# Writing Your Own Tool Set
# A Summary of Information from the Tool Locator
# ERS

Steven E. Glass
February 18, 1986

## What's in a Tool Set?

Every tool set has the following components:

Function Pointer Table
Individual Functions

Some tool sets have

Auxilary routines called by the individual functions

## The Function Pointer Table

The Function Pointer Table (FPT) is a table with four byte entries. It has the following format:

| | | |
|---|---|---|
| Count+1 | 4 bytes | Number of Functions plus one |
| Addr of F1 - 1 | 4 bytes | Pointer to Boot init function minus one |
| Addr of F2 - 1 | 4 bytes | Pointer to Startup function minus one |
| Addr of F3 - 1 | 4 bytes | Pointer to Shutdown function minus one |
| Addr of F4 - 1 | 4 bytes | Pointer to Version function minus one |
| Addr of F5 - 1 | 4 bytes | Pointer to Reset function minus one |
| NotImp-1 | 4 bytes | Pointer to NotImp minus one |
| NotImp-1 | 4 bytes | Pointer to NotImp minus one |
| NotImp-1 | 4 bytes | Pointer to NotImp minus one |
| Addr of F9 - 1 | 4 bytes | Pointer to first non required function in tool set minus one |

And so on...

## Installing Your Tool Set

So you will write code that has an FPT like this with pointers to your code. After you have this you will want to install your tool into the system. You do this by calling the tool locator function SetTSPtr. SetTSPtr takes three inputs as follows:

| | | |
|---|---|---|
| SystemOrUser | word | $0000 for system tool, $8000 for user tool set. |
| Tool Set Number | word | a number between 1 and 255 |
| Pointer to FPT | long | A four byte pointer to the FTP described above. |

When SetTSPtr is called, your tools is installed in the system and its boot initialization function call is executed.

A handy way to do this in the current environment is to assemble code that contains your tool set and preceed it with a routine to install it in the system. (An example of this follows.) This way you can BRUN your code file from Applesoft BASIC and have your tool installed. If you want to make sure that your code runs outside of bank zero, you can have the installation routine move it to another bank before the tool set is installed.

## What about Memory Use?

There are three kinds of tools: 1) those that do not have any information that must live between calls. 2) those that have information that must live between calls but that information can be anywhere in memory; and 3) those that must have information that lives between calls in bank zero. The first case is easy, there is no extra work to do. Cases 2 and 3 require that the tool to do a little work to prevent it from using fixed memory locations.

The Tool Locator provides a facility for doing this. It maintains a Work Area Pointer Table (WAPT). This is a table with entries for each tool in the system. The tool can put any value it wants in the table. So if a tool needs memory of type 2, it can ask the memory manager for memory and put the handle to that memory in the WAPT. Case 3 is a little trickier. We have a convention that tools do not use memory in bank zero unless that memory is given to it by the Application. Tools that work this way are QuickDraw II and the event manager. Both these tools require that the application pass the address of memory in bank zero that can be used for zero page. Both these tools put this address in the WAPT.

The tool locator provides two calls to help tool sets maintain work area information: GetWAP and SetWAP.

### Function Execution Environment

When your function is called, the machine is in full native mode and the 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 |

The stack is as follows on function entry.

| | | |
|---|---|---|
| Params | 7 | TOS just before call |
| RTL from Call | 4 | |
| Another RTL | 1 | |
| | 0 | Current TOS |

(TOS is top of stack.) It is the functions responsiblity to clean of any input parameters from the stack before executing an RTL (unless the function is documented as doing something different).

### Signaling Errors

By convention, functions return an error code in the a register and signal the error with the carry flag. (Carry clear and zero in the a register indicates no error.) Error codes have the following format:

| | |
|---|---|
| Tool Set Number | High Byte |
| Msg Number | Low Byte |

The high byte of the error code is set to the tool set number generating the error. This way a quick draw call can pass on a error message from the memory manager in an intelligent way.

**The Example**

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

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

            rep #$30                ; 16 bit registers

            PushWord #0             ; signal a system 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 0                   ; system 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 0                   ; 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 if # is used).
; The PushLong macro pushes a long on the stack (either from memory
;               or with two peas if # is used).
```