GUIDE TO APW C RELEASE V1.0B7
A Note From Apple Engineers

This APW C release consists of three major parts:

- a disk containing the APW C software
- one reference manual covering APW C
- these release notes.

We have included these release notes with your software to cover the discrepancies between the software and documentation. The notes are divided into sections roughly corresponding to the chapters in the APW C manual. Each section is divided into two parts. The first part corrects or supplements the documentation. The second part of each section lists currently known software problems, ordered by severity (the most severe first). You can read these notes ahead of time, but we would suggest reading them again after you become familiar with the chapter describing a given piece of the system and before you actually use the software. The paragraphs starting with **NOTE** are particularly significant. Please read them carefully.

All of the page references in these notes refer to the following documents:

- "Apple IIGS Programmer's Workshop Reference", APDA Draft, 27 October 1986
- "Apple IIGS Programmer's C Reference", APDA Draft, 9 March 1987
- "GUIDE TO APW RELEASE V1.0B4. A Note From Apple Engineers".

The APW software was tested with the Apple IIGS System Disk V1.1 and APW V1.0B4.
NOTE: the previous release of APW, APW V1.0B1, cannot be used with this software.

We have gone through many cycles of testing and bug fixing that culminated in this Beta release. However, we have been developing software for too long to believe that we have detected all the problems . We need your help to make it better. Please report any problems that you find with the APW software or documentation on the enclosed "APW BUG REPORT" form. Feel free to use the enclosed "APW SUGGESTIONS and EVALUATION" form to communicate your recommendations for future APW software.

Development Systems Group

GETTING STARTED

Minimum suggested configuration:

Apple IIGS.
Two 3 1/2" 800 K disk drives (by default in slot 5, drives 1 and 2).
512K Apple IIGS Memory Expansion card (memory expansion slot).
Apple monitor.

Recommended configuration:

Apple IIGS.
1 Mb Apple IIGS Memory Expansion card (memory expansion slot).
Two 3 1/2" 800 K disk drives (by default in slot 5, drives 1 and 2).
Apple monitor.
a hard disk (interface card in slot 7).
1 Mb Apple II Memory Expansion card, "slinky", used for a RAM disk (in slot 2).

Software diskettes needed:

APW C V1.0ß7 disk containing APW C release
APW V1.0ß4 disk containing the basic APW system

Installation procedures

The following instructions are intended to help the first time user install and use the APW C V1.0ß7 release. If you have been using previous releases, you can probably skip these instructions.

Backup your original APWC disks following the instructions contained in the "Backing up Your APW Disk" section of the APW Reference manual (the manual refers to the APW disk, but follow the instructions as they would apply to APWC). While installing APW C use the disk copy that you have made.

Caution: Do not create different volumes with the same volume name, otherwise the system may become confused.

Installation on a Two 800K Disk Drive Apple IIGS system

1. Place the APW V1.0ß4 diskette in the first drive (it is a bootable diskette) and insert the APWC V1.0ß7 diskette in the second drive. Boot your Apple IIGS system.
2. Execute `EDIT SYSTEM/LOGIN` and remove the comment characters (the asterisks) from the appropriate lines (see the comments in the SYSTEM/LOGIN file).
3. Copy /APW/LANGUAGES/LINKED to /APWC/LANGUAGES and delete /APW/LANGUAGES directory by executing the following commands:
```
COPY -C /APW/LANGUAGES/LINKED /APWC/LANGUAGES
DELETE /APW/LANGUAGES/=
DELETE /APW/LANGUAGES
```
4. Boot your Apple IIGS system. You are ready to create your own applications and APW utilities.

Installation on a Hard Disk

1. Install APW V1.0B4 on your hard disk as described in the APW Release Notes.
2. Copy all of the C release files to your hard disk by executing the following commands:
```
    COPY -C /APWC/LANGUAGES/= /hardisk/APW/LANGUAGES/
    COPY -C /APWC/LIBRARIES/= /hardisk/APW/LIBRARIES/
```
3. Boot your Apple IIGS system.

APW C LANGUAGE AND COMPILER

C LANGUAGE - DOCUMENTATION CHANGES

• The Library Index in Appendix D is only a partial index. It cross-references only those items DOCUMENTED in the manual. Many other routines exist in CLIB and are documented in the Apple IIGS Toolbox Reference and the Apple Numerics Manual.

• If you are writing a C function to be called by the ROMs, for example a floating-point halt handler, don't reference C variables in `~globals` because they use 16-bit addressing and won't resolve properly. Instead, refer only to variables in `~string` or `~arrays`; they use 24-bit addressing and work fine. An easy way of making a variable reside in `~arrays` instead of `~globals` is to declare it as an one-element array. For example, use
```
              char c[1]; int n[1]; double x[1];
```
instead of
```
              char c; int n; double x;
```
• The APW C documentation lists two predefined preprocessor symbols *APW* and *WD65816*. In the current implementation of APW C these symbols are not predefined by the compiler. Suggested workaround: the symbol '**Megamax**' is erroneously predefined, so if you do:
```
    #ifdef Megamax
    #define APW
    #define WD65816
    #endif
```
at the beginning of your source code, then you can subsequently do
```
    #ifdef APW
                     ...
    #endif
```
and subsequent changes to the C compiler will not hurt you.

• #APPEND directive requires a quote, i.e., the correct format is
```
    #APPEND "pathname"
```

C COMPILER - FAILED OPERATION

• **NOTE:** For arrays whose size in bytes is greater than 64K, array indexing calculations are performed using 'short' or 'y-index' math. Pointer math, however, is correctly done using '32-bit' math. So, if you have a big array, you may be forced to do array references via pointers, as in:

`*(a + i)` instead of `a[i]`

In the future, the fix will be: if the array is > 64K in size, then all indexing will be done in 'long math'. If the array is known to be < 64K in size, then 'short math' is acceptable. This means that '`extern int arr[64]`' in the future will produce better code than '`extern int arr[]`', because this latter case will force use of long math to index into arr.

There is a temporary workaround, **which we hereby deprecate (so do not do the following!):**

It has been suggested that using a 'long int' expression to index into arrays would be better. While this currently will cause the compiler to use long math in array indexing calculations, which temporarily works, developers should not count on this in the future. The proper type for array indexing expressions is 'int', not 'long int'. This restricts the number of array elements to 64K. (But since each array element may be of any size, the size of the whole array is not limited to 64K.)

So, the correct thing to do is this:
(i) use a[i] when i is int, and sizeof(a) < 64K
(ii) use *(a+i) when i is int and sizeof(a) > 64K
(iii) do not use a[l] where l is long.

• **NOTE:** The addressing used when arrays are indexed by longs is incorrect. The following function will not work when the "num" parameter is greater than $8000:

```
zero(arr,num)
char *arr;
long num;
{
     long i;
     for (i=0;i<num;i++) arr[i]=0;
}
```

The first $8000 bytes are stored correctly, but all subsequent bytes are stored into the wrong memory locations. The code generated for "arr+i" decrements the bank byte of the generated address when the "i" is greater than equal to $8000. This causes the problem.
Suggested workaround: use `unsigned long` instead of `long`.

• `enumvar = IntConstant` is not currently allowed by APW C. APW C wants you to use a typecast, of the form `enumvar = (enum whatever) Intvar`.
• Results of the % operator are always positive (i.e.,. (-31 % 2) == 1).
• `p(0xffffffff)` pushes a short parameter, not a long. If you really want 32 bits, do `p(0xffffffffL)`. In most other places, the distinction between whether `0xffffffff` is short or long will not break programs.
• `int a[-10];` is accepted as a legal declaration.
• Contrary to the Language Specifications for SANE C Numerics, the C compiler is doing too much floating-point constant folding at compile-time; expressions such as `x = 0.0/0.0` and `y = 1.0/3.0`, which should signal, respectively, invalid and inexact at run-time, don't.
• the shell variable `{status}` is not set by the compiler errors.
• the APW C compiler creates duplicate labels in `~globals` when the same global name is declared twice.
• **NOTE**: the three problems described above suggest that **"#APPEND" directive should not be used** with this release of the C compiler.
• when the C source contains an "#APPEND" directive referring to an ASM65816 file then an incorrect object file is generated.
• Using the commands `RUN` or `COMPILE` on a C program that has an "#APPEND" directive to an ASM65816 file halts after the assembly process. Appending a C program to an ASM65816 program works OK.
• C writes over the .A file produced by the Assembler if it is appended to an ASM65816 file with more than one segment.
• Multiplicative assignment operators can cause system to crash when used with floating point values. Thus val *= 1.0009 will not work; use val = val * 1.0009.

STANDARD C LIBRARY

C LIBRARY - DOCUMENTATION CHANGES

• The standard C library function `ioctl` has the following additional arguments:
FIOGETEOF
FIOSETEOF
FIOGETMARK
FIOSETMARK.

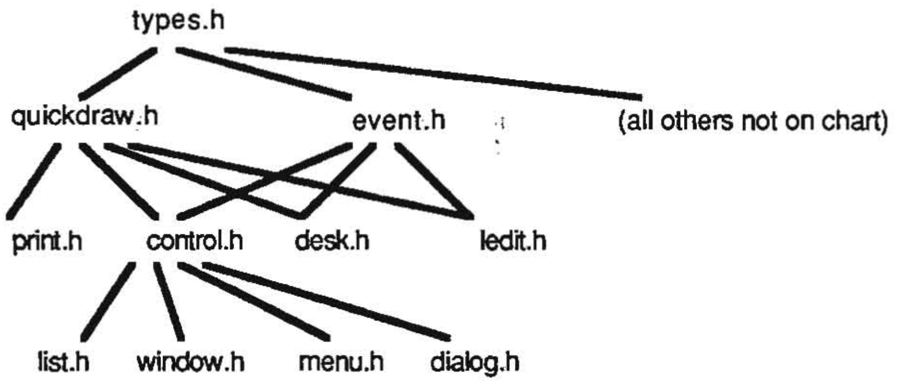
The two "SET" commands set the respective items to the value in the parameter "arg," which must be a "`long`". The two "GET" commands return the respective values via the parameter "arg," which must be a "`long *`" (pointer to long).
• On page 36 (`malloc`—memory allocator), the description of `calloc` should read:
Function `calloc` allocates space for an array of nelem elements of size elsize. The function will fail and return NULL if the product of nelem and elsize exceeds 32766.

C LIBRARY - INCORRECT OPERATION

• Console I/O has several display problems involving special characters (tab, carriage return, backspace). These characters do not have the expected effect on the screen, or on an input line.
• The `envp` parameter to `main()` is not implemented and will not be in the foreseeable future.
•`open()` and `fopen()` do not permit multiple read-only paths to the same file. This is a ProDOS limitation; no workaround is possible.
• The ProDOS device name (e.g., .d3) is not currently expanded when used in a library function.
• **NOTE:** Console input is processed a line at a time. No matter how many characters are requested from the console, the input routine will read until it receives a carriage return or an end-of-file indication. The APW Shell uses control-@ (ASCII 0) to signify end of file; the libraries return the usual C EOF value (-1). The final release of APW C will support both single character input and full input line editing. Limited input editing (with the Delete key only) is supported in this release.
• **NOTE:** A call to `calloc` function with a negative size causes crash.
• Function `lseek` does not return the correct long int value if it is greater than or equal to +32K.

C SYSTEM INTERFACE

• The C header files have been set up to automatically reflect the following inheritance scheme. Any #include'd file will automatically bring in its ancestors if they are not already present. Because the nested inclusion is conditional, no problems of duplicate definitions will occur even if you explicitly #include all of them in your source code.



The inheritance is based upon the reference to data structures either as fields of new data structures, or parameters, or return types of toolbox calls. For example, if you only need windows and menus, you can just specify those header files, and the rest will be taken care of.
• A data structure that uses bit fields, ColorValue, is provided in QuickDraw.h. Its fields are nibbles named blueBits, greenBits, redBits, and reserved. We had to add the 'Bits' to the names to avoid conflict with the color constants blue, green, and red, which are also defined in QuickDraw.h.
• The type `AnSCBByte` is defined as a struct with bitfields. Since the compiler automatically allocates a Word for up to 16 bits of bitfields we have changed the name to `AnSCB`. The structure `LocInfo` has been modified by removing the "reserved" byte, to accommodate the larger size of the PortSCB field. So now `LocInfo.reserved` is `LocInfo.PortSCB.reservedByte`. `AnSCBByte` is now defined as `AnSCB`. You can still use the old name, bearing in mind that in C, it is, and always was, 2 bytes in size .
• Some Dialog Manager calls expect a pointer to a DialogTemplate or an AlertTemplate as a parameter. The templates are defined as having a NULL terminated array of ItemTempPtr at the end. Since the length of the array will vary, we have defined that part of the Dialog and Alert template definitions as :

```
   ItemTempPtr dtItemList[dtItemListLength];   and
```

```
    ItemTempPtr atItemList[atItemListLength];
```
respectively. If, for example, you wish to allow up to 10 items as part of your dialog templates then do:
```
    #define dtItemListLength 11
    #include <dialog.h>
```
The array needs to be one bigger to allow for the terminating null pointer. If you do not predefine dtItemListLength and/or atItemListLength, they will default to 8 and 5 respectively, thus allowing your dialog templates to have up to 7 items and your alert templates to have up to 4 items. If you allocate your templates externally, the ItemTempPtr's will be automatically initialized to NULL, so you only need to set as many as you are using to point to item templates (leaving at least the last element null), and you will automatically have a null terminated list, as required by the Dialog Manager routines.
• In types.h, we have defined `Word` and `word` as `unsigned int`. `LongWord` and `DblWord` as `unsigned long`. Only tool calls and structure fields that can have a negative numerical value are defined as int, or long. Use of unsigned types wherever possible has an optimizing affect on the APW C Compiler.
• In types.h we also have:
```
    typedef char *Ptr;
    typdef Ptr *Handle;
```
In general we have used these types elsewhere in the headers only where the object being referenced is a char or string, or when the definition of the object is not public. Wherever a field in a data structure or a function return type is a pointer or a handle to a publicly defined object (i.e., an object defined in the headers), it has been specifically typed to reflect this. For example: `NewControl` returns a CtlRecHndl, not merely a Handle. The advantage is that if the compiler knows that something is a handle to a particular data structure it will allow use of the dot operator to access the fields once the handle is dereferenced.
• Certain calls will require pointers to a result record or result space to be made from C. To use them you will have to add type definitions as required by the parameter types listed with each call. Unless otherwise noted you should not add the call definitions themselves. The necessary function declarations are provided in the appropriate header files.
In Integer Math: (You probably won't need to make these calls from C)
```
    typedef struct LDivRec {
        Longint quotient;
        Longint remainder;
    } LDivRec, *LDivRecPtr;
    LongDivide(numerator,denominator,resultPtr)
    Longint numerator;
    Longint denominator;
    LDivRecPtr resultPtr;
    LongMul(multiplicand, multiplier, resultPtr)
    Longint multiplicand;
    Longint multiplier;
    comp  *resultPtr;
    typedef struct SDivRec {
        int quotient;
        int remainder;
    } SDivRec, *SDivRecPtr; /* no additional typedef is required */
    SDivide(numerator,denominator,resultPtr)
    int numerator;
    int demominator;
    SDivRecPtr resultPtr;
    UDivide(numerator,denominator, resultPtr)
    word numerator;
    word denominator;
    SDivRecPtr resultPtr;
```
You should not need to use the Multiply call from C, but if you do you will need to change the definition in intmath.h to:
```
    extern pascal Longint Multiply() inline(0x090B, dispatcher);
```

and leave out the resultRecPtr parameter when you call it.
In Text Tools:

```
   typedef struct GlobalsRec {
      Word orMask;
      Word andMask;
   } GlobalsRec, *GlobalsRecPtr;
   void GetInGlobals(resultPtr)
   GlobalsRecPtr resultPtr;
   void GetOutGlobals(resultPtr)
   GlobalsRecPtr resultPtr;
   void GetErrGlobals(resultPtr)
   GlobalsRecPtr resultPtr;
   typedef struct DeviceRec {
      LongWord ptrOrSlot;
      Word deviceType;
   } DeviceRec, *DeviceRecPtr;
   void GetInputDevice(resultPtr)
   DeviceRecPtr resultPtr;
   void GetOutputDevice(resultPtr)
   DeviceRecPtr resultPtr;
   void GetErrorDevice(resultPtr)
   DeviceRecPtr resultPtr;
```

In Miscellaneous Tools all the necessary result record pointer types are provided in misctool.h. The calls which use them are:

```
   void ReadTimeHex(resultPtr)
   TimeRecPtr resultPtr;
   void FWEntry(aRegValue,xRegValue,yRegValue, eModeEntryPt, resultPtr)
   Word aRegValue;
   Word xRegValue;
   Word yRegValue;
   Word eModeEntryPoint;
   FWRecPtr resultPtrp;
   void GetMouseClamp(resultPtr)
   ClampRecPtr resultPtr;
   void ReadMouse(resultPtr)
   MouseRecPtr resultPtr;
   void GetAbsClamp(resultPtr)
   ClampRecPtr(resultPtr);
```

APW SHELL INTERFACE

SHELL INTERFACE - DOCUMENTATION CHANGES

The shell interface has been completely rewritten. It works as documented in the current APW C manual. Several of the names of the routines have been changed. The names are correct in the documentation. All shell calls are upper case letters only with the underscore '_' character used to separate the different words. This makes the shell calls follow the same naming convention as the ProDOS calls.

CALLING CONVENTIONS

CALLING CONVENTIONS - DOCUMENTATION CHANGES

The only two routines that use parameter blocks in the shell interface are GET_LINE_INFO and SET_LINE_INFO. The other routines are called with the required parameters. If the shell routines are called from assembly language, then the shell macros require a parameter block for all calls. The shell routines are called exactly the same as ProDOS calls.

CONTENTS OF APWC V1.0B4 DISK

LIBRARIES	
CINCLUDE	directory containing standard C headers and Apple IIGS tool interfaces
MEMORY.H	
DIALOG.H	
EVENT.H	
FONT.H	
INTMATH.H	
LINEEDIT.H	
LIST.H	
LOADER.H	
LOCATOR.H	
MENU.H	
MISCTOOL.H	
NOTESYN.H	
PRINT.H	
PRODOS.H	
QDAUX.H	
QUICKDRAW.H	
SCHEDULER.H	
SCRAP.H	
SHELL.H	
SOUND.H	
STDFILE.H	
TEXTTOOL.H	
TYPES.H	
WINDOW.H	
CONTROL.H	
DESK.H	
CTYPE.H	
ERRNO.H	
ERRORS.H	
FCNTL.H	
FILES.H	
IOCTL.H	
SANE.H	
SIGNAL.H	
STDIO.H	
VALUES.H	
VARARGS.H	
SETJMP.H	
DEBUG.H	
MALLOC.H	
MATH.H	
STRING.H	
CLIB	standard C library
SYSINTERFACE	Apple IIGS tool interface library
START.ROOT	module containing C initialization code
LANGUAGES	
CC	APW C compiler

LIBRARY INTERNALS

START.ROOT. START.ROOT initializes the **Memory Manager, Text Tools, and SANE**, and shuts them down on exit. **A program should neither initialize nor shut down those managers.** All other managers must be initialized and shut down by the application, as needed.
The global variable `_ownerid` (unsigned) contains the user ID for the program. Memory allocated with this ID will automatically be recovered by the system when the program exits. If a programmer asks for a new ID number, he must recover the memory himself.
The C library terminates with a ProDOS QUIT call. QUIT accepts a flag indicating that the program is restartable. Unfortunately, this version of the system does not support restartable programs. Future versions of the C library will allow a programmer to specify whether a program is restartable.

HOW TO CHANGE THE DEFAULT STACK SIZE. The system loader allocates 1K of memory in Bank 0 for a program's run-time stack. This is adequate for small programs. Most programs will need more than 1K; at least 8K is recommended. Here is how to tell the loader to allocate more stack space for your program.
You can specify the amount of stack space you want, from 1K to 31K, in 1K increments. The directory /APWC/LIBRARIES contains files STACK1K.ROOT, STACK2K.ROOT, STACK4K.ROOT, STACK8K.ROOT, and STACK16K.ROOT. These provide stack space of 1K, 2K, 4K, 8K, and 16K, respectively. Simply link in the combination of STACK??K.ROOT files that add up to the amount of space you need. For example, the following command will provide 9K of stack space for MYPROG:

```
        LINK 2/START MYPROG 2/STACK8K 2/STACK1K KEEP=MYPROG
```

COMPARISON WITH MPW C

BIT FIELDS. The APW C Compiler will not take 'int' for bit field. The MPW C compiler allows it. In APW C you should use 'unsigned int' for your bit fields. (Actually this is more portable and is the preferred use in any C dialect.)

CHARACTERS. Characters are inherently '`unsigned`' on APW C. That means that when a char variable is promoted to int, as in '`foo(mychar)`' then an unsigned extension is performed. On MPW C, characters are promoted to '`int`' via signed extensions.

MACRO EXPANSION. When the following macro

```
                    #define mesg(x) printf("x is %d", s)
```

is defined and used somewhere in the program as `mesg(i)`, some C compilers will print out "i is 50" but APW C always prints out "x is 50". That is, macro substitution does not and will not ever take place inside of quoted strings.

PREPROCESSOR. '#' on the first column followed by nothing represents an illegal preprocessor command to the APW C compiler. MPW C will not complain.

CHANGES TO THE C SYSTEM INTERFACE

This section of the release notes is intended for the developers who were previously 'seeded' by Apple with the earlier versions of APW. It describes the differences between the headers contained in the current release and their previous versions. If you have not used previous releases of APW, please skip this section.

The names have been modified to conform with the Inside Mac conventions. The conventions used are as follows:

(i) All constants start with a lowercase letter.
(ii) All fields within structures start with a lowercase letter.
(iii) All procedure names start with an uppercase letter.
(iv) All variable names start with an uppercase letter.

We have tried to follow these conventions in all the header files. To make the transition easier for the developers who have been using the old names, we are providing a file called CANON.DICT (which can be found in the UTILITIES subdirectory of the /APW disk), which has the correct spelling for each name in the header files. This file may be used with the CANON utility to modify the names in existing source files.

NOTE: the current release of the system interface corresponds to the toolbox interfaces described in the "Apple IIGS Toolbox Reference" published by APDA in April of 1987 and may not work with the same publication released in November of 1986.

APW SUGGESTIONS AND EVALUATION

After you have had a chance to become familiar with the APW software you are using, please complete this brief form.

What is your company name?______________________________

How long did it take for you to feel relatively comfortable with the APW environment?

What do you like best about APW ?

What do you like least about APW?

What would you like to see added to future versions of APW?

Would you use a MPW based cross-development system for the Apple IIGS?

General comments

Please return this completed form to:

APW Product Manager, MS 27-S
Apple Computer, Inc.
10500 N. DeAnza Blvd.
Cupertino, CA 95014

APW BUG REPORT

Date______________**APW Version**______**System Disk Version**______

Area (circle as many as required):

Compiler: C
Assembler
Library
Shell/Editor
Utility____
Performance
Documentation

Configuration:

Hard Disk? NO YES-Slot#____-Brand________________________
Apple IIGS Memory Expansion card size_________________________
Apple II Memory Expansion Card? NO YES-Slot#___-Size_____

Bug Description:

____

Contact Information:

NAME__________________________**Phone#**__________________
Company Name____
Address____
City, State, Zip____

Please return this completed form to:

APW Testing, MS 27-G
Apple Computer, Inc.
10500 N. DeAnza Blvd.
Cupertino, CA 95014

```
*********** Control Manager Procedures ********************

void CtlBootInit()

void CtlNewRes()

void CtlReset()

void CtlShutDown()

void CtlStartUp(userID,dPageAddr)
Word userID;
Word dPageAddr;

Boolean CtlStatus()

unsigned int CtlVersion()

void DisposeControl(theControlHandle)
CtlRecHndl theControlHandle;

void DragControl(startPoint,limitRectPtr,slopRectPtr,axis,theControlHandle)
Point startPoint;
RectPtr limitRectPtr;
RectPtr slopRectPtr;
Word axis;
CtlRecHndl theControlHandle;

unsigned long DragRect(actionProcPtr,dragPatternPtr,start,dragRectPtr,limitRectPtr,slopRectPtr,axis)
ProcPtr actionProcPtr;
Pattern dragPatternPtr;
Point start;
RectPtr dragRectPtr;
RectPtr limitRectPtr;
RectPtr slopRectPtr;
Word axis;

void DrawControls(theWindowPtr)
GrafPortPtr theWindowPtr;

void DrawOneCtl(theControlHandle)
CtlRecHndl theControlHandle;

void EraseControl(theControlHandle)
CtlRecHndl theControlHandle;

unsigned int FindControl(foundCtlPtr,foundPoint,theWindowPtr)
Pointer foundCtlPtr;
Point foundPoint;
GrafPortPtr theWindowPtr;

ProcPtr GetCtlAction(theControlHandle)
CtlRecHndl theControlHandle;

unsigned int GetCtlDPage()

unsigned long GetCtlParams(theControlHandle)
CtlRecHndl theControlHandle;

unsigned long GetCtlRefCon(theControlHandle)
CtlRecHndl theControlHandle;

Ptr GetCtlTitle(theControlHandle)
CtlRecHndl theControlHandle;

unsigned int GetCtlValue(theControlHandle)
CtlRecHndl theControlHandle;

unsigned long GrowSize()

void HideControl(theControlHandle)
CtlRecHndl theControlHandle;

void HiliteControl(hiliteState,theControlHandle)
Word hiliteState;
CtlRecHndl theControlHandle;

void KillControls(theWindowPtr)
GrafPortPtr theWindowPtr;

void MoveControl(newPoint,theControlHandle)
Point newPoint;
CtlRecHndl theControlHandle;

CtlRecHndl NewControl(theWindowPtr,boundsRectPtr,titlePtr,flag,value,param1,param2,defProcPtr,
refCon,colorTablePtr)
GrafPortPtr theWindowPtr;
RectPtr boundsRectPtr;
Pointer titlePtr;
Word flag;
Word value;
Word param1;
Word param2;
ProcPtr defProcPtr;
LongInt refCon;
CtlColorTablePtr colorTablePtr;

void SetCtlAction(newActionPtr,theControlHandle)
ProcPtr newActionPtr;
CtlRecHndl theControlHandle;

FontHndl SetCtlIcons(newFontHandle)
FontHndl newFontHandle;

void SetCtlParams(param1,param2,theControlHandle)
Word param1;
Word param2;
CtlRecHndl theControlHandle;

void SetCtlRefCon(newRefCon,theControlHandle)
DblWord newRefCon;
CtlRecHndl theControlHandle;

void SetCtlTitle(titlePtr,theControlHandle)
Pointer titlePtr;
CtlRecHndl theControlHandle;

void SetCtlValue(curValue,theControlHandle)
Word curValue;
CtlRecHndl theControlHandle;

void ShowControl(theControlHandle)
CtlRecHndl theControlHandle;

unsigned int TestControl(testPoint,theControlHandle)
Point testPoint;
CtlRecHndl theControlHandle;

unsigned int TrackControl(Start,actionProcPtr,theControlHandle)
Point Start;
```

```
ProcPtr actionProcPtr;
CtlRecHndl theControlHandle;

**************** Desk Manager Procedures ********************


void ChooseCDA()

void CloseAllNDAs()

void CloseNDA(refNum)
Word refNum;

void CloseNDAByWinPtr(theWindowPtr)
GrafPortPtr theWindowPtr;

void DeskBootInit()

void DeskReset()

void DeskShutDown()

void DeskStartUp()

Boolean DeskStatus()

unsigned int DeskVersion()

void FixAppleMenu(startingID)
Word startingID;

Ptr GetDAStrPtr(daIDNum)
Word daIDNum;

int GetNumNDAs()

void InstallCDA(idHandle)
Handle idHandle;

void InstallNDA(idHandle)
Handle idHandle;

unsigned int OpenNDA(idNum)
Word idNum;

void RestAll()

void RestScrn()

void SaveAll()

void SaveScrn()

void SetDAStrPtr(altDispHandle,stringTablePtr)
Handle altDispHandle;
Pointer stringTablePtr;

void SystemClick(eventRecPtr,theWindowPtr,findWndwResult)
EventRecordPtr eventRecPtr;
GrafPortPtr theWindowPtr;
Word findWndwResult;

Boolean SystemEdit(editType)
Word editType;

Boolean SystemEvent(eventWhat,eventMessage,eventWhen,eventWhere,eventMods)
Word eventWhat;
LongWord eventMessage;
LongWord eventWhen;
Point eventWhere;
Word eventMods;

void SystemTask()

**************** Dialog Manager Procedures ********************


unsigned int Alert(alertTemplatePtr,filterProcPtr)
AlertTempPtr alertTemplatePtr;
ProcPtr filterProcPtr;

unsigned int CautionAlert(alertTemplatePtr,filterProcPtr)
AlertTempPtr alertTemplatePtr;
ProcPtr filterProcPtr;

void CloseDialog(theDialogPtr)
GrafPortPtr theDialogPtr;

Boolean DefaultFilter(theDialogPtr,theEventPtr,itemHitPtr)
GrafPortPtr theDialogPtr;
EventRecordPtr theEventPtr;
WordPtr itemHitPtr;

void DialogBootInit()

void DialogReset()

Boolean DialogSelect(theEventPtr,theDialogPtr,itemHitPtr)
EventRecordPtr theEventPtr;
GrafPortHndl theDialogPtr;
WordPtr itemHitPtr;

void DialogShutDown()

void DialogStartUp(userID)
Word userID;

Boolean DialogStatus()

unsigned int DialogVersion()

void DisableDItem(theDialogPtr,itemID)
GrafPortPtr theDialogPtr;
Word itemID;

void DlgCopy(theDialogPtr)
GrafPortPtr theDialogPtr;

void DlgCut(theDialogPtr)
GrafPortPtr theDialogPtr;

void DlgDelete(theDialogPtr)
GrafPortPtr theDialogPtr;

void DlgPaste(theDialogPtr)
GrafPortPtr theDialogPtr;

void DrawDialog(theDialogPtr)
GrafPortPtr theDialogPtr;

void EnableDItem(theDialogPtr,itemID)
GrafPortPtr theDialogPtr;
```

```
Word itemID;

void ErrorSound(soundProcPtr)
ProcPtr soundProcPtr;

unsigned int FindDItem(theDialogPtr,thePoint)
GrafPortPtr theDialogPtr;
Point thePoint;

unsigned int GetAlertStage()

CtlRecHndl GetControlDItem(theDialogPtr,itemID)
GrafPortPtr theDialogPtr;
Word itemID;

unsigned int GetDefButton(theDialogPtr)
GrafPortPtr theDialogPtr;

void GetDItemBox(theDialogPtr,itemID,itemBoxPtr)
GrafPortPtr theDialogPtr;
Word itemID;
RectPtr itemBoxPtr;

unsigned int GetDItemType(theDialogPtr,itemID)
GrafPortPtr theDialogPtr;
Word itemID;

unsigned int GetDItemValue(theDialogPtr,itemID)
GrafPortPtr theDialogPtr;
Word itemID;

unsigned int GetFirstDItem(theDialogPtr)
GrafPortPtr theDialogPtr;

void GetIText(theDialogPtr,itemID,theStringPtr)
GrafPortPtr theDialogPtr;
Word itemID;
Pointer theStringPtr;

void GetNewDItem(theDialogPtr,itemTemplatePtr)
GrafPortPtr theDialogPtr;
ItemTempPtr itemTemplatePtr;

GrafPortPtr GetNewModalDialog(dialogTemplatePtr)
DlgTempPtr dialogTemplatePtr;

unsigned int GetNextDItem(theDialogPtr,itemID)
GrafPortPtr theDialogPtr;
Word itemID;

void HideDItem(theDialogPtr,itemID)
GrafPortPtr theDialogPtr;
Word itemID;

Boolean IsDialogEvent(theEventPtr)
EventRecordPtr theEventPtr;

unsigned int ModalDialog(filterProcPtr)
ProcPtr filterProcPtr;

unsigned long HodalDialog2(filterProcPtr)
ProcPtr filterProcPtr;

void NewDItem(theDialogPtr,itemID,itemRectPtr,itemType,itemDescr,itemValue,itemFlag,itemColorPtr)
GrafPortPtr theDialogPtr;
Word itemID;
RectPtr itemRectPtr;
Word itemType;
pointer itemDescr;
Word itemValue;
Word itemFlag;
CtlColorTablePtr itemColorPtr;

GrafPortPtr NewModalDialog(dBoundsRectPtr,dVisibleFlag,dRefCon)
RectPtr dBoundsRectPtr;
Boolean dVisibleFlag;
DblWord dRefCon;

GrafPortPtr NewModelessDialog(dBoundsRectPtr,dTitlePtr,dBehindPtr,dFlag,dRefCon,dFullSizePtr)
RectPtr dBoundsRectPtr;
Pointer dTitlePtr;
GrafPortPtr dBehindPtr;
Word dFlag;
DblWord dRefCon;
RectPtr dFullSizePtr;

unsigned int NoteAlert(alertTemplatePtr,filterProcPtr)
GrafPortPtr alertTemplatePtr;
ProcPtr filterProcPtr;

void ParamText(param0Ptr,param1Ptr,param2Ptr,param3Ptr)
Pointer param0Ptr;
Pointer param1Ptr;
Pointer param2Ptr;
Pointer param3Ptr;

void RemoveDItem(theDialogPtr,itemID)
GrafPortPtr theDialogPtr;
Word itemID;

void ResetAlertStage()

void SelIText(theDialogPtr,itemID,startSel,endSel)
GrafPortPtr theDialogPtr;
Word itemID;
Word startSel;
Word endSel;

void SetDAFont(fontHandle)
FontHndl fontHandle;

void SetDefButton(defButtonID,theDialogPtr)
Word defButtonID;
GrafPortPtr theDialogPtr;

void SetDItemBox(theDialogPtr,itemID,itemBoxPtr)
GrafPortPtr theDialogPtr;
Word itemID;
RectPtr itemBoxPtr;

void SetDItemType(itemType,theDialogPtr,itemID)
Word itemType;
GrafPortPtr theDialogPtr;
Word itemID;

void SetDItemValue(itemValue,theDialogPtr,itemID)
Word itemValue;
GrafPortPtr theDialogPtr;
Word itemID;

void SetIText(theDialogPtr,itemID,theStringPtr)
GrafPortPtr theDialogPtr;
```

```
Word itemID;
Pointer theStringPtr;

void ShowDItem(theDialogPtr,itemID)
GrafPortPtr theDialogPtr;
Word itemID;

unsigned int StopAlert(alertTemplatePtr,filterProcPtr)
GrafPortPtr alertTemplatePtr;
ProcPtr filterProcPtr;

void UpdateDialog(theDialogPtr,updateRgnHandle)
GrafPortPtr theDialogPtr;
RgnHandle updateRgnHandle;

****************** Event Manager Procedures ************

Boolean Button(buttonNum)
Word buttonNum;

unsigned int DoWindows()

void EMBootInit()

void EMResut()

void EMShutDown()

void EMStartUp(dPageAddr,queueSize,xMinClamp,xMaxClamp,yMinClamp,yMaxClamp,userID)
Word dPageAddr;
Integer queueSize;
Integer xMinClamp;
Integer xMaxClamp;
Integer yMinClamp;
Integer yMaxClamp;
Word userID;

Boolean EMStatus()

unsigned int EMVersion()

Boolean EventAvail(eventMask,eventPtr)
Word eventMask;
EventRecordPtr eventPtr;

void FakeMouse(changedFlag,modLatch,padding,absPos,buttonStatus)
Word changedFlag;
Byte modLatch;
Byte padding;
Point absPos;
Word buttonStatus;

unsigned int FlushEvents(eventMask,stopMask)
Word eventMask;
Word stopMask;

unsigned long GetCaretTime()

unsigned long GetDblTime()

void GetMouse(mouseLocPtr)
PointPtr mouseLocPtr;

Boolean GetNextEvent(eventMask,eventPtr)
Word eventMask;
EventRecordPtr eventPtr;

Boolean GetOSEvent(eventMask,eventPtr)
Word eventMask;
EventRecordPtr eventPtr;

Boolean OSEventAvail(eventMask,eventPtr)
Word eventMask;
EventRecordPtr eventPtr;

unsigned int PostEvent(eventCode,eventMsg)
Word eventCode;
DblWord eventMsg;

void SetEventMask(theMask)
Word theMask;

void SetSwitch()

Boolean StillDown(buttonNum)
Word buttonNum;

unsigned long TickCount()

Boolean WaitMouseUp(buttonNum)
Word buttonNum;

**************** Font Manager Procedures ********************

void AddFamily(famNum,namePtr)
Word famNum;
Pointer namePtr;

void AddFontVar(fontHandle,newSpecs)
FontHndl fontHandle;
Word newSpecs;

unsigned long ChooseFont(currentID,famSpecs)
Dblword currentID;
Word famSpecs;

unsigned int CountFamilies(famSpecs)
Word famSpecs;

unsigned int CountFonts(desiredID,fontSpecs)
Dblword desiredID;
Word fontSpecs;

unsigned int FindFamily(famSpecs,positionNum,namePtr)
Word famSpecs;
Word positionNum;
Pointer namePtr;

void FindFontStats(desiredID,fontSpecs,positionNum,resultPtr)
Dblword desiredID;
Word fontSpecs;
Word positionNum;
FontStatRecPtr resultPtr;

void FixFontMenu(menuID,startingID,famSpecs)
Word menuID;
Word startingID;
Word famSpecs;

void FMBootInit()

unsigned long FMGetCurFID()
unsigned long FMGetSysFID()

void FMReset()

void FMSetSysFont(fontID)
DblWord fontID;

void FMShutDown()

void FMStartUp(userID,dPageAddr)
Word userID;
Word dPageAddr;

Boolean FMStatus()

unsigned int FMVersion()

unsigned int GetFamInfo(famNum,namePtr)
Word famNum;
Pointer namePtr;

unsigned int GetFamNum(namePtr)
Pointer namePtr;

void InstallFont(desiredID,scaleWord)
Dblword desiredID;
Word scaleWord;

unsigned int itemID2FamNum(itemID)
Word itemID;

void LoadFont(desiredID,fontSpecs,positionNum,resultPtr)
Dblword desiredID;
Word fontSpecs;
Word positionNum;
FontStatRecPtr resultPtr;

void LoadSysFont()

void SetPurgeStat(fontID,purgeStat)
Dblword fontID;
Word purgeStat;

***************** IntMath Procedures ******************** *********************

int Dec2Int(strPtr,strLength,signedFlag)
Pointer strPtr;
Integer strLength;
Boolean signedFlag;

long Dec2Long(strPtr,strLength,signedFlag)
Pointer strPtr;
Integer strLength;
Boolean signedFlag;

long Fix2Frac(fixedValue)
Longint fixedValue;

long Fix2Long(fixedValue)
Longint fixedValue;

void Fix2X(fixedValue,extendPtr)
Longint fixedValue;
Pointer extendPtr;

long FixATan2(input1,input2)
Longint input1;
Longint input2;

long FixDiv(quotient,divisor)
Longint quotient;
Longint divisor;

long FixMul(multiplicand,multiplier)
Word multiplicand;
Word multiplier;

long FixRatio(numerator,denominator)
Word numerator;
Word denominator;

int FixRound(fixedValue)
Longint fixedValue;

long Frac2Fix(fracValue)
Longint fracValue;

void Frac2X(fracValue,extendPtr)
Longint fracValue;
Pointer extendPtr;

long FracCos(angle)
Longint angle;

long FracDiv(quotient,divisor)
Longint quotient;
Longint divisor;

long FracMul(multiplicand,multiplier)
Word multiplicand;
Word multiplier;

long FracSin(angle)
Longint angle;

long FracSqrt(fracValue)
Longint fracValue;

int Hex2Int(strPtr,strLength)
Pointer strPtr;
Integer strLength;

long Hex2Long(strPtr,strLength)
Pointer strPtr;
Integer strLength;

unsigned long HexIt(intValue)
Word intValue;

unsigned int HiWord(longValue)
Longint longValue;

void IMBootInit()

void IMReset()

void IMShutDown()

void IMStartUp()

Boolean IMStatus()
```

```
unsigned int IMVersion()

void Int2Dec(wordValue,strPtr,strLength,signedFlag)
Integer wordValue;
Pointer strPtr;
Integer strLength;
Boolean signedFlag;

void Int2Hex(intValue,strPtr,strLength)
Word intValue;
Pointer strPtr;
Integer strLength;

void Long2Dec(longValue,strPtr,strLength,signedFlag)
LongInt longValue;
Pointer strPtr;
Integer strLength;
Boolean signedFlag;

long Long2Fix(longValue)
LongInt longValue;

void Long2Hex(longValue,strPtr,strLength)
LongInt longValue;
Pointer strPtr;
Integer strLength;

void LongDivide(numerator,denominator,resultRecPtr)
Long numerator;
Long denominator;
Pointer resultRecPtr;

void LongMul(multiplicand,multiplier,resultRecPtr)
Long multiplicand;
Long multiplier;
Pointer resultRecPtr;

unsigned int LoWord(longValue)
LongInt longValue;

void Multiply(multiplicand,multiplier,resultRecPtr)
Word multiplicand;
Word multiplier;
Pointer resultRecPtr;

void SDivide(numerator,denominator,resultRecPtr)
Word numerator;
Word denominator;
Pointer resultRecPtr;

void UDivide(numerator,denominator,resultRecPtr)
Word numerator;
Word denominator;
Pointer resultRecPtr;

long X2Fix(extendPtr)
Pointer extendPtr;

long X2Frac(extendPtr)
Pointer extendPtr;

***************** LineEdit Procedures ********************* *********************

void LEActivate(handleLE)
Handle handleLE;
```

```
void LEBootInit()

void LEClick(eventPtr,handleLE)
Pointer eventPtr;
Handle handleLE;

void LECopy(handleLE)
Handle handleLE;

void LECut(handleLE)
Handle handleLE;

void LEDeactivate(handleLE)
Handle handleLE;

void LEDelete(handleLE)
Handle handleLE;

void LEDispose(handleLE)
Handle handleLE;

void LEFromScrap()

unsigned int LEGetScrapLen()

void LEIdle(handleLE)
Handle handleLE;

void LEInsert(textPtr,textLength,handleLE)
Pointer textPtr;
Integer textLength;
Handle handleLE;

void LEKey(theKey,modifiers,handleLE)
Word theKey;
Word modifiers;
Handle handleLE;

LERecHndl LENew(destRectPtr,viewRectPtr,maxTextLen)
Pointer destRectPtr;
Pointer viewRectPtr;
Integer maxTextLen;

void LEPaste(handleLE)
Handle handleLE;

void LEReset()

Handle LEScrapHandle()

void LESetCaret(caretAddrPtr,handleLE)
Pointer caretAddrPtr;
Handle handleLE;

void LESetHilite(hiliteAddrPtr,handleLE)
Pointer hiliteAddrPtr;
Handle handleLE;

void LESetScrapLen(newLength)
Integer newLength;

void LESetSelect(selStart,selEnd,handleLE)
Integer selStart;
Integer selEnd;
Handle handleLE;
```

```
void LESetText(textPtr,textLength,handleLE)
Pointer textPtr;
Integer textLength;
Handle handleLE;

void LEShutDown()

void LEStartUp(userID,dPageAddr)
Word userID;
Word dPageAddr;

Boolean LEStatus()

void LETextBox(textPtr,textLength,boxPtr,textJustify)
Pointer textPtr;
Integer textLength;
Pointer boxPtr;
Integer textJustify;

void LETextBox2()

void LEToScrap()

void LEUpdate(handleLE)
Handle handleLE;

unsigned int LEVersion()

**************** List Manager Procedures ********************


ListCtlRecHndl CreateList()

void DrawMember()

ProcPtr GetListDefProc()

void ListBootInit()

void ListReset()

void ListShutDown()

void ListStartup()

Boolean ListStatus()

unsigned int ListVersion()

void NewList()

Ptr NextMember()

Ptr ResetMember()

void SelectMember()

void SortList()

**************** Loader Procedures ******************** ********************

void GetLoadSegInfo(userID,loadFileNum,loadSegNum,bufferPtr)
Word userID;
Word loadFileNum;
Word loadSegNum;
Pointer bufferPtr;

unsigned int GetUserID(pathNamePtr)
Pointer pathNamePtr;

void InitialLoad(userID,loadFileNamePtr,spMemFlag,resultRecPtr)
Word userID;
Pointer loadFileNamePtr;
Boolean spMemFlag;
Pointer resultRecPtr;

Ptr LGetPathname(userID,fileNumber)
Word userID;
Word fileNumber;

void LoaderInitialization()

void LoaderReset()

void LoaderShutDown()

void LoaderStartUp()

Boolean LoaderStatus()

unsigned int LoaderVersion()

void LoadSegName(userID,loadFileNamePtr,loadSegNamePtr,resultRecPtr)
Word userID;
Pointer loadFileNamePtr;
Pointer loadSegNamePtr;
Pointer resultRecPtr;

Ptr LoadSegNum(userID,loadFileNum,loadSegNum)
Word userID;
Word loadFileNum;
Word loadSegNum;

void Restart(userID,resultRecPtr)
Word userID;
Pointer resultRecPtr;

void UnloadSeg(segmentPtr,resultRecPtr)
Pointer segmentPtr;
Pointer resultRecPtr;

void UnloadSegNum(userID,loadFileNum,loadSegNum)
Word userID;
Word loadFileNum;
Word loadSegNum;

unsigned int UserShutDown(userID,restartFlag)
Word userID;
Word restartFlag;

**************** Locator Procedures ******************** ********************

Ptr GetFuncPtr(userOrSystem,funcTSNum)
Integer userOrSystem;
Word funcTSNum;

Ptr GetTSPtr(userOrSystem,tSNum)
Integer userOrSystem;
Word tSNum;

Ptr GetWAP(userOrSystem,tSNum)
```

```
Integer userOrSystem;
Word tSNum;

void LoadOneTool(toolNumber,minVersion)
Integer toolNumber;
Word minVersion;

void LoadTools(toolTablePtr)
Pointer toolTablePtr;

void MessageCenter()

void RestoreTextState()

Handle SaveTextState()

void SetTSPtr(userOrSystem,tSNum,fPTPtr)
Integer userOrSystem;
Word tSNum;
Pointer fPTPtr;

void SetWAP(userOrSystem,tSNum,wAPTPtr)
Integer userOrSystem;
Word tSNum;
Pointer wAPTPtr;

void TLBootInit()

unsigned int TLMountVolume(whereX,whereY,line1Ptr,line2Ptr,but1Ptr,but2Ptr)
Integer whereX;
Integer whereY;
Pointer line1Ptr;
Pointer line2Ptr;
Pointer but1Ptr;
Pointer but2Ptr;

void TLReset()

void TLShutDown()

void TLStartUp()

Boolsan TLStatus()

unsigned int TLTextMountVolume(line1Ptr,line2Ptr,but1Ptr,but2Ptr)
Pointer line1Ptr;
Pointer line2Ptr;
Pointer but1Ptr;
Pointer but2Ptr;

unsigned int TLVersion()

void UnloadOneTool(toolNumber)
Integer toolNumber;

********** Memory Manager Procedures ********************

void BlockMove(sourcePtr,destPtr,count)
Pointer sourcePtr;
Pointer destPtr;
LongInt count;

void CheckHandle(theHandle)
Handle theHandle;

void CompactMem()

void DisposeAll(userID)
Word userID;

void DisposeHandle(theHandle)
Handle theHandle;

Handle FindHandle(locationPtr)
Pointer locationPtr;

unsigned long FreeMem()

unsigned long GetHandleSize(theHandle)
Handle theHandle;

void HandToHand(sourceHandle,destHandle,count)
Handle sourceHandle;
Handle destHandle;
LongInt count;

void HandToPtr(sourceHandle,destPtr,count)
Handle sourceHandle;
Pointer destPtr;
LongInt count;

void HLock(theHandle)
Handle theHandle;

void HLockAll(userID)
Word userID;

void HUnlock(theHandle)
Handle theHandle;

void HUnlockAll(userID)
Word userID;

unsigned long MaxBlock()

void MMBootInit()

void MMReset()

void MMShutDown(userID)
Word userID;

unsigned int MMStartUp()

Boolean MMStatus()

unsigned int MMVersion()

Handle NewHandle(blockSize,userID,attributes,locationPtr)
LongInt blockSize;
Word userID;
Word attributes;
Pointer locationPtr;

void PtrToHand(sourcePtr,destHandle,count)
Pointer sourcePtr;
Handle destHandle;
LongInt count;

void PurgeAll(userID)
Word userID;
```

```
void PurgeHandle(theHandle)
Handle theHandle;

void ReallocHandle(blockSize,userID,attributes,locationPtr,theHandle)
LongInt blockSize;
Word userID;
Word attributes;
Pointer locationPtr;
Handle theHandle;

void RestoreHandle(theHandle)
Handle theHandle;

void SetHandleSize(newSize,theHandle)
LongInt newSize;
Handle theHandle;

void SetPurge(newPurgeLevel,theHandle)
Word newPurgeLevel;
Handle theHandle;

void SetPurgeAll(newPurgeLevel,userID)
Word newPurgeLevel;
Word userID;

unsigned long TotalMem()

**************** Menu Manager Procedures ********************


void CalcMenuSize(newWidth,newHeight,menuNum)
Word newWidth;
Word newHeight;
Word menuNum;

void CheckMItem(checkedFlag,itemNum)
Boolean checkedFlag;
Word itemNum;

unsigned int CountMItems(menuNum)
Word menuNum;

void DeleteMenu(menuNum)
Word menuNum;

void DeleteMItem(itemNum)
Word itemNum;

void DisableMItem(itemNum)
Word itemNum;

void DisposeMenu(menuHandle)
Handle menuHandle;

void DrawMenuBar()

void EnableMItem(itemNum)
Word itemNum;

unsigned int FixMenuBar()

void FlashMenuBar()

unsigned long GetBarColors()

CtlRecHndl GetMenuBar()

unsigned int GetMenuFlag(menuNum)
Word menuNum;

GrafPortPtr GetMenuMgrPort()

Ptr GetMenuTitle(menuNum)
Word menuNum;

MenuRecHndl GetMHandle(menuNum)
Word menuNum;

Ptr GetMItem(itemNum)
Word itemNum;

unsigned int GetMItemFlag(itemNum)
Word itemNum;

unsigned int GetMItemMark(itemNum)
Word itemNum;

TextStyle GetMItemStyle(itemNum)
Word itemNum;

unsigned int GetMTitleStart()

unsigned int GetMTitleWidth(menuNum)
Word menuNum;

CtlRecHndl GetSysBar()

void HiliteMenu(hiliteFlag,menuNum)
Boolean hiliteFlag;
Word menuNum;

void InitPalette()

void InsertMenu(addMenuHandle,insertAfter)
Handle addMenuHandle;
Word insertAfter;

void InsertMItem(addItemPtr,insertAfter,menuNum)
Pointer addItemPtr;
Word insertAfter;
Word menuNum;

void MenuBootInit()

unsigned int MenuGlobal()

void MenuKey(taskRecPtr,barHandle)
Pointer taskRecPtr;
Handle barHandle;

void MenuNewRes()

void MenuRefresh(redrawRoutinePtr)
Pointer redrawRoutinePtr;

void MenuReset()

void MenuSelect(taskRecPtr,barHandle)
Pointer taskRecPtr;
Handle barHandle;

void MenuShutDown()
void MenuStartUp(userID,dPageAddr)
Word userID;
Word dPageAddr;

Boolean MenuStatus()

unsigned int MenuVersion()

MenuRecHndl NewMenu(menuStringPtr)
Pointer menuStringPtr;

CtlRecHndl NewMenuBar(theWindowPtr)
Pointer theWindowPtr;

void SetBarColors(newBarColor,newInvertColor,newOutColor)
Word newBarColor;
Word newInvertColor;
Word newOutColor;

void SetMenuBar(barHandle)
Handle barHandle;

void SetMenuFlag(newValue,menuNum)
Word newValue;
Word menuNum;

void SetMenuID(newMenuNum,curMenuNum)
Word newMenuNum;
Word curMenuNum;

void SetMenuTitle(newStrPtr,menuNum)
Pointer newStrPtr;
Word menuNum;

void SetMItem(newStrPtr,itemNum)
Pointer newStrPtr;
Word itemNum;

void SetMItemBlink(count)
Word count;

void SetMItemFlag(newValue,itemNum)
Word newValue;
Word itemNum;

void SetMItemID(newItemNum,curItemNum)
Word newItemNum;
Word curItemNum;

void SetMItemMark(mark,itemNum)
Word mark;
Word itemNum;

void SetMItemName(strPtr,itemNum)
Pointer strPtr;
Integer itemNum;

void SetMItemStyle(textStyle,itemNum)
Word textStyle;
Word itemNum;

void SetMTitleStart(xStart)
Word xStart;

void SetMTitleWidth(newWidth,menuNum)
Word newWidth;
Word menuNum;

void SetSysBar(barHandle)
Handle barHandle;

**************** MiscTool Procedures ********************

void ClampMouse(xMinClamp,xMaxClamp,yMinClamp,yMaxClamp)
Word xMinClamp;
Word xMaxClamp;
Word yMinClamp;
Word yMaxClamp;

void ClearMouse()

void ClrHeartBeat()

void DeleteID(iDTag)
Word iDTag;

void DelHeartBeat(taskPtr)
Pointer taskPtr;

void FWEntry(aRegValue,xRegValue,yRegValue,eModeEntryPt,resultRecPtr)
Word aRegValue;
Word xRegValue;
Word yRegValue;
Word eModeEntryPt;
Pointer resultRecPtr;

void GetAbsClamp(resultRecPtr)
Pointer resultRecPtr;

Ptr GetAddr(refNum)
Word refNum;

unsigned int GetIRQEnable()

void GetMouseClamp(resultRecPtr)
Pointer resultRecPtr;

unsigned int GetNewID(iDTag)
Word iDTag;

unsigned long GetTick()

Ptr GetVector(vectorRefNum)
Word vectorRefNum;

void HomeMouse()

void InitMouse(mouseSlot)
Word mouseSlot;

void IntSource(srcRefNum)
Word srcRefNum;

void MTBootInit()

void MTReset()

void MTShutDown()

void MTStartUp()
```

```
Boolean MTStatus()

unsigned int MTVersion()

unsigned int Munger(destPtr,destLenPtr,targPtr,targLen,replPtr,replLen,padPtr)
Pointer destPtr;
Pointer destLenPtr;
Pointer targPtr;
Integer targLen;
Pointer replPtr;
Integer replLen;
Pointer padPtr;

unsigned int PackBytes(startHandle,sizePtr,bufferPtr,bufferSize)
Handle startHandle;
Pointer sizePtr;
Pointer bufferPtr;
Word bufferSize;

void PosMouse(xPos,yPos)
Word xPos;
Word yPos;

void ReadAsciiTime(bufferAddrPtr)
Pointer bufferAddrPtr;

unsigned int ReadBParam(paramRefNum)
Word paramRefNum;

void ReadBRam(bufferAddrPtr)
Pointer bufferAddrPtr;

void ReadMouse(resultRecPtr)
Pointer resultRecPtr;

void ReadTimeHex(resultRecPtr)
Pointer resultRecPtr;

unsigned int ServeMouse()

void SetAbsClamp(xMinClamp,xMaxClamp,yMinClamp,yMaxClamp)
Word xMinClamp;
Word xMaxClamp;
Word yMinClamp;
Word yMaxClamp;

void SetHeartBeat(taskPtr)
Pointer taskPtr;

void SetMouse(mouseMode)
Word mouseMode;

void SetVector(vectorRefNum,vectorAddrPtr)
Word vectorRefNum;
Pointer vectorAddrPtr;

void StatusID(iDTag)
Word iDTag;

void SysBeep()

void SysFailMgr(errorCode,strPtr)
Word errorCode;
Pointer strPtr;

unsigned int UnPackBytes(bufferPtr,bufferSize,startHandle,sizePtr)
Pointer bufferPtr;
Word bufferSize;
Handle startHandle;
Pointer sizePtr;

void WriteBParam(theData,paramRefNum)
Word theData;
Word paramRefNum;

void WriteBRam(bufferAddrPtr)
Pointer bufferAddrPtr;

void WriteTimeHex(month,day,curYear,hour,minute,second)
Byte month;
Byte day;
Byte curYear;
Byte hour;
Byte minute;
Byte second;

**************** NoteSyn Procedures ********************

void AllNotesOff()

unsigned int AllocGen(requestPriority)
Word requestPriority;

void DeallocGen(genNumber)
Word genNumber;

void NoteOff(genNumber,semitone)
Word genNumber;
Word semitone;

void NoteOn(genNumber,semitone,volume,instrumentPtr)
Word genNumber;
Word semitone;
Word volume;
Pointer instrumentPtr;

void NSBootInit()

void NSReset()

void NSShutDown()

void NSStartUp(updateRate,userUpdateRtnPtr)
Word updateRate;
Pointer userUpdateRtnPtr;

Boolean NSStatus()

unsigned int NSVersion()

**************** Print Manager Procedures ********************

void PMBootInit()

void PMReset()

void PMShutDown()

void PMStartUp(userID,directPage)
Integer userID;
Integer directPage;
```

```
Boolean PMStatus()

unsigned int PMVersion()

Boolean PrChooser()

void PrCloseDoc(printGrafPortPtr)
Pointer printGrafPortPtr;

void PrClosePage(printGrafPortPtr)
Pointer printGrafPortPtr;

void PrControl()

void PrDefault(printRecordHandle)
Handle printRecordHandle;

unsigned int PrError()

Boolean PrJobDialog(printRecordHandle)
Handle printRecordHandle;

GrafPortPtr PrOpenDoc(printRecordHandle,printerPortPtr)
Handle printRecordHandle;
Pointer printerPortPtr;

void PrOpenPage(grafPortPtr,pageFramePtr)
Pointer grafPortPtr;
Pointer pageFramePtr;

void PrPicfile(printRecordHandle,printGrafPortPtr,statusRecPtr)
Handle printRecordHandle;
Pointer printGrafPortPtr;
Pointer statusRecPtr;

void PrSetError(errorNumber)
Word errorNumber;

Boolean PrStlDialog(printRecordHandle)
Handle printRecordHandle;

Boolean PrValidate(printRecordHandle)
Handle printRecordHandle;

**************** ProDos Procedures ********************

void ALLOC_INTERRUPT(InterruptRecPtr)
Pointer interruptRecPtr;

void CHANGE_PATH(PathNameRecPtr)
Pointer PathNameRecPtr;

void CLEAR_BACKUP_BIT(PathNameRecPtr)
Pointer PathNameRecPtr;

void CLOSE(FileIORecPtr)
Pointer FileIORecPtr;

void CREATE(FileRecPtr)
Pointer FileRecPtr;

void DEALLOC_INTERRUPT(InterruptRecPtr)
Pointer InterruptRecPtr;

void DESTROY(PathNameRecPtr)
Pointer PathNameRecPtr;

void D_INFO(PBlockPtr)
Pointer PBlockPtr;

void ERASE_DISK(PBlockPtr)
Pointer PBlockPtr;

void FLUSH(FileIORecPtr)
Pointer FileIORecPtr;

void FORMAT(FormatRecPtr)
Pointer FormatRecPtr;

void GET_BOOT_VOL(PathNameRecPtr)
Pointer PathNameRecPtr;

void GET_DEV_NUM(DevNumRecPtr)
Pointer DevNumRecPtr;

void GET_DIR_ENTRY(PBlockPtr)
Pointer PBlockPtr;

void GET_EOF(EOFRecPtr)
Pointer EOFRecPtr;

void GET_FILE_INFO(FileRecPtr)
Pointer FileRecPtr;

void GET_LAST_DEV(PBlockPtr)
Pointer PBlockPtr;

void GET_LEVEL(LevelRecPtr)
Pointer LevelRecPtr;

void GET_MARK(MarkRecPtr)
Pointer MarkRecPtr;

void GET_PATH_NAME(PathNameRecPtr)
Pointer PathNameRecPtr;

void GET_PREFIX(PrefixRecPtr)
Pointer PrefixRecPtr;

void GET_VERSION(VersionRecPtr)
Pointer VersionRecPtr;

void NEWLINE(NewlineRecPtr)
Pointer NewlineRecPtr;

void OPEN(OpenRecPtr)
Pointer OpenRecPtr;

void QUIT(PathNameRecPtr)
Pointer PathNameRecPtr;

void READ(FileIORecPtr)
Pointer FileIORecPtr;

void READ_BLOCK(BlockRecPtr)
Pointer BlockRecPtr;

void SET_EOF(EOFRecPtr)
Pointer EOFRecPtr;

void SET_FILE_INFO(FileRecPtr)
Pointer FileRecPtr;
```

```
void SET_LEVEL(LevelRecPtr)
Pointer LevelRecPtr;

void SET_MARK(MarkRecPtr)
Pointer MarkRecPtr;

void SET_PREFIX(PrefixRecPtr)
Pointer PrefixRecPtr;

void VOLUME(VolumeRecPtr)
Pointer VolumeRecPt;

void WRITE(FileIORecPtr)
Pointer FileIORecPtr;

void WRITE_BLOCK(BlockRecPtr)
Pointer BlockRecPtr;

**************** QDAux Procedures ********************

void CopyPixels(srcLocPtr,destLocPtr,srcRect,destRect,xferMode,makeRgn)
Pointer srcLocPtr;
Pointer destLocPtr;
Pointer srcRect;
Pointer destRect;
Word xferMode;
Handle makeRgn;

void QDAuxBootInit()

void QDAuxReset()

void QDAuxShutDown()

void QDAuxStartUp()

Boolean QDAuxStatus()

unsigned int QDAuxVersion()

void WaitCursor()

**************** QuickDraw Procedures ********************

void AddPt(srcPtPtr,destPtPtr)
Pointer xrcPtPtr;
Pointer destPtPtr;

void CharBounds(theChar,rectPtr)
Word theChar;
Pointer rectPtr;

int CharWidth(theChar)
Word theChar;

void ClearScreen(colorWord)
Word colorWord;

void ClipRect(rectPtr)
Pointer rectPtr;

void ClosePicture()

void ClosePoly()

void ClosePort(portPtr)
Pointer portPtr;

void CloseRgn(rgnHandle)
RgnHandle rgnHandle;

void CopyRgn(srcRgnHandle,destRgnHandle)
RgnHandle srcRgnHandle;
RgnHandle destRgnHandle;

void CStringBounds(cStringPtr,rectPtr)
Pointer cStringPtr;
Pointer rectPtr;

int CStringWidth(cStringPtr)
Pointer cStringPtr;

void DiffRgn(rgn1Handle,rgn2Handle,diffRgnHandle)
RgnHandle rgn1Handle;
RgnHandle rgn2Handle;
RgnHandle diffRgnHandle;

void DisposeRgn(rgnHandle)
RgnHandle rgnHandle;

void DrawChar(theChar)
Word theChar;

void DrawCString(cStringPtr)
Pointer cStringPtr;

void DrawPicture(picHandle,destRect)
Handle picHandle;
Pointer destRect;

void DrawString(stringPtr)
Pointer stringPtr;

void DrawText(textPtr,textLength)
Pointer textPtr;
Integer textLength;

Boolean EmptyRect(rectPtr)
Pointer rectPtr;

Boolean EmptyRgn(rgnHandle)
RgnHandle rgnHandle;

Boolean EqualPt(srcPtPtr,destPtPtr)
Pointer srcPtPtr;
Pointer destPtPtr;

Boolean EqualRect(rect1Ptr,rect2Ptr)
Pointer rect1Ptr;
Pointer rect2Ptr;

Boolean EqualRgn(rgn1Handle,rgn2Handle)
RgnHandle rgn1Handle;
RgnHandle rgn2Handle;

void EraseArc(rectPtr,startAngle,arcAngle)
Pointer rectPtr;
Integer startAngle;
Integer arcAngle;

void EraseOval(rectPtr)
```

```
Pointer rectPtr;

void ErasePoly(polyHandle)
Handle polyHandle;

void EraseRect(rectPtr)
Pointer rectPtr;

void EraseRgn(rgnHandle)
RgnHandle rgnHandle;

void EraseRRect(rectPtr,ovalWidth,ovalHeight)
Pointer rectPtr;
Integer ovalWidth;
Integer ovalHeight;

void FillArc(rectPtr,startAngle,arcAngle,patternPtr)
Pointer rectPtr;
Integer startAngle;
Integer arcAngle;
Pointer patternPtr;

void FillOval(rectPtr,patternPtr)
Pointer rectPtr;
Pointer patternPtr;

void FillPoly(polyHandle,patternPtr)
Handle polyHandle;
Pointer patternPtr;

void FillRect(rectPtr,patternPtr)
Pointer rectPtr;
Pointer patternPtr;

void FillRgn(rgnHandle,patternPtr)
RgnHandle rgnHandle;
Pointer patternPtr;

void FillRRect(rectPtr,ovalWidth,ovalHeight,patternPtr)
Pointer rectPtr;
Integer ovalWidth;
Integer ovalHeight;
Pointer patternPtr;

void ForceBufDims(maxWidth,maxFontHeight,maxFBRExtent)
Integer maxWidth;
Integer maxFontHeight;
Integer maxFBRExtent;

void FrameArc(rectPtr,startAngle,arcAngle)
Pointer rectPtr;
Integer startAngle;
Integer arcAngle;

void FrameOval(rectPtr)
Pointer rectPtr;

void FramePoly(polyHandle)
Handle polyHandle;

void FrameRect(rectPtr)
Pointer rectPtr;

void FrameRgn(rgnHandle)
RgnHandle rgnHandle;

void FrameRRect(rectPtr,ovalWidth,ovalHeight)
Pointer rectPtr;
Integer ovalWidth;
Integer ovalHeight;

Ptr GetAddress(tableID)
Integer tableID;

unsigned int GetArcRot()

unsigned int GetBackColor()

void GetBackPat(patternPtr)
Pointer patternPtr;

Fixed GetCharExtra()

void GetClip(rgnHandle)
RgnHandle rgnHandle;

Handle GetClipHandle()

unsigned int GetColorEntry(tableNumber,entryNumber)
Integer tableNumber;
Integer entryNumber;

void GetColorTable(tableNumber,destTablePtr)
Integer tableNumber;
Pointer destTablePtr;

Ptr GetCursorAdr()

int GetFGSize()

FontHndl GetFont()

TextStyle GetFontFlags()

void GetFontGlobals(fGRecPtr)
Pointer fGRecPtr;

unsigned long GetFontID()

void GetFontInfo(fontInfoRecPtr)
Pointer fontInfoRecPtr;

int GetFontLore()

unsigned int GetForeColor()

ProcPtr GetGrafProcs()

SCBword GetMasterSCB()

void GetPen(pointPtr)
Pointer pointPtr;

void GetPenMask(maskPtr)
Pointer maskPtr;

unsigned int GetPenMode()

void GetPenPat(patternPtr)
Pointer patternPtr;

void GetPenSize(pointPtr)
```

```
Pointer pointPtr;

void GetPenState(penStatePtr)
Pointer penStatePtr;

long GetPicSave()

unsigned int GetPixel(h,v)
Integer h;
Integer v;

long GetPolySave()

GrafPortPtr GetPort()

void GetPortLoc(locInfoPtr)
Pointer locInfoPtr;

void GetPortRect(rectPtr)
Pointer rectPtr;

long GetRgnSave()

void GetROMFont()

SCBword GetSCB(scanLine)
Integer scanLine;

Fixed GetSpaceExtra()

SCBword GetStandardSCB()

long GetSysField()

FontHndl GetSysFont()

unsigned int GetTextFace()

unsigned int GetTextMode()

int GetTextSize()

long GetUserField()

Handle GetVisHandle()

void GetVisRgn(rgnHandle)
RgnHandle rgnHandle;

void GlobalToLocal(pointPtr)
Pointer pointPtr;

void GrafOff()

void GrafOn()

void HideCursor()

void HidePen()

void InflateTextBuffer()

void InitColorTable(tablePtr)
Pointer tablePtr;

void InitCursor()
```

```
void InitPort(portPtr)
Pointer portPtr;

void InsetRect(rectPtr,dH,dV)
Pointer rectPtr;
Integer dH;
Integer dV;

void InsetRgn(rgnHandle,dH,dV)
RgnHandle rgnHandle;
Integer dH;
Integer dV;

void InvertArc(rectPtr,startAngle,arcAngle)
Pointer rectPtr;
Integer startAngle;
Integer arcAngle;

void InvertOval(rectPtr)
Pointer rectPtr;

void InvertPoly(polyHandle)
Handle polyHandle;

void InvertRect(rectPtr)
Pointer rectPtr;

void InvertRgn(rgnHandle)
RgnHandle rgnHandle;

void InvertRRect(rectPtr,ovalWidth,ovalHeight)
Pointer rectPtr;
Integer ovalWidth;
Integer ovalHeight;

void KillPicture(pichandle)
Handle pichandle;

void KillPoly(polyHandle)
Handle polyHandle;

void Line(dH,dV)
Integer dH;
Integer dV;

void LineTo(h,v)
Integer h;
Integer v;

void LocalToGlobal(pointPtr)
Pointer pointPtr;

void MapPoly(polyHandle,srcRectPtr,destRectPtr)
Handle polyHandle;
Pointer srcRectPtr;
Pointer destRectPtr;

void MapPt(pointPtr,srcRectPtr,destRectPtr)
Pointer pointPtr;
Pointer srcRectPtr;
Pointer destRectPtr;

void MapRect(rectPtr,srcRectPtr,destRectPtr)
Pointer rectPtr;
Pointer srcRectPtr;
Pointer destRectPtr;

void MapRgn(mapRgnHandle,srcRectPtr,destRectPtr)
RgnHandle mapRgnHandle;
Pointer srcRectPtr;
Pointer destRectPtr;

void Move(dH,dV)
Integer dH;
Integer dV;

void MovePortTo(h,v)
Integer h;
Integer v;

void MoveTo(h,v)
Integer h;
Integer v;

Handle NewRgn()

void ObscureCursor()

void OffsetPoly(polyHandle,dH,dV)
Handle polyHandle;
Integer dH;
Integer dV;

void OffsetRect(rectPtr,dH,dV)
Pointer rectPtr;
Integer dH;
Integer dV;

void OffsetRgn(rgnHandle,dH,dV)
RgnHandle rgnHandle;
Integer dH;
Integer dV;

Handle OpenPicture(picFrame)
Pointer picFrame;

Handle OpenPoly()

void OpenPort(portPtr)
Pointer portPtr;

void OpenRgn()

void PaintArc(rectPtr,startAngle,arcAngle)
Pointer rectPtr;
Integer startAngle;
Integer arcAngle;

void PaintOval(rectPtr)
Pointer rectPtr;

void PaintPixels(paintParamPtr)
Pointer paintParamPtr;

void PaintPoly(polyHandle)
Handle polyHandle;

void PaintRect(rectPtr)
Pointer rectPtr;

void PaintRgn(rgnHandle)
RgnHandle rgnHandle;

void PaintRRect(rectPtr,ovalWidth,ovalHeight)
Pointer rectPtr;
Integer ovalWidth;
Integer ovalHeight;

void PenNormal()

void PicComment(kind,dataSize,dataHandle)
Integer kind;
Integer dataSize;
Handle dataHandle;

void PPToPort(srcLocPtr,srcRectPtr,destX,destY,transferMode)
Pointer srcLocPtr;
Pointer srcRectPtr;
Integer destX;
Integer destY;
Word transferMode;

void Pt2Rect(point1Ptr,point2Ptr,rectPtr)
Pointer point1Ptr;
Pointer point2Ptr;
Pointer rectPtr;

Boolean PtInRect(pointPtr,rectPtr)
Pointer pointPtr;
Pointer rectPtr;

Boolean PtInRgn(pointPtr,rgnHandle)
Pointer pointPtr;
RgnHandle rgnHandle;

void QDBootInit()

void QDReset()

void QDShutDown()

void QDStartUp(dPageAddr,masterSCB,maxWidth,userID)
Word dPageAddr;
Word masterSCB;
Integer maxWidth;
Word userID;

Boolean QDStatus()

unsigned int QDVersion()

int Random()

Boolean RectInRgn(rectPtr,rgnHandle)
Pointer rectPtr;
RgnHandle rgnHandle;

void RectRgn(rgnHandle,rectPtr)
RgnHandle rgnHandle;
Pointer rectPtr;

void RestoreBufDims(sizeInfoPtr)
Pointer sizeInfoPtr;

void SaveBufDims(sizeInfoPtr)
Pointer sizeInfoPtr;
```

```
void ScalePt(pointPtr,srcRectPtr,destRectPtr)
Pointer pointPtr;
Pointer srcRectPtr;
Pointer destRectPtr;

void ScrollRect(rectPtr,dH,dV,updateRgnHandle)
Pointer rectPtr;
Integer dH;
Integer dV;
RgnHandle updateRgnHandle;

Boolean SectRect(rect1Ptr,rect2Ptr,intersectRectPtr)
Pointer rect1Ptr;
Pointer rect2Ptr;
Pointer intersectRectPtr;

void SectRgn(rgn1Handle,rgn2Handle,destRgnHandle)
RgnHandle rgn1Handle;
RgnHandle rgn2Handle;
RgnHandle destRgnHandle;

void SetAllSCBs(newSCB)
SCBword newSCB;

void SetArcRot()

void SetBackColor(backColor)
Word backColor;

void SetBackPat(patternPtr)
Pointer patternPtr;

void SetBufDims(maxWidth,maxFontHeight,maxFBRExtent)
Integer maxWidth;
Integer maxFontHeight;
Integer maxFBRExtent;

void SetCharExtra(charExtra)
LongInt charExtra;

void SetClip(rgnHandle)
RgnHandle rgnHandle;

void SetClipHandle(rgnHandle)
RgnHandle rgnHandle;

void SetColorEntry(tableNumber,entryNumber,newColor)
Integer tableNumber;
Integer entryNumber;
Integer newColor;

void SetColorTable(tableNumber,srcTablePtr)
Integer tableNumber;
Pointer srcTablePtr;

void SetCursor(cursorPtr)
Pointer cursorPtr;

void SetEmptyRgn(rgnHandle)
RgnHandle rgnHandle;

void SetFont(newFontHandle)
FontHandle newFontHandle;

void SetFontFlags(fontFlags)
Word fontFlags;
```

```
void SetFontID(fontIDHandle)
Handle fontIDHandle;

void SetForeColor(foreColor)
Word foreColor;

void SetGrafProcs(grafProcsPtr)
GrafProcsPtr grafProcsPtr;

void SetIntUse(useInt)
Word useInt;

void SetMasterSCB(masterSCB)
SCBword
 masterSCB;

void SetOrigin(h,v)
Integer h;
Integer v;

void SetPenMask(maskPtr)
Pointer maskPtr;

void SetPenMode(penMode)
Word penMode;

void SetPenPat(patternPtr)
Pointer patternPtr;

void SetPenSize(penWidth,penHeight)
Integer penWidth;
Integer penHeight;

void SetPenState(penStatePtr)
Pointer penStatePtr;

void SetPicSave(picSaveValue)
LongInt picSaveValue;

void SetPolySave(polySaveValue)
LongInt polySaveValue;

void SetPort(portPtr)
Pointer portPtr;

void SetPortLoc(locInfoPtr)
Pointer locInfoPtr;

void SetPortRect(rectPtr)
Pointer rectPtr;

void SetPortSize(portWidth,portHeight)
Integer portWidth;
Integer portHeight;

void SetPt(srcPtPtr,h,v)
Pointer srcPtPtr;
Integer h;
Integer v;

void SetRandSeed(randomSeed)
LongInt randomSeed;

void SetRect(rectPtr,left,top,right,bottom)
Pointer rectPtr;
```

```
Integer left;
Integer top;
Integer right;
Integer bottom;

void SetRectRgn(rgnHandle,left,top,right,bottom)
RgnHandle rgnHandle;
Integer left;
Integer top;
Integer right;
Integer bottom;

void SetRgnSave(rgnSaveValue)
LongInt rgnSaveValue;

void SetSCB(scanLine,newSCB)
Integer scanLine;
Integer newSCB;

void SetSolidBackPat(colorNum)
Integer colorNum;

void SetSolidPenPat(colorNum)
Integer colorNum;

void SetSpaceExtra(spaceExtra)
Fixed spaceExtra;

void SetStdProcs(stdProcRecPtr)
Pointer stdProcRecPtr;

void SetSysField(sysFieldValue)
LongInt sysFieldValue;

void SetSysFont(fontHandle)
FontHandle fontHandle;

void SetTextFace(textFace)
Word textFace;

void SetTextMode(textMode)
Word textMode;

void SetTextSize()

void SetUserField(userFieldValue)
LongInt userFieldValue;

void SetVisHandle(rgnHandle)
RgnHandle rgnHandle;

void SetVisRgn(rgnHandle)
RgnHandle rgnHandle;

void ShowCursor()

void ShowPen()

void SolidPattern(patternPtr,colorNum)
Pointer patternPtr;
Integer colorNum;

void StringBounds(stringPtr,rectPtr)
Pointer stringPtr;
Pointer rectPtr;

int StringWidth(stringPtr)
Pointer stringPtr;

void SubPt(srcPtPtr,destPtPtr)
Pointer srcPtPtr;
Pointer destPtPtr;

void TextBounds(textPtr,textLength,rectPtr)
Pointer textPtr;
Integer textLength;
Pointer rectPtr;

int TextWidth(textPtr,textLength)
Pointer textPtr;
Integer textLength;

void UnionRect(rect1Ptr,rect2Ptr,unionRectPtr)
Pointer rect1Ptr;
Pointer rect2Ptr;
Pointer unionRectPtr;

void UnionRgn(rgn1Handle,rgn2Handle,unionRgnHandle)
RgnHandle rgn1Handle;
RgnHandle rgn2Handle;
RgnHandle unionRgnHandle;

void XorRgn(rgn1Handle,rgn2Handle,xorRgnHandle)
RgnHandle rgn1Handle;
RgnHandle rgn2Handle;
RgnHandle xorRgnHandle;

**************** Scheduler Procedures ********************

Boolean SchAddTask(taskPtr)
Pointer taskPtr;

void SchBootInit()

void SchFlush()

void SchReset()

void SchShutDown()

void SchStartUp()

Boolean SchStatus()

unsigned int SchVersion()

**************** Scrap Manager Procedures ********************

void GetScrap(destHandle,scrapType)
Handle destHandle;
Word scrapType;

int GetScrapCount()

Handle GetScrapHandle(scrapType)
Word scrapType;

Ptr GetScrapPath()

long GetScrapSize(scrapType)
Word scrapType;
```

```
unsigned int GetScrapState()

void LoadScrap()

void PutScrap(numBytes,scrapType,srcPtr)
LongInt numBytes;
Word scrapType;
Pointer srcPtr;

void ScrapBootInit()

void ScrapReset()

void ScrapShutDown()

void ScrapStartUp()

Boolean ScrapStatus()

unsigned int ScrapVersion()

void SetScrapPath(pathPtr)
Pointer pathPtr;

void UnloadScrap()

void ZeroScrap()

**************** Shell Procedures ********************

void DIRECTION(device,direct)
integer device;
pointer direct;

void ERROR()

void EXECUTE(flag,comm)
integer flag;
pointer comm;

int GET_LANG()

void GET_LINE_INFO(PblockPtr)
GetLinfoPB PblockPtr;

void GET_VAR(varname,value)
pointer varname;
pointer value;

void INIT_WILDCARD(file,flags)
pointer file;
integer flags;

Char NEXT_WILDCARD(nextfile)
pointer nextfile;

void READ_INDEXED(varname,value,index)
pointer varname;
pointer value;
integer index;

void REDIRECT(device,app,file)
integer device;
integer app;
pointer file;

void SET_LANG(language)
integer language;

void SET_LINE_INFO(PblockPtr)
GetLinfoPB PblockPtr;

void SET_VAR(varname,value)
pointer varname;
pointer value;

int STOP()

int VERSION()

**************** Sound Manager Procedures **************

unsigned int FFGeneratorStatus(genNumber)
Word genNumber;

unsigned int FFSoundDoneStatus(genNumber)
Word genNumber;

unsigned int FFSoundStatus()

void FFStartSound(genNumFFSynth,pBlockPtr)
Word genNumFFSynth;
Pointer pBlockPtr;

void FFStopSound(genMask)
Word genMask;

unsigned int GetSoundVolume(genNumber)
Word genNumber;

Ptr GetTableAddress()

void ReadRamBlock(destPtr,dOCStart,byteCount)
Pointer destPtr;
Word dOCStart;
Word byteCount;

void SetSoundMIRQV(sMasterIRQ)
LongInt sMasterIRQ;

void SetSoundVolume(volume,genNumber)
Word volume;
Word genNumber;

Ptr SetUserSoundIRQV(userIRQVector)
LongInt userIRQVector;

void SoundBootInit()

void SoundReset()

void SoundShutDown()

void SoundStartUp(wAPTPtr)
Word wAPTPtr;

Boolean SoundToolStatus()

unsigned int SoundVersion()

void WriteRamBlock(srcPtr,dOCStart,byteCount)
Pointer srcPtr;
```

```
Word dOCStart;
Word byteCount;

**************** StdFile Procedures ********************
void SFAllCaps(allCapsFlag)
Boolean allCapsFlag;

void SFBootInit()

void SFGetFile(whereX,whereY,promptPtr,filterProcPtr,typeListPtr,replyPtr)
Integer whereX;
Integer whereY;
Pointer promptPtr;
Pointer filterProcPtr;
Pointer typeListPtr;
Pointer replyPtr;

void SFPGetFile(whereX,whereY,promptPtr,filterProcPtr,typeListPtr,dialogPtr,dialogHookPtr,replyPtr)
Integer whereX;
Integer whereY;
Pointer promptPtr;
Pointer filterProcPtr;
Pointer typeListPtr;
Pointer dialogPtr;
Pointer dialogHookPtr;
Pointer replyPtr;

void SFPPutFile(whereX,whereY,promptPtr,origNamePtr,maxLen,dialogPtr,dialogHookPtr,replyPtr)
Integer whereX;
Integer whereY;
Pointer promptPtr;
Pointer origNamePtr;
Integer maxLen;
Pointer dialogPtr;
Pointer dialogHookPtr;
Pointer replyPtr;

void SFPutFile(whereX,whereY,promptPtr,origNamePtr,maxLen,replyPtr)
Integer whereX;
Integer whereY;
Pointer promptPtr;
Pointer origNamePtr;
Integer maxLen;
Pointer replyPtr;

void SFReset()

void SFShutDown()

void SFStartUp(userID,dPageAddr)
Word userID;
Word dPageAddr;

Boolean SFStatus()

unsigned int SFVersion()

**************** TextTool Procedures ********************

void CtlTextDev(controlDevice,controlCode)
Word controlDevice;
Word controlCode;

void ErrWriteBlock(textPtr,offset,count)
Pointer textPtr;
Word offset;
```

```
Word count;

void ErrWriteChar(theChar)
Word theChar;

void ErrWriteCString(cStrPtr)
Pointer cStrPtr;

void ErrWriteLine(strPtr)
Pointer strPtr;

void ErrWriteString(strPtr)
Pointer strPtr;

void GetErrGlobals(resultRecPtr)
Pointer resultRecPtr;

void GetErrorDevice(resultRecPtr)
Pointer resultRecPtr;

void GetInGlobals(resultRecPtr)
Pointer resultRecPtr;

void GetInputDevice(resultRecPtr)
Pointer resultRecPtr;

void GetOutGlobals(resultRecPtr)
Pointer resultRecPtr;

void GetOutputDevice(resultRecPtr)
Pointer resultRecPtr;

void InitTextDev(deviceNum)
Word deviceNum;

unsigned int ReadChar(echoFlag)
Word echoFlag;

unsigned int ReadLine(bufferPtr,maxCount,eOLChar,echoFlag)
Pointnr bufferPtr;
Word maxCount;
Word eOLChar;
Word echoFlag;

void SetErrGlobals(aNDMask,oRMask)
Word aNDMask;
Word oRMask;

void SetErrorDevice(deviceType,ptrOrSlot)
Word deviceType;
Longint ptrOrSlot;

void SetInGlobals(aNDMask,oRMask)
Word aNDMask;
Word oRMask;

void SetInputDevice(deviceType,ptrOrSlot)
Word deviceType;
Longint ptrOrSlot;

void SetOutGlobals(aNDMask,oRMask)
Word aNDMask;
Word oRMask;

void SetOutputDevice(deviceType,ptrOrSlot)
Word deviceType;
```

```
void ShowHide(showFlag,theWindowPtr)
Boolean showFlag;
Pointer theWindowPtr;

void ShowWindow(theWindowPtr)
Pointer theWindowPtr;

void SizeWindow(newWidth,newHeight,theWindowPtr)
Word newWidth;
Word newHeight;
Pointer theWindowPtr;

void StartDrawing(theWindowPtr)
Pointer theWindowPtr;

void StartInfoDrawing(infoRectPtr,theWindowPtr)
Pointer infoRectPtr;
Pointer theWindowPtr;

unsigned int TaskMaster(eventMask,taskRecPtr)
Word eventMask;
Pointer taskRecPtr;

Boolean TrackGoAway(startX,startY,theWindowPtr)
Integer startX;
Integer startY;
Pointer theWindowPtr;

Boolean TrackZoom(startX,startY,theWindowPtr)
Integer startX;
Integer startY;
Pointer theWindowPtr;

void ValidRect(goodRectPtr)
Pointer goodRectPtr;

void ValidRgn(goodRgnHandle)
Handle goodRgnHandle;

void WindBootInit()

unsigned long WindDragRect()

void WindNewRes()

unsigned int WindowGlobal()

void WindReset()

void WindShutDown()

void WindStartUp(userID)
Word userID;

Boolean WindStatus()

unsigned int WindVersion()

void ZoomWindow(theWindowPtr)
Pointer theWindowPtr;
```