sp)+,a5
    rts

;-----
;
; here begins the system interface commands.  When the user calls
; OpenLibrary/CloseLibrary/RemoveLibrary, this eventually gets translated
; into a call to the following routines (Open/Close/Expunge).  Exec
; has already put our library pointer in A6 for us.  Exec has turned
; off task switching while in these routines (via Forbid/Permit), so
; we should not take too long in them.
;-----

; Open returns the library pointer in d0 if the open
; was successful.  If the open failed then null is returned.
; It might fail if we allocated memory on each open, or
; if only open application could have the library open
; at a time...

Open:          ; ( libptr:a6, version:d0 )

;----- mark us as having another opener
addq.w  #1,LIB_OPENCNT(a6)

;----- prevent delayed expunges
bclr   #LIBB_DELEXP,ml_Flags(a6)

```

```

move.l a6,d0
rts

; There are two different things that might be returned from
; the Close routine. If the library is no longer open and
; there is a delayed expunge then Close should return the
; segment list (as given to Init). Otherwise close should
; return NULL.

Close:      ; ( libptr:a6 )

;----- set the return value
CLEAR      d0

;----- mark us as having one fewer openers
subq.w #1,LIB_OPENCNT(a6)

;----- see if there is anyone left with us open
bne.s 1$

;----- see if we have a delayed expunge pending
btst #LIBB_DELEXP,ml_Flags(a6)
beq.s 1$

;----- do the expunge
bsr Expunge

1$:
rts

; There are two different things that might be returned from
; the Expunge routine. If the library is no longer open
; then Expunge should return the segment list (as given to
; Init). Otherwise Expunge should set the delayed expunge
; flag and return NULL.
;
; One other important note: because Expunge is called from
; the memory allocator, it may NEVER Wait() or otherwise
; take long time to complete.

Expunge:    ; ( libptr: a6 )

movem.l d2/a5/a6,-(sp)
move.l a6,a5
move.l ml_SysLib(a5),a6

;----- see if anyone has us open
tst.w LIB_OPENCNT(a5)
beq 1$

;----- it is still open. set the delayed expunge flag
bset #LIBB_DELEXP,ml_Flags(a5)
CLEAR d0
bra.s Expunge_End

```

```

l$:
;----- go ahead and get rid of us.  Store our seglist in d2
move.l ml_SegList(a5),d2

;----- unlink from library list
move.l a5,a0
CALLSYS Remove

;
; device specific closings here...
;

;----- close the dos library
move.l ml_DosLib(a5),a1
CALLSYS CloseLibrary

;----- free our memory
move.l a5,a1
move.l LIB_NEGSIZE(a5),d0

sub.l d0,a1
add.l LIB_POSSIZE(a5),d0

CALLSYS FreeMem

;----- set up our return value
move.l d2,d0

```

```

Expunge_End:
movem.l (sp)+,d2/a5/a6
rts

```

```

Null:
CLEAR d0
rts

```

```

;-----
; here begins the library specific commands
;-----

```

```

MyFunc0:
CLEAR d0
rts

```

```

MyFunc1:
moveq #1,d0
rts

```

```

; EndCode is a marker that show the end of your code.
; Make sure it does not span sections nor is before the
; rom tag in memory! It is ok to put it right after
; the rom tag -- that way you are always safe. I put

```

```
    ; it here because it happens to be the "right" thing  
    ; to do, and I know that it is safe in this case.  
EndCode:
```

END

```
*****
*
*      Copyright (C) 1985, Commodore Amiga Inc.  All rights reserved.  *
*
*****/
```

```
*****
*
* mydev.asm -- skeleton device code
*
* Source Control
* -----
*
* $Header: amain.asm,v 31.3 85/10/18 19:04:04 neil Exp $
*
* $Locker: neil $
*
* $Log: amain.asm,v $
*
```

```
*****/
SECTION section
```

```
NOLIST
include "exec/types.i"
include "exec/nodes.i"
include "exec/lists.i"
include "exec/libraries.i"
include "exec/devices.i"
include "exec/io.i"
include "exec/alerts.i"
include "exec/initializers.i"
include "exec/memory.i"
include "exec/resident.i"
include "exec/ables.i"
include "exec/errors.i"
include "libraries/dos.i"
include "libraries/dosexterns.i"
```

```
include "asmsupp.i"
```

```
include "mydev.i"
```

```
LIST
```

```
;----- These don't have to be external, but it helps some
;----- debuggers to have them globally visible
```

```
XDEF Init
XDEF Open
XDEF Close
XDEF Expunge
XDEF Null
XDEF myName
XDEF BeginIO
XDEF AbortIO
```

```

XREF   _AbsExecBase

XLIB   OpenLibrary
XLIB   CloseLibrary
XLIB   Alert
XLIB   FreeMem
XLIB   Remove
XLIB   FindTask
XLIB   AllocMem
XLIB   CreateProc
XLIB   PutMsg
XLIB   RemTask
XLIB   ReplyMsg
XLIB   Signal
XLIB   GetMsg
XLIB   Wait
XLIB   WaitPort
XLIB   AllocSignal
XLIB   SetTaskPri
    
```

INT_ABLES

```

; The first executable location. This should return an error
; in case someone tried to run you as a program (instead of
; loading you as a library).
FirstAddress:
    CLEAR    d0
    rts
    
```

```

-----
; A romtag structure. Both "exec" and "ramlib" look for
; this structure to discover magic constants about you
; (such as where to start running you from...).
-----
    
```

```

; Most people will not need a priority and should leave it at zero.
; the RT_PRI field is used for configuring the roms. Use "mods" from
; wack to look at the other romtags in the system
MYPRI EQU 0
    
```

initDDescrip:

```

DC.W   RTC_MATCHWORD           ;STRUCTURE RT,0
DC.L   initDDescrip           ; UWORD RT_MATCHWORD
DC.L   EndCode                 ; APTR RT_MATCHTAG
DC.B   RTE_AUTOINIT           ; APTR RT_ENDSKIP
DC.B   VERSION                 ; UBYTE RT_FLAGS
DC.B   NT_DEVICE              ; UBYTE RT_VERSION
DC.B   MYPRI                  ; UBYTE RT_TYPE
DC.L   myName                 ; BYTE RT_PRI
DC.L   idString               ; APTR RT_NAME
DC.L   Init                   ; APTR RT_IDSTRING
                                ; APTR RT_INIT
                                ; LABEL RT_SIZE
    
```

```

; this is the name that the device will have
subSysName:
myName:      MYDEVNAME

; a major version number.
VERSION:     EQU      1

; A particular revision. This should uniquely identify the bits in the
; device. I use a script that advances the revision number each time
; I recompile. That way there is never a question of which device
; that really is.
REVISION:    EQU      17

; this is an identifier tag to help in supporting the device
; format is 'name version.revision (dd MON yyyy)', <cr>, <lf>, <null>
idString:    dc.b      'mydev 1.0 (31 Oct 1985)',13,10,0

dosName:     DOSNAME

; force word alignment
ds.w        0

; The romtag specified that we were "RTE_AUTOINIT". This means
; that the RT_INIT structure member points to one of these
; tables below. If the AUTOINIT bit was not set then RT_INIT
; would point to a routine to run.

Init:
DC.L        MyDev_Sizeof      ; data space size
DC.L        funcTable        ; pointer to function initializers
DC.L        dataTable        ; pointer to data initializers
DC.L        initRoutine      ; routine to run

funcTable:

;----- standard system routines
dc.l        Open
dc.l        Close
dc.l        Expunge
dc.l        Null

;----- my device definitions
dc.l        BeginIO
dc.l        AbortIO

;----- function table end marker
dc.l        -1

; The data table initializes static data structures.
; The format is specified in exec/InitStruct routine's
; manual pages. The INITBYTE/INITWORD/INITLONG routines
; are in the file "exec/initializers.i". The first argument

```



```

; is the offset from the device base for this byte/word/long.
; The second argument is the value to put in that cell.
; The table is null terminated
dataTable:
INITBYTE      LH_TYPE,NT_DEVICE
INITLONG      LN_NAME,myName
INITBYTE      LIB_FLAGS,LIBF_SUMUSED!LIBF_CHANGED
INITWORD      LIB_VERSION,VERSION
INITWORD      LIB_REVISION,REVISION
INITLONG      LIB_IDSTRING,idString
DC.L          0

; This routine gets called after the device has been allocated.
; The device pointer is in D0. The segment list is in a0.
; If it returns non-zero then the device will be linked into
; the device list.
initRoutine:

;----- get the device pointer into a convenient A register
move.l  a5,-(sp)
move.l  d0,a5

;----- save a pointer to exec
move.l  a6,md_SysLib(a5)

;----- save a pointer to our loaded code
move.l  a0,md_SegList(a5)

;----- open the dos library
lea     dosName(pc),a1
CLEAR  d0
CALLSYS OpenLibrary

move.l  d0,md_DosLib(a5)
bne.s  init_DosOK

;----- can't open the dos! what gives
ALERT  AG_OpenLib!AO_DOSLib

init_DosOK:
;----- now build the static data that we need
;
; put your initialization here...
;

move.l  a5,d0
move.l  (sp)+,a5
rts

;-----
; here begins the system interface commands. When the user calls
; OpenLibrary/CloseLibrary/RemoveLibrary, this eventually gets translated
; into a call to the following routines (Open/Close/Expunge). Exec

```

```

; has already put our device pointer in a6 for us. Exec has turned
; off task switching while in these routines (via Forbid/Permit), so
; we should not take too long in them.
;
;-----

```

```

; Open sets the IO_ERROR field on an error. If it was successfull,
; we should set up the IO_UNIT field.

```

```

Open:      ; ( device:a6, iob:a1, unitnum:d0, flags:d1 )
movem.l   d2/a2/a3/a4,-(sp)

```

```

move.l    a1,a2      ; save the iob

```

```

;----- see if the unit number is in range

```

```

moveq     #MD_NUMUNITS,d2

```

```

cmp.l     d2,d0

```

```

bcc.s     Open_Error ; unit number out of range

```

```

;----- see if the unit is already initialized

```

```

move.l    d0,d2      ; save unit number

```

```

lsl.l     #2,d0

```

```

lea.l     md_Units(a6,d0.1),a4

```

```

move.l    (a4),d0

```

```

bne.s     Open_UnitOK

```

```

;----- try and conjure up a unit

```

```

bsr       InitUnit

```

```

;----- see if it initialized OK

```

```

move.l    (a4),d0

```

```

beq.s     Open_Error

```

```

Open_UnitOK:

```

```

move.l    d0,a3      ; unit pointer in a3

```

```

move.l    d0,IO_UNIT(a2)

```

```

;----- mark us as having another opener

```

```

addq.w    #1,LIB_OPENCNT(a6)

```

```

addq.w    #1,UNIT_OPENCNT(a3)

```

```

;----- prevent delayed expunges

```

```

bclr      #LIBB_DELEXP,md_Flags(a6)

```

```

Open_End:

```

```

movem.l   (sp)+,d2/a2/a3/a4

```

```

rts

```

```

Open_Error:

```

```

move.b    #IOERR_OPENFAIL,IO_ERROR(a2)

```

```

bra.s     Open_End

```

```

; There are two different things that might be returned from

```

```

; the Close routine.  If the device is no longer open and
; there is a delayed expunge then Close should return the
; segment list (as given to Init).  Otherwise close should
; return NULL.

```

```

Close:          ; ( device:a6, iob:a1 )
movem.l a2/a3,-(sp)

move.l a1,a2

move.l IO_UNIT(a2),a3

;----- make sure the iob is not used again
moveq.l #-1,d0
move.l d0,IO_UNIT(a2)
move.l d0,IO_DEVICE(a2)

;----- see if the unit is still in use
subq.w #1,UNIT_OPENCNT(a3)
bne.s Close_Device

bsr ExpungeUnit

```

```

Close_Device:
;----- mark us as having one fewer openers
subq.w #1,LIB_OPENCNT(a6)

;----- see if there is anyone left with us open
bne.s Close_End

;----- see if we have a delayed expunge pending
btst #LIBB_DELEXP,md_Flags(a6)
beq.s Close_End

;----- do the expunge
bsr Expunge

```

```

Close_End:
movem.l (sp)+,a2/a3
rts

```

```

; There are two different things that might be returned from
; the Expunge routine.  If the device is no longer open
; then Expunge should return the segment list (as given to
; Init).  Otherwise Expunge should set the delayed expunge
; flag and return NULL.
;
; One other important note: because Expunge is called from
; the memory allocator, it may NEVER Wait() or otherwise
; take long time to complete.

```

```

Expunge:        ; ( device: a6 )

movem.l d2/a5/a6,-(sp)
move.l a6,a5

```

```

    move.l  md_SysLib(a5),a6

    ;----- see if anyone has us open
    tst.w  LIB_OPENCNT(a5)
    beq    1$

    ;----- it is still open.  set the delayed expunge flag
    bset   #LIBB_DELEXP,md_Flags(a5)
    CLEAR  d0
    bra.s  Expunge_End

1$:
    ;----- go ahead and get rid of us.  Store our seglist in d2
    move.l  md_SegList(a5),d2

    ;----- unlink from device list
    move.l  a5,a0
    CALLSYS Remove

    ;
    ; device specific closings here...
    ;

    ;----- close the dos library
    move.l  md_DosLib(a5),a1
    CALLSYS CloseLibrary

    ;----- free our memory
    move.l  a5,a1
    move.l  LIB_NEGSIZE(a5),d0

    sub.l   d0,a1
    add.l   LIB_POSSIZE(a5),d0

    CALLSYS FreeMem

    ;----- set up our return value
    move.l  d2,d0

Expunge_End:
    movem.l (sp)+,d2/a5/a6
    rts

Null:
    CLEAR  d0
    rts

InitUnit:
    ; ( d2:unit number, a3:scratch, a6:devptr )
    movem.l d2/d3/d4,-(sp)

    ;----- allocate unit memory
    move.l  #MyDevUnit_Sizeof,d0
    move.l  #MEMF_PUBLIC!MEMF_CLEAR,d1
    LINKSYS AllocMem,md_SysLib(a6)

```

```

tst.l    d0
beq      InitUnit_End

move.l   d0,a3
move.b   d2,mdu_UnitNum(a3)      ; initialize unit number

;----- start up the unit process. We do a trick here --
;----- we set his message port to PA_IGNORE until the
;----- new process has a change to set it up.
;----- We cannot go to sleep here: it would be very nasty
;----- if someone else tried to open the unit
;----- (exec's OpenDevice has done a Forbid() for us --
;----- we depend on this to become single threaded).
move.l   #MYPROCSTACKSIZE,d4    ; stack size
move.l   #myproc_seglist,d3     ; segment list
lsr.l    #2,d3                   ; change to bcpl pointer
moveq    #MYPROCPRI,d2          ; pick out its priority
move.l   #myName,d1             ; name is the device's
LINKSYS  CreateProc,md_DosLib(a6)

tst.l    d0
beq      InitUnit_FreeUnit

;----- set up the unit structures for the new process
move.l   d0,mdu_Process(a3)
move.l   d0,a0
lea      -pr_MsgPort(a0),a0
move.l   a0,MP_SIGTASK(a3)
move.b   #PA_IGNORE,MP_FLAGS(a3)

;----- send a startup message to the new process
lea      mdu_Msg(a3),a1
move.l   a3,mdm_Unit(a1)
move.l   a6,mdm_Device(a1)
move.l   d0,a0                  ; message port is new process port
LINKSYS  PutMsg,md_SysLib(a6)

;----- mark us as ready to go
move.l   d2,d0                  ; unit number
lsl.l    #2,d0
move.l   a3,md_Units(a6,d0.l)   ; set unit table

InitUnit_End:
movem.l  (sp)+,d2/d3/d4
rts

;----- got an error. free the unit structure that we allocated.
InitUnit_FreeUnit:
bsr      FreeUnit
bra.s    InitUnit_End

FreeUnit:
; ( a3:unitptr, a6:deviceptr )
move.l   a3,a1
move.l   #MyDevUnit_Sizeof,d0

```

```
LINKSYS FreeMem,md_SysLib(a6)
rts
```

```
ExpungeUnit: ; ( a3:unitptr, a6:deviceptr )
             move.l d2,-(sp)

             ;----- get rid of the unit's task. We know this is safe
             ;----- because the unit has an open count of zero, so it
             ;----- is 'guaranteed' not in use.
             move.l mdu_Process(a3),a1
             lea -(pr_MsgPort)(a1),a1
             LINKSYS RemTask,md_SysLib(a6)

             ;----- save the unit number
             CLEAR d2
             move.b mdu_UnitNum(a3),d2

             ;----- free the unit structure.
             bsr FreeUnit

             ;----- clear out the unit vector in the device
             lsl.l #2,d2
             clr.l md_Units(a6,d2.l)

             move.l (sp)+,d2

             rts
```

```
-----
;
; here begins the device specific functions
;
-----
```

```
; cmdtable is used to look up the address of a routine that will
; implement the device command.
```

```
cmdtable:
DC.L Invalid ; $00000001
DC.L MyReset ; $00000002
DC.L Read ; $00000004
DC.L Write ; $00000008
DC.L Update ; $00000010
DC.L Clear ; $00000020
DC.L MyStop ; $00000040
DC.L Start ; $00000080
DC.L Flush ; $00000100
DC.L Foo ; $00000200
DC.L Bar ; $00000400
```

```
cmdtable_end:
```

```
; this define is used to tell which commands should not be queued
; command zero is bit zero.
```

```
; The immediate commands are Invalid, Reset, Stop, Start, Flush
IMMEDIATES EQU $000001c3
```

```

;
; BeginIO starts all incoming io. The IO is either queued up for the
; unit task or processed immediately.
;
BeginIO:      ; ( iob: a1, device:a6 )
              move.l  a3,-(sp)

              ;----- bookkeeping
              move.l  IO_UNIT(a1),a3

              ;----- see if the io command is within range
              move.w  IO_COMMAND(a1),d0
              cmp.w   #MYDEV_END,d0
              bcc.s   BeginIO_NoCmd

              DISABLE a0

              ;----- process all immediate commands no matter what
              move.w  #IMMEDIATES,d1
              btst   d0,d1
              bne.s  BeginIO_Immediate

              ;----- see if the unit is STOPPED. If so, queue the msg.
              btst   #MDUB_STOPPED,UNIT_FLAGS(a3)
              bne.s  BeginIO_QueueMsg

              ;----- this is not an immediate command. see if the device is
              ;----- busy.
              bset   #UNITB_ACTIVE,UNIT_FLAGS(a3)
              beq.s  BeginIO_Immediate

              ;----- we need to queue the device. mark us as needing
              ;----- task attention. Clear the quick flag
BeginIO_QueueMsg:
              BSET   #UNITB_INTASK,UNIT_FLAGS(a3)
              bclr  #IOB_QUICK,IO_FLAGS(a1)

              ENABLE a0

              move.l  a3,a0
              LINKSYS PutMsg,md_SysLib(a6)
              bra.s  BeginIO_End

BeginIO_Immediate:
              ENABLE a0

              bsr    PerformIO

BeginIO_End:
              move.l  (sp)+,a3
              rts

BeginIO_NoCmd:
              move.b  #IOERR_NOCMD,IO_ERROR(a1)
              bra.s  BeginIO_End

```

```

;
; PerformIO actually dispatches an io request. It expects a3 to already
; have the unit pointer in it. a6 has the device pointer (as always).
; a1 has the io request. Bounds checking has already been done on
; the io request.
;

```

```

PerformIO:      ; ( iob:a1, unitptr:a3, devptr:a6 )
                move.l  a2,-(sp)
                move.l  a1,a2

                move.w  IO_COMMAND(a2),d0
                lea    cmdtable(pc),a0
                move.l  0(a0,d0.w),a0

                jsr     (a0)

                move.l  (sp)+,a2
                rts

```

```

;
; TermIO sends the IO request back to the user. It knows not to mark
; the device as inactive if this was an immediate request or if the
; request was started from the server task.
;

```

```

TermIO:         ; ( iob:a1, unitptr:a3, devptr:a6 )
                move.w  IO_COMMAND(a1),d0
                move.w  #IMMEDIATES,d1
                btst   d0,d1
                bne.s  TermIO_Immediate

                ;----- we may need to turn the active bit off.
                btst   #UNITB_INTASK,UNIT_FLAGS(a3)
                bne.s  TermIO_Immediate

                ;----- the task does not have more work to do
                bclr   #UNITB_ACTIVE,UNIT_FLAGS(a3)

```

```

TermIO_Immediate:
                ;----- if the quick bit is still set then we don't need to reply
                ;----- msg -- just return to the user.
                btst   #IOB_QUICK,IO_FLAGS(a1)
                bne.s  TermIO_End

                LINKSYS ReplyMsg,md_SysLib(a6)

```

```

TermIO_End:
                rts

```

```

AbortIO:       ; ( iob: a1, device:a6 )

```

```

;-----

```



```

;
; here begins the functions that implement the device commands
; all functions are called with:
;   a1 -- a pointer to the io request block
;   a2 -- another pointer to the iob
;   a3 -- a pointer to the unit
;   a6 -- a pointer to the device
;
; Commands that conflict with 68000 instructions have a "My" prepended
; to them.
-----

```

```

Invalid:
    move.b #IOERR_NOCMD,IO_ERROR(a1)
    bsr   TermIO
    rts

```

```

MyReset:
    ; !!! fill me in !!!
    ; !!! fill me in !!!
    ; !!! fill me in !!!
    ; !!! fill me in !!!

```

```

;
; the Read command acts as an infinite source of nulls. It clears
; the user's buffer and marks that many bytes as having been read.
;

```

```

Read:
    move.l IO_DATA(a1),a0
    move.l IO_LENGTH(a1),d0
    move.l d0,IO_ACTUAL(a1)

    ;----- deal with a zero length read
    beq.s  Read_End

    ;----- now copy the data
    CLEAR  d1

```

```

Read_Loop:
    move.b d1,(a0)+
    subq.l #1,d0
    bne.s  Read_Loop

```

```

Read_End:
    bsr   TermIO
    rts

```

```

;
; the Write command acts as bit bucket. It clears acknowledges all
; the bytes the user has tried to write to it.
;

```

Write:

```

    move.l  IO_LENGTH(a1),IO_ACTUAL(a1)

    bsr    TermIO
    rts

```

```

;
; Update and Clear are internal buffering commands. Update forces all
; io out to its final resting spot, and does not return until this is
; done. Clear invalidates all internal buffers. Since this device
; has no internal buffers, these commands do not apply.
;

```

Update:

```

Clear:
    bra    Invalid

```

```

;
; the Stop command stop all future io requests from being
; processed until a Start command is received. The Stop
; command is NOT stackable: e.g. no matter how many stops
; have been issued, it only takes one Start to restart
; processing.
;

```

MyStop:

```

    bset    #MDUB_STOPPED,UNIT_FLAGS(a3)

    bsr    TermIO
    rts

```

Start:

```

    bsr    InternalStart

    move.l  a2,a1
    bsr    TermIO

    rts

```

InternalStart:

```

;----- turn processing back on
bclr    #MDUB_STOPPED,UNIT_FLAGS(a3)
;
;----- kick the task to start it moving
move.l  a3,a1
CLEAR   d0
move.l  MP_SIGBIT(a3),d1
bset    d1,d0
LINKSYS Signal,md_SysLib(a3)

    rts

```

```

;
; Flush pulls all io requests off the queue and sends them back.
; We must be careful not to destroy work in progress, and also
; that we do not let some io requests slip by.

```

```
;  
; Some funny magic goes on with the STOPPED bit in here. Stop is  
; defined as not being reentrant. We therefore save the old state  
; of the bit and then restore it later. This keeps us from  
; needing to DISABLE in flush. It also fails miserably if someone  
; does a start in the middle of a flush.  
;
```

Flush:

```
movem.l d2/a6, -(sp)  
  
move.l md_SysLib(a6), a6  
  
bset #MDUB_STOPPED, UNIT_FLAGS(a3)  
sne d2
```

Flush_Loop:

```
move.l a3, a0  
CALLSYS GetMsg  
  
tst.l d0  
beq.s Flush_End  
  
move.l d0, a1  
move.b #IOERR_ABORTED, IO_ERROR(a1)  
CALLSYS ReplyMsg  
  
bra.s Flush_Loop
```

Flush_End:

```
move.l d2, d0  
movem.l (sp)+, d2/a6  
  
tst.b d0  
beq.s 1$
```

```
bsr InternalStart
```

1\$:

```
move.l a2, a1  
bsr TermIO  
  
rts
```

```
;  
; Foo and Bar are two device specific commands that are provided just  
; to show you how to add your own commands. The currently return that  
; no work was done.  
;
```

Foo:

Bar:

```
CLEAR d0  
move.l d0, IO_ACTUAL(a1)
```

```

    bsr   TermIO
    rts

```

```

-----
; here begins the process related routines
; A Process is provided so that queued requests may be processed at
; a later time.

```

```

; Register Usage
; a3 -- unit pointer
; a6 -- syslib pointer
; a5 -- device pointer
; a4 -- task (NOT process) pointer
; d7 -- wait mask
-----

```

```

; some dos magic. A process is started at the first executable address
; after a segment list. We hand craft a segment list here. See the
; the DOS technical reference if you really need to know more about this.

```

```

        cnop      0,4           ; long word align
        DC.L      16           ; segment length -- any number will do
myproc_seglist:
        DC.L      0           ; pointer to next segment

```

```

; the next instruction after the segment list is the first executable address

```

```
Proc_Begin:
```

```

    move.l  _AbsExecBase,a6

;----- wait for our first packet
    SUB.L  a1,a1
    CALLSYS FindTask           ; <my task> = FindTask(0)
    move.l  d0,a0
    move.l  d0,a4             ; save task in a4
    lea    pr_MsgPort(a0),a0  ; get msg port for my processes
    CALLSYS WaitPort

;----- take msg off the port
    move.l  d0,a1
    move.l  d0,a2             ; save the message
    CALLSYS Remove

;----- get our parameters out of it
    move.l  mdm_Device(a2),a5  ; a5 is now our device
    move.l  mdm_Unit(a2),a3

;----- Allocate the right signal
    moveq   #-1,d0
    CALLSYS AllocSignal       ; -1 is any signal at all

```

```

move.b d0,MP_SIGBIT(a3)
move.b #PA_SIGNAL,MP_FLAGS(a3)

;----- change the bit number into a mask, and save in d7
CLEAR d7
bset d0,d7

;-----
;----- OK, kids, we are done with initialization. We now
;----- can start the main loop of the driver. It goes
;----- like this. Because we had the port marked PA_IGNORE
;----- for a while (in InitUnit) we jump to the getmsg
;----- code on entry.
;-----
;----- wait for a message
;----- lock the device
;----- get a message. if no message unlock device and loop
;----- dispatch the message
;----- loop back to get a message
;-----

bra.s Proc_CheckStatus

;----- main loop: wait for a new message
Proc_MainLoop:
move.l d7,d0
CALLSYS Wait

Proc_CheckStatus:
;----- see if we are stopped
btst #MDUB_STOPPED,UNIT_FLAGS(a3)
bne.s Proc_MainLoop ; device is stopped

;----- lock the device
bset #UNITB_ACTIVE,UNIT_FLAGS(a3)
bne.s Proc_MainLoop ; device in use

;----- get the next request
Proc_NextMessage:
move.l a3,a0
CALLSYS GetMsg
tst.l d0
beq.s Proc_Unlock ; no message?

;----- do this request
move.l d0,a1
exg a5,a6 ; put device ptr in right place
bsr PerformIO
exg a5,a6 ; get syslib back in a6

bra.s Proc_NextMessage

;----- no more messages. back ourselves out.
Proc_Unlock:

```

```
and.b    #fff&(UNITB_ACTIVE!UNITB_INTASK),UNIT_FLAGS(a3)
bra      Proc_MainLoop
```

Proc_Fail:

```
;----- we come here on initialization failures
bsr      FreeUnit
rts
```

```
-----
; EndCode is a marker that show the end of your code.
; Make sure it does not span sections nor is before the
; rom tag in memory! It is ok to put it right after
; the rom tag -- that way you are always safe. I put
; it here because it happens to be the "right" thing
; to do, and I know that it is safe in this case.
-----
EndCode:
```

END

Appendix L

Disk Format Information

This appendix contains information useful to the developer who desires to take over the entire machine. That is, rather than use AmigaDOS to load the code, the application is to boot directly after Kickstart, not using AmigaDOS or Intuition. This appendix provides two pieces of information: the disk boot block format, and the format of the actual data on disk.

THE BOOT PROCESS

The first two sectors are read into the system at an arbitrary position; therefore, the code MUST be PC-relative. The first three longwords are as in devices/bootblock.h. The type should be BBID_DOS; the checksum must be correct (as in additive carry wraparound sum of 0xffffffff). Execution starts at location 12 of the sectors that were read in.

The code is called with an open disk I/O request in A1 (see the TrackDisk chapter for the format of this IOREquest block). The boot code is free to use it as it wishes (it may trash A1, but must not trash the io block itself).

The boot code returns two values: D0 and A0. D0 is a failure code -- if it is non-zero then a system alert will be called, then the boot code falls into the debugger.

If D0 is null then A0 contains the start address to jump to. The strap module will free the boot sectors, close the I/O block, do any other cleanup that is required, and jump to the location pointed to by A0.

COMMODORE-AMIGA DISK FORMAT

The following are details about how the bits on the Commodore-Amiga disk are actually written.

Gross Data Organization:

3 1/2 inch disk
double-sided
80 cylinders/160 tracks

Per-track Organization:

Nulls written as a gap, then 11 sectors of data.
No gaps written between sectors.

Per-sector Organization:

All data is MEM encoded. This is the pre-encoded contents of each sector:

| | |
|--|--|
| two bytes of 00 data | (MEM = AAAA each) |
| two bytes of A1* | ("standard sync byte" -- MEM encoded A1 without a clock pulse) |
| | (MEM = 4489 each) |
| one byte of format-byte | (Amiga 1.0 format = FF) |
| one byte of track number | |
| one byte of sector number | |
| one byte of sectors until end of write | (NOTE 1) |

[above 4 bytes treated as one longword
for purposes of MEM encoding]

16 bytes of OS recovery info (NOTE 2)
 [treated as a block of 16 bytes for encoding]
four bytes of header checksum
 [treated as a longword for encoding]
four bytes of data-area checksum
 [treated as a longword for encoding]
512 bytes of data
 [treated as a block of 512 bytes for encoding]

NOTES:

NOTE 1.

The track number and sector number are constant for each particular sector. However, the sector offset byte changes each time we rewrite the track.

The Amiga does a full track read starting at a random position on the track and going for slightly more than a full track read to assure that all data gets into the buffer. The data buffer is examined to determine where the first sector of data begins as compared to the start of the buffer. The track data is block moved to the beginning of the buffer so as to align some sector with the first location in the buffer.

Because we start reading at a random spot, the read data may be divided into three chunks: a series of sectors, the track gap, and another series of sectors. The sector offset value tells the disk software how many more sectors remain before the gap. From this the software can figure out the buffer memory location of the last byte of legal data in the buffer. It can then search past the gap for the next sync byte and, having found it, can block move the rest of the disk data so that all 11 sectors of data are contiguous.

Example:

first-ever write of the track from a buffer like this:

<GAP> |sector0|sector1|sector2|.....|sector10|

sector offset values:

11 10 9 1

(if I find this one at the start of my read buffer,
then I know there are this many more sectors
with no intervening gaps before I hit a gap).

sample read of this track:

<junk>|sector9|sector10|<gap>|sector0|...|sector8|<junk>

value of 'sectors till end of write':

2 1 11 3

result of track realligning:

<GAP>|sector9|sector10|sector0|...|sector8|

new sectors till end of write:

11 10 9 ... 1

so that when the track is rewritten, the sector offsets are adjusted to match the way the data was written.

NOTE 2. This is operating systems dependent data and relates to how AmigaDos assigns sectors to files.

Reserved for future use.

GENERAL:

When data is MFM encoded, the encoding is performed on the basis of a data block-size. In the sector encoding described above, there are bytes individually encoded; three segments of 4 bytes of data each, treated as longwords; one segment of 16 bytes treated as a block; two segments of longwords for the header and data checksums; and the data area of 512 bytes treated as a block.

When the data is encoded, the odd bits are encoded first, then the even bits of the block.

(Make a block of bytes formed from all odd bits of the block, encode as MFM.

Make a block of bytes formed from all even bits of the block, encode as MFM. Even bits are shifted left one bit position before being encoded.)

SOURCE CODE FOR DATA ENCODE/DECODE

```

decodeBlock( mfmbuffer, userbuffer, numwords )
WORD *mfmbuffer;      /* the encoded data */
WORD *userbuffer;     /* where to put the decoded data */
int numwords;         /* the number of WORDS of data (not bytes) */
{
    WORD *oddptr, *evenptr, oddbits, evenbits;

    oddptr = mfmbuffer;

    /* the even region starts right after the odd one */
    evenptr = &mfmbuffer[numwords];

    while( numwords-- > 0 ) {
        /* mask off the mfm clock bits, and shift the word */
        oddbits = ((*oddptr++ << 1) & 0xAAAA);

        /* even bits are already in the right place. Just mask off clock */
        evenbits = ((*evenptr++) & 0x5555);

        /* recombine the two sections */
        *userbuffer++ = oddbits | evenbits;
    }
}

encodeBlock( mfmbuffer, userbuffer, numwords )
WORD *mfmbuffer;     /* where to put the encoded data */
WORD *userbuffer;    /* the user data, before encoding */
int numwords;        /* the number of WORDS of data (not bytes) */
{
    WORD *oddptr, *evenptr;
    WORD *ubuf;

    oddptr = mfmbuffer;

    /* the even region starts right after the odd one */
    evenptr = &mfmbuffer[numwords];

    /* mfmencode takes one word of mfm data can correctly sets
     * the clock bits
     */

    /* encode the odd bits */
    for( ubuf = userbuffer, i = numwords; i > 0; i-- ) {
        oddptr++ = mfmencode( (*ubuf++ >> 1) & 0x5555 );
    }

    /* encode the even bits */
    for( ubuf = userbuffer, i = numwords; i > 0; i-- ) {
        evenptr++ = mfmencode( *ubuf++ & 0x5555 );
    }
}

```


Commodore Business Machines, Inc.
1200 Wilson Drive, West Chester, PA 19380

Commodore Business Machines, Limited
370 Pharmacy Avenue, Agincourt, Ontario, M1W 2K4

Copyright 1985 © Commodore-Amiga, Inc.