Appendix A

Writing Your Own Tool Set

Overview

The Tool Locator System is so flexible that you can write their own tool sets to use in your applications. The Tool Locator System supports both System Tools and User Tools.

There are some factors which you must consider when writing your own tool set:

- Tool sets must use Full Native mode.
- Work space must be dynamically assigned. New tool sets should not use any fixed RAM locations for work space, but must obtain work space from the Memory Manager. This avoids memory conflicts such as those caused by fixed usage of "screen holes." A limited set of exceptions to this rule will be published in the final release of this manual.
- A simple interrupt environment must be provided. All new functions must either be reentrant or must disable interrupts during execution. Because each approach has significant costs, the designer must consider this decision very carefully. Most functions, especially those that execute in less than 500µs, will probably choose to disable interrupts. More time-consuming functions should probably also choose to disable interrupts, especially if they are executed rarely.
- Routines must restore the caller's execution environment before returning control to the caller.
- Routines may not assume the presence of any operating system unless the operating system is directly relevant; for example, a routine that reads or writes a file, where other considerations demand that the file type be known anyway.

Structure of the Tool Locator

The Tool Locator requires no fixed ROM locations and a few fixed RAM locations. All functions are accessed through the tool locator via their tool set number and function number. The Tool Locator uses the tool set number to find an entry in the Tool Pointer Table (TPT). This table contains pointers to Function Pointer Tables (FPT). Each tool set has an FPT containing pointers to the individual routines in the tool. The Tool Locator uses the function number to find the address of the routine being called.

Each tool in ROM has an FPT in ROM. There is also a TPT in ROM pointing to all the FPT's in ROM. One fixed RAM location is used to point to this TPT in ROM. This location is initialized at power up and warm boot by the firmware. In this way the address of the TPT in ROM does not ever have to be fixed.

The TPT has the following form:

Table X-X: Tool Pointer Table Structure

Count	(4 bytes)	Number of tool sets plus one
Pointer to TS 1 FPT	(4 bytes)	Pointer to Function Pointer Table for TSNum 1
Pointer to TS 2 FPT	(4 bytes)	Pointer to Function Pointer Table for TSNum 2
...	...	...

A Function Pointer Table has the following form:

Table X-X: Function Pointer Table Structure

Count	(4 bytes)	Number of routines plus one
Address of F1 - 1	(4 bytes)	Pointer to BootInit routine minus one
Address of F2 - 1	(4 bytes)	Pointer to Startup routine minus one
Address of F3 - 1	(4 bytes)	Pointer to Shutdown routine minus one
Address of F4 - 1	(4 bytes)	Pointer to Version routine minus one
Address of F5 - 1	(4 bytes)	Pointer to Reset routine minus one
Address of F6 - 1	(4 bytes)	Pointer to reserved routine minus one
Address of F7 - 1	(4 bytes)	Pointer to reserved routine minus one
Address of F8 - 1	(4 bytes)	Pointer to reserved routine minus one
Address of F9 - 1	(4 bytes)	Pointer to first non-required routine minus one

Tool Set Numbers and Function Numbers

Each tool is assigned a permanent tool number. Assignment starts at one and continues with each successive integer.

Each function within a tool set is assigned a permanent function number. For the functions within each tool, assignment starts at one and continues with each successive integer. Thus, each function has a unique, permanent identifier of the form (TSNum,FuncNum). Both the TSNum and FuncNum are 8-bit numbers.

The Tool Set numbers assigned to the Apple tools are as follows:

Table X-X: Tool Set Numbers

TSNum	Tool Set
1	Tool Locator
2	Memory Manager
3	Miscellaneous. Tools
4	QuickDraw II

5	Desk Manager
6	Event Manager
7	Scheduler
8	Sound Manager
9	Apple Desktop Bus Tools
10	SANE

For each tool, certain standard calls must be present. Each tool must have a boot initialization function that is executed at boot time by either the ROM startup code or when the tool is installed in the system. In addition, each tool has an application startup function, an application shutdown function to allow an application to turn each tool "on" and "off", and a version call that returns information about the version of the tool.

All tools must return version information in the form of a word. The high byte of the word indicates the major release number (starting with 1). The low byte of the word indicates the minor release number (starting with 0). The most significant bit of the word indicates whether the code is an official release or a prototype (no distinction is made between alpha, beta, or other prototype releases).

The standard calls are summarized in the following table:

Table X-X: Required Tool Calls

FuncNum	Descriptions
1	Boot initialization function for each tool
2	Application startup function for each tool
3	Application shutdown function for each tool
4	Version information
5	Reset
6	Reserved for future use
7	Reserved for future use
8	Reserved for future use

Obtaining Memory

Tools are to obtain any memory they need dynamically (using as little fixed memory as possible) through the Memory Manager. In order to do that, a tool needs some way to find out the location of its data structures. The Tool Locator maintains a table of work area pointers for the individual tools. The Work Area Pointer Table (WAPT) is a table of pointers to the work areas of individual tools.

Each tool will have an entry in the WAPT for its own use. Entries are assigned by tool set number (tool four has entry four and so on). A pointer to the WAPT must be kept in RAM at a fixed memory location so that space for the table can be allocated dynamically. At firmware initialization time, the pointer to the WAPT is set to zero.

The Tool Locator system permanently reserves some space in bank $E1 for the following purposes:

Table X-X: Tool Locator Permanent Ram Space (Bank E1)

(4 bytes)	Pointer to the active TPT. The pointer is to the ROM-based TPT if there are no RAM-based tool sets and no RAM-based ROM patches. Otherwise, it will point to a RAM-based TPT.
(4 bytes)	Pointer to the active user's TPT. This pointer is zero initially, indicating that no user tools are present.
(4 bytes)	Pointer to the Work Area Pointer Table (WAPT). The WAPT parallels the TPT. Each WAPT entry is a pointer to a work area assigned to the corresponding tool set. At startup time, each WAPT entry is set to zero, indicating no assigned work area.
(4 bytes)	Pointer to the user's Work Area Pointer Table (WAPT).
(16 bytes)	Entry points to the dispatcher.

This is the only RAM permanently reserved by the tool locator system.

Tool Locator System Initialization

Each tool set must be initialized before use by application programs. Two types of initialization are needed: boot initialization and application initialization. Boot initialization occurs at system startup time (boot time); regardless of the applications to be executed, the system calls the boot initialization function of every tool set. Thus, each tool set must have a boot initialization routine (FuncNum = 1), even if it does nothing. This function has no input or output parameters.

Application initialization occurs during application execution. The application calls the application startup function (FuncNum=2) of each tool set that it will use. The application startup function performs the chores needed to start up the tool set so the application can use it. This function may have inputs and outputs, as defined by the individual tool set..

The application shutdown function (FuncNum=3) should be executed as soon as the application no longer needs to use the tool. The shutdown releases the resources used by the tool. As a precaution against applications that forget to execute the shutdown function, the startup function should either execute the shutdown function itself or do something else to assure a reasonable startup state.

The provision of two initialization times reflects the needs of currently envisioned tools. For example, the Memory Manager requires boot time initialization because it must operate properly even before any application has been loaded. On the other hand, SANE needs to be initialized only if the system executes some application or desk accessory that uses it. Initializing only the tool sets that will be used saves resources, particularly RAM.

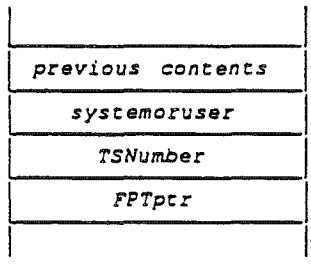
Disk and RAM Structure of Tools

This section will eventually discuss additional details of dynamically loaded, RAM-based tool sets. The exact form of tools on disk is undecided at this time.

Installing Your Tool Set

Before you make any calls to your own tool set, you have to install your tool into the system. You do this by calling the tool locator function SetTSPtr. SetTSPtr takes three inputs on the stack as follows:

Stack Before SetTSPtr



Word specifying the tool as either a system ($0000) or user ($8000) tool.

Word specifying tool set number of the tool set whose pointer is to be set.

Pointer to the Function Pointer Table for the tool.

When SetTSPtr is called, your tool is installed in the system and its boot initialization function call is executed. The following example illustrates the installation:

A Tool Set Installation Example

```
;-------------------------------------------------------------------------------
Install     START

            clc                     ; switch to full native mode and
            xce                     ; save initial state
            php

            rep  #$30               ; 16 bit registers

            PushWord  $8000         ; signal a user tool
            PushWord  #$23          ; Put the tool number on the stack
            PushLong  #CallTable    ; Point to call table
            _SetTSPtr

            plp                     ; restore machine state
            xce
            rts

            END
;-------------------------------------------------------------------------------
CallTable   START

            long   (TheEnd-CallTable)/4

            long  MyBootInit-1
            long  MyStartUp-1
            long  MyShutDown-1
            long  MyVersion-1
            long  MyReset-1
            long  NotImp-1
            long  NotImp-1
            long  NotImp-1

            long  FirstFunc-1
            long  LastFunc-1

TheEND      ;

            END
```

```
;-------------------------------------------------------------------------------
MyBootInit    START                 ; called when installed

              lda #0
              clc
              rtl

              END
;-------------------------------------------------------------------------------
MyStartUp     START                 ; user passes me the loc to use in
                                    ; bank zero as word.

RTL1          equ 1
RTL2          equ RTL1+3
ZPToUse       equ RTL2+3

              lda ZPToUse,s         ; get users value

              pea $8000             ; user call
              pea $23               ; tool set number
              pea 0                 ; high word is zero
              pha                   ; low word is user's value
              _SetWAP               ; set it

              lda #0
              clc
              rtl

              END
```

```
;-------------------------------------------------------------------------------
MyShutDown    START

              cmp #0
              beq nevermind

              pea $8000              ; clear out the WAPT entry
              pea $23
              pea 0
              pea 0
              _SetWAP

nevermind     lda #0
              clc
              rtl

              END
;-------------------------------------------------------------------------------
MyVersion     START
RTL1          equ 1
RTL2          equ RTL1+3
VerNum        equ RTL2+3

              lda #$90               ; version 1.0 prototype
              sta VerNum,s
              lda #0
              clc
              rtl

              END

;-------------------------------------------------------------------------------
MyReset       START

              lda #0
              clc
              rtl

              END
```

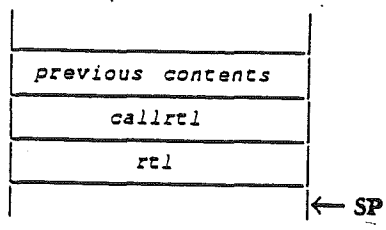
```
;-------------------------------------------------------------------------------
NotImp          START

                lda  #$23FF
                sec
                rtl

                END
;-------------------------------------------------------------------------------
FirstFunc       START

                lda  #0
                clc
                rtl

                END
;-------------------------------------------------------------------------------
LastFunc        START

                lda  #0
                clc
                rtl

                END
;-------------------------------------------------------------------------------
;   Notes
;
;The long directive.deposits a 4-byte value in memory low bytes first
;The PushWord macro pushes a word onto the stack (either from a memory
;             location or with a pea instruction if # is used).
;The PushLong macro pushes a long on the stack (either from memory
;             or with two pea instructions if # is used).
```

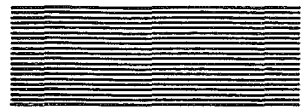
Function Execution Environment

When your function is called, the machine is in full native mode and the following three registers are set with specific information to make the function's job easier:

- A-Reg low word of entry in WAPT for tool
- Y-Reg high word of entry in WAPT for tool
- X-Reg Function number and Tool number

When the function is called, the stack looks like this:


previous contents
callrtl
rtl
SP



Appendix B

Error Codes

No Errors

$0000 in accumulator and carry flag not set.

System Death Codes

$0001	ProDOS'16 - Unclaimed interrupt
$0004	Divide by zero
$000A	ProDOS'16 - Volume Control Block unusable
$000B	ProDOS'16 - File Control Block unusable
$000C	ProDOS'16 - Block zero allocated illegally
$000D	ProDOS'16 - Interrupt with I/O shadowing off
$0015	Segment Loader error
$0017-24	Can't load a package
$0025	Out of memory
$0026	Segment Loader error
$0027	File map trashed
$0028	Stack overflow error
$0030	Please insert disk (file manager alert)
$0032-53	Memory manager error
$0100	Can't mount system startup volume

Memory Manager Error Codes

$0201	Memory full error.
$0202	Illegal operation on a NIL handle.
$0203	NIL handle expected for this operation.
$0204	Illegal operation on a locked or immovable block.
$0205	Attempt to purge an unpurgeable block.
$0206	Invalid handle given.
$0207	Invalid owner ID given.

Miscellaneous Tool Set Error Codes

$0301	Bad Input Parameter.

$0302	No Device for Input Parameter.
$0303	Task is already in Heartbeat queue.
$0304	No signature in task header was detected during insert or delete.
$0305	Damaged queue was detected during insert or delete.
$0306	Task was not found during delete.
$0307	Firmware task was unsuccessful.
$0308	Detected damaged HeartBeat Queue.
$0309	Attempted dispatch to a device that is not connected.
$030A	Undefined.
$030B	ID tag not available.

Event Manager Error Codes

$0601	Duplicate EMStartUp call
$0602	Reset Error.
$0603	Event Manager not active.
$0604	Illegal event code.
$0605	Illegal button number.
$0606	Queue size too large.
$0607	Not enough memory available for queue.
$0681	Fatal system error - event queue damaged.
$0682	Fatal system error - queue handle damaged.

Sound Manager Error Codes

$0804	DOC address range error.
$0810	No DOC chip found
$0812	NO SoundStartup call made
$0813	Invalid generator number
$0814	Synthesizer mode error
$0815	Generator busy
$0817	Master IRQ not assigned
$0818	Sound tools already started

Integer Math Tool Set Error Codes

$0B01	Bad input parameter

$0B02	Illegal character in string
$0B03	Integer or Long Integer overflow
$0B04	String overflow

Text Tool Set Error Codes

$0C01	Illegal device type.
	Note: the following errors should occur only for Pascal Devices
$0C02	Illegal device number.
$0C03	Bad mode: illegal operation.
$0C04	Undefined hardware error.
$0C05	Lost device: device is no longer on-line.
$0C06	Lost file: file is no longer in the diskette directory.
$0C07	Bad title: illegal filename.
$0C08	No room: insufficient space on the specified diskette.
$0C09	No device: the specified volume is not on-line.
$0C0A	No file: the specified file is not in the directory of the specified volume.
$0C0B	Duplicate file: attempt to rewrite a file when a file of that name already exists.
$0C0C	Not closed: attempt to open an already-open file.
$0C0D	Not open: attempt to access a closed file.
$0C0E	Bad format: error in reading real or integer.
$0C0F	Ring buffer overflow: characters are arriving faster than the input buffer can accept them.
$0C10	Write-protect error: the specified diskette is write-protected.
$0C40	Device error: failed to complete a read or write correctly.