TO: Software Developers
FROM: CPW Team
SUBJECT: CPW C Release V1.0 D4
cc: D. Eyes
N. Johnson
J. Merritt
R. Metzker
S. McDonald

CONTENTS OF THE RELEASE DISK

- **PRODOS** SYS — an operating system/application loader
- **SYSTEM** DIR — directory containing operating system, loader, tools, etc.
- **CORTLINK** S16 — A **temporary** linker, which should be used when developing C applications. It will be replaced by the CPW linker. It should be loaded from the loader demonstration (*demo.sys16*) program. It will prompt you for the names of the files to link. The file *start.o* , listed below, should always be included as the first file in the list. The syntax for the CORTLINK is as following:

  start.o {<list of files to be linked>} -o {<result of the link process>}
- **ERRORS** TXT — a C compiler internal file
- **HELLO** S16
- **LOADSTUFF** DIR — directory containing loader files
  - **LOAD** BIN
  - **P16.1** BIN
  - **P16.0** BIN
  - **LOADER1** BIN
  - **LOADER2** BIN
- **INCLUDE** DIR — directory containing standard Cortland include files
  - **MEM.H** TXT
  - **OS.H** TXT
  - **FCNTL.H** TXT
  - **CTYPE.H** TXT
  - **STRING.H** TXT
  - **TOOLDEFS.H** TXT
  - **SANE.H** TXT
  - **STDIO.H** TXT
  - **MATH.H** TXT
  - **TOOLS.H** TXT — an include file containing definitions for all of the interfaces to Cortland system tools
- **MMCC** S16 — MegaMax C compiler. It should be loaded from the loader demonstration (*demo.sys16*) program. It will prompt you for the name of the file to compile. The **name** of your file **should have .c suffix**. This is a temporary requirement and will change as soon as we integrate the compiler with the rest of the development system. The result of your compilation will be placed in the file name created by replacing .c in your source file name with .o.
- **START.O** OBJ — C compiler run time initialization segment. Its purpose is to initialize the memory manager and set up the stack space.
- **SYSLIB** OBJ — an object module treated as a library by CORTLINK; contains the functions usually provided by the standard C library and UNIX interface library. It will be replaced by the Apple developed library.
- **DEMO.SYS16** S16 — loader demonstration program
- **MYSAMPLE** DIR
  - **MYSAMPLE.C** — source program for the sample
  - **MENUS.A** — assembly language module containing data structure definitions

- **MENUS.O** object module created by assembling MENUS.A; needs to be linked together with MYSAMPLE.O and START.O (see above) to create MYSAMPLE.
- **MYSAMPLE** an executable version
- **PIC1, PIC2, PIC3, PIC4** picture files loaded by MYSAMPLE

IMPLEMENTED FEATURES OF THE C COMPILER

• All of the features described in the C compiler and language ERS document, unless listed in the NOT YET IMPLEMENTED section, are implemented. In addition, the following features are implemented:

• The source level segmentation directive is called **"overlay"**.

USAGE: overlay "name"

The directive must appear between the functions. "name" is the name of the load segment that all code, following the directive until the end of the file or the next such directive, will be placed in.

• The format for an **inline** declaration is:

[extern] pascal [result-type] func-name () inline(m,n) ;

where m is the trap number, n is the entry point. The inline declaration is used internally at Apple to call all of the Cortland tools. I do not think you will have to use it.

• A **pascal** function specification has one of the following formats:

[extern] pascal [result-type] func-name () ;

or

pascal [result-type] func-name (<formal-parameters>) {<statement-list> }

This specification should be used to call functions and procedures that conform to the specifications listed in the ERS document.

• **In-line assembly** has one of the following formats:

asm { <assembly-statements> }

or

asm (<external-name>) { <assembly-statements> }

The first format can appear anywhere a statement is legal. The second can only appear where a function definition is legal. In this case the external name is defined as the entry point for a segment containing the assembly. The assembler mnemonic names are exactly the same as names defined in the 65816 specification (ie. exactly the same as in the CPW assembler). You can use any valid C expression as your assembler expression. However, you cannot allocate any writeable data within your code, therefore writing self modifying code is not possible. Note, that there is no specific macro facility, other than the one associated with the C compiler. Three additional assembler instructions are allowed by the **in-line assembly**. Their mnemonic names are DCW, DCB and DCL. These instructions insert a word, byte or long constant(s) into your code. They have the following format:

DC*x* {<number list separated by comma>}

where *x* is W, B, or L.

NOT YET IMPLEMENTED

- ENUM clause
- floating point code generation and SANE support
- bit fields in the structures
- full standard C library support (for the implemented subset see below)
- full Cortland tool interface library (for the implemented subset see below)

PERFORMANCE AND LIMITATIONS

• The initial performance of the code generated by the compiler seems to be quite good, taking into consideration fact that it is just a developmental release without many possible code generation improvements. The performance figure for this compiler, as measured in dhrystones per second, is 185. This places our compiler at about 66% of performance of any compiler running on IBM PC. You should expect that we will improve this performance. Note, that due to the specific 65816 architecture **operations on unsigned int** variables **are much faster** than on signed quantities.

• The arrays and global variables in release D4 are stored in the same area, therefore you cannot have more than 64K allocated to them. Future releases of this compiler will remove the limitation on arrays, but the limitation on size of global scalar variables will remain.
• Due to the current performance of the loader, it takes about 1 minute 40 seconds to load the compiler. The group responsible for the loader is aware of the situation and they working on improving this performance Note, that the maximum limit for the physical transfer of data from 3.5" disk drive is about 7.6 K per second, therefore the minimum time to load C compiler (173 K currently) will not go below 1.5 * 23 = 35 seconds, where 1.5 is how many times the loader reads file to be loaded.

SUGGESTION ON DEVELOPING USING C COMPILER

A useful Cortland configuration, using RAM disk to improve the performance, is described in the APPENDIX LIFE IN THE FAST LANE.

CONTENTS OF THE C LIBRARY (SYSLIB)

The release disk contains an object file named SYSLIB. This file is the system library. It is scanned by CORTLINK to retrieve any functions you would expect to be implemented by the C compiler. The following is the list of functions that are contained there:

_exit	calloc
close	creat
exit	fclose
fflush	fgetc
fgets	fopen
fprintf	fputc
fputs	fread
free	freopen
fscanf	fseek
ftell	fwrite
gets	getw
lseek	malloc
open	printf
puts	putw
read	rewind
scanf	setbuf
sprintf	sscanf
strcat	strcmp
strcpy	strlen
strncat	strncmp
strncpy	ungetc
unlink	write

The interface to this library is exactly the same as specified by the documentation accompanying the Mac version of the MegaMax C compiler with two exceptions - **fopen** and **creat**. These functions can have an optional third argument (a byte) indicating ProDOS file type (ProDOS file types are defined in OS.H include file). This optional argument is indicated by:

- 'F' as a first character in the type string (ie. second argument) in fopen
- O_MODE bit is set in the oflag (ie. second argument) in creat (O_MODE is defined in FCNTL.H include file).

NOTE: the current **implementation** of the above interfaces is **temporary** in its nature. They are provided to us courtesy of the compiler developer (MegaMax) and are shipped with the compiler until Apple developed libraries are completed. The above libraries are used by the compiler.

INTERFACE LIBRARIES

• All of the interfaces to the Cortland system tools that do not require any 'glue' code (ie. code necessary to make sure that the interface is compatible with the C language constructs) are available.
• The Cortland tools by convention always return an error result. This result is stored by the C compiler in a global int variable named _ermo. This variable can be checked after each tool call - zero always indicates a successful completion. For now, it is your responsibility to define this variable in your program if you call any of the system tools.
• APPENDIX GLUE contains the list of all calls available in the current interface specification.
• The system interfaces are fully supported, although they were not tested as thoroughly as we would like.

SAMPLES

See the disk for MYSAMPLE and APPENDIX MYSAMPLE.

KNOWN PROBLEMS

• The following program **won't** compile on the Cortland C compiler:

```
main()
{
 int i,j;
 i = ("Hello",j);
}
```

The type of this comma operation in release D4 is that of "Hello", not of j. The remedy for this problem is to use type casting to force expressions to be of the same type.
• There is a problem with passing structures to procedures. For example, the following program will fail:

```
typedef struct pair {short one, two} PAIR;
void check(x) PAIR x;
{
        if (x.one != 1) puts("error1.");
        if (x.two != 2) puts("error2.");
}

main()
{       PAIR x;
        x.one = 1; x.two = 2;
        check(x);
}
```

NEW PROBLEMS

If you find new problems with the compiler or the system released here, feel free to contact Apple Developer Technical Support at (408) 973-4357, MS 27T (if you are a seeded developer) or Ron Metzker at MS 27G (if you are an Apple internal user).

ACKNOWLEDGMENTS

I want to thank David Eyes, Neal Johnson, Stuart McDonald, and (of course) MegaMax whose work provided material for this release.

Sincerely yours,

Marek Z. Jeziorek

July 30, 1986

About the PICS.C sample program

Pics.c is a version of the pics program written in assembly language by Steve Glass and found on the ProDOS/16 distribution disks. It was converted to C with a basically identical program structure, allowing for easy comparison.

The first thing you need to know about the pics.c program is the use of the *tools.h* include file. This contains most of the headers required to access the Cortland tools. The header definitions illustrate two CPW C extensions to the standard C syntax. The first of these extensions allows specification of a "Pascal" type function, which means that the parameters get passed on the stack in the order that the tools expect them in, and that space for a return value is reserved on the stack – and specification of "inline" tool function call generation. Note that because the "Pascal" style function call stores room for the result on the stack, functions that return no values (i. e., procedures in pascal nomenclature) must be declared as returning the C type "void". The header definition allows specification of the tool number and the dispatcher entry point (normally the tool locater).

The "include" referencing the tools.h file uses an absolute pathname because of the funky way in which angle-bracket includes work right now. Although these header files seem to be in fairly reasonable shape, they are of course subject to change. Ultimately, the header files will include definition of all the relevant data structures needed; the ones found in the sample program are strictly provisional and probably bear no relation, as far as the structure and field names are concerned, to the final structure definitions.

The second important thing to be aware of is that the menu definitions are included in a separate assembly language source file, menus.a (the object is in menus.o). This is because the menu definitions are very long strings, which are fairly painful to initialize in C. The good news is that in future versions of the window manager, menu definitions will be allowed which optionally allow use of the null character instead of the return character to separate individual menu definitions; this will allow initialization of the menu definitions as an array of string pointers.

The last tricky thing to be aware of is the fact that a pointer to a C function is passed (in the Winblock structure) to the window manager when the NewWindow function is called. This routine updates the content region whenever the window needs updating. It is called directly from

within the window manager, rather than from your own code. The environment that exists when the function is called is a little different, most notably, the data bank register value is probably pointing somewhere else than where C expects it to be. The C code generation scheme expects the data bank register to point to the global data area so that sixteen-bit addressing can be used to access global scalars. In a nutshell, this means that you can't use global variables in functions called directly by a tool. The two ways around this are either to reference globals by first taking their address and storing it (or passing it, as in the example) in a pointer, or to use global arrays or structures rather than global scalars. Both usages will cause long addresses to be used.

Otherwise, the program is fairly straightforward and does things pretty much the same way they were done in the original assembly language example.

To compile and link the program, in a two-drive configuration, with /CPW in one drive and /RAM5 in the other, do the following:

First, make sure that the contents of /RAM5/LOADSTUFF have been copied onto the /CPW disk (they should wind up in the top level directory, that is, /CPW). No room? Make some! Boot into /CPW. The current prefix should be /CPW.

Type **load /ram5/mmcc** on the CPW command line.

Please wait.

Type Y to the question "do you want to run the program just loaded?"

When prompted for the name of the file to compile, enter **/ram5/pics/pics.c.**
The compiler will do its thing, and create /ram5/pics/pics.o.

When asked if you want to shut down the previous application, re-boot the machine (/CPW should be in the boot drive).

Now type **load /ram5/cortlink.**

Please wait, but not as long.

Type Y, etc.

When prompted for files to link, type:
/ram5/start.o /ram5/pics/pics.o /ram5/menus.o -o /ram5/pic

This creates the load file /ram5/pic. start.o is the initialization code; this needs to be the first linked file in every C application. menus.o is the menu definition (remember?).

When the compiler is done, reboot /ram5 -- this will load ProDOS/16, and you can launch your program from the ProDOS/16 loader program. Enjoy. See? It can be done. Things will be getting better soon.

APPENDIX

L I F E I N T H E F A S T L A N E :
U S I N G T H E C C O M P I L E R F R O M A R A M D I S K

Dependencies:
    Hardware: extra 1 Mbyte RAM card;
    Software: CPW A10, C Compiler D4, Cortland System Loader D8.

Background:
    Currently, there are two hard-wired files: the C compiler, MMCC, loads angled bracket include files from /ORCA/INCLUDE; the C linker, CORTLINK, links with the library /RAM5/SYSLIB.

Initialize a 1 Mbyte RAM card, other than the Cortland Memory card, and name it /CPW . It must be initialized to be bootable.

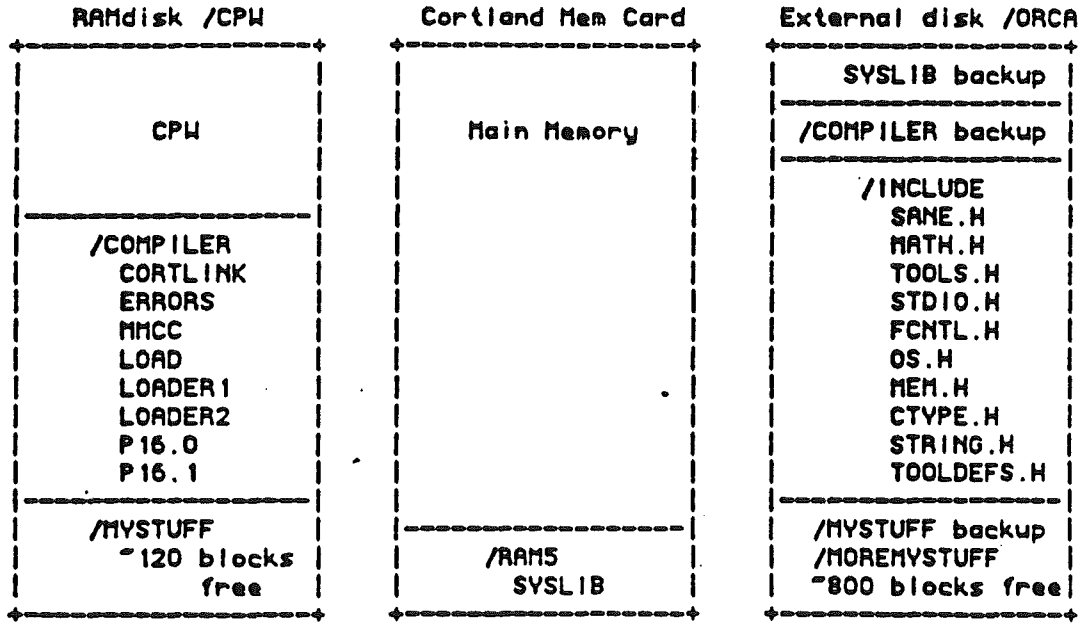
Copy all the files and directories from the CPW A10 disk onto the RAM card.

Create a directory of your own choosing on the RAM card to hold the C compiler; copy the files CORTLINK, ERRORS, MMCC, START.O, /LOADSTUFF/LOAD, /LOADSTUFF/LOADER1, /LOADSTUFF/LOADER2, /LOADSTUFF/P16.O, and /LOADSTUFF/P16.1 there from the C disk.

Initialize an external disk to /ORCA; copy the file SYSLIB and the INCLUDE directory, along with its contents, there from the C disk.

Enable the Cortland Memory Card's /RAM5 by setting the Maximum RAM Disk Size to 800K from the Control Panel; copy /ORCA/SYSLIB to /RAM5. See note.

Set the Startup Slot to the location of your RAMdisk /CPW using the Control Panel. Re-boot. Your configuration will look something like this.



A typical compile, link, run, is

```
    *prefix /cpw/compiler
    *load mmcc                                  ...problems? reboot before load
    File to compile:
    /mystuff/hello.c
    *load cortlink                              ...problems? reboot before load
    Enter object files:
    start.o /cpw/mystuff/hello.o -o /cpw/mystuff/hello
    *load /cpw/mystuff/hello                    ...problems? reboot before load
```

Note: /RAM5/SYSLIB occasionally disappears; when it does, copy it back.

APPENDIX GLUE

```
#define dispatcher 0xe10000

typedef int word;
typedef int boolean;
typedef char *ptr;
typedef ptr *handle;

extern pascal void  CrtlBootInit() inline(0x0110, dispatcher);     /* Control Manager */
extern pascal void  CtrlStartUp() inline(0x0210, dispatcher);     /* Control Manager */
extern pascal void  CtrlShutDown() inline(0x0310, dispatcher);     /* Control Manager */
extern pascal word  CtrlVersion() inline(0x0410, dispatcher);     /* Control Manager */
extern pascal void  CtrlReset() inline(0x0510, dispatcher);     /* Control Manager */
extern pascal void  CtrlStatus() inline(0x0610, dispatcher);     /* Control Manager */
extern pascal handle  NewControl() inline(0x0910, dispatcher);     /* Control Manager */
extern pascal void  DisposeControl() inline(0x0A10, dispatcher);     /* Control Manager */
extern pascal void  KillControl() inline(0x0B10, dispatcher);     /* Control Manager */
extern pascal void  SetCTitle() inline(0x0C10, dispatcher);     /* Control Manager */
extern pascal ptr  GetCTitle() inline(0x0D10, dispatcher);     /* Control Manager */
extern pascal void  HideControl() inline(0x0E10, dispatcher);     /* Control Manager */
extern pascal void  ShowControl() inline(0x0F10, dispatcher);     /* Control Manager */
extern pascal void  DrawControls() inline(0x1010, dispatcher);     /* Control Manager */
extern pascal void  HiliteControl() inline(0x1110, dispatcher);     /* Control Manager */
extern pascal void  CtrlNewRes() inline(0x1210, dispatcher);     /* Control Manager */
extern pascal word  FindControl() inline(0x1310, dispatcher);     /* Control Manager */
extern pascal word  TestControl() inline(0x1410, dispatcher);     /* Control Manager */
extern pascal word  TrackControl() inline(0x1510, dispatcher);     /* Control Manager */
extern pascal void  MoveControl() inline(0x1610, dispatcher);     /* Control Manager */
extern pascal void  DragControl() inline(0x1710, dispatcher);     /* Control Manager */
extern pascal void  SizeControl() inline(0x1810, dispatcher);     /* Control Manager */
extern pascal void  SetCtlValue() inline(0x1910, dispatcher);     /* Control Manager */
extern pascal word  GetCtlValue() inline(0x1A10, dispatcher);     /* Control Manager */
extern pascal void  SetCtlParams() inline(0x1B10, dispatcher);     /* Control Manager */
extern pascal long  GetCtlParams() inline(0x1C10, dispatcher);     /* Control Manager */
extern pascal long  DragRect() inline(0x1D10, dispatcher);     /* Control Manager */
extern pascal word  GrowSize() inline(0x1E10, dispatcher);     /* Control Manager */
extern pascal word  GetCtrlZpage() inline(0x1F10, dispatcher);     /* Control Manager */
extern pascal void  EMBootInit() inline(0x0106, dispatcher);     /* Event Manager */
extern pascal void  EMStartUp() inline(0x0206, dispatcher);     /* Event Manager */
extern pascal void  EMShutDown() inline(0x0306, dispatcher);     /* Event Manager */
extern pascal word  EMVersion() inline(0x0406, dispatcher);     /* Event Manager */
extern pascal void  EMReset() inline(0x0506, dispatcher);     /* Event Manager */
extern pascal boolean  EMActive() inline(0x0606, dispatcher);     /* Event Manager */
extern pascal boolean  GetNextEvent() inline(0x0A06, dispatcher);     /* Event Manager */
extern pascal boolean  EventAvail() inline(0x0B06, dispatcher);     /* Event Manager */
extern pascal void  GetMouse() inline(0x0C06, dispatcher);     /* Event Manager */
extern pascal boolean  Button() inline(0x0D06, dispatcher);     /* Event Manager */
extern pascal boolean  StillDown() inline(0x0E06, dispatcher);     /* Event Manager */
extern pascal boolean  WaitMouseUp() inline(0x0F06, dispatcher);     /* Event Manager */
extern pascal long  TickCount() inline(0x1006, dispatcher);     /* Event Manager */
extern pascal long  GetDblTime() inline(0x1106, dispatcher);     /* Event Manager */
extern pascal long  GetCaretTime() inline(0x1206, dispatcher);     /* Event Manager */
extern pascal word  PostEvent() inline(0x1406, dispatcher);     /* Event Manager */
extern pascal word  FlushEvents() inline(0x1506, dispatcher);     /* Event Manager */
extern pascal boolean  GetOSEvent() inline(0x1606, dispatcher);     /* Event Manager */
extern pascal boolean  OSEventAvail() inline(0x1706, dispatcher);     /* Event Manager */
```

```
extern pascal word  GetUserID() inline(0x0E11, dispatcher);      /* Loader */
extern pascal void  GetLoadSegInfo() inline(0x0F11, dispatcher);      /* Loader */
extern pascal void  LockSeg() inline(0x1011, dispatcher);      /* Loader */
extern pascal void  UnlockSeg() inline(0x1111, dispatcher);      /* Loader */
extern pascal word  UserShutDown() inline(0x1211, dispatcher);      /* Loader */
extern pascal void  MMBootInit() inline(0x0102, dispatcher);      /* Memory Manager */
extern pascal word  MMStartUp() inline(0x0202, dispatcher);      /* Memory Manager */
extern pascal void  MMShutDown() inline(0x0302, dispatcher);      /* Memory Manager */
extern pascal word  MMGetVersion() inline(0x0402, dispatcher);      /* Memory Manager */
extern pascal void  MMReset() inline(0x0502, dispatcher);      /* Memory Manager */
extern pascal boolean  MMStatus() inline(0x0602, dispatcher);      /* Memory Manager */
extern pascal handle  NewHandle() inline(0x0902, dispatcher);      /* Memory Manager */
extern pascal void  ReallocHandle() inline(0x0A02, dispatcher);      /* Memory Manager */
extern pascal void  DisposHandle() inline(0x1002, dispatcher);      /* Memory Manager */
extern pascal void  DisposAll() inline(0x1102, dispatcher);      /* Memory Manager */
extern pascal void  PurgeHandle() inline(0x1202, dispatcher);      /* Memory Manager */
extern pascal void  PurgeAll() inline(0x1302, dispatcher);      /* Memory Manager */
extern pascal long  GetHandleSize() inline(0x1802, dispatcher);      /* Memory Manager */
extern pascal void  SetHandleSize() inline(0x1902, dispatcher);      /* Memory Manager */
extern pascal handle  Findhandle() inline(0x1A02, dispatcher);      /* Memory Manager */
extern pascal long  FreeMem() inline(0x1B02, dispatcher);      /* Memory Manager */
extern pascal long  MaxBlock() inline(0x1C02, dispatcher);      /* Memory Manager */
extern pascal long  TotalMem() inline(0x1D02, dispatcher);      /* Memory Manager */
extern pascal void  HLock() inline(0x2002, dispatcher);      /* Memory Manager */
extern pascal void  HLockAll() inline(0x2102, dispatcher);      /* Memory Manager */
extern pascal void  HUnlock() inline(0x2202, dispatcher);      /* Memory Manager */
extern pascal void  HUnlockAll() inline(0x2302, dispatcher);      /* Memory Manager */
extern pascal void  SetPurge() inline(0x2402, dispatcher);      /* Memory Manager */
extern pascal void  SetPurgeAll() inline(0x2502, dispatcher);      /* Memory Manager */
extern pascal void  BlockMove() inline(0x2B02, dispatcher);      /* Memory Manager */
extern pascal void  MenuBootInit() inline(0x010F, dispatcher);      /* Menu Manager */
extern pascal void  MenuStartUp() inline(0x020F, dispatcher);      /* Menu Manager */
extern pascal void  MenuShutDown() inline(0x030F, dispatcher);      /* Menu Manager */
extern pascal word  MenuVersion() inline(0x040F, dispatcher);      /* Menu Manager */
extern pascal void  MenuReset() inline(0x050F, dispatcher);      /* Menu Manager */
extern pascal word  MenuStatus() inline(0x060F, dispatcher);      /* Menu Manager */
extern pascal void  MenuKey() inline(0x090F, dispatcher);      /* Menu Manager */
extern pascal handle  GetMenuBar() inline(0x0A0F, dispatcher);      /* Menu Manager */
extern pascal void  MenuRefresh() inline(0x0B0F, dispatcher);      /* Menu Manager */
extern pascal void  FlashMenuBar() inline(0x0C0F, dispatcher);      /* Menu Manager */
extern pascal void  InsertMenu() inline(0x0D0F, dispatcher);      /* Menu Manager */
extern pascal void  DeleteMenu() inline(0x0E0F, dispatcher);      /* Menu Manager */
extern pascal void  InsertItem() inline(0x0F0F, dispatcher);      /* Menu Manager */
extern pascal void  DeleteItem() inline(0x100F, dispatcher);      /* Menu Manager */
extern pascal handle  GetSysBar() inline(0x110F, dispatcher);      /* Menu Manager */
extern pascal void  SetSysBar() inline(0x120F, dispatcher);      /* Menu Manager */
extern pascal word  FixMenuBar() inline(0x130F, dispatcher);      /* Menu Manager */
extern pascal word  CountMItems() inline(0x140F, dispatcher);      /* Menu Manager */
extern pascal void  NewMenuBar() inline(0x150F, dispatcher);      /* Menu Manager */
extern pascal void  SetBarColors() inline(0x170F, dispatcher);      /* Menu Manager */
extern pascal long  GetBarColors() inline(0x180F, dispatcher);      /* Menu Manager */
extern pascal void  SetTitleStart() inline(0x190F, dispatcher);      /* Menu Manager */
extern pascal word  GetTitleStart() inline(0x1A0F, dispatcher);      /* Menu Manager */
extern pascal void  CalcMenuSize() inline(0x1C0F, dispatcher);      /* Menu Manager */
extern pascal void  SetTitleWidth() inline(0x1D0F, dispatcher);      /* Menu Manager */
extern pascal word  GetTitleWidth() inline(0x1E0F, dispatcher);      /* Menu Manager */
extern pascal void  SetMenuFlag() inline(0x1F0F, dispatcher);      /* Menu Manager */
```

```
extern pascal void  DelHeartBeat() inline(0x1303, dispatcher);     /* Misc. Tools -Heartbe
extern pascal void  ClrHeartBeat() inline(0x1403, dispatcher);     /* Misc. Tools -Heartbe
extern pascal void  InitMouse() inline(0x1803, dispatcher);     /* Misc. Tools -Mouse Tool
extern pascal void  SetMouse() inline(0x1903, dispatcher);     /* Misc. Tools -Mouse Tool
extern pascal void  HomeMouse() inline(0x1A03, dispatcher);     /* Misc. Tools -Mouse Tool
extern pascal void  ClearMouse() inline(0x1B03, dispatcher);     /* Misc. Tools -Mouse Too
extern pascal void  ClampMouse() inline(0x1C03, dispatcher);     /* Misc. Tools -Mouse Too
extern pascal void  PosMouse() inline(0x1E03, dispatcher);     /* Misc. Tools -Mouse Tool
extern pascal word  ServeMouse() inline(0x1F03, dispatcher);     /* Misc. Tools -Mouse Too
extern pascal void  SysDeathMgr() inline(0x1503, dispatcher);     /* Misc. Tools -System D
extern pascal void  QDStartUp() inline(0x0204, dispatcher);     /* Quickdraw II */
extern pascal void  QDShutDown() inline(0x0304, dispatcher);     /* Quickdraw II */
extern pascal word  QDVersion() inline(0x0404, dispatcher);     /* Quickdraw II */
extern pascal boolean  QDStatus() inline(0x0604, dispatcher);     /* Quickdraw II */
extern pascal void  GetAddress() inline(0x0904, dispatcher);     /* Quickdraw II */
extern pascal void  GrafOn() inline(0x0A04, dispatcher);     /* Quickdraw II */
extern pascal void  GrafOff() inline(0x0B04, dispatcher);     /* Quickdraw II */
extern pascal word  GetStandardSCB() inline(0x0c04, dispatcher);     /* Quickdraw II */
extern pascal void  InitColorTable() inline(0x0D04, dispatcher);     /* Quickdraw II */
extern pascal void  SetColorTable() inline(0x0E04, dispatcher);     /* Quickdraw II */
extern pascal void  GetColorTable() inline(0x0F04, dispatcher);     /* Quickdraw II */
extern pascal void  SetColorEntry() inline(0x1004, dispatcher);     /* Quickdraw II */
extern pascal word  GetColorEntry() inline(0x1104, dispatcher);     /* Quickdraw II */
extern pascal void  SetSCB() inline(0x1204, dispatcher);     /* Quickdraw II */
extern pascal word  GetSCB() inline(0x1304, dispatcher);     /* Quickdraw II */
extern pascal void  SetAllSCBs() inline(0x1404, dispatcher);     /* Quickdraw II */
extern pascal void  ClearScreen() inline(0x1504, dispatcher);     /* Quickdraw II */
extern pascal void  SetMasterSCB() inline(0x1604, dispatcher);     /* Quickdraw II */
extern pascal word  GetMasterSCB() inline(0x1704, dispatcher);     /* Quickdraw II */
extern pascal void  OpenPort() inline(0x1804, dispatcher);     /* Quickdraw II */
extern pascal void  InitPort() inline(0x1904, dispatcher);     /* Quickdraw II */
extern pascal void  ClosePort() inline(0x1A04, dispatcher);     /* Quickdraw II */
extern pascal void  SetPort() inline(0x1B04, dispatcher);     /* Quickdraw II */
extern pascal ptr  GetPort() inline(0x1C04, dispatcher);     /* Quickdraw II */
extern pascal void  SetPortLoc() inline(0x1D04, dispatcher);     /* Quickdraw II */
extern pascal void  GetPortLoc() inline(0x1E04, dispatcher);     /* Quickdraw II */
extern pascal void  SetPortRec() inline(0x1F04, dispatcher);     /* Quickdraw II */
extern pascal void  GetPortRec() inline(0x2004, dispatcher);     /* Quickdraw II */
extern pascal void  SetPortSize() inline(0x2104, dispatcher);     /* Quickdraw II */
extern pascal void  MovePortTo() inline(0x2204, dispatcher);     /* Quickdraw II */
extern pascal void  SetOrigin() inline(0x2304, dispatcher);     /* Quickdraw II */
extern pascal void  SetClip() inline(0x2404, dispatcher);     /* Quickdraw II */
extern pascal void  GetClip() inline(0x2504, dispatcher);     /* Quickdraw II */
extern pascal void  ClipRect() inline(0x2604, dispatcher);     /* Quickdraw II */
extern pascal void  HidePen() inline(0x2704, dispatcher);     /* Quickdraw II */
extern pascal void  ShowPen() inline(0x2804, dispatcher);     /* Quickdraw II */
extern pascal void  GetPen() inline(0x2904, dispatcher);     /* Quickdraw II */
extern pascal void  SetPenState() inline(0x2A04, dispatcher);     /* Quickdraw II */
extern pascal void  GetPenState() inline(0x2B04, dispatcher);     /* Quickdraw II */
extern pascal void  SetPenSize() inline(0x2C04, dispatcher);     /* Quickdraw II */
extern pascal void  GetPenSize() inline(0x2D04, dispatcher);     /* Quickdraw II */
extern pascal void  SetPenMode() inline(0x2E04, dispatcher);     /* Quickdraw II */
extern pascal word  GetPenMode() inline(0x2F04, dispatcher);     /* Quickdraw II */
extern pascal void  SetPenPat() inline(0x3004, dispatcher);     /* Quickdraw II */
extern pascal void  GetPenPat() inline(0x3104, dispatcher);     /* Quickdraw II */
extern pascal void  SetPenMask() inline(0x3204, dispatcher);     /* Quickdraw II */
extern pascal void  GetPenMask() inline(0x3304, dispatcher);     /* Quickdraw II */
```

```
extern pascal void  InvertRgn() inline(0x7C04, dispatcher);      /* Quickdraw II */
extern pascal void  FillRgn() inline(0x7D04, dispatcher);      /* Quickdraw II */
extern pascal void  ScrollRect() inline(0x7E04, dispatcher);      /* Quickdraw II */
extern pascal void  PaintPixels() inline(0x7F04, dispatcher);      /* Quickdraw II */
extern pascal void  AddPt() inline(0x8004, dispatcher);      /* Quickdraw II */
extern pascal void  SubPt() inline(0x8104, dispatcher);      /* Quickdraw II */
extern pascal void  SetPt() inline(0x8204, dispatcher);      /* Quickdraw II */
extern pascal boolean  EqualPt() inline(0x8304, dispatcher);      /* Quickdraw II */
extern pascal void  LocalToGlobal() inline(0x8404, dispatcher);      /* Quickdraw II */
extern pascal void  GlobalToLocal() inline(0x8504, dispatcher);      /* Quickdraw II */
extern pascal int  Random() inline(0x8604, dispatcher);      /* Quickdraw II */
extern pascal void  SetRandSeed() inline(0x8704, dispatcher);      /* Quickdraw II */
extern pascal word  GetPixel() inline(0x8804, dispatcher);      /* Quickdraw II */
extern pascal void  ScalePt() inline(0x8904, dispatcher);      /* Quickdraw II */
extern pascal void  MapPt() inline(0x8A04, dispatcher);      /* Quickdraw II */
extern pascal void  MapRect() inline(0x8B04, dispatcher);      /* Quickdraw II */
extern pascal void  MapRgn() inline(0x8C04, dispatcher);      /* Quickdraw II */
extern pascal void  SetStdProcs() inline(0x8D04, dispatcher);      /* Quickdraw II */
extern pascal void  SetCursor() inline(0x8E04, dispatcher);      /* Quickdraw II */
extern pascal ptr  GetCursorAdr() inline(0x8F04, dispatcher);      /* Quickdraw II */
extern pascal void  HideCursor() inline(0x9004, dispatcher);      /* Quickdraw II */
extern pascal void  ShowCursor() inline(0x9104, dispatcher);      /* Quickdraw II */
extern pascal void  ObscureCursor() inline(0x9204, dispatcher);      /* Quickdraw II */
extern pascal void  SetFont() inline(0x9404, dispatcher);      /* Quickdraw II */
extern pascal handle  GetFont() inline(0x9504, dispatcher);      /* Quickdraw II */
extern pascal void  GetFontInfo() inline(0x9604, dispatcher);      /* Quickdraw II */
extern pascal void  SetFontFlags() inline(0x9804, dispatcher);      /* Quickdraw II */
extern pascal word  GetFontFlags() inline(0x9904, dispatcher);      /* Quickdraw II */
extern pascal void  SetTextFace() inline(0x9A04, dispatcher);      /* Quickdraw II */
extern pascal word  GetTextFace() inline(0x9B04, dispatcher);      /* Quickdraw II */
extern pascal void  SetTextMode() inline(0x9C04, dispatcher);      /* Quickdraw II */
extern pascal word  GetTextMode() inline(0x9D04, dispatcher);      /* Quickdraw II */
extern pascal void  SetSpaceExtra() inline(0x9E04, dispatcher);      /* Quickdraw II */
extern pascal word  GetSpaceExtra() inline(0x9F04, dispatcher);      /* Quickdraw II */
extern pascal void  SetForeColor() inline(0xA004, dispatcher);      /* Quickdraw II */
extern pascal word  GetForeColor() inline(0xA104, dispatcher);      /* Quickdraw II */
extern pascal void  SetBackColor() inline(0xA204, dispatcher);      /* Quickdraw II */
extern pascal word  GetBackColor() inline(0xA304, dispatcher);      /* Quickdraw II */
extern pascal void  DrawChar() inline(0xA404, dispatcher);      /* Quickdraw II */
extern pascal void  DrawString() inline(0xA504, dispatcher);      /* Quickdraw II */
extern pascal void  DrawCString() inline(0xA604, dispatcher);      /* Quickdraw II */
extern pascal void  DrawText() inline(0xA704, dispatcher);      /* Quickdraw II */
extern pascal int  CharWidth() inline(0xA804, dispatcher);      /* Quickdraw II */
extern pascal int  StringWidth() inline(0xA904, dispatcher);      /* Quickdraw II */
extern pascal int  CStringWidth() inline(0xAA04, dispatcher);      /* Quickdraw II */
extern pascal int  TextWidth() inline(0xAB04, dispatcher);      /* Quickdraw II */
extern pascal void  CharBounds() inline(0xAC04, dispatcher);      /* Quickdraw II */
extern pascal void  StringBounds() inline(0xAD04, dispatcher);      /* Quickdraw II */
extern pascal void  CStringBounds() inline(0xAE04, dispatcher);      /* Quickdraw II */
extern pascal void  TextBounds() inline(0xAF04, dispatcher);      /* Quickdraw II */
extern pascal void  SetSysFont() inline(0xB204, dispatcher);      /* Quickdraw II */
extern pascal handle  GetSysFont() inline(0xB304, dispatcher);      /* Quickdraw II */
extern pascal void  SetVisRgn() inline(0xB404, dispatcher);      /* Quickdraw II */
extern pascal void  GetVisRgn() inline(0xB504, dispatcher);      /* Quickdraw II */
extern pascal void  SetIntUse() inline(0xB604, dispatcher);      /* Quickdraw II */
extern pascal void  FramePoly() inline(0xBC04, dispatcher);      /* Quickdraw II */
extern pascal void  PaintPoly() inline(0xBD04, dispatcher);      /* Quickdraw II */
```

```
extern pascal void  ErrWriteLine() inline(0x1B0C, dispatcher);      /* Text Tools */
extern pascal void  WriteString() inline(0x1C0C, dispatcher);      /* Text Tools */
extern pascal void  ErrWriteString() inline(0x1D0C, dispatcher);      /* Text Tools */
extern pascal void  WriteBlock() inline(0x1E0C, dispatcher);      /* Text Tools */
extern pascal void  ErrWriteBlock() inline(0x1F0C, dispatcher);      /* Text Tools */
extern pascal void  WriteCString() inline(0x200C, dispatcher);      /* Text Tools */
extern pascal void  ErrWriteCString() inline(0x210C, dispatcher);      /* Text Tools */
extern pascal word  ReadChar() inline(0x220C, dispatcher);      /* Text Tools */
extern pascal void  ReadBlock() inline(0x230C, dispatcher);      /* Text Tools */
extern pascal word  ReadLine() inline(0x240C, dispatcher);      /* Text Tools */
extern pascal void  TLStartUp() inline(0x0201, dispatcher);      /* Tool Locator */
extern pascal void  TLShutDown() inline(0x0301, dispatcher);      /* Tool Locator */
extern pascal word  TLVersion() inline(0x0401, dispatcher);      /* Tool Locator */
extern pascal void  TLReset() inline(0x0501, dispatcher);      /* Tool Locator */
extern pascal ptr  GetTSPtr() inline(0x0901, dispatcher);      /* Tool Locator */
extern pascal void  SetTSPtr() inline(0x0A01, dispatcher);      /* Tool Locator */
extern pascal ptr  GetFuncPtr() inline(0x0B01, dispatcher);      /* Tool Locator */
extern pascal ptr  GetWAP() inline(0x0C01, dispatcher);      /* Tool Locator */
extern pascal void  SetWAP() inline(0x0D01, dispatcher);      /* Tool Locator */

extern pascal void LoadTools() inline(0x0E01, dispatcher);      /* Tool Locator */

extern pascal void  WindStartUp() inline(0x020E, dispatcher);      /* Window Manager */
extern pascal void  WindShutDown() inline(0x030E, dispatcher);      /* Window Manager */
extern pascal word  WindVersion() inline(0x040E, dispatcher);      /* Window Manager */
extern pascal void  WindReset() inline(0x050E, dispatcher);      /* Window Manager */
extern pascal void  WindStatus() inline(0x060E, dispatcher);      /* Window Manager */
extern pascal ptr  NewWindow() inline(0x090E, dispatcher);      /* Window Manager */
extern pascal boolean  CheckUpdate() inline(0x0A0E, dispatcher);      /* Window Manager */
extern pascal void  CloseWindow() inline(0x0B0E, dispatcher);      /* Window Manager */
extern pascal long  Desktop() inline(0x0C0E, dispatcher);      /* Window Manager */
extern pascal void  SetWTitle() inline(0x0D0E, dispatcher);      /* Window Manager */
extern pascal ptr  GetWTitle() inline(0x0E0E, dispatcher);      /* Window Manager */
extern pascal void  SetFrameColor() inline(0x0F0E, dispatcher);      /* Window Manager */
extern pascal void  GetFrameColor() inline(0x100E, dispatcher);      /* Window Manager */
extern pascal void  SelectWindow() inline(0x110E, dispatcher);      /* Window Manager */
extern pascal void  HideWindow() inline(0x120E, dispatcher);      /* Window Manager */
extern pascal void  ShowWindow() inline(0x130E, dispatcher);      /* Window Manager */
extern pascal void  SetInfoDraw() inline(0x130E, dispatcher);      /* Window Manager */
extern pascal void  SendBehind() inline(0x140E, dispatcher);      /* Window Manager */
extern pascal ptr  FrontWindow() inline(0x150E, dispatcher);      /* Window Manager */
extern pascal word  FindWindow() inline(0x170E, dispatcher);      /* Window Manager */
extern pascal boolean  TrackGoAway() inline(0x180E, dispatcher);      /* Window Manager */
extern pascal void  MoveWindow() inline(0x190E, dispatcher);      /* Window Manager */
extern pascal void  DragWindow() inline(0x1A0E, dispatcher);      /* Window Manager */
extern pascal long  GrowWindow() inline(0x1B0E, dispatcher);      /* Window Manager */
extern pascal void  SizeWindow() inline(0x1C0E, dispatcher);      /* Window Manager */
extern pascal word  TaskMaster() inline(0x1D0E, dispatcher);      /* Window Manager */
extern pascal void  BeginUpdate() inline(0x1E0E, dispatcher);      /* Window Manager */
extern pascal void  EndUpdate() inline(0x1F0E, dispatcher);      /* Window Manager */
extern pascal ptr  GetWMgrPort() inline(0x200E, dispatcher);      /* Window Manager */
extern pascal long  PinRect() inline(0x210E, dispatcher);      /* Window Manager */
extern pascal void  HiliteWindow() inline(0x220E, dispatcher);      /* Window Manager */
extern pascal void  WNewRes() inline(0x250E, dispatcher);      /* Window Manager */
extern pascal boolean  TrackZoom() inline(0x250E, dispatcher);      /* Window Manager */
extern pascal void  ZoomWindow() inline(0x260E, dispatcher);      /* Window Manager */
extern pascal void  SetWRefCon() inline(0x280E, dispatcher);      /* Window Manager */
```