THE MAC ALMANAC II

MAC ALMANAC II, VERSION 1.0 - 4 JUNE 1990

You may have wondered, as I have, why nobody has ever come up with any kind of comprehensive listing of the Mac's ROM traps and global variables. So far, the lists in existence are scattered in many widely varying sources. For example, Apple's Inside Macintosh contains ROM trap listings in both alphabetical and numeric orders, but its global variable list is both incomplete and in alphabetic order only. Having lists ordered by memory address helps make debugging with TMON and Macsbug much easier. It also helps programmers using 68000 assembly language, such as myself.

Apple's official policy is that programmers should use the names of variables when writing their programs. Their reasoning is that the use of names will prevent incompatibilities with future Macs and/or system files. I don't follow that policy for two reasons. First, the only way to create incompatibilities is for Apple to change the locations of some or all of the global variables in a future release of system software. Doing this would immediately invalidate the entire Macintosh software base, so I don't think Apple would seriously think of it. Secondly, encoding the actual addresses helps debugging since my source code becomes closer to the object code. This is efficient because I only have to look up variable names once when writing source code and eliminates multiple lookups when debugging with Macsbug. Normally I use variable addresses in my source code and place variable names in comment lines.

After struggling with the separated lists, I decided to improve my productivity by centralizing all that information in a master file. This master file, MacAlmanac, has since helped me greatly. Its distribution as PD (Public Domain) made sense, so others could benefit as well.

It would be a good idea to print this file in its entirety and three-hole punch it for placement in a binder. This file has been reformatted for printing on the LaserWriter, and for completeness, the laser font Times has been substituted for the screen font Geneva used in Almanac I. The master list will be a time-saver for you, regardless of what development system you are using (Pascal, C, BASIC, FORTRAN, Lisp, Forth, etc.). It will also help you if you disassemble bits of ROM code or other hacker stuff.

Revision History:

VERSION		DATE	PAGES	CHANGES
I	1.0	28 January 1987	26	Created, first release
I	1.1	12 May 1987	27	Added: Errors on Powerup
I	1.2	25 October 1987	29	Added: Traps & Keyboards
I	1.3	16 December 1987	29	Altered pages 19, 27, and 28
I	1.4	30 December 1987	29	Fonts table reorganized (Page 26)
I	1.5	18 April 1988	30	Added: new global variables
II	1.0	4 June 1990	32	Total rewrite of Almanac I, first rerelease

If you need to contact me, I can be reached at my home address:

James H. Olson
4515 Hurley Street
Philadelphia, PA 19120-4528
(215) 457-7114 (answering machine)

***** MAC ALMANAC IS IN THE PUBLIC DOMAIN *****

PART 1

Global Variables in Memory Address Order

Simply, global variables on the Macintosh™ are variables that hold values that are independent of any running program and don't usually change value when one program quits and another starts. These variables are all assigned as a group to one or two large blocks of RAM located between location 0 (the start of RAM) and the system and application areas of RAM. Each variable is assigned a specific memory location. Values stored in these variables usually contain information such as the amount of RAM installed in the Mac, the version of ROM, system, and Finder installed, mouse location, serial port settings, and the like. Programs can read and write to any of these variables, and writes will have lasting effects on the system (until changed again). Upon startup, and again if the Mac is reset, just about all variables are initialized by the ROM startup code after any memory tests (see Part 6).

Since the Mac was introduced in 1984, Apple has encouraged people writing Mac software to read and write to global variables by using their names. When language compilers see these names, they get the equivalent addresses from an "include" file specified during the compilation process. The final program only has the memory addresses. Also since 1984, Apple has made press announcements and other talk about getting rid of global variables, or at least making major changes. This verbal noise was particularly evident when Switcher™ (later MultiFinder™) was released. It turns out that changing existing global variables would create a mass confusion big enough to jeopardize the entire Mac product line. Because all existing Mac software (system, application, DA, INIT, etc.) depends on globals, even if the software doesn't access globals directly, changing them would instantly invalidate the entire Mac software base. Apple cannot arbitrarily change the locations of these variables without creating problems with just about every Macintosh program on the market. Because of this restraining effect, Apple only makes improves existing variables or adds new ones. A good example of this is the variable BootDrive (located at $210). Under the original 64K ROM it contained the drive number where the boot floppy was located. When HFS was introduced in 1986, BootDrive was changed to hold the volume/directory reference number of the boot disk. This change made BootDrive more useful now that hard disks are frequently used as the boot disk, yet remains compatible with floppies.

I have attempted to give the most complete listings possible, drawing from several sources. For instance, Apple's *Inside Macintosh* gives only a partial listing of global variables in alphabetical order. My list is in memory address order, which is better suited to debugging with MacsBug and other monitor-like debuggers. The list also exposes "holes": locations Apple hasn't yet made public. While compiling the list, I noticed some variables had more than one name. From what I can decipher, many of these variables were used during the original development of the Macintosh before its release in 1984 and fell into disuse afterwards. Some even have strange names like "MonkeyLives" ($100) and "MrMacHook" ($A2C). Microsoft even had a location named after it ($A78- now ApplScratch), and a location was named after the infamous "Twiggy" 5.25-inch floppy disk drives on the original Lisa (TwiggyVars, $128). Alternate names appear probably because Apple re-used these locations for new variables while developing the 128K ROM. Another good example is ROM85. Under the old 64K ROM, this was an non-public location that held -1. Under all newer ROMs, this location always holds a positive value.

In the following list, old names from the 64K ROM are noted by a dot "•" in the left column before their names. Note that certain sections of memory have special names declared. These area-designators look like variables but are easy to pick out- their size fields contain hyphens. All locations that haven't been made public are marked with "[????]".

Variable	Location	Size	Description
SysCom	$100	-	start of System communication area
MonkeyLives	$100	word	monkey lives if nonzero
ScrVRes	$102	word	screen vertical dots/inch
ScrHRes	$104	word	screen horizontal dots/inch
ScreenRow	$106	word	rowBytes of screen
MemTop	$108	long	ptr to end of RAM

	BufPtr	$10C	long	ptr to end of jump table
	StkLowPt	$110	long	lowest stack pointer value as measured in VBL task
	HeapEnd	$114	long	ptr to end of application heap
	TheZone	$118	long	ptr to current heap zone
	UTableBase	$11C	long	ptr to unit I/O table
	MacJmp	$120	long	ptr to jump vector table used by MacsBug
	DskRtnAdr	$124	long	temporary pointer used by Disk Driver
•	TwiggyVars	$128	long	ptr to 'other' driver variables (Lisa 5.25" drive)
	PollRtnAddr	$128	long	ptr to 'other' driver variables (Lisa 5.25" drive)
	DskVerify	$12C	byte	used by Mac 3.5" Disk Driver for read/verify
	LoadTrap	$12D	byte	trap before launch?
	MmInOK	$12E	byte	Initial Memory Manager checks ok?
•	DskWr11	$12F	byte	try 1-1 disk writes?
	CPUFlag	$12F	byte	code for installed CPU: 0=68000, 1=68010, 2=68020, 3=68030
	ApplLimit	$130	long	address of application heap limit
	SonyVars	$134	long	ptr to Mac 3.5" Disk Driver variables
	PWMValue	$138	word	current PWM value
	PollStack	$13A	long	address of SCC poll data start stack location
	PollProc	$13E	long	ptr to SCC poll data procedure
	DskErr	$142	word	disk routine result code
	SysEvtMask	$144	word	system event mask
	SysEvtBuf	$146	long	ptr to system event queue element buffer
	EventQueue	$14A	10	event queue header
	EvtBufCnt	$154	word	maximum #of events in SysEvtBuf minus 1
	RndSeed	$156	long	random number seed
	SysVersion	$15A	word	System file version number (e.g. System 4.1=$0410)
	SEvtEnb	$15C	byte	0 = SysEvent always returns FALSE
	DSWndUpdate	$15D	byte	GetNextEvent not to paint behind System error dialog?
	FontFlag	$15E	byte	font manager loop flag
	Filler3	$15F	byte	1 byte of filler
	VBLQueue	$160	10	VBL queue header
	Ticks	$16A	long	Tick count: time since system startup (tick=1/60 sec)
	MBTicks	$16E	long	tick count when mouse button was last pressed
	MBState	$172	byte	current mouse button state
	Tocks	$173	byte	Lisa sub-tick count
	KeyMap	$174	8	bitmap of the keyboard
	KeypadMap	$17C	long	bitmap for numeric keypad (uses 18 bits)
	[????]	$180	long	<unknown location>
	KeyLast	$184	word	ASCII code for last valid keycode
	KeyTime	$186	long	tickcount when KEYLAST was received
	KeyRepTime	$18A	long	tick count when key was last repeated
	KeyThresh	$18E	word	threshold for key repeat
	KeyRepThresh	$190	word	key repeat speed
	Lv11DT	$192	32	Level-1 secondary interrupt vector table
	Lv12DT	$1B2	32	Level-2 secondary interrupt vector table
	UnitNtryCnt	$1D2	word	count of entries in unit table
	VIA	$1D4	long	base address of 6522 VIA chip
	SCCRd	$1D8	long	address of Z8530 SCC chip (used when reading the chip)
	SCCWr	$1DC	long	address of Z8530 SCC chip (used when writing the chip)
	IWM	$1E0	long	base address of IWM chip (floppy drive controller)
	scratch20	$1E4	20	general scratch area
	SysParam	$1F8	-	System parameter RAM vars (PRAM info)
	SPValid	$1F8	byte	validation: $A8 if last write to clock chip was good
	SPATalkA	$1F9	byte	AppleTalk node ID for modem port
	SPATalkB	$1FA	byte	AppleTalk node ID for printer port
	SPConfig	$1FB	byte	serial-port-in-use flags for both ports
	SPPortA	$1FC	word	modem port configuration (baud, parity, bits)
	SPPortB	$1FE	word	printer port configuration (baud, parity, bits)

	SPAlarm	$200	long	alarm clock setting
	SPFont	$204	word	font number of application font minus 1
•	SPKbdPrint	$206	word	auto-key threshold/rate and printer connection
	SPKbd	$206	byte	auto-key threshold and rate
	SPPrint	$207	byte	printer connection
•	SPVolClik	$208	word	speaker volume; double click and caret flash times
	SPVolCtl	$208	byte	speaker volume
	SPClikCaret	$209	byte	double click and caret flash times
	SPMisc	$20A	byte	reserved for future use
	SPMisc2	$20B	byte	mouse tracking, startup floppy drive, menu blink
	Time	$20C	long	current date/time (seconds since midnight 1 JAN 1904)
•	BootDrive	$210	word	drive number of boot drive
	BootDrive	$210	word	working directory reference number of boot disk
	JShell	$212	word	journaling shell state
•	Filler3A	$214	word	negative of vRefNum last seen by Standard File Package
	SFSaveDisk	$214	word	negative of vRefNum last seen by Standard File Package
	KbdVars	$216	-	Keyboard manager variables
	[????]	$216	word	<unknown location>
	KbdLast	$218	byte	ADB address of keyboard last used
	[????]	$219	byte	<unknown location>
	JKybdTask	$21A	long	ptr to keyboard VBL task hook
	KbdType	$21E	byte	keyboard model number
	AlarmState	$21F	byte	alarm clock: Bit7=parity, Bit6=beeped, Bit0=enable
•	CurIOTrap	$220	word	current I/O trap being executed
	MemErr	$220	word	Memory Manager error code
	DiskVars	$222	-	Disk driver variables (60 bytes)
	[????]	$222	60	<unknown locations>
	FlEvtMask	$25E	word	mask of flushable events (FlushEvents)
	SdVolume	$260	byte	Current speaker volume (bits 0 through 2 only)
	SdEnable	$261	byte	Sound enabled?
	SoundVars	$262	-	Sound driver variables (32 bytes)
	SoundPtr	$262	long	pointer to 4-voice sound definition (SynthRec)
	SoundBase	$266	long	ptr to free-form sound definition (SynthRec)
	SoundVBL	$26A	16	vertical retrace control element
	SoundDCE	$27A	long	pointer to Sound Driver's device control entry
	SoundActive	$27E	byte	sound is active?
	SoundLevel	$27F	byte	current amplitude in 740-byte sound buffer
	CurPitch	$280	word	current value of COUNT in square-wave SynthRec
	SoundLast	$282	long	address past last sound variable
	[????]	$286	long	<unknown location>
	[????]	$28A	long	<unknown location>
	ROM85	$28E	word	holds a positive value if 128K or later ROM in Mac
	PortAUse	$290	byte	Port A usage: if zero, port available
	PortBUse	$291	byte	Port B usage: if zero, port available
	ScreenVars	$292	-	Screen driver variables (8 bytes)
	[????]	$292	long	<unknown location>
	[????]	$296	long	<unknown location>
	JGNEFilter	$29A	long	ptr to GetNextEvent filter procedure
	Key1Trans	$29E	long	ptr to keyboard translator procedure
	Key2Trans	$2A2	long	ptr to numeric keypad translator procedure
	SysZone	$2A6	long	starting address of system heap zone
	ApplZone	$2AA	long	starting address of application heap zone
	ROMBase	$2AE	long	base address of ROM (Trap Dispatcher)
	RAMBase	$2B2	long	base address of RAM (Trap Dispatcher)
	BasicGlob	$2B6	long	ptr to BASIC globals
	DSAlertTab	$2BA	long	ptr to system error alert table in use
	ExtStsDT	$2BE	16	External/status interrupt vector table
	SCCASts	$2CE	byte	SCC read register 0 last external/status interrupt - A

	SCCBSts	$2CF	byte	SCC read register 0 last external/status interrupt - B
	SerialVars	$2D0	-	async driver variables (16 bytes)
	[????]	$2D0	long	<unknown location>
	[????]	$2D4	long	<unknown location>
	ABusVars	$2D8	long	ptr to AppleTalk variables
	[????]	$2DC	long	<unknown location>
	FinderName	$2E0	16	name of the shell, usually "Finder" (STRING[15])
	DoubleTime	$2F0	long	double click interval in ticks
	CaretTime	$2F4	long	caret blink interval in ticks
	ScrDmpEnb	$2F8	byte	screen dump enable - zero disables FKEY processing
	ScrDmpType	$2F9	byte	$FF dumps screen, $FE dumps front window (FKEY 4)
	TagData	$2FA	-	sector tag info for disk drivers (14 bytes)
	[????]	$2FA	word	<unknown location>
	BufTgFNum	$2FC	long	File tags buffer: file number
	BufTgFFlg	$300	word	File tags buffer: flags (bit1=1 if resource fork)
	BufTgFBkNum	$302	word	File tags buffer: logical block number
	BufTgDate	$304	long	File tags buffer: last modification date/time
	DrvQHdr	$308	10	queue header of drives in system
	PWMBuf2	$312	long	ptr to PWM buffer 1 (or 2 if sound)
	HpChk	$316	long	heap check RAM code
•	MaskBC	$31A	long	Memory Manager byte count mask
•	MaskHandle	$31A	long	Memory Manager handle mask
•	MaskPtr	$31A	long	Memory Manager pointer mask
	Lo3Bytes	$31A	long	holds the constant $00FFFFFF
	MinStack	$31E	long	minimum stack size used in InitApplZone
	DefltStack	$322	long	default size of stack
	MMDefFlags	$326	word	default zone flags
	GZRootHnd	$328	long	root handle for GrowZone
	GZRootPtr	$32C	long	root pointer for GrowZone
	GZMoveHnd	$330	long	moving handle for GrowZone
	DSDrawProc	$334	long	ptr to alternate system error draw procedure
	EjectNotify	$338	long	ptr to eject notify procedure
	IAZNotify	$33C	long	ptr to world swaps notify procedure
	FileVars	$340	-	file system vars (184 bytes)
	CurDB/CkdDB	$340	word	current dir block/used for searches
•	FSCallAsync	$342	word	"One byte free"
	NxtDB	$342	word	<no description available>
	MaxDB	$344	word	<no description available>
	FlushOnly	$346	byte	flag used by UnMountVol and FlushVol
	RegRsrc	$347	byte	flag used by OpenRF and FileOpen
	FLckUnlck	$348	byte	flag used by SetFilLock and RstFilLock
	FrcSync	$349	byte	when set, all file system calls are synchronized
	NewMount	$34A	byte	used by MountVol to flag new mounts
	NoEject	$34B	byte	used by Eject and Offline
	DrMstrBlk	$34C	word	master directory block in a volume
	HFS Globals	$34E	-	HFS global variables (168 bytes)
	FCBSPtr	$34E	long	ptr to file control block buffer
	DefVCBPtr	$352	long	ptr to default volume control block
	VCBQHdr	$356	10	volume control block queue header
	FSQHdr	$360	10	file I/O queue header
	HFSVars	$36A	-	Start of TFS variables (RAM version)
	HFSStkTop	$36A	long	Temp location of stack ptr during async calls
	HFSStkPtr	$36E	long	Temporary location of HFS stack ptr
	WDCBsPtr	$372	long	Working Directory queue header
	HFSFlags	$376	byte	Internal HFS flags
•	SysCRefCnt	$377	byte	system cache usage count (#of vols)
	CacheFlag	$377	byte	system cache usage count now used as cache flag
	SysBMCPtr	$378	long	System-wide bitmap cache pointer

	SysVolCPtr	$37C	long	System-wide volume cache pointer
	SysCtlCPtr	$380	long	System-wide control cache pointer
	DefVRefNum	$384	word	Default volume's VRefNum/WDRefNum
	PMSPPtr	$386	long	ptr to list of directories on PMSP
	HFSDSErr	$392	word	Final gasp - error that caused IOErr
	HFSVarEnd	$394	-	End of HFS variable area
	CacheVars	$394	8	<no description available>
	CurDirStore	$398	word	ID of last directory opened
	[????]	$39A	word	<unknown location>
	CacheCom	$39C	long	<no description available>
	[????]	$3A0	word	<unknown location>
	ErCode	$3A2	word	report errors here during async routines
	Params	$3A4	-	File Mgr I/O ParamBlock (50 bytes)
	[????]	$3A4	50	<unknown locations>
	FSTemp8	$3D6	8	used by Rename
•	FSTemp4	$3DE	word	used by Rename and CkFilMod
	FSIOErr	$3DE	word	last I/O error
	[????]	$3E0	word	<unknown location>
	FSQueueHook	$3E2	long	ptr to hook to capture all FS calls
	ExtFSHook	$3E6	long	ptr to command done hook
	DskSwtchHook	$3EA	long	ptr to hook for disk-switch dialog
	ReqstVol	$3EE	long	ptr to offline or external file system volume VCB
	ToExtFS	$3F2	long	ptr to external file system
	FSVarEnd	$3F6	-	end of file system variables
	FSFCBLen	$3F6	word	size of file control block; contains -1 on 64K ROM Macs
	DSAlertRect	$3F8	8	rectangle for system error and disk-switch alerts
•	DispatchTab	$400	1024	OS & Toolbox trap dispatch table (64K ROM) (512 words)
	OSDispTable	$400	1024	OS trap dispatch table (128K and later ROM) (256 longs)
	GRAFBEGIN	$800	-	graf (QuickDraw) global area
	JHideCursor	$800	long	<no description available>
	JShowCursor	$804	long	<no description available>
	JShieldCursor	$808	long	<no description available>
	JScrnAddr	$80C	long	<no description available>
	JScrnSize	$810	long	<no description available>
	JInitCrsr	$814	long	<no description available>
	JSetCrsr	$818	long	<no description available>
	JCrsrObscure	$81C	long	<no description available>
	JUpdateProc	$820	long	<no description available>
	LGrafJump	$824	long	<no description available>
	GrafVar	$824	-	QuickDraw variables
	ScrnBase	$824	long	base address of main screen
	MTemp	$828	long	low-level interrupt mouse location
	RawMouse	$82C	long	un-jerked mouse coordinates
	NMouse	$830	long	processed mouse coordinate
	CrsrPin	$834	8	cursor pinning rectangle
	CrsrRect	$83C	8	cursor hit rectangle
	TheCrsr	$844	68	cursor data, mask & hotspot
	CrsrAddr	$888	long	address of data under cursor
•	CrsrSave	$88C	64	data under the cursor
	CrsrSave	$88C	long	ptr to data under the cursor
	[????]	$890	20	<unknown locations>
	MainDevice	$8A4	long	handle to current main device
	DeviceList	$8A8	long	handle to first element in device list
	[????]	$8AC	long	<unknown location>
	QDColors	$8B0	28	default QuickDraw colors
	CrsrVis	$8CC	byte	cursor visible?

	CrsrBusy	$8CD	byte	cursor locked out?
	CrsrNew	$8CE	byte	cursor changed?
	CrsrCouple	$8CF	byte	cursor coupled to mouse?
	CrsrState	$8D0	word	cursor nesting level
	CrsrObscure	$8D2	byte	Cursor obscure semaphore
	CrsrScale	$8D3	byte	cursor scaled?
	[????]	$8D4	word	<unknown location>
	MouseMask	$8D6	long	V-H mask for ANDing with mouse
	MouseOffset	$8DA	long	V-H offset for adding after ANDing
	JournalFlag	$8DE	word	journaling state
	JSwapFont	$8E0	long	jump entry for FMSwapFont
•	JFontInfo	$8E4	long	jump entry for FMFontMetrics
	WidthListHand	$8E4	long	handle to a list of handles of recently-used width tables
	JournalRef	$8E8	word	Journalling driver's refnum
	[????]	$8EA	word	<unknown location>
	CrsrThresh	$8EC	word	delta threshold for mouse scaling
	JCrsrTask	$8EE	long	address of CrsrVBLTask
	GRAFEND	$8F2	-	End of graphics globals
	WWExist	$8F2	byte	window manager initialized?
	DExist	$8F3	byte	QuickDraw is initialized
	JFetch	$8F4	long	ptr to fetch-a-byte routine for drivers
	JStash	$8F8	long	ptr to stash-a-byte routine for drivers
	JIODone	$8FC	long	ptr to IODone routine for drivers
	LoadVars	$900	-	Segment Loader variables (68 bytes)
	CurApRefNum	$900	word	refNum of current application's resFile
	LaunchFlag	$902	byte	Tells whether Launch or Chain was last called
	[????]	$903	byte	<unknown location>
	CurrentA5	$904	long	current value of register A5
	CurStackBase	$908	long	ptr to the base (beginning) of the stack
	[????]	$90C	long	<unknown location>
	CurApName	$910	32	name of current application (STRING[31])
	SaveSegHandle	$930	long	handle to segment 0 (CODE 0)
	CurJTOffset	$934	word	current jump table offset from register A5
	CurPageOption	$936	word	current page 2 configuration (screen/sound buffers)
	HiliteMode	$938	word	set to -1 if hilighting mode is on, 0 otherwise
	LoaderPBlock	$93A	10	param block for ExitToShell
•	PrintVars	$944	16	print code variables
•	LastLGlobal	$944	long	address past last loader global
	PrintErr	$944	word	Print Manager error code
	[????]	$946	14	<unknown locations>
•	CoreEditVars	$954	12	core edit variables
	LastPGlobal	$954	long	address of last printer global
	[????]	$958	long	<unknown location>
	[????]	$95C	long	<unknown location>
	scrapVars	$960	-	Scrap Manager variables (32 bytes)
•	scrapInfo	$960	long	scrap length
	scrapSize	$960	long	scrap length
	scrapHandle	$964	long	handle to RAM scrap
	scrapCount	$968	word	count changed by ZeroScrap
	scrapState	$96A	word	scrap state: tells if scrap exists in RAM or on disk
	scrapName	$96C	long	pointer to scrap file name (normally "Clipboard File")
	scrapTag	$970	16	scrap file name (STRING[15])
	scrapEnd	$980	-	End of scrap vars
	ToolGBase	$980	-	base address of toolbox globals
	ToolVars	$980	-	toolbox variables
	RomFont0	$980	long	handle to system font

•	ApFontID	$984	word	resource ID of application font
	ApFontID	$984	word	font number of application font
	GotStrike	$986	byte	Do we have the strike?
	FMDefaultSize	$987	byte	default size
•	CurFMInput	$988	long	ptr to QuickDraw FMInput record
	CurFMFamily	$988	word	current font family
	CurFMSize	$98A	word	current font size
	CurFMFace	$98C	byte	current font face
	CurFMNeedBits	$98D	byte	boolean telling whether it needs strike
	CurFMDevice	$98E	word	current font device
	CurFMNumer	$990	long	current numerator of scale factor
	CurFMDenom	$994	long	current denominator of scale factor
	FMgrOutRec	$998	long	ptr to QuickDraw FontOutput record
	FOutError	$998	word	Font Manager error code
	TFOutFontHandle	$99A	long	handle to font bits
	FOutBold	$99E	byte	bolding factor
	FOutItalic	$99F	byte	italic factor
	FOutULOffset	$9A0	byte	underline offset
	FOutULShadow	$9A1	byte	underline halo
	FOutULThick	$9A2	byte	underline thickness
	FOutShadow	$9A3	byte	shadow factor
	FOutExtra	$9A4	byte	extra horizontal width
	FOutAscent	$9A5	byte	height above baseline
	FOutDescent	$9A6	byte	height below baseline
	FOutWidMax	$9A7	byte	maximum width of character
	FOutLeading	$9A8	byte	space between lines
	FOutUnused	$9A9	byte	unused (padding) byte -must have even number
	FOutNumer	$9AA	long	point for numerators of scale factor
	FOutDenom	$9AE	long	point for denominators of scale factor
	FMDotsPerInch	$9B2	long	h,v dotsPerInch (resolution) of current device
	FMStyleTab	$9B6	24	style heuristic table given by device
	ToolScratch	$9CE	8	scratch area
	WindowList	$9D6	long	ptr to Z-ordered linked list of windows
	SaveUpdate	$9DA	word	Enable update events?
	PaintWhite	$9DC	word	erase windows before update event?
	WMgrPort	$9DE	long	ptr to window manager's grafport
	DeskPort	$9E2	long	ptr to Desk grafPort (Whole screen)
	OldStructure	$9E6	long	handle to saved structure region
	OldContent	$9EA	long	handle to saved content region
	GrayRgn	$9EE	long	handle to rounded-corner region drawn as the desktop
	SaveVisRgn	$9F2	long	handle to temporarily saved visRegion
	DragHook	$9F6	long	ptr to user hook called during dragging
	scratch8	$9FA	8	general scratch area
	TempRect	$9FA	8	scratch rectangle
	OneOne	$A02	long	holds the constant $00010001
	MinusOne	$A06	long	holds the constant $FFFFFFFF
	TopMenuItem	$A0A	word	pixel value of top of scrollable menu
	AtMenuBottom	$A0C	word	flag for menu scrolling
	IconBitmap	$A0E	14	scratch bitmap used for plotting things
	MenuList	$A1C	long	handle to current menuBar list structure
	MBarEnable	$A20	word	menuBar enable for desk acc's that own the menu bar
	CurDeKind	$A22	word	window kind of deactivated window
	MenuFlash	$A24	word	flash feedback count
	TheMenu	$A26	word	resource ID of hilited menu
	SavedHandle	$A28	long	handle to data under a menu
•	MrMacHook	$A2C	long	Mr. Macintosh hook
	MBarHook	$A2C	long	ptr to MenuSelect hook called before menu is drawn

MenuHook	$A30	long	ptr to user hook called during MenuSelect
DragPattern	$A34	8	pattern used to draw outlines of dragged regions
DeskPattern	$A3C	8	pattern used for the desktop
DragFlag	$A44	word	implicit parameter to DragControl
CurDragAction	$A46	long	ptr to implicit actionProc for dragControl
FPState	$A4A	6	floating point state
TopMapHndl	$A50	long	handle to map of most recently opened resource file
SysMapHndl	$A54	long	handle to map of System resourc file
SysMap	$A58	word	reference number of System resource file
CurMap	$A5A	word	reference number of current resource file
ResReadOnly	$A5C	word	Read-only flag
ResLoad	$A5E	word	Auto-load feature
ResErr	$A60	word	Resource Manager error code
TaskLock	$A62	byte	re-entering SystemTask
FScaleDisable	$A63	byte	disable font scaling?
CurActivate	$A64	long	ptr to window slated for activate event
CurDeactive	$A68	long	ptr to window slated for deactivate event
DeskHook	$A6C	long	ptr to hook for painting the desk
TEDoText	$A70	long	ptr to textEdit doText proc hook
TERecal	$A74	long	ptr to textEdit recalText proc hook
• MicroSoft	$A78	12	ApplScratch - for Seattle font
ApplScratch	$A78	12	application scratch area
GhostWindow	$A84	long	ptr to window never to be considered frontmost
CloseOrnHook	$A88	long	ptr to hook for closing desk ornaments
ResumeProc	$A8C	long	ptr to Resume procedure (System error dialog)
SaveProc	$A90	long	address of Save failsafe procedure
SaveSP	$A94	long	Safe stack ptr for restart or save
ANumber	$A98	word	resID of last alert
ACount	$A9A	word	number of times last alert was called (0 through 3)
DABeeper	$A9C	long	ptr to current beep routine
DAStrings	$AA0	16	paramText substitution strings (4 handles)
TEScrpLengt	$AB0	long	textEdit Scrap Length
TEScrpHandl	$AB4	long	handle to textEdit Scrap
AppPacks	$AB8	32	Handles to PACK resources (ID's from 0 to 7)
SysResName	$AD8	20	name of system resource file (STRING[19])
AppParmHandle	$AEC	long	handle to hold application parameters
DSErrCode	$AF0	word	last (or current) system error alert ID
ResErrProc	$AF2	long	ptr to Resource Manager error procedure
TEWdBreak	$AF6	long	ptr to default word break routine
DlgFont	$AFA	word	current font number for dialogs and alerts
LastTGLobal	$AFC	long	address of last global

The contents of memory locations from $B00 on up depend on the ROM version installed.

• 64K ROM Macs: these Macs have a single trap dispatch table located at $400 through $7FF. The system heap begins at $B00, leaving no room for additional globals.

• 128K and later ROM: the original trap dispatch table at $400 through $7FF was converted for use by the Operating System, and a second trap dispatch table for Toolbox traps was created at $C00. Its length is 2K on the Mac Plus and SE, and 4K on the SE/30 and II series. The intervening space from $B00 through $BFF was reserved for new global variables. Note that some globals were defined within the trap table, with addresses above $C00. While I do not understand why Apple did this, the effect is that a few table entries have been pre-empted, reducing the table's capacity. As explained in the Trap Dispatcher section below, each A-trap expects a routine address to be stored in the trap table. Since some of these locations were used for globals, the affected A-traps cannot be used (more on this later).

• HeapStart	$B00	-	start of the System Heap on 64K ROM Macs
TrapAgain	$B00	long	use 4 bytes here for another trap
[????]	$B04	word	<unknown location>

ROMMapHndl	$B06	long	handle to ROM resource map
PWMBuf1	$B0A	long	ptr to PWM buffer
BootMask	$B0E	word	needed during boot
WidthPtr	$B10	long	ptr to global width table
AtalkHk1	$B14	long	ptr to Appletalk hook 1
AtalkHk2	$B18	long	ptr to Appletalk hook 2
[????]	$B1C	long	<unknown location>
[????]	$B20	word	<unknown location>
HWCfgFlags	$B22	word	hardware configuration flags (two names for this global)
SCSIFlag	$B22	word	SCSI configuration word (bit 15=1 if SCSI installed)
[????]	$B24	6	<unknown locations>
WidthTabHandle	$B2A	long	handle to global width table
[????]	$B2E	6	<unknown locations>
BtDskRfn	$B34	word	boot drive driver reference number
BootTmp8	$B36	8	temporary space needed by StartBoot
[????]	$B3E	byte	<unknown location>
T1Arbitrate	$B3F	byte	holds $FF if Timer T1 up for grabs
[????]	$B40	20	<unknown locations>
MenuDisable	$B54	long	resID and menuItem of last chosen menu item
[????]	$B58	40	<unknown locations>
<switched vars>	$B80	-	switched variables (128 bytes)
RMGRHiVars	$B80	-	RMGR variables (32 bytes)
[????]	$B80	14	<unknown locations>
RomMapInsert	$B9E	byte	flag: insert map to the ROM resources
TmpResLoad	$B9F	byte	temp SetResLoad state for calls using ROMMapInsert
IntlSpec	$BA0	long	international software installed if not -1
[????]	$BA4	word	<unknown location>
SysFontFam	$BA6	word	if nonzero, the font # for system font
SysFontSize	$BA8	word	if nonzero, the system font size
MBarHeight	$BAA	word	pixel height of menu bar
[????]	$BAC	long	<unknown location>
NewUnused	$BC0	word	formerly FlEvtMask
LastFOND	$BC2	long	handle to last family record used
[????]	$BC4	48	<unknown locations>
FractEnable	$BF4	byte	enables fractional widths if not zero
[????]	$BF5	byte	<unknown location>
[????]	$BF6	10	<unknown locations>
ToolDispTable	$C00	2048	Toolbox trap dispatch table (Mac Plus & SE)
ToolDispTable	$C00	4096	Toolbox trap dispatch table (Mac II series and SE/30)
HeapStart	$1400	-	start of system heap (Mac Plus & SE)
HeapStart	$1C00	-	start of system heap (Mac II series and SE/30)

Locations defined within the trap dispatch tables:

JADBProc	$6B8	long	ptr to ADBReInit pre-/post-processing routine
MMU32Bit	$CB2	byte	current address mode
TheGDevice	$CC8	long	handle to current active device
AuxWinHead	$CD0	long	auxiliary window list header
JVBLTask	$D28	long	jump vector to DoVBLTask routine
SynListHandle	$D32	long	handle to synthetic font list
MenuCInfo	$D50	[??]	menu color info table header
DTQueue	$D92	10	deferred task queue header
JDTInstall	$D9C	long	jump vector to DTInstall routine
HiliteRGB	$DA0	[??]	default hilighting color

These variables pre-empt the following A-traps (in same order as lost above): A0AE (unassigned), A82C (Pack10), A832 (unassigned), A834 (SetFScaleDisable), A94A (SetMFlash), A954 (NewControl), A964 through A968 (SetMinCtl, SetMaxCtl, TestControl, DragControl, TrackControl). Note that many of these traps are assigned to ROM calls that have existed since 1984. After checking these locations myself, they appear to be set up for these traps, not as global variables. **Until** I can verify the accuracy of my source list, treat this group of variables with **extreme caution**.

PART 2A

MC68000-series Microprocessor Exception Errors

$00	reset: initial stack pointer
$04	reset: initial program counter
$08	bus error
$0C	address error
$10	illegal instruction
$14	zero divide
$18	CHK bounds check failed
$1C	TRAPV overflow trap
$20	priveleged instruction
$24	trace mode
$28	$Axxx Trap dispatcher
$2C	$Fxxx coprocessor instructions
$30-3B	[reserved by Motorola]
$3C	uninitialized interrupt
$40-5F	[reserved by Motorola]
$60	spurious interrupt
$64	level 1 interrupt - VIA (SY6522)
$68	level 2 interrupt - SCC (Z8530)
$6C	level 3 interrupt - VIA and SCC together
$70	level 4 interrupt - debug switch interrupt
$74	level 5 interrupt - debug switch and VIA
$78	level 6 interrupt - debug switch and SCC
$7C	level 7 interrupt - debug switch, VIA, SCC
$80-BF	TRAP instruction vectors
$C0-FF	[reserved by Motorola]
$100-3FF	user interrupt vectors (Mac global vars)

PART 2B

Macintosh™ System Errors

NOTE: Certain MC68000 exceptions listed in part A will generate equivalent system errors.

1	bus (illegal hardware address)
2	address
	PACK 2 not found (Disk Initialization)
3	illegal instruction
4	zero divide
5	CHK bounds check failed
6	TRAPV overflow trap
7	privileged instruction
8	trace mode
9	line 1010 trap (A-trap)
10	line 1111 trap (F-trap)
11	miscellaneous hardware exception
12	unimplemented core routine (bad trap)
13	uninstalled interrupt (debug switch)
14	IO core
	"Welcome to Macintosh" alert
15	segment loader: no such CODE resource
16	floating point
17	PACK 0 not found (List Manager)
18	PACK 1 not found (reserved)
	19
20	PACK 3 not found (Standard File)
21	PACK 4 not found (Floating-point)
22	PACK 5 not found (Transcendental)
23	PACK 6 not found (International Utils)
24	PACK 7 not found (Binary/Decimal)
25	out of memory
26	segment loader: can't start application
27	"System" resource file map was clobbered
28	stack overran the application heap
30	"Please insert the disk: " alert
31	wrong disk (switch disk alert)
	40
41	"Can't load the Finder" alert
42	"You may turn off your Macintosh" alert
32767	General system error

The following system errors are reported under all ROM versions except the original 64K ROM:

32	Memory Manager: current zone is inconsistent
33	Memory Manager: value of ZcbFree is negative
51	Slot Manager: unserviceable slot interrupt
81	SANE: bad opcode passed to FP68K routine
84	Menu Manager: menu resource has been purged
85	Menu Manager: can't find menu bar
86	Menu Manager: hierarchical menu error
99	HFS stack overflowed

PART 2C

Toolbox and Operating System Errors

The errors in this section are generated by the various Mac system routines, either at the Operating System level or at the higher Toolbox level. These errors are then passed to the program (application, DA, INIT, etc.) which called the system routine. It is up to the program to handle these errors.

Printing Manager Error

128	iPrAbort	Printing Manager: user aborted printing

Slot Manager Initialization Errors

14	sdmPriInitErr	initialize error: installed cards
13	sdmPRAMInitErr	initialize error: Slot PRAM
12	sdmSRTInitErr	initialize error: Slot Resource Table
11	sdmInitErr	initialize error: SDM
10	sdmJTInitErr	initialize error: SDM Jump Table
3	siInitSPTblErr	initialize error: slot priority table
2	siInitVBLQsErr	initialize error: slot VBL queues
1	siInitSDTblErr	initialize error: slot interrupt dispatch table

SCSI Manager Errors

10	scComplPhaseErr	SCSIComplete failed; bus not in status phase
9	scBusTOErr	bus timed out before data was ready for SCSI "blind" operations SCSIRBlind and SCSIWBlind
8	scSequenceErr	current operation was started out of proper sequence
7	scMgrBusyErr	SCSI Manager busy when SCSIGet was called
6	scCompareErr	SCSI Manager busy when SCSIGet was called
5	scPhaseErr	bus in wrong phase for attempted operation
4	scBadparmsErr	bad parameter or TIB opcode
3	scArbNBErr	arbitration failed during SCSIGet; bus busy
2	scCommErr	communications error (operations timeout)

Miscellaneous Errors

1	evtNotEnb	event type not defined in system event mask
0	noErr	the operation was completed normally
-1	qErr	queue element not found during deletion
-1	iPrSavPFil	Printing Manager: saving spool file
-2	vTypErr	invalid queue element
-3	corErr	[obsolete] core routine number out of range
-4	unimpErr	[obsolete] unimplemented core routine
-8	seNoDB	no debugger installed to handle debugger traps

Color Manager Errors

-9	iTabPurgErr	[no description available] generated by: Color2Index/ITabMatch
-10	noColMatch	[no description available] generated by: Color2Index/ITabMatch
-11	qAllocErr	[no description available] generated by: MakeITable
-12	tblAllocErr	[no description available] generated by: MakeITable
-13	overRun	[no description available] generated by: MakeITable
-14	noRoomErr	[no description available] generated by: MakeITable
-15	seOutOfRange	[no description available] generated by: SetEntry

-16	seProtErr	[no description available] generated by: SetEntry
-17	i2CRangeErr	[no description available] generated by: SetEntry
-18	gdBadDev	[no description] generated by: SetEntry
-19	reRangeErr	[no description] generated by: SetEntry
-20	seInvRequest	[no description] generated by: SetEntry
-21	seNoMemErr	[no description] generated by: SetEntry

Device Manager Errors

-17	controlErr	driver can't handle control calls
-18	statusErr	driver can't handle status calls
-19	readErr	driver can't handle read calls
-20	writErr	driver can't handle write calls
-21	badUnitErr	driver refnum isn't in unit table
-22	unitEmptyErr	driver refnum unit table entry is an empty (zero) handle
-23	openErr	R/W permission conflicts with open permission
-23	openErr	couldn't open RAM serial driver (64K ROM)
-24	closeErr	[obsolete]
-25	dRemovErr	tried to remove an open driver
-26	dInstErr	DrvrInstall couldn't find driver in resource file
-27	abortErr	IO call aborted by KillIO
-27	iIOAbort	Printing Manager: I/O abort error
-28	notOpenErr	driver not open
-29	unitTblFullErr	unit table full
-30	dceExtErr	dce extension error

File System Errors

-33	dirFulErr	MFS directory full
-34	dskFulErr	disk full
-35	nsvErr	no such volume
-36	ioErr	I/O error
-37	bdNamErr	bad name (name is zero-length or contains a colon)
-38	fnOpnErr	file not open
-39	eofErr	end of file reached during a read
-40	posErr	tried to position before start of file
-41	memErr	[obsolete] memory full on open
-42	tmfoErr	too many files open
-43	fnfErr	file not found
-44	wPrErr	disk is locked (hardware)
-45	fLckdErr	file is locked
-46	vLckdErr	disk is locked (software)
-47	fBsyErr	file busy (delete); one or more files are open
-48	dupFNErr	file with same name & version# already exists
-49	opWrErr	file already open with write permission
-50	paramErr	error in user parameter list (plus other errors) File Manager: no such disk (and no default disk exists) Disk Driver: bad positioning information Disk Init Package: bad drive number
-51	rfNumErr	refnum specifies a nonexistent access path
-52	gfpErr	error during Get file position call (GetFPos)
-53	volOffLinErr	volume not online (was ejected)
-54	permErr	tried to open a locked file for writing
-55	volOnLinErr	volume already in drive
-56	nsDrvErr	no such drive (specified drive number not in drive queue)
-57	noMacDskErr	not a Mac diskette; disk doesn't have Mac directory

-58	extFSErr	volume belongs to an external file system
-59	fsRnErr	rename ran into problems
-60	badMDBErr	bad master directory block - reinit the disk!
-61	wrPermErr	access path doesn't allow writing

Font Manager Errors

-64	fontDecError	error during font declaration
-65	fontNotDeclared	font not declared
-66	fontSubErr	font substitution occured

Disk Errors (range: -64 through -84 inclusive)

-64	lastDskErr	last of the low-level disk errors (in reverse order)
-64	noDriveErr	drive not installed/connected
-65	offLinErr	operation requested for an offline disk
-66	noNybErr	couldn't find 5 nibbles in 200 tries (blank disk)
-67	noAdrMkErr	couldn't find address mark
-68	dataVerErr	read-verify compare failed
-69	badCksmErr	address mark checksum didn't check
-70	badBtSlpErr	bad address mark bit slip nibbles
-71	noDtaMkErr	couldn't find a data mark header
-72	badDCksum	bad data mark checksum
-73	badDBtSlp	bad data mark bit slip nibbles
-74	wrUnderrun	write underrun occurred
-75	cantStepErr	step handshake failed (drive fault)
-76	tk0BadErr	can't find track 0
-77	initIWMErr	unable to initialize IWM (disk controller chip)
-78	twoSideErr	tried to read 2nd side on a single sided drive
-79	spdAdjErr	unable to correctly adjust 400K drive speed
-80	seekErr	track number bad on address mark (drive fault)
-81	sectNFErr	sector not found on track
-82	fmt1Err	can't find sector 0 after track format
-83	fmt2Err	can't get enough sync
-84	verErr	track failed to verify
-84	firstDskErr	first of the low-level disk errors (in reverse order)

Clock Chip Errors

-85	clkRdErr	unable to read clock
-86	clkWrErr	time written did not verify
-87	prWrErr	Parameter RAM written didn't verify
-88	prInitErr	InitUtil found the PRAM uninitialized (status not $A8)

Device Driver Errors

-89	rcvrErr	[obsolete] SCC receiver error (framing, parity, OR)
-90	breakRecd	[obsolete] Break received (SCC)

AppleTalk Errors

-91	ddpSktErr	DDP socket error: socket already active; not a known socket; socket table full; all dynamic socket numbers in use
-92	ddpLenErr	DDP datagram or ALAP data too long
-93	noBridgeErr	no such bridge
-94	lapProtErr	ALAP protocol errors, attach or detach
-95	excessCollsns	ALAP too many (over 32) collisions or line sensed in use

-97	portInUse	driver open error: port is already in use
-98	portNotCf	driver open error: port not configured for this connection

Memory Manager Errors (not available on 64K ROM)

-99	memROZErr	hard error in read-only zone
-99	memROZWarn	soft error in read-only zone

Scrap Manager Errors

-100	noScrapErr	no scrap exists
-102	noTypeErr	no data of that type in scrap

Memory Manager Errors

-108	memFullErr	not enough room in heap zone
-109	nilHandleErr	NIL (zero) master pointer (handle is empty)
-110	memAdrErr	[obsolete] address was odd or out of range
-111	memWZErr	attempt to operate on a free block
-112	memPurErr	attempt to purge a locked or non-purgable block
-113	memAZErr	[obsolete] address in zone check failed
-114	memPCErr	[obsolete] pointer check failed
-115	memBCErr	[obsolete] block check failed
-116	memSCErr	[obsolete] size check failed
-117	memLockedErr	block is locked

File System Errors (not available on 64K ROM)

-120	dirNFErr	directory not found
-121	tmwdoErr	too many working directories open
-122	badMovErr	tried to move into offspring
-123	wrgVolTypErr	tried to do an HFS operation on a nonHFS volume
-124	volGoneErr	Server volume has been disconnected
-126	mBarNFnd	Menu Manager: MBDF not found
-127	hMenuFindErr	could not find Hierarchical menu's parent (MenuKey)
-127	fsDSIntErr	internal file system fault

Color Quickdraw & Color Manager Errors

-150	cMatchErr	Color2Index failed to find an index
-151	cTempMemErr	failed to allocate memory for temporary structures
-152	cNoMemErr	failed to allocate memory for structure
-153	cRangeErr	range error on colorTable request
-154	cProtectErr	colorTable entry protection violation
-155	cDevErr	invalid graphics device type
-156	cResErr	invalid resolution for MakeITable

Resource Manager Errors

-192	resNotFound	resource not found
-193	resFNotFound	resource file not found
-194	addResFailed	AddResource failed
-195	addRefFailed	[obsolete] AddReference failed
-196	rmvResFailed	RmveResource failed
-197	rmvRefFailed	[obsolete] RmveReference failed

Resource Manager Errors (not available on 64K ROM)

-198	resAttrErr	attribute prohibits the operation
-199	mapReadErr	resource map is garbled

Sound Manager Errors

-200	noHardware	supporting hardware for selected synthesizer doesn't exist
-201	notEnoughHardware	no more channels for selected synthesizer
-203	queueFull	queue is full
-204	resProblem	problems encountered while loading resource
-205	badChannel	bad channel queue length
-206	badFormat	bad handle to 'snd ' resource

Slot Manager Errors

-290	smSDMInitErr	SDM could not be initialized.
-290	smSRTInitErr	Slot Resource Table could not be initialized.
-290	smPRAMInitErr	Slot Resource Table could not be initialized.
-290	smPriInitErr	Cards could not be initialized.
-300	smEmptySlot	No card in slot
-301	smCRCFail	CRC check failed for declaration data
-302	smFormatErr	bad FHeader format in declaration ROM
-303	smRevisionErr	bad revison number in declaration ROM
-304	smNoDir	Directory offset is zero
-305	smLWTstBad	Long Word Test field was not $5A932BC7.
-306	smNosInfoArray	SDM couldn't obtain memory for the sInfo array
-307	smResrvErr	Reserved field not zero (fatal error)
-308	smUnExBusErr	Unexpected bus error occurred
-309	smBLFieldBad	bad ByteLanes field
-310	smFHBlockRdErr	FHeader block couldn't be read
-311	smFHBlkDispErr	FHeader block couldn't be deleted (disposed of)
-312	smDisposePErr	_DisposPointer error
-313	smNoBoardsRsrc	No Board sResource.
-314	smGetPRErr	Error occured during _sGetPRAMRec (See SIMStatus)
-315	smNoBoardId	No Board ID
-316	smIntStatVErr	The InitStatus_V field was negative after primary init
-317	smIntTblVErr	Slot Resource Table initialization failed
-318	smNoJmpTbl	SDM jump table could not be created
-319	smBadBoardId	bad Board ID, re-init the PRAM record
-320	smBusErrTO	BusError timeout
-330	smBadRefId	Reference ID not found in List
-331	smBadsList	ID's in sList aren't in ascending order
-332	smReservedErr	Reserved field not zero
-333	smCodeRevErr	Code revision is wrong (sExec)
-334	smCPUErr	CPU field is wrong (sExec)
-335	smsPointerNil	sPointer is zero: no sList specified
-336	smNilsBlockErr	Nil (zero-length) sBlock error
-337	smSlotOOBErr	Slot out of bounds error
-338	smSelOOBErr	Selector out of bounds error
-339	smNewPErr	_NewPointer error
-340	smBlkMoveErr	_BlockMove error
-341	smCkStatusErr	bad slot status (InitStatus_A,V)
-342	smGetDrvrNamErr	Error occured during _sGetDrvrName.
-343	smDisDrvrNamErr	Error occured during _sDisDrvrName.
-344	smNoMoresRsrcs	No more sResources
-345	smsGetDrvrErr	Error occurred during _sGetDrvr
-346	smBadsPtrErr	Bad sPointer was passed to an SDM routine
-347	smByteLanesErr	bad byteLanes value was passed to an SDM routine
-348	smOffsetErr	Offset was too big

-349	smNoGoodOpens	No opens were successful in the loop.
-350	smSRTOvrFlErr	sResource table overflowed
-351	smRecNotFnd	Record not found in the sResource table

Slot Manager Error

-360	slotNumErr	bad slot number

AppleTalk Errors

-1024	nbpBuffOvr	NBP buffer overflow
-1025	nbpNoConfirm	NBP name not confirmed
-1026	nbpConfDiff	NBP name confirmed for different socket
-1027	nbpDuplicate	NBP duplicate name found
-1028	nbpNotFound	NBP name does not exist
-1029	nbpNISErr	NBP names information socket error
-1066	aspBadVersNum	Server cannot support this ASP version
-1067	aspBufTooSmall	Buffer too small
-1068	aspNoMoreSess	No more sessions on server
-1069	aspNoServers	No servers at that address
-1070	aspParamErr	Parameter error
-1071	aspServerBusy	Server cannot open another session
-1073	aspSessClosed	Session closed
-1073	aspSizeErr	Command block too big
-1074	aspTooMany	Too many clients (server error)
-1075	aspNoAck	No ACK on attention request (server err)
-1096	reqFailed	ATPSndRequest failed: retry count exhausted
-1097	tooManyReqs	ATP too many concurrent requests
-1098	tooManySkts	ATP too many responding sockets
-1099	badATPSkt	ATP bad responding socket
-1100	badBuffNum	ATP bad sequence number
-1101	noRelErr	ATP no release received
-1102	cbNotFound	ATP control block does not exist
-1103	noSendResp	ATPAddRsp issued before ATPSndRsp
-1104	noDataArea	too many active ATP calls
-1105	reqAborted	request was aborted

Application Errors (range: -2000 through -2999 inclusive)

reserved for use by the current application

AppleTalk Errors

-3101	buf2SmallErr	DDP datagram or ALAP frame too big for buffer
-3102	noMPPErr	MPP driver not installed
-3103	cksumErr	DDP bad checksum
-3104	extractErr	NBP tuple doesn't exist in buffer
-3105	readQErr	socket or protocol type bad or doesn't exist in table
-3106	atpLenErr	ATP response message too long
-3107	atpBadRsp	bad response from ATPRequest
-3108	recNotFnd	ABRecord doesn't exist
-3109	sktClosedErr	async call aborted before completion: socket was closed

Printing Manager Errors (occur with Laserwriters)

-4096	[no name declared]	no free Connect Control Blocks available
-4097	[no name declared]	bad connection refNum
-4098	[no name declared]	request already active

-4099	[no name declared]	write request too big
-4100	[no name declared]	connection just closed
-4101	[no name declared]	printer closed or does not exist

AppleTalk Errors

-5000	afpAccessDenied	[no description available]
-5001	afpAuthContinue	[no description available]
-5002	afpBadUAM	[no description available]
-5003	afpBadVersNum	[no description available]
-5004	afpBitmapErr	[no description available]
-5005	afpCantMove	[no description available]
-5006	afpDenyConflict	[no description available]
-5007	afpDirNotEmpty	[no description available]
-5008	afpDiskFull	[no description available]
-5009	afpEofError	[no description available]
-5010	afpFileBusy	[no description available]
-5011	afpFlatVol	[no description available]
-5012	afpItemNotFound	[no description available]
-5013	afpLockErr	[no description available]
-5014	afpMiscErr	[no description available]
-5015	afpNoMoreLocks	[no description available]
-5016	afpNoServer	[no description available]
-5017	afpObjectExists	[no description available]
-5018	afpObjectNotFound	[no description available]
-5019	afpParmErr	[no description available]
-5020	afpRangeNotLocked	[no description available]
-5021	afpRangeOverlap	[no description available]
-5022	afpSessClosed	[no description available]
-5023	afpUserNotAuth	[no description available]
-5024	afpCallNotSupported	[no description available]
-5025	afpObjectTypeErr	[no description available]
-5026	afpTooManyFilesOpen	[no description available]
-5027	afpServerGoingDown	[no description available]
-5028	afpCantRename	[no description available]
-5029	afpDirNotFound	[no description available]
-5030	afpIconTypeError	[no description available]

_SysEnvirons Errors

-5500	envNotPresent	returned by glue - _SysEnvirons trap does not exist
-5501	envBadVers	Version non-positive; no information was returned
-5502	envVersTooBig	Version bigger than _SysEnvirons routine can handle

PART 3A

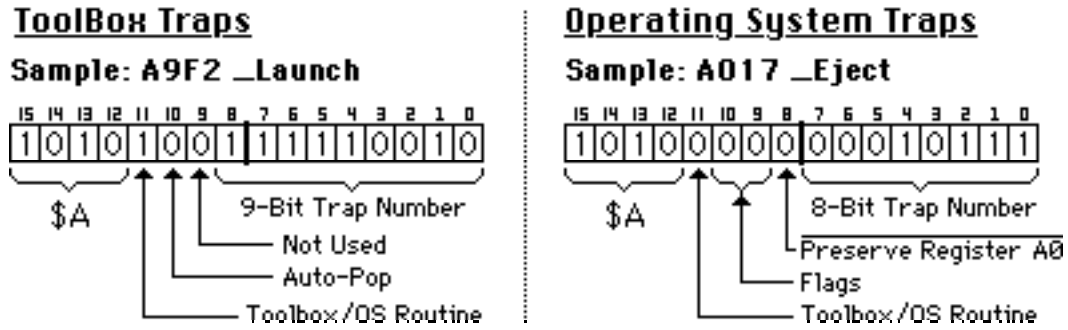
Macintosh™ Trap Dispatcher

The Trap Dispatcher is the part of the Macintosh Operating System that routes system calls to the actual routines. The Trap Dispatcher has been revised two times since its initial release, totaling 3 versions. The earliest version was part of the original 64K ROM shipped with the 128K and 512K Macs. The first revision appeared in the Mac Plus 128K ROM and the second (and last) revision appeared in all later Mac models.

The Trap Dispatcher takes advantage of a special feature of the 68000 microprocessor called *unimplemented instructions* . These instructions, like all other 68000 instructions, are 16 bits long (2 bytes) and can be represented by four hex digits. Unimplemented instructions always begin with the hex digits A or F and can contain anything in the last 3 digits. Whenever the 68000 gets such an instruction, it automatically performs a *trap* . The 68000 fetches an address from a memory location predetermined by Motorola and then jumps to that address. The Mac ROM startup code automatically stores the starting location of the Trap Dispatcher into this specific memory location when the system is powered up or rebooted (the Trap Dispatcher itself is also located in the ROM). Unimplemented instructions are called A-traps or F-traps, depending on the first hex digit of the instruction.

Motorola reserved all F-traps for math coprocessor instructions, and allowed the OEM, i.e. Apple, free use of A-traps. Apple in turn built the whole Macintosh Operating System around A-traps. All system calls were numbered and were split into two broad categories, called Toolbox Traps and Operating System (OS) Traps. Each system call was issued its own A-trap, or trap word, containing its own trap (system call) number. Apple built the Trap Dispatcher to handle A-traps on the Macintosh.

Since any A-trap causes the 68000 to invoke the Trap Dispatcher, the Trap Dispatcher must decode the last 3 digits of the A-trap to figure out which system call to execute. The diagram below shows how the Trap Dispatcher decodes A-traps:



Trap word bits are defined as follows.

Bits 15 through 12 are set to the binary pattern 1010 (hexdecimal A) for all traps.

Bit 11 determines the trap's category: 1 for Toolbox traps and 0 for Operating System traps.

For OS traps, **bits 10 and 9** are used for flags, whose meanings depend on the routine being called. For Toolbox traps, bit 10 is the auto-pop bit. If it is set, the Trap Dispatcher will remove, or "pop", the return address from the top of the stack and throw it out. It will then pop the stack again and use that address as the return address. This feature supports languages such as Lisa Pascal that always JSR'd to a table of trap words (instead of inserting the trap words inline with the code).

Bit 9 is a reserved bit in Toolbox trap words in the 64K and 128K ROMs. Apple added the bit to the trap number on all later ROMs.

Bit 8 is used by OS traps to preserve the 68000's register A0 across a system call. If the bit is 0, register A0 is saved before the system call begins execution and restored after it finishes. If the bit is 1, register A0 is not saved and restored (allows OS routines that return a value in A0 to work).

Bits 7 through 0 comprise all or part of the trap number for all traps.

PART 3B

Macintosh™ Traps

With the exception of the original 64K ROM found in 128K and 512K Macintoshes, all Mac ROM sets are based on the 128K ROM found in the Mac 512KE and Plus. The ROMs in all later Mac models contain some additions not in the 128K ROM. These additions are marked by asterisk (*) to the left of their name. Apple has tried to retrofit these additional traps into older Macs (that don't have them in ROM) by using the System software to install them in RAM, one reason newer systems are so huge. However, Apple excluded certain goups of traps from the retrofit process. A good example is Color QuickDraw, which is available only on Macs that have it in ROM (SE/30 and II series).

While Apple has made many additions to the ROM trap set, only two traps have been removed. Existing only in the original 64K ROM, the obscure Resource Manager traps AddReference and RemoveReference never found much use, leading Apple to declare them "obsolete" and not include them in all later ROMs. These routines haven't been supported since 1985 and should not be used (in the list these two routines are marked with a black dot "•"). In fact, they were described only in the 3-ring binder and phone book editions of Inside Macintosh and weren't included in the final release.

All ROM traps are given in 68000 assembly language format. Traps in boldface are bundles of several subroutines. Listings of the subroutines follow the general list. Be aware that some trap names differ from their high-level language counterparts. The reason: the original Lisa assembler limited trap names to 8 significant letters (including the initial underscore), and would thus see the two names _UnloadSeg and _UnloadScrap as representing the same trap. Changing the spelling of one of the two traps resolves the name conflict (thus, _UnloadSeg becomes _UnlodeScrap). In the MDS and MPW assemblers, Apple increased the character limit to 31. While the new limit would eliminate all conflicts, Apple made the misspelled names permanent.

For a detailed description of the trap word mechanism, see Part 3A.

	Trap name.....	Trap		Trap name.....	Trap		Trap name.....	Trap
*	_ADBOp	$A07C	•	_AddReference	$A9AC		_Allocate	$A010
*	_ADBReInit	$A07B		_AddResMenu	$A94D	*	_AllocCursor	$AA1D
*	_AddComp	$AA3B		_AddResource	$A9AB		_AngleFromSlope	$A8C4
	_AddDrive	$A04E	*	_AddSearch	$AA3A		_AppendMenu	$A933
	_AddPt	$A87E		_Alert	$A985	*	_AttachVBL	$A071
	_BackColor	$A863		_BitClr	$A85F		_BitTst	$A85D
	_BackPat	$A87C		_BitNot	$A85A		_BitXor	$A859
*	_BackPixPat	$AA0B		_BitOr	$A85B		_BlockMove	$A02E
	_BeginUpdate	$A922		_BitSet	$A85E		_BringToFront	$A920
	_BitAnd	$A858		_BitShift	$A85C		_Button	$A974
*	_CalcCMask	$AA4F	*	_CloseCPort	$A87D	*	_CopyPixMap	$AA05
	_CalcMask	$A838		_CloseDeskAcc	$A9B7	*	_CopyPixPat	$AA09
	_CalcMenuSize	$A948		_CloseDialog	$A982		_CopyRgn	$A8DC
	_CalcVBehind	$A90A		_ClosePgon	$A8CC		_CouldAlert	$A989
	_CalcVis	$A909		_ClosePicture	$A8F4		_CouldDialog	$A979
	_CautionAlert	$A988		_ClosePort	$A87D		_Count1Resources	$A80D
	_Chain	$A9F3		_CloseResFile	$A99A		_Count1Types	$A81C
	_ChangedResource	$A9AA		_CloseRgn	$A8DB	*	_CountADBs	$A077
*	_CharExtra	$AA23		_CloseWindow	$A92D		_CountMItems	$A950
	_CharWidth	$A88D		_CmpString	$A03C		_CountResources	$A99C
	_CheckItem	$A945	*	_Color2Index	$AA33		_CountTypes	$A99E
	_CheckUpdate	$A911		_ColorBit	$A864		_Create	$A008
	_ClearMenuBar	$A934		_CompactMem	$A04C		_CreateResFile	$A9B1

_ClipAbove	$A90B	_Control	$A004	_CurResFile	$A994
_ClipRect	$A87B	_CopyBits	$A8EC		
_Close	$A001	_CopyMask	$A817		

	Trap name.....	Trap		Trap name.....	Trap		Trap name.....	Trap
	_Date2Sec	$A9C7	*	_DisposCIcon	$AA25		_DragWindow	$A925
	_Delay	$A03B		_DisposControl	$A955		_Draw1Control	$A96D
*	_DelComp	$AA4D	*	_DisposCTable	$AA24		_DrawChar	$A883
	_Delete	$A009		_DisposDialog	$A983		_DrawControls	$A969
	_DeleteMenu	$A936	*	_DisposGDevice	$AA30		_DrawDialog	$A981
*	_DelMCEntries	$AA60		_DisposHandle	$A023		_DrawGrowIcon	$A904
	_DelMenuItem	$A952		_DisposMenu	$A932		_DrawMenuBar	$A937
*	_DelSearch	$AA4C	*	_DisposPixMap	$AA04		_DrawNew	$A90F
	_DeltaPoint	$A94F	*	_DisposPixPat	$AA08		_DrawPicture	$A8F6
	_Dequeue	$A96E		_DisposPtr	$A01F		_DrawString	$A884
	_DetachResource	$A992		_DisposRgn	$A8D9		_DrawText	$A885
	_DialogSelect	$A980		_DisposWindow	$A914		_DrvrInstall	$A03D
	_DiffRgn	$A8E6	*	_DoVBLTask	$A072		_DrvrRemove	$A03E
	_DisableItem	$A93A		_DragControl	$A967	*	_DTInstall	$A082
*	_DispMCEntries	$AA63		_DragGrayRgn	$A905			
*	_DisposCCursor	$AA26		_DragTheRgn	$A926			
	_Eject	$A017		_Enqueue	$A96F		_EraseRect	$A8A3
	_Elems68K	$A9EC		_EqualPt	$A881		_EraseRgn	$A8D4
	_EmptyHandle	$A02B		_EqualRect	$A8A6		_EraseRoundRect	$A8B2
	_EmptyRect	$A8AE		_EqualRgn	$A8E3		_ErrorSound	$A98C
	_EmptyRgn	$A8E2		_EraseArc	$A8C0		_EventAvail	$A971
	_EnableItem	$A939		_EraseOval	$A8B9		_ExitToShell	$A9F4
	_EndUpdate	$A923		_ErasePoly	$A8C8			
	_FillArc	$A8C2		_Fix2Long	$A840		_Frac2X	$A845
*	_FillCArc	$AA11		_Fix2X	$A843		_FracCos	$A847
*	_FillCOval	$AA0F		_FixAtan2	$A818		_FracDiv	$A84B
*	_FillCPoly	$AA13		_FixDiv	$A84D		_FracMul	$A84A
*	_FillCRect	$AA0E		_FixMul	$A868		_FracSin	$A848
*	_FillCRgn	$AA12		_FixRatio	$A869		_FracSqrt	$A849
*	_FillCRoundRect	$AA10		_FixRound	$A86C		_FrameArc	$A8BE
	_FillOval	$A8BB		_FlashMenuBar	$A94C		_FrameOval	$A8B7
	_FillPoly	$A8CA		_FlushEvents	$A032		_FramePoly	$A8C6
	_FillRect	$A8A5		_FlushFile	$A045		_FrameRect	$A8A1
	_FillRgn	$A8D6		_FlushVol	$A013		_FrameRgn	$A8D2
	_FillRoundRect	$A8B4		_FMSwapFont	$A901		_FrameRoundRect	$A9B0
	_FindControl	$A96C		_FontMetrics	$A835		_FreeAlert	$A98A
	_FindDItem	$A984		_ForeColor	$A862		_FreeDialog	$A97A
	_FindWindow	$A92C		_FP68K	$A9EB		_FreeMem	$A01C
	_Fix2Frac	$A841		_Frac2Fix	$A842		_FrontWindow	$A924
	_Get1IxResource	$A80E	*	_GetGDevice	$AA32		_GetOSEvent	$A031
	_Get1IxType	$A80F		_GetHandleSize	$A025		_GetPattern	$A9B8
	_Get1NamedResource	$A820		_GetIcon	$A9BB		_GetPen	$A89A
	_Get1Resource	$A81F	*	_GetIndADB	$A078		_GetPenState	$A898
*	_GetADBInfo	$A079		_GetIndResource	$A99D		_GetPicture	$A9BC
	_GetAppParms	$A9F5		_GetIndType	$A99F		_GetPixel	$A865
*	_GetAuxCtl	$AA44		_GetItem	$A946	*	_GetPixPat	$AA0C
*	_GetAuxWin	$AA42	*	_GetItemCmd	$A84E		_GetPort	$A874
*	_GetBackColor	$AA1A		_GetIText	$A990		_GetPtrSize	$A021
*	_GetCCursor	$AA1B		_GetItmIcon	$A93F		_GetResAttrs	$A9A6

	Trap name.....	Trap
*	_GetCIcon	$AA1E
	_GetClip	$A87A
*	_GetCPixel	$AA17
	_GetCRefCon	$A95A
*	_GetCTable	$AA18
	_GetCTitle	$A95E
	_GetCtlAction	$A96A
	_GetCtlValue	$A960
	_GetCursor	$A9B9
*	_GetCVariant	$A809
*	_GetCWMgrPort	$AA48
*	_GetDefaultStartup	$A07D
*	_GetDeviceList	$AA29
	_GetDItem	$A98D
	_GetEOF	$A011
	_GetFileInfo	$A00C
	_GetFName	$A8FF
	_GetFNum	$A900
	_GetFontInfo	$A88B
*	_GetForeColor	$AA19
	_GetFPos	$A018
	_HandAndHand	$A9E4
	_HandleZone	$A126
	_HandToHand	$A9E1
	_HClrRBit	$A068
	_HFSDispatch	**$A260**
	_HGetState	$A069
	_HideControl	$A958
	_HideCursor	$A852
*	_Index2Color	$AA34
	_InfoScrap	$A9F9
	_InitAllPacks	$A9E6
	_InitApplZone	$A02C
*	_InitCport	$AA01
	_InitCursor	$A850
	_InitDialogs	$A97B
	_InitFonts	$A8FE
*	_InitGDevice	$AA2E
	_InitGraf	$A86E
	_InitMenus	$A930
	_InitPack	$A9E5
*	_KeyTrans	$A9C3
	_KillControls	$A956
	_Launch	$A9F2
	_Line	$A892
	_LineTo	$A891
	_LoadResource	$A9A2

	Trap name.....	Trap
	_GetItmMark	$A943
	_GetItmStyle	$A941
	_GetKeys	$A976
*	_GetMainDevice	$AA2A
	_GetMaxCtl	$A962
*	_GetMaxDevice	$AA27
*	_GetMCEntry	$AA64
*	_GetMCInfo	$AA61
	_GetMenuBar	$A93B
	_GetMHandle	$A949
	_GetMinCtl	$A961
	_GetMouse	$A972
	_GetNamedResource	$A9A1
	_GetNewControl	$A9BE
*	_GetNewCWindow	$AA46
	_GetNewDialog	$A97C
	_GetNewMBar	$A9C0
	_GetNewWindow	$A9BD
*	_GetNextDevice	$AA2B
	_GetNextEvent	$A970
*	_GetOSDefault	$A084
	_HideDItem	$A827
	_HidePen	$A896
	_HideWindow	$A916
*	_HiliteColor	$AA22
	_HiliteControl	$A95D
	_HiliteMenu	$A938
	_HiliteWindow	$A91C
	_HiWord	$A86A
	_InitPort	$A86D
*	_InitProcMenu	$A808
	_InitQueue	$A016
	_InitResources	$A995
	_InitUtil	$A03F
	_InitWindows	$A912
	_InitZone	$A019
	_InsertMenu	$A935
	_InsertResMenu	$A951
	_InsetRect	$A8A9
	_InsetRgn	$A8E1
	_InsMenuItem	$A826
	_KillIO	$A006
	_KillPicture	$A8F5
	_LoadSeg	$A9F0
	_LocalToGlobal	$A870
	_LodeScrap	$A9FB
	_Long2Fix	$A83F

	Trap name.....	Trap
	_GetResFileAttrs	$A9F6
	_GetResInfo	$A9A8
	_GetResource	$A9A0
	_GetRMenu	$A9BF
	_GetScrap	$A9FD
	_GetString	$A9BA
*	_GetSubTable	$AA37
	_GetTrapAddress	$A146
*	_GetVideoDefault	$A080
	_GetVol	$A014
	_GetVolInfo	$A007
	_GetWindowPic	$A92F
	_GetWMgrPort	$A910
	_GetWRefCon	$A917
	_GetWTitle	$A919
*	_GetWVariant	$A80A
	_GetZone	$A11A
	_GlobalToLocal	$A871
	_GrafDevice	$A872
	_GrowWindow	$A92B
	_HLock	$A029
	_HNoPurge	$A04A
	_HomeResFile	$A9A4
	_HPurge	$A049
	_HSetRBit	$A067
	_HSetState	$A06A
	_HUnlock	$A02A
*	**_InternalWait**	**$A07F**
	_InvalRect	$A928
	_InvalRgn	$A927
	_InverRect	$A8A4
	_InverRgn	$A8D5
	_InverRoundRect	$A8B3
	_InvertArc	$A8C1
*	_InvertColor	$AA35
	_InvertOval	$A8BA
	_InvertPoly	$A8C9
	_IsDialogEvent	$A97F
	_KillPoly	$A8CD
	_LongMul	$A867
	_LoWord	$A86B

	Trap name.....	Trap		Trap name.....	Trap		Trap name.....	Trap
*	_MakeITable	$AA39		_MaxMem	$A11D		_MountVol	$A00F
*	_MakeRGBPat	$AA0D		_MaxSizeRsrc	$A821		_Move	$A894
	_MapPoly	$A8FC		_MeasureText	$A837		_MoveControl	$A959
	_MapPt	$A8F9	*	_MenuChoice	$AA66		_MoveHHi	$A064
	_MapRect	$A8FA		_MenuKey	$A93E		_MovePortTo	$A877
	_MapRgn	$A8FB		_MenuSelect	$A93D		_MoveTo	$A893
	_MaxApplZone	$A063		_ModalDialog	$A991		_MoveWindow	$A91B
	_MaxBlock	$A061		_MoreMasters	$A036		_Munger	$A9E0
*	_NewCDialog	$AA4B	*	_NewGDevice	$AA2F		_NewPtr	$A11E
	_NewControl	$A954		_NewHandle	$A122		_NewRgn	$A8D8
*	_NewCWindow	$AA45		_NewMenu	$A931		_NewString	$A906
	_NewDialog	$A97D	*	_NewPixMap	$AA03		_NewWindow	$A913
	_NewEmptyHandle	$A066	*	_NewPixPat	$AA07		_NoteAlert	$A987
	_ObscureCursor	$A856		_Open	$A000		_OpenResFile	$A997
	_Offline	$A035	*	_OpenCport	$AA00		_OpenRF	$A00A
	_OffsetPoly	$A8CE		_OpenDeskAcc	$A9B6		_OpenRFPerm	$A9C4
	_OffsetRect	$A8A8		_OpenPicture	$A8F3		_OpenRgn	$A8DA
	_OfsetRgn	$A8E0		_OpenPoly	$A8CB		_OSEventAvail	$A030
*	_OpColor	$AA21		_OpenPort	$A86F			
	_Pack0	$A9E7		_PaintArc	$A8BF		_PlotIcon	$A94B
	_Pack1	$A9E8		_PaintBehind	$A90D	*	_PopUpMenuSelect	$A80B
	_Pack2	$A9E9		_PaintOne	$A90C		_PortSize	$A876
	_Pack3	$A9EA		_PaintOval	$A8B8		_PostEvent	$A02F
	_Pack4	$A9EB		_PaintPoly	$A8C7		_PPostEvent	$A12F
	_Pack5	$A9EC		_PaintRect	$A8A2	*	_ProtectEntry	$AA3D
	_Pack6	$A9ED		_PaintRgn	$A8D3		_Pt2Rect	$A8AC
	_Pack7	$A9EE		_PaintRoundRect	$A8B1		_PtInRect	$A8AD
	_Pack8	$A816		_ParamText	$A98B		_PtInRgn	$A8E8
	_Pack9	$A82B		_PenMode	$A89C		_PtrAndHand	$A9EF
	_Pack10	$A82C		_PenNormal	$A89E		_PtrToHand	$A9E3
	_Pack11	$A82D		_PenPat	$A89D		_PtrToXHand	$A9E2
	_Pack12	$A82E	*	_PenPixPat	$AA0A		_PtrZone	$A148
	_Pack13	$A82F		_PenSize	$A89B		_PtToAngle	$A8C3
	_Pack14	$A830		_PicComment	$A8F2		_PurgeMem	$A04D
	_Pack15	$A831		_PinRect	$A94E		_PurgeSpace	$A062
	_PackBits	$A8CF	*	_PlotCIcon	$AA1F		_PutScrap	$A9FE
*	_QDError	$AA40						
	_Random	$A861		_RectRgn	$A8DF	*	_RGBForeColor	$AA14
	_RDrvrInstall	$A04F		_ReleaseResource	$A9A3	*	_RGetResource	$A80C
	_Read	$A002		_RelString	$A050	•	_RmveReference	$A9AE
	_ReadDateTime	$A039		_Rename	$A00B		_RmveResource	$A9AD
*	_RealColor	$AA36		_ResError	$A9AF		_RsrcMapEntry	$A9C5
	_RealFont	$A902	*	_ReserveEntry	$AA3E		_RsrcZoneInit	$A996
	_ReallocHandle	$A027		_ResrvMem	$A040		_RstFilLock	$A042
	_RecoverHandle	$A128	*	_RestoreEntries	$AA4A			
	_RectInRgn	$A8E9	*	_RGBBackColor	$AA15			

	Trap name.....	Trap
*	_SaveEntries	$AA49
	_SaveOld	$A90E
	_ScalePt	$A8F8
*	**_ScriptUtil**	**$A8B5**
	_ScrollRect	$A8EF
	_SCSIDispatch	**$A815**
	_Secs2Date	$A9C6
	_SectRect	$A8AA
	_SectRgn	$A8E4
*	_SeedCFill	$AA50
	_SeedFill	$A839
	_SelectWindow	$A91F
	_SelIText	$A97E
	_SendBehind	$A921
*	_SetADBInfo	$A07A
	_SetAppBase	$A057
	_SetApplLimit	$A02D
*	_SetCCursor	$AA1C
*	_SetClientID	$AA3C
	_SetClip	$A879
*	_SetCPixel	$AA16
*	_SetCPortPix	$AA06
	_SetCRefCon	$A95B
	_SetCTitle	$A95F
	_SetCtlAction	$A96B
*	_SetCtlColor	$AA43
	_SetCtlValue	$A963
	_SetCursor	$A851
	_SetDateTime	$A03A
*	_SetDefaultStartup	$A07E
*	_SetDeskCPat	$AA47
*	_SetDeviceAttribute	$AA2D
	_SetDItem	$A98E
	_SetEmptyRgn	$A8DD
*	_SetEntries	$AA3F
	_SetEOF	$A012
	_SetFileInfo	$A00D
	_SetFilLock	$A041
	_SetFilType	$A043
	_SetFontLock	$A903
	_SetFPos	$A044
	_SetFScaleDisable	$A834
	_SetGrowZone	$A04B

	Trap name.....	Trap
*	_SetGDevice	$AA31
	_SetHandleSize	$A024
	_SetItem	$A947
*	_SetItemCmd	$A84F
	_SetIText	$A98F
	_SetItmIcon	$A940
	_SetItmMark	$A944
	_SetItmStyle	$A942
	_SetMaxCtl	$A965
*	_SetMCEntries	$AA65
*	_SetMCInfo	$AA62
	_SetMenuBar	$A93C
	_SetMFlash	$A94A
	_SetMinCtl	$A964
	_SetOrigin	$A878
*	_SetOSDefault	$A083
	_SetPBits	$A875
	_SetPenState	$A899
	_SetPort	$A873
	_SetPt	$A880
	_SetPtrSize	$A020
	_SetRecRgn	$A8DE
	_SetRect	$A8A7
	_SetResAttrs	$A9A7
	_SetResFileAttrs	$A9F7
	_SetResInfo	$A9A9
	_SetResLoad	$A99B
	_SetResPurge	$A993
	_SetStdProcs	$A8EA
	_SetString	$A907
	_SetTrapAddress	$A047
*	_SetVideoDefault	$A081
	_SetVol	$A015
*	_SetWinColor	$AA41
	_SetWindowPic	$A92E
	_SetWRefCon	$A918
	_SetWTitle	$A91A
	_SetZone	$A01B
	_ShieldCursor	$A855
	_ShowControl	$A957
	_ShowCursor	$A853
	_ShowDItem	$A828
	_ShowHide	$A908

	Trap name.....	Trap
	_ShowPen	$A897
	_ShowWindow	$A915
*	**_Shutdown**	**$A895**
*	_SIntInstall	$A075
*	_SIntRemove	$A076
	_SizeControl	$A95C
	_SizeRsrc	$A9A5
	_SizeWindow	$A91D
	_SlopeFromAngle	$A8BC
*	**_SlotManager**	**$A06E**
*	_SlotVInstall	$A06F
*	_SlotVRemove	$A070
	_SpaceExtra	$A88E
	_StackSpace	$A065
	_Status	$A005
	_StdArc	$A8BD
	_StdBits	$A8EB
	_StdComment	$A8F1
	_StdGetPic	$A8EE
	_StdLine	$A890
	_StdOval	$A8B6
	_StdPoly	$A8C5
	_StdPutPic	$A8F0
	_StdRect	$A8A0
	_StdRgn	$A8D1
	_StdRRect	$A8AF
	_StdText	$A882
	_StdTxMeas	$A8ED
	_StillDown	$A973
	_StopAlert	$A986
	_StringWidth	$A88C
*	_StripAddress	$A055
	_StuffHex	$A866
	_SubPt	$A87F
*	_SwapMMUMode	$A05D
	_SysBeep	$A9C8
	_SysEdit	$A9C2
*	_SysEnvirons	$A090
	_SysError	$A9C9
	_SystemClick	$A9B3
	_SystemEvent	$A9B2
	_SystemMenu	$A9B5
	_SystemTask	$A9B4

	_TEActivate	$A9D8
	_TEAutoView	$A813
	_TECalText	$A9D0
	_TEClick	$A9D4
	_TECopy	$A9D5
	_TECut	$A9D6
	_TEDeactivate	$A9D9
	_TEDelete	$A9D7
	_TEInit	$A9CC
	_TEInsert	$A9DE
	_TEKey	$A9DC
	_TENew	$A9D2
	_TEPaste	$A9DB
	_TEPinScroll	$A812
	_TEScroll	$A9DD
	_TESelView	$A811
*	_TEStyleNew	$A83E
	_TEUpdate	$A9D3
	_TextBox	$A9CE
	_TextFace	$A888
	_TextFont	$A887
	_TextMode	$A889
	_TextSize	$A88A
	_TextWidth	$A886

*	_TEDispatch	$A83D		_TESetJust	$A9DF		_TickCount	$A975
	_TEDispose	$A9CD		_TESetSelect	$A9D1		_TrackBox	$A83B
*	_TEGetOffset	$A83C		_TESetText	$A9CF		_TrackControl	$A968
	Trap name.....	Trap		Trap name.....	Trap		Trap name.....	Trap
	_TEGetText	$A9CB		_TestControl	$A966		_TrackGoAway	$A91E
	_TEIdle	$A9DA	*	_TestDeviceAttribute	$AA2C			
	_UnionRect	$A8AB		_UnlodeScrap	$A9FA		_UpdtDialog	$A978
	_UnionRgn	$A8E5		_UnmountVol	$A00E		_UprString	$A054
	_Unique1ID	$A810		_UnpackBits	$A8D0		_UseResFile	$A998
	_UniqueID	$A9C1		_UpdateResFile	$A999			
	_UnloadSeg	$A9F1		_UpdtControl	$A953			
	_ValidRect	$A92A		_VInstall	$A033			
	_ValidRgn	$A929		_VRemove	$A034			
	_WaitMouseUp	$A977		_WriteParam	$A038			
	_Write	$A003		_WriteResource	$A9B0			
	_X2Fix	$A844		_X2Frac	$A846		_XorRgn	$A8E7
	_ZeroScrap	$A9FC		_ZoomWindow	$A83A			

Below is a list of all system traps which represent multiple routines. These traps decide which routine to execute by looking for a *routine number* either on the stack (stack-based) or in register D0 (register-based). All traps with the exception of InternalWait expect the routine number to be passed as a 16-bit unsigned word. InternalWait looks for a 32-bit routine number. To call an individual routine, the routine number must be either pushed on the stack or placed in register D0 (following the method expected by its parent trap). Note that all Macintosh Packages (PACK) are stack-based.

****

_PACK 0 - $A9E7 **LIST MANAGER** Stack-based

NOTE: First appeared in System 3.2

LActivate	0	LDelRow	36	LNextCell	72
LAddColumn	4	LDispose	40	LRect	76
LAddRow	8	LDoDraw	44	LScroll	80
LAddToCell	12	LDraw	48	LSearch	84
LAutoScroll	16	LFind	52	LSetCell	88
LCellSize	20	LGetCell	56	LSetSelect	92
LClick	24	LGetSelect	60	LSize	96
LClrCell	28	LLastClick	64	LUpdate	100
LDelColumn	32	LNew	68		

****

_PACK 1 - $A9E8 **Reserved**

****

_PACK 2 - $A9E9 **DISK INITIALIZATION** Stack-based

DIBadMount	0	DILoad	2	DIVerify	8
DIFormat	6	DIUnload	4	DIZero	10

****

_PACK 3 - $A9EA **STANDARD FILE** Stack-based

SFGetFile	2	SFPPutFile	3
SFPGetFile	4	SFPutFile	1

****

****

_PACK 4 - $A9EB **FLOATING-POINT MATH** Stack-based

NOTE: The Standard Apple Numerics Environment (SANE) declares a second name for this trap: _FP68K. SANE also requires the data type of routine parameters; this is accomplished by setting certain high bits of the routine number (and altering the 16-bit value). The numbers listed below, with all high bits clear, will work for 80-bit extended-precision floating- point parameters. See the Apple Numerics Manual for more information.

FOABS	15	FOLOGB	26	FOGETENV	3
FOADD	0	FOMUL	4	FOSETHV	5
FOB2D	11	FONEG	13	FOSETXCP	21
FOCLASS	28	FONEXT	19	FOSQRT	18
FOCMP	8	FOPROCENTRY	23	FOSUB	2
FOCPX	10	FOPROCEXIT	25	FOTESTXCP	27
FOCPYSGN	17	FOREM	12	FOTTI	22
FOD2B	9	FORTI	20	FOX2Z	16
FODIV	6	FOSCALB	24	FOZ2X	14
FOGETHV	7	FOSETENV	1		

****

_PACK 5 - $A9EC **TRANSCENDENTAL FUNCTIONS** Stack-based

NOTE: SANE declares a second name for this trap: _Elems68K. Also, SANE treats these routines in the same manner as FP68K / Pack4. See the Apple Numerics Manual for more information.

FOANNUITY	49174	FOEXP1X	12	FORANDX	32
FOATANX	30	FOEXP21X	14	FOSINX	24
FOCOMPOUND	49172	FOLNX	0	FOTANX	28
FOCOSX	26	FOLN1X	4	FOXPWRI	32784
FOEXPX	8	FOLOG21X	6	FOXPWRY	32786
FOEXP2X	10	FOLOG2X	2		

****

_PACK 6 - $A9ED **INTERNATIONAL UTILITIES** Stack-based

IUDatePString	14	IUMagIDString	12	IUSetIntl	8
IUDateString	0	IUMagString	10	IUTimePString	16
IUGetIntl	6	IUMetric	4	IUTimeString	2

****

_PACK 7 = $A9EE **BINARY/DECIMAL CONVERSION** Stack-based

NOTE: SANE declares a second name for this trap: _DecStr68K, and uses routines 2, 3, and 4. No high bits are required by SANE for these routines, unlike packages 4 and 5.

CStr2Dec	4	__NumToString	0	StringToNum	1
Dec2Str	3	PStr2Dec	2		

****

_HFSDispatch = $A260 Register-based

CloseWD	2	GetFCBInfo	8	SetCatInfo	10
CatMove	5	GetWDInfo	7	SetVolInfo	11
DirCreate	6	LockRng	16	UnlockRng	17
GetCatInfo	9	OpenWD	1		

****

_SCSIDispatch = $A815 Stack-based

SCSICmd	3	* SCSIMsgOut	13	SCSISelect	2
SCSIComplete	4	SCSIRBlind	8	SCSIStat	10
SCSIGet	1	SCSIRead	5	SCSIWBlind	9
SCSIInstall	7	SCSIReset	0	SCSIWrite	6
* SCSIMsgIn	12	* SCSISelAtn	11		

****

*** _InternalWait** $A07F basis unknown

NOTE: This trap was listed in Inside Macintosh without an accompanying description.

SetTimeout 0 GetTimeout 1

****

****

* **_ScriptUtil**	$A8B5			stack-based [long]	
NOTE: Routine number expected to be a 32-bit long integer on stack					
smChar2Pixel	22	smFontScript	0	smMeasureJust	32
smCharByte	16	smGetEnvirons	8	smPixel2Char	20
smCharType	18	smGetScript	12	smSetEnvirons	10
smDrawJust	30	smHiliteText	28	smSetScript	14
smFindWord	26	smIntlScript	2	smTranslit	24
smFont2Script	6	smKybdScript	4		

****

* **_Shutdown**	$A895			Stack-based	
ShutDwnInstall	3	ShutDwnRemove	4		
ShutDwnPower	1	ShutDwnStart	2		

****

* **_SlotManager**	$A06E			Register-based	
InitPRAMRecs	37	sFindsRsrcPtr	48	sReadByte	0
InitSDec1Mgr	32	sFindStruct	6	sReadDrvrName	25
InitsRsrcTable	41	sGetBlock	5	sReadInfo	16
sCalcsPointer	44	sGetcString	3	sReadLong	2
sCalcStep	40	sGetDriver	45	sReadPBSize	38
sCardChanged	34	sNextRsrc	20	sReadPRAMRec	17
sCkCardStatus	24	sNextTypesRsrc	21	sReadStruct	7
sdeleteSRTRec	49	sOffsetData	36	sReadWord	1
sDisposePtr	23	sPrimaryInit	33	sRsrcInfo	22
sExec	35	sPtrToSlot	46	sSearchSRT	42
sFindDevBase	27	sPutPRAMRec	18	sUpdateSRT	43
sFindsInfoRecPtr	47	sReadFHeader	19		

****

PART 4

Unit Table Entries (Device Driver)

0	[reserved]
1	hard disk driver (MacXL/HD20)
2	.Print
3	.Sound
4	.Sony (disk driver)
5	.AIn (Modem port async in)
6	.AOut (Modem port async out)
7	.BIn (Printer port async in)
8	.BOut (Printer port async out)
9	.MPP (AppleTalk)
10	.ATP (AppleTalk)
11	[reserved]
12-26	desk accessories in System file
27-31	desk accessories in application files
32-39	SCSI drivers 0-7
40-47	[reserved]

PART 5

Font Numbers for Apple® Screen Fonts

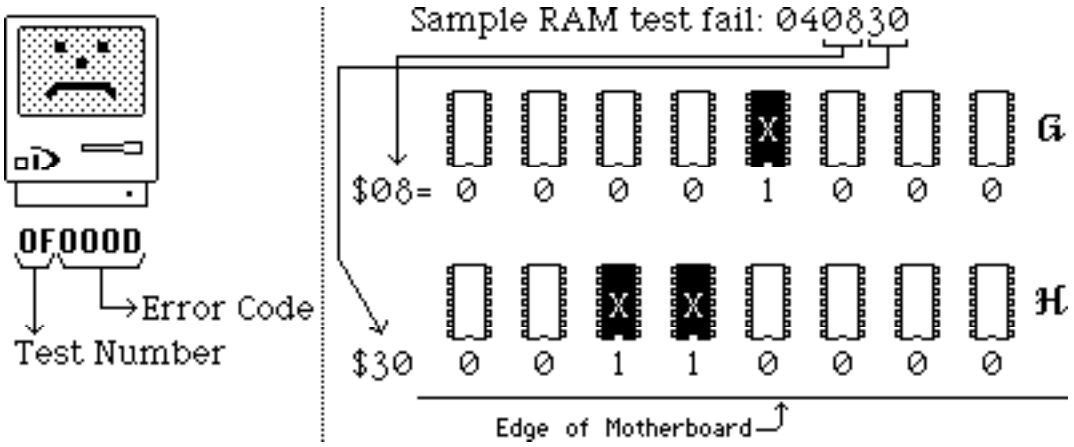
Font	Font No.	Res ID	Font	Font No.	Res ID
Chicago	0	0	[reserved]	13	1664
application font	1	128	[reserved]	14	1792
New York	2	256	[reserved]	15	1920
Geneva	3	384	[reserved]	16	2048
Monaco	4	512	[reserved]	17	2176
Venice	5	640	[reserved]	18	2304
London	6	768	[reserved]	19	2432
Athens	7	896	Times Roman	20	2560
San Francisco	8	1024	Helvetica	21	2688
Toronto	9	1152	Courier	22	2816
Seattle	10	1280	Symbol	23	2944
Cairo	11	1408	Taliesin	24	3072
Los Angeles	12	1536	[reserved]	25	3200

NOTES:

1. Font numbers 0-127 are reserved for future Apple fonts.
2. Font numbers 128-383 are reserved for vendor assignments.
3. Font numbers 384-511 available for use by anyone.
4. Old copies of Apple's Font/DA Mover (older than version 3.x) will not handle fonts that have negative resource ID's (i.e. font numbers 256 and up), affecting half the font numbers reserved for vendors and all of the general-use font numbers.

PART 6

Power-up/Reset Memory Tests



When the Mac is first turned on, it performs several internal tests before displaying the insert disk icon. Although Apple never published details about these tests, this information was leaked to the public soon after the Mac's introduction in 1984.

Every time the Mac performs a cold start, usually when power is first applied (or even after a bad system crash), the boot code in the ROM executes 5 memory tests. The first one is a ROM self-test, and the remaining four test the RAM. If any of these tests fail, obviously the Mac in question will need logic board repairs. To help their technicians diagnose the trouble, Apple distributed a diagram similar to the one above.

The diagram was released for the first Mac motherboard, used in the old 128K and 512K Macs. To the best of my knowledge, Apple has not released diagrams for Macs built with SIMM RAM modules (Mac Plus and all later models). But I do understand that Apple used a chip-numbering system for SIMMs similar to the one used for the soldered-in RAM chips of the original motherboards.

The sad-Mac screen is used to report memory test failures on all Mac models. The sad-Mac screen itself is generated by the System Error Handler when it doesn't have the normal system error box loaded (such as before the Mac can boot, or after a bad enough crash). Below the sad-Mac icon are 6 hexadecimal digits which describe the system error.

For memory test failures, the first two hex digits always report the test number. The second pair of digits represent the chips in row 'G' on the motherboard (see diagram), and the last pair of digits describe row 'H'. Starting from the left, each bit corresponds to a particular chip. If the chip is bad, the bit is set to '1', otherwise the bit is zero.

For system errors, the first pair of hex digits will always be a 0F. The last four digits represent the error code (which will be the normal system error ID code converted into hex). The table below lists both the 0F and its associated error IDs (listed as processor errors):

Test Number and Description	Error Code
01 ROM test failed	(meaningless)
02 Memtest - Bus subtest	Identity of suspected bad RAM chip(s)
03 Memtest - ByteWrite	"
04 Memtest - Mod3test	"
05 Memtest - Add uniqueness	"
06 through 0E - not used	

0F Exception - MC68000 error	Processor error (listed below)
0001 Bus Error	0007 privilege violation
0002 address error	0008 trace
0003 illegal instruction	0009 A-trap
0004 zero divide	000A F-trap

0005 check instruction
0006 trapv instruction
000B other exceptions
000D NMI (interrupt button)

PART 7A

Standard ASCII Chart

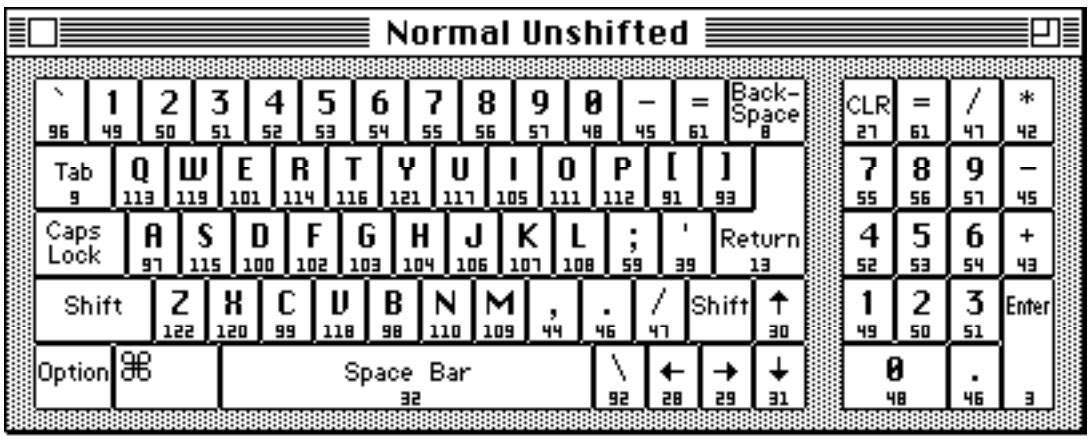
This ASCII chart is labeled in hexadecimal. The ASCII character code for any character is obtained by adding the character's row and column numbers, e.g. ASCII code for "g" is 60+7 or $67 hex.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	NUL						ACK	BEL	BS	HT	LF		FF	CR	SO	SI
10		XON		XOFF		NAK						ESC				
20	SP	!	"	#	$	%	&	'	(	)	*	+	,	−	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	DEL

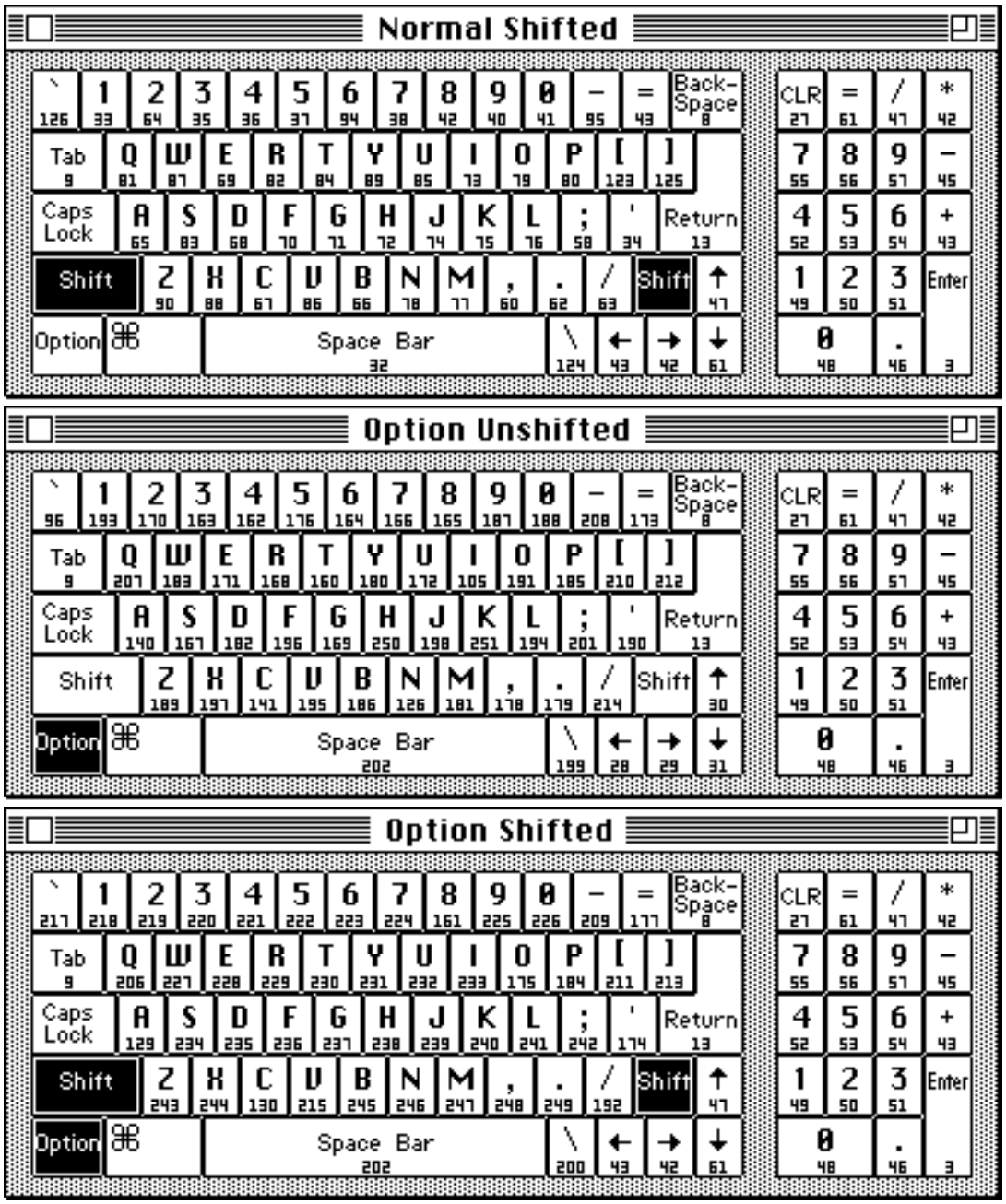
PART 7B

ASCII Character Layout, Mac Plus Keyboard

NOTE: Although there are several other Apple keyboard designs, the ASCII codes generated by their keys will largely follow the pattern used by the Macintosh Plus keyboard.



(continued on next page)


Normal Shifted
` 126 1 33 2 64 3 35 4 36 5 37 6 94 7 38 8 42 9 40 0 41 - 95 = 43 Back-Space 8
Tab 9 Q 81 W 87 E 69 R 82 T 84 Y 89 U 85 I 73 O 79 P 80 [123] 125
Caps Lock A 65 S 83 D 68 F 70 G 71 H 72 J 74 K 75 L 76 ; 58 ' 34 Return 13
Shift Z 90 X 88 C 67 V 86 B 66 N 78 M 77 , 60 . 62 / 63 Shift ↑ 47
Option ⌘ Space Bar 32 \ 124 ← 43 → 42 ↓ 61
CLR 27 = 61 / 47 * 42
7 55 8 56 9 57 - 45
4 52 5 53 6 54 + 43
1 49 2 50 3 51 Enter 3
0 48 . 46
Option Unshifted
` 96 1 193 2 170 3 163 4 162 5 176 6 164 7 166 8 165 9 187 0 188 - 208 = 173 Back-Space 8
Tab 9 Q 207 W 183 E 171 R 168 T 160 Y 180 U 172 I 105 O 191 P 185 [210] 212
Caps Lock A 140 S 167 D 182 F 196 G 169 H 250 J 198 K 251 L 194 ; 201 ' 190 Return 13
Shift Z 189 X 197 C 141 V 195 B 186 N 126 M 181 , 178 . 179 / 214 Shift ↑ 30
Option ⌘ Space Bar 202 \ 199 ← 28 → 29 ↓ 31
CLR 27 = 61 / 47 * 42
7 55 8 56 9 57 - 45
4 52 5 53 6 54 + 43
1 49 2 50 3 51 Enter 3
0 48 . 46
Option Shifted
` 217 1 218 2 219 3 220 4 221 5 222 6 223 7 224 8 161 9 225 0 226 - 209 = 177 Back-Space 8
Tab 9 Q 206 W 227 E 228 R 229 T 230 Y 231 U 232 I 233 O 175 P 184 [211] 213
Caps Lock A 129 S 234 D 235 F 236 G 237 H 238 J 239 K 240 L 241 ; 242 ' 174 Return 13
Shift Z 243 X 244 C 130 V 215 B 245 N 246 M 247 , 248 . 249 / 192 Shift ↑ 47
Option ⌘ Space Bar 202 \ 200 ← 43 → 42 ↓ 61
CLR 27 = 61 / 47 * 42
7 55 8 56 9 57 - 45
4 52 5 53 6 54 + 43
1 49 2 50 3 51 Enter 3
0 48 . 46