Hidden Gems of the Apple IIgs Toolbox

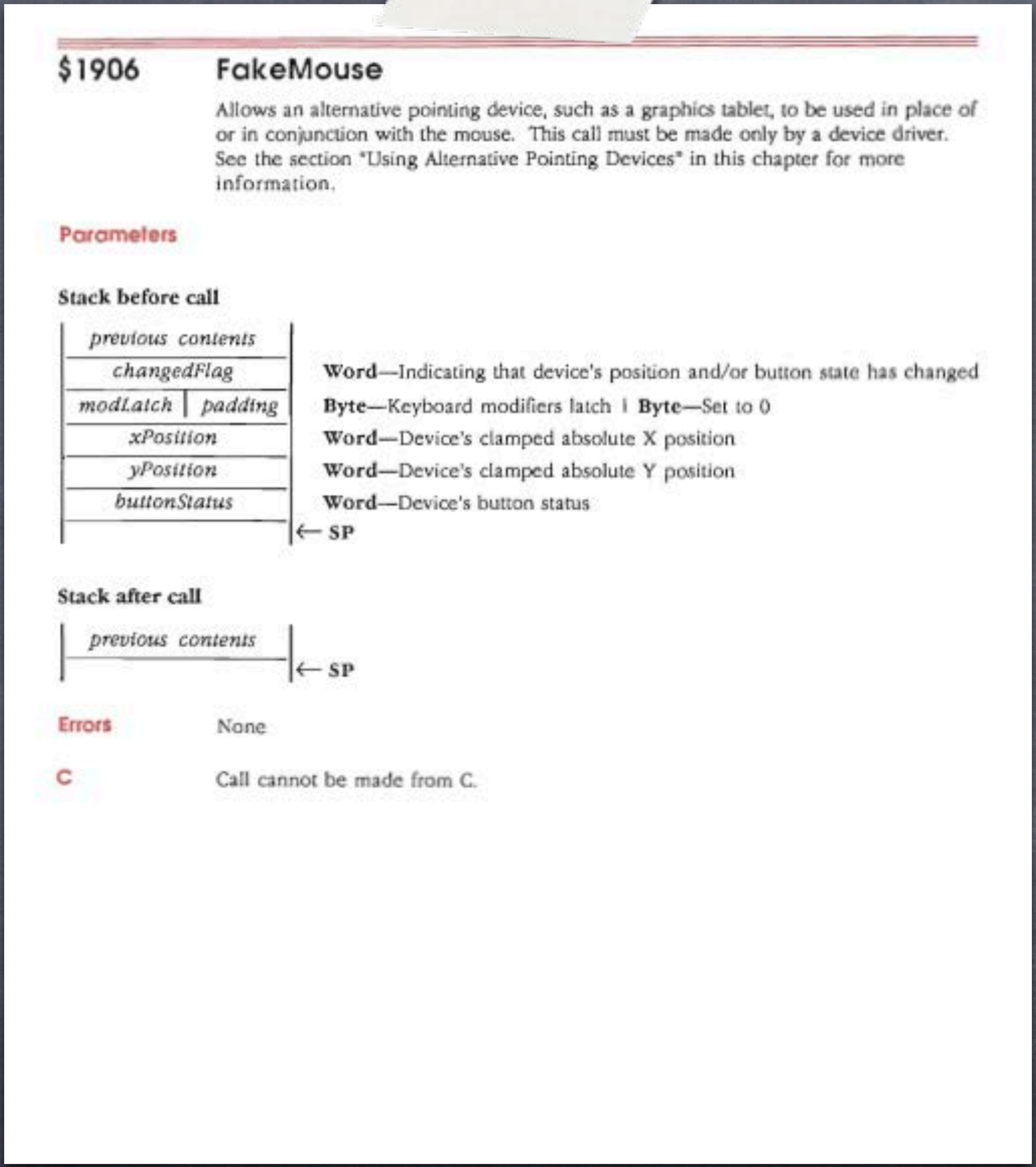
Eric Shepherd

An Overview

- 32 toolsets.
- Some 1200 or so tool calls.
- That's a lot!
- Let's look at some stuff you might not have seen.

Make the Mouse Do Your Bidding

- Have you ever wanted to automate something there wasn’t support for in the Toolbox?
- Take a look at FakeMouse!



$1906 FakeMouse

Allows an alternative pointing device, such as a graphics tablet, to be used in place of or in conjunction with the mouse. This call must be made only by a device driver. See the section "Using Alternative Pointing Devices" in this chapter for more information.

Parameters

Stack before call

previous contents	
changedFlag	**Word**—Indicating that device's position and/or button state has changed
modLatch | *padding*	**Byte**—Keyboard modifiers latch | **Byte**—Set to 0
xPosition	**Word**—Device's clamped absolute X position
yPosition	**Word**—Device's clamped absolute Y position
buttonStatus	**Word**—Device's button status
	← **SP**

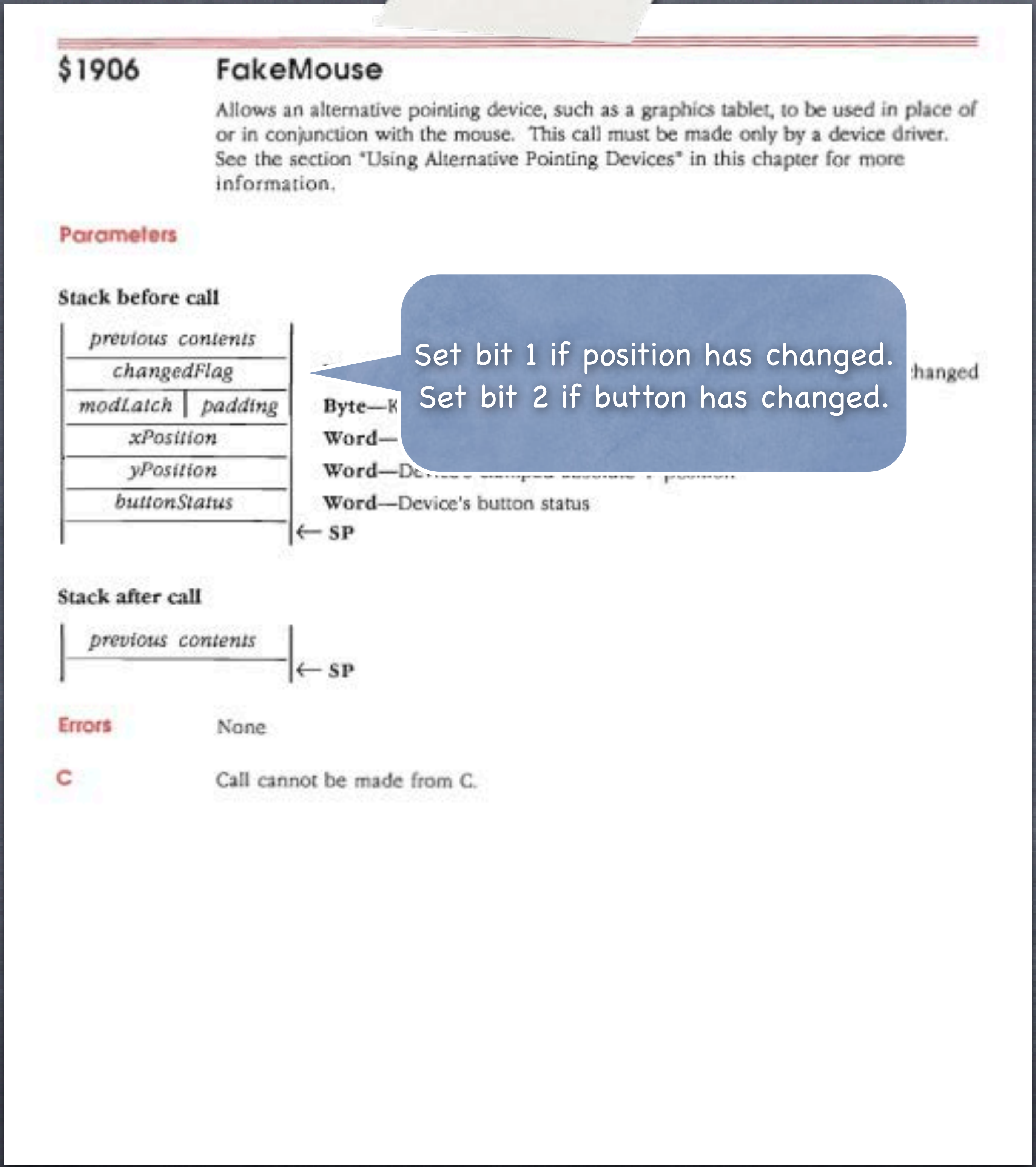
Stack after call

previous contents	
	← **SP**

Errors None

C Call cannot be made from C.

FakeMouse



$1906 **FakeMouse**

Allows an alternative pointing device, such as a graphics tablet, to be used in place of or in conjunction with the mouse. This call must be made only by a device driver. See the section "Using Alternative Pointing Devices" in this chapter for more information.

Parameters

Stack before call

previous contents	
changedFlag	hanged
modLatch | *padding*	**Byte**—K
xPosition	**Word**—
yPosition	**Word**—De
buttonStatus	**Word**—Device's button status
	← **SP**

Stack after call

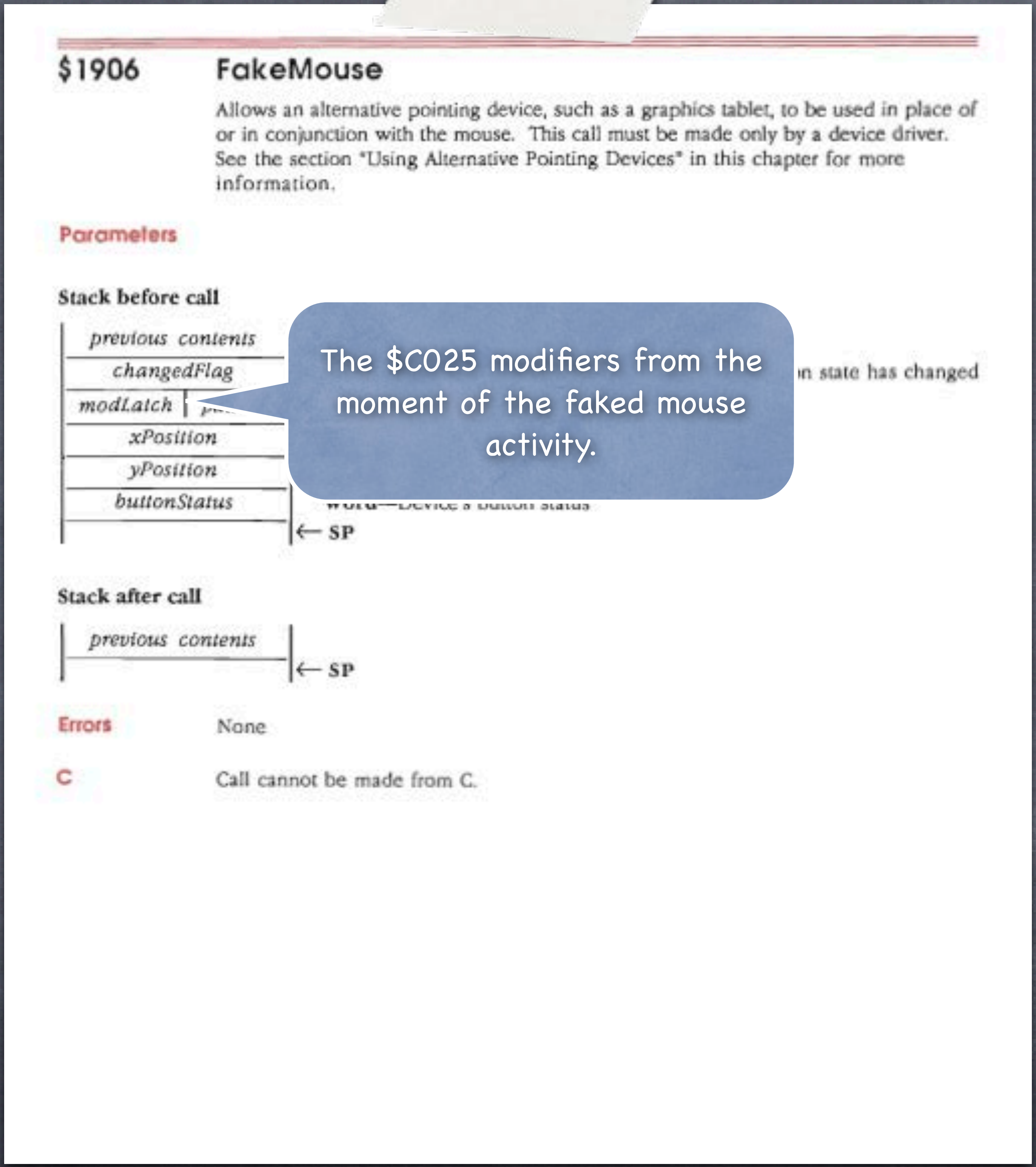
previous contents	
	← **SP**

Errors None

C Call cannot be made from C.

Set bit 1 if position has changed.
Set bit 2 if button has changed.

FakeMouse



$1906 **FakeMouse**

Allows an alternative pointing device, such as a graphics tablet, to be used in place of or in conjunction with the mouse. This call must be made only by a device driver. See the section "Using Alternative Pointing Devices" in this chapter for more information.

Parameters

Stack before call

```
| previous contents |
|   changedFlag     |          ...n state has changed
| modLatch |  pa... |
|    xPosition      |
|    yPosition      |
|   buttonStatus    |          Word—Device's button status
|                   | ← SP
```

The $C025 modifiers from the moment of the faked mouse activity.

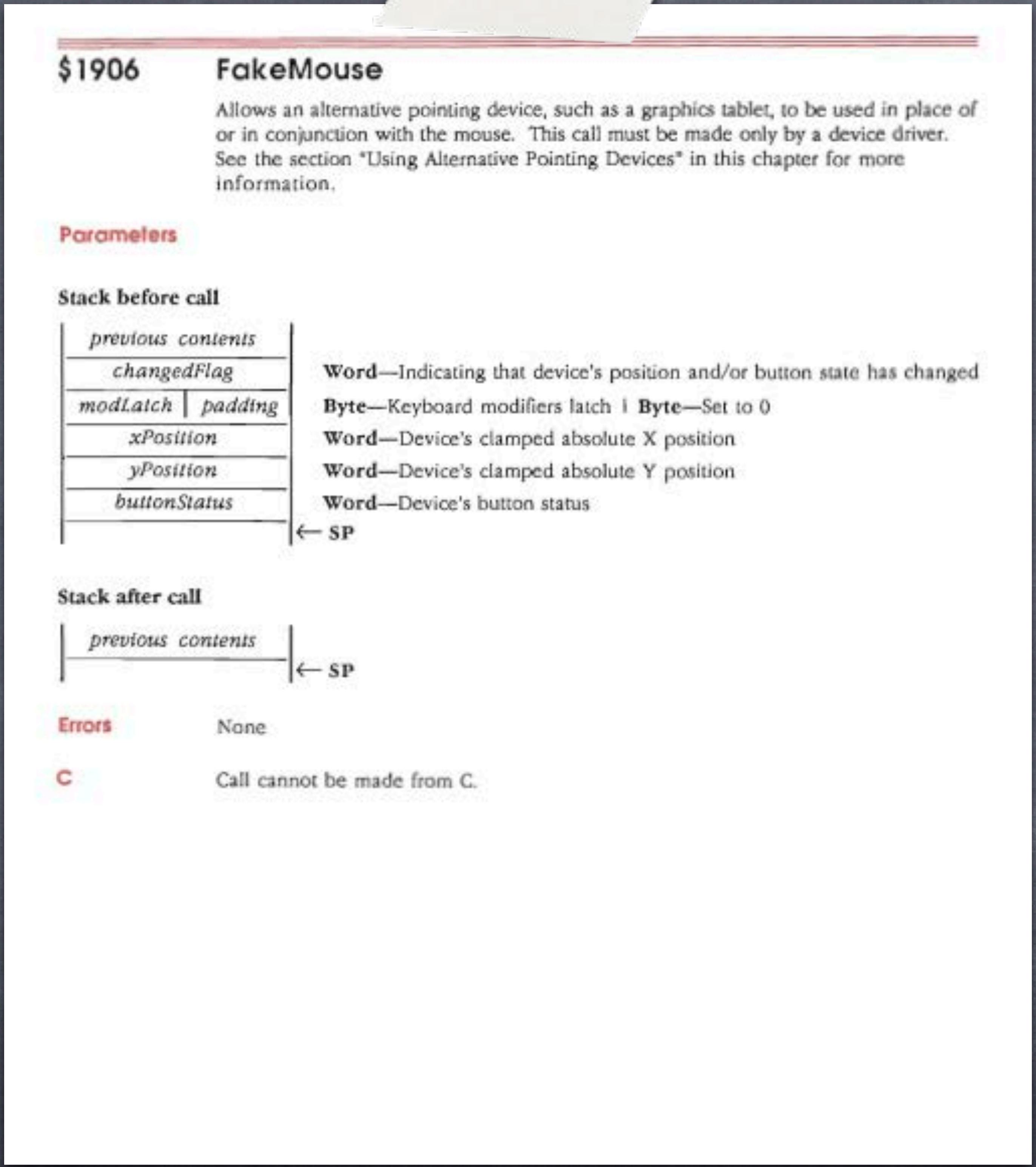
Stack after call

```
| previous contents |
|                   | ← SP
```

Errors None

C Call cannot be made from C.

FakeMouse



$1906 FakeMouse

Allows an alternative pointing device, such as a graphics tablet, to be used in place of or in conjunction with the mouse. This call must be made only by a device driver. See the section "Using Alternative Pointing Devices" in this chapter for more information.

Parameters

Stack before call

previous contents	
changedFlag	**Word**—Indicating that device's position and/or button state has changed
modLatch | *padding*	**Byte**—Keyboard modifiers latch | **Byte**—Set to 0
xPosition	**Word**—Device's clamped absolute X position
yPosition	**Word**—Device's clamped absolute Y position
buttonStatus	**Word**—Device's button status
	← **SP**

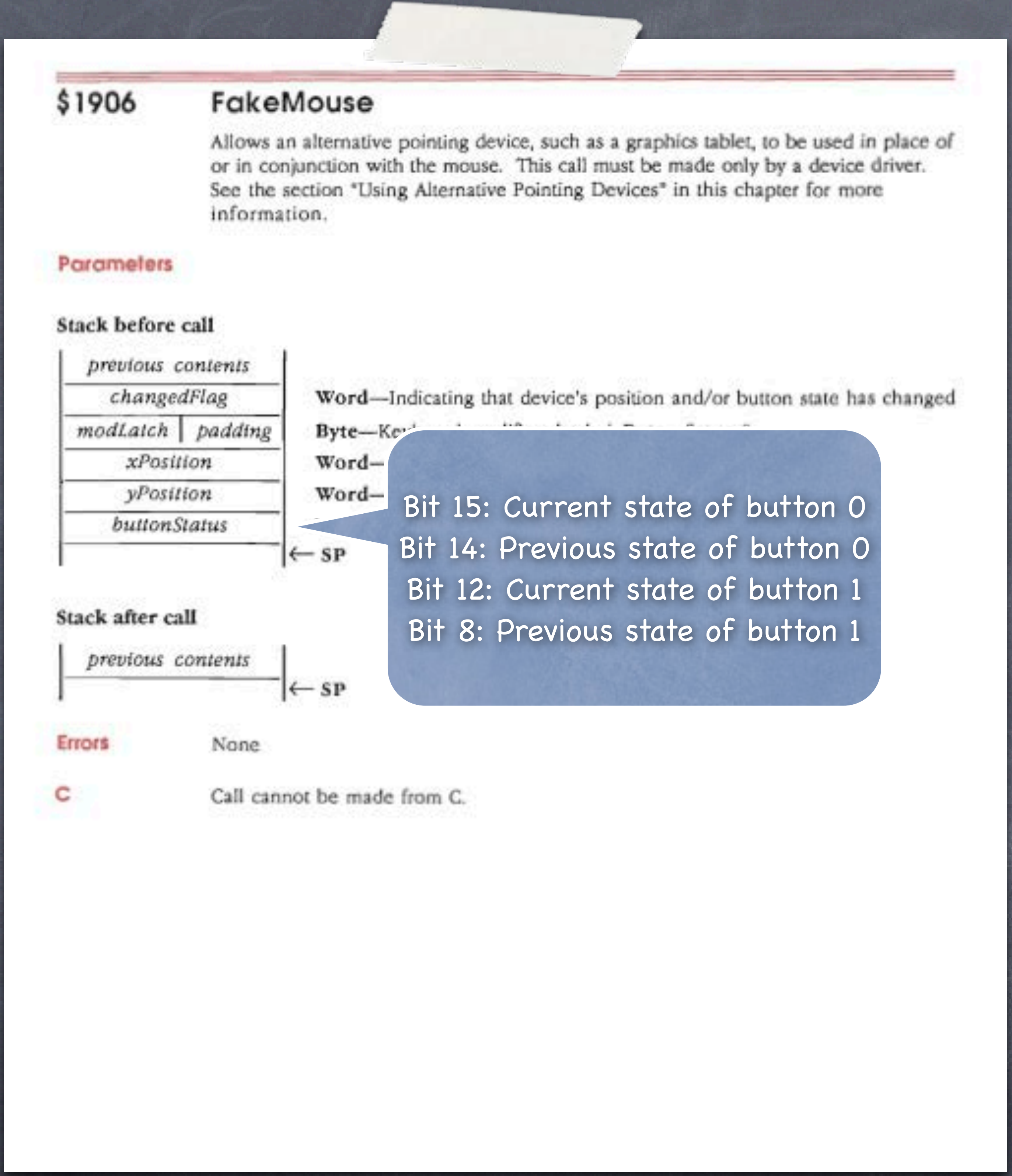
Stack after call

previous contents	
	← **SP**

Errors None

C Call cannot be made from C.

FakeMouse



$1906 FakeMouse

Allows an alternative pointing device, such as a graphics tablet, to be used in place of or in conjunction with the mouse. This call must be made only by a device driver. See the section "Using Alternative Pointing Devices" in this chapter for more information.

Parameters

Stack before call

previous contents	
changedFlag	**Word**—Indicating that device's position and/or button state has changed
modLatch | *padding*	**Byte**—Ke
xPosition	**Word**—
yPosition	**Word**—
buttonStatus	
← SP	

Stack after call

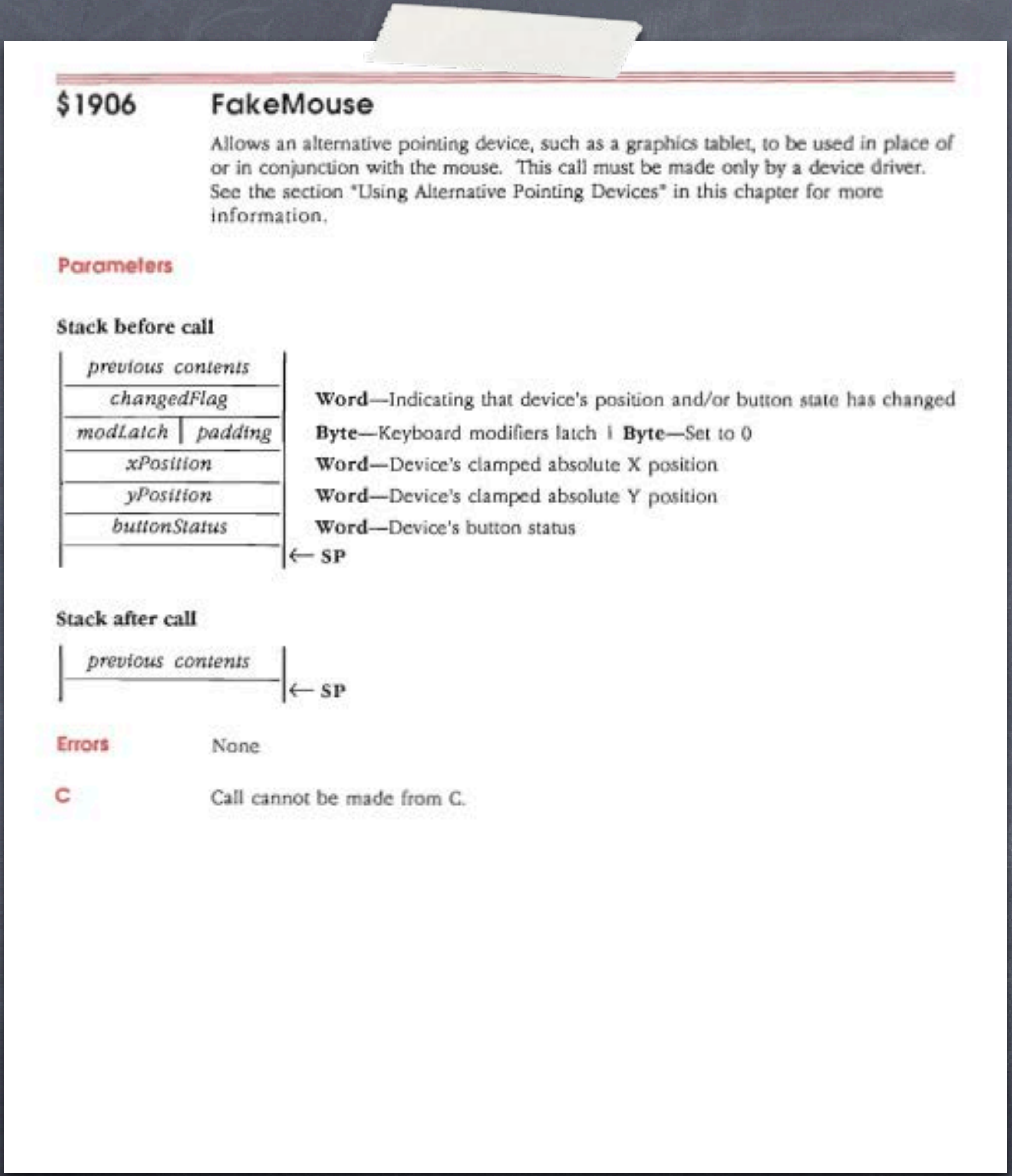
previous contents
← SP

Errors None

C Call cannot be made from C.

Bit 15: Current state of button 0
Bit 14: Previous state of button 0
Bit 12: Current state of button 1
Bit 8: Previous state of button 1

FakeMouse



$1906 FakeMouse

Allows an alternative pointing device, such as a graphics tablet, to be used in place of or in conjunction with the mouse. This call must be made only by a device driver. See the section "Using Alternative Pointing Devices" in this chapter for more information.

Parameters

Stack before call

previous contents	
changedFlag	**Word**—Indicating that device's position and/or button state has changed
modLatch | *padding*	**Byte**—Keyboard modifiers latch | **Byte**—Set to 0
xPosition	**Word**—Device's clamped absolute X position
yPosition	**Word**—Device's clamped absolute Y position
buttonStatus	**Word**—Device's button status
	← **SP**

Stack after call

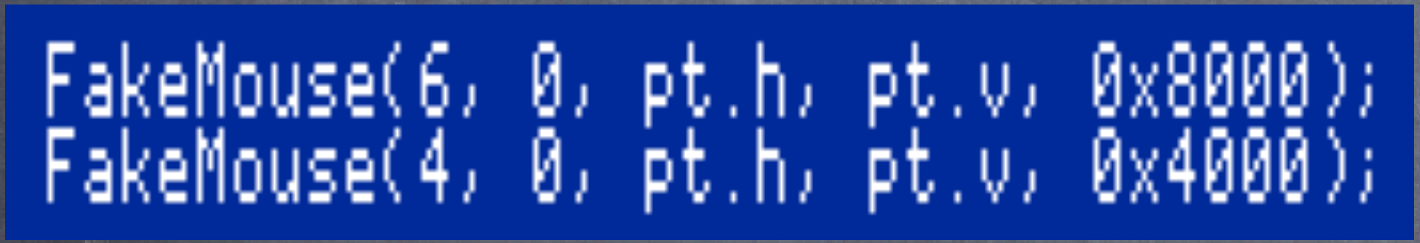
previous contents	
	← **SP**

Errors None

C Call cannot be made from C.

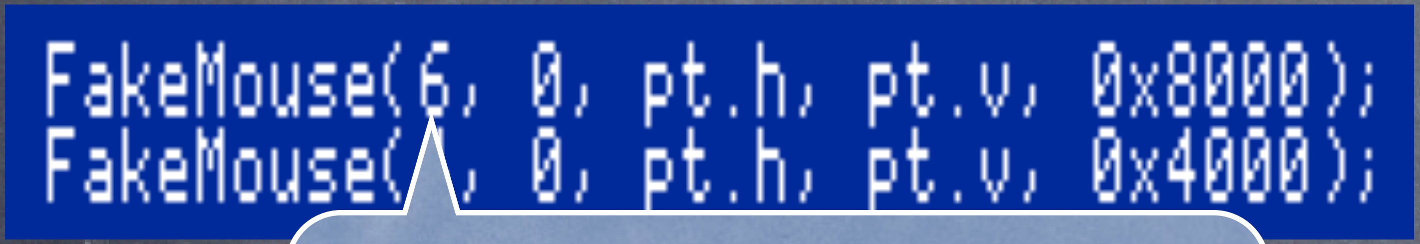
FakeMouse

Click! Click, I say!



```
FakeMouse(6, 0, pt.h, pt.v, 0x8000);
FakeMouse(4, 0, pt.h, pt.v, 0x4000);
```

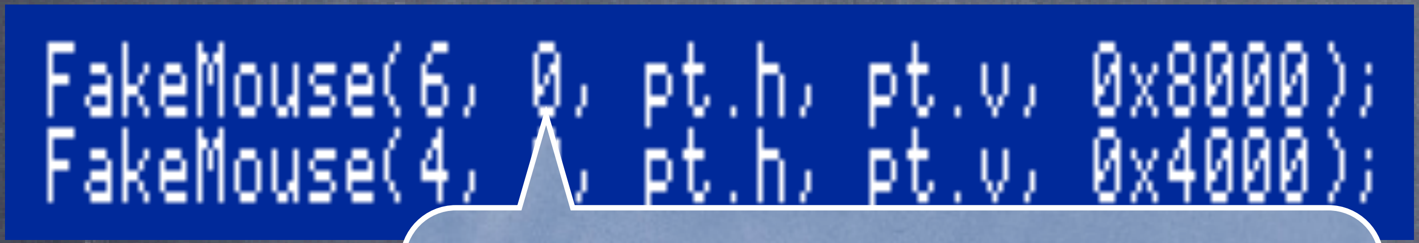
Click! Click, I say!



```
FakeMouse(6, 0, pt.h, pt.v, 0x8000);
FakeMouse([illegible], 0, pt.h, pt.v, 0x4000);
```

Button and mouse position both changed

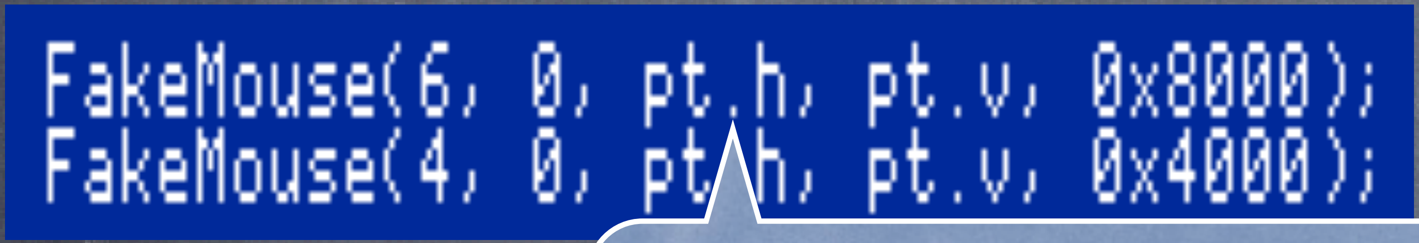
Click! Click, I say!



```
FakeMouse(6, 0, pt.h, pt.v, 0x8000);
FakeMouse(4, 0, pt.h, pt.v, 0x4000);
```

Modifiers; we have none here.

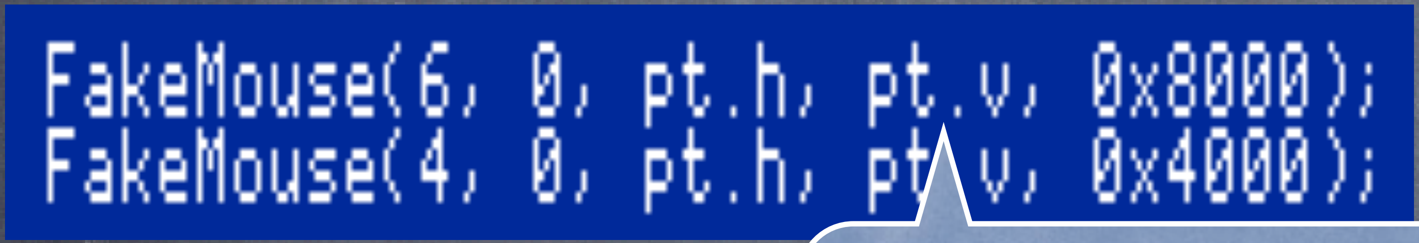
Click! Click, I say!



```
FakeMouse(6, 0, pt.h, pt.v, 0x8000);
FakeMouse(4, 0, pt.h, pt.v, 0x4000);
```

Mouse X coordinate.

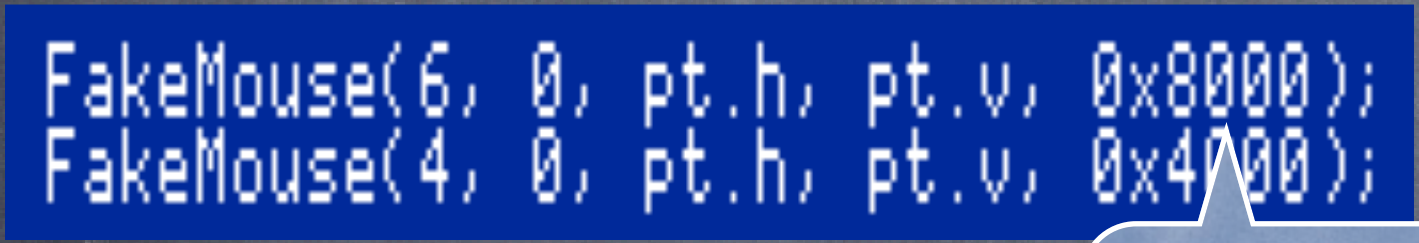
Click! Click, I say!



```
FakeMouse(6, 0, pt.h, pt.v, 0x8000);
FakeMouse(4, 0, pt.h, pt.v, 0x4000);
```

Mouse Y coordinate.

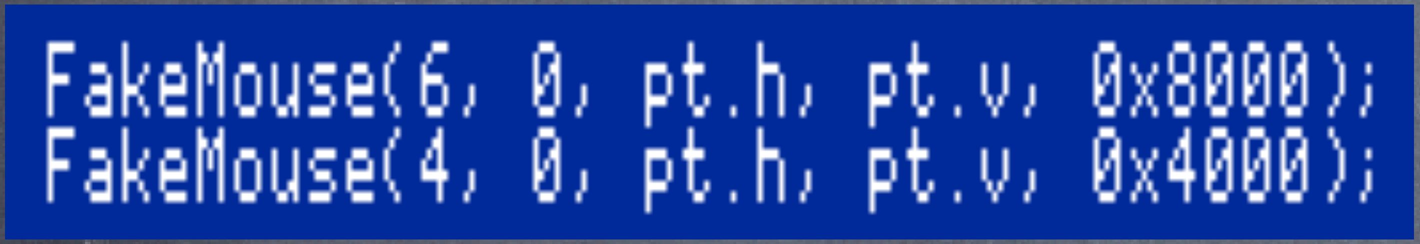
Click! Click, I say!



```
FakeMouse(6, 0, pt.h, pt.v, 0x8000);
FakeMouse(4, 0, pt.h, pt.v, 0x4000);
```

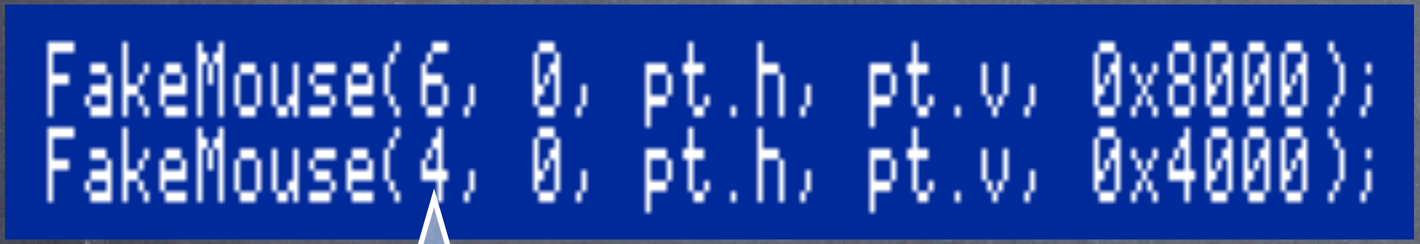
Mouse button was up but is now down.

Click! Click, I say!



```
FakeMouse(6, 0, pt.h, pt.v, 0x8000);
FakeMouse(4, 0, pt.h, pt.v, 0x4000);
```

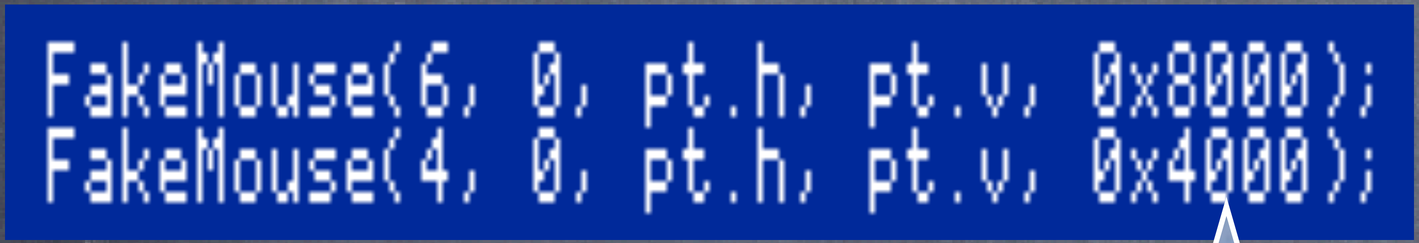
Click! Click, I say!



```
FakeMouse(6, 0, pt.h, pt.v, 0x8000);
FakeMouse(4, 0, pt.h, pt.v, 0x4000);
```

Button state has changed; position has not.

Click! Click, I say!

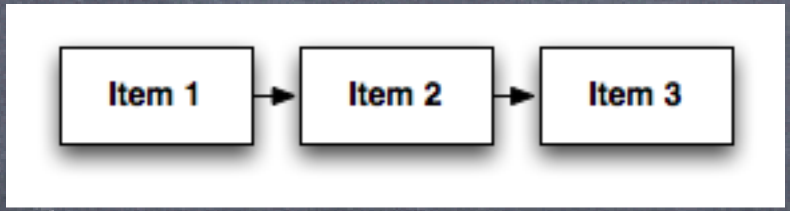


```
FakeMouse(6, 0, pt.h, pt.v, 0x8000);
FakeMouse(4, 0, pt.h, pt.v, 0x4000);
```

Mouse button was down but is now up.

Queue Up for Queues

- Queues are a common data structure
- Records pointing to each other in a chain

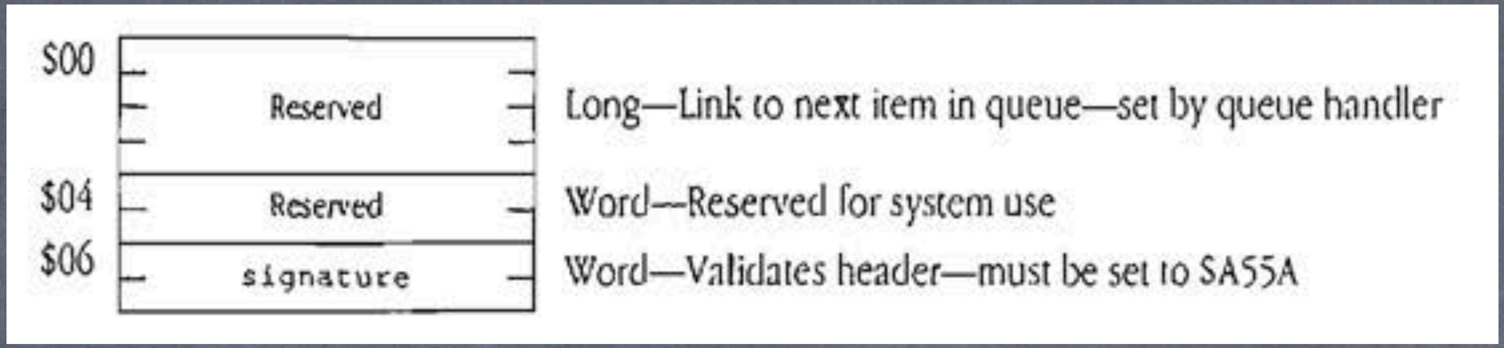


- Last item has a NULL pointer

Misc Tools FTW!

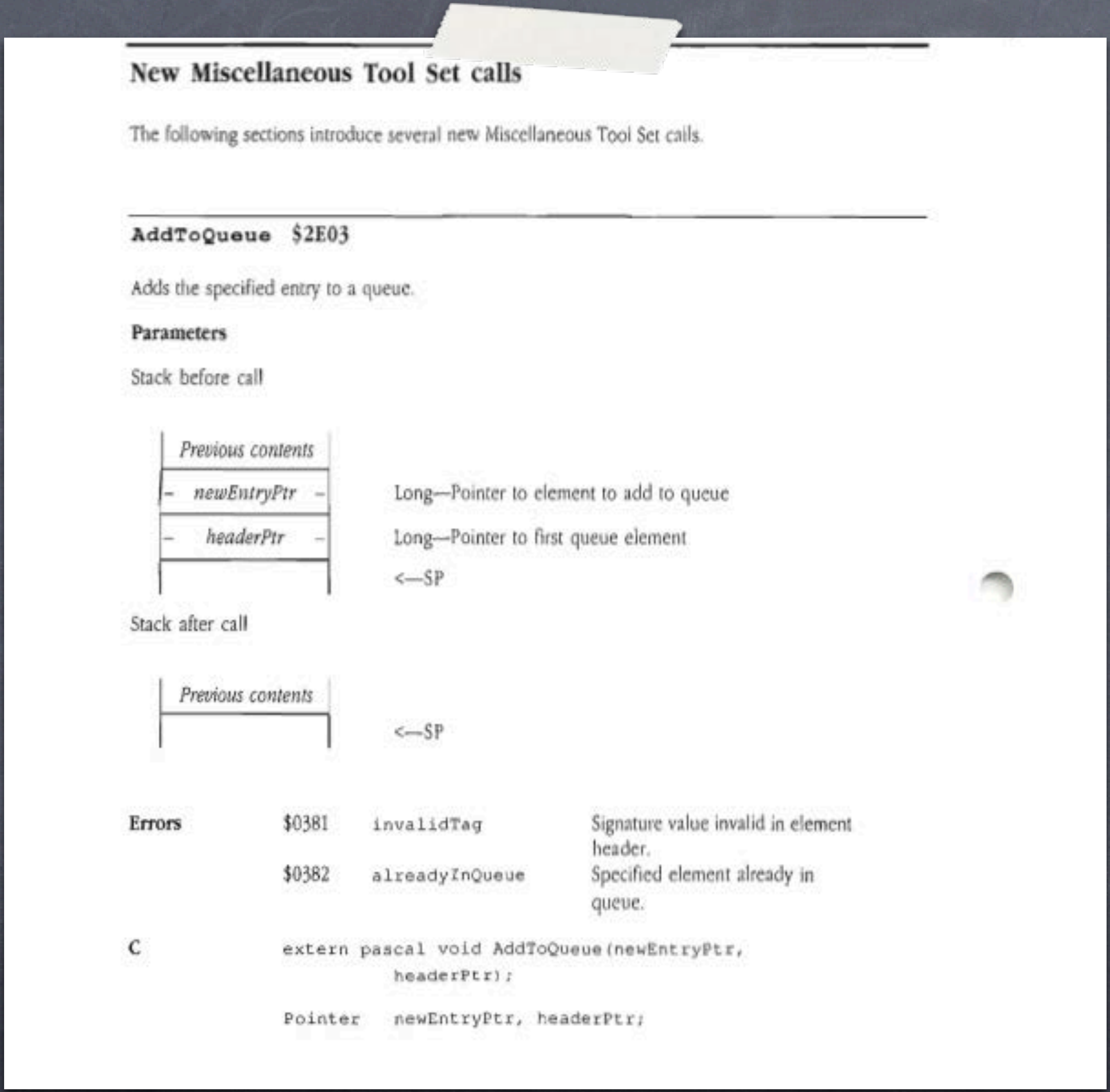
- System 5.0 added AddToQueue and DeleteFromQueue
- Let's take a look

Queue Structure



$00	Reserved	Long—Link to next item in queue—set by queue handler
$04	Reserved	Word—Reserved for system use
$06	signature	Word—Validates header—must be set to $A55A

- Your records need to start with these fields.
- Set the first 6 bytes to 0.
- Signature is used to verify the queue is valid.
- Create your first record yourself, then...


New Miscellaneous Tool Set calls

The following sections introduce several new Miscellaneous Tool Set calls.

AddToQueue $2E03

Adds the specified entry to a queue.

Parameters

Stack before call

```
|  Previous contents  |
|---------------------|
|–  newEntryPtr     –|      Long—Pointer to element to add to queue
|–  headerPtr       –|      Long—Pointer to first queue element
|                     |      <—SP
```

Stack after call

```
|  Previous contents  |
|---------------------|
|                     |      <—SP
```

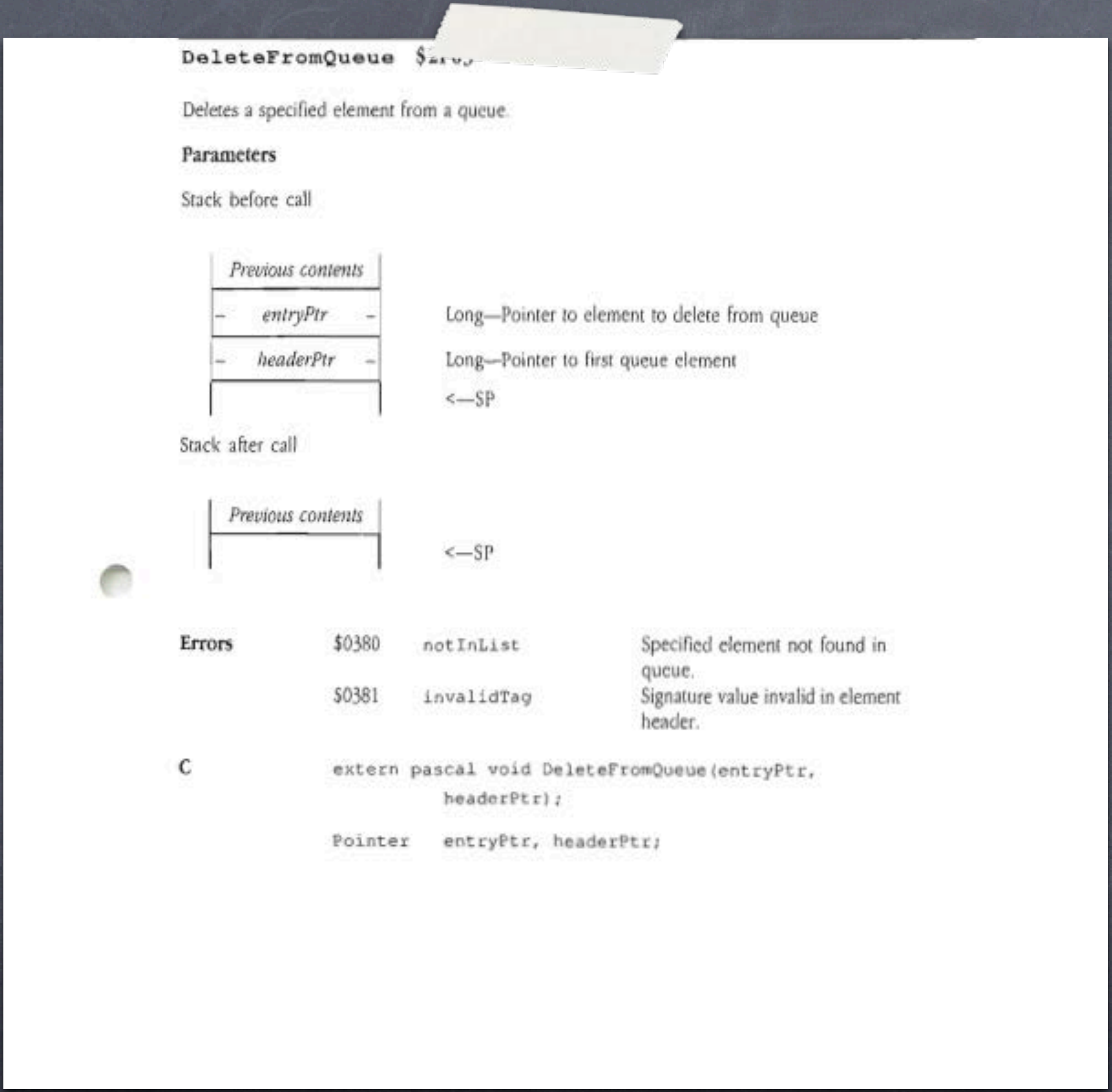
Errors	$0381	`invalidTag`	Signature value invalid in element header.
	$0382	`alreadyInQueue`	Specified element already in queue.

C

```
extern pascal void AddToQueue(newEntryPtr,
          headerPtr);

Pointer   newEntryPtr, headerPtr;
```

AddToQueue



DeleteFromQueue $[illegible]

Deletes a specified element from a queue.

Parameters

Stack before call

Previous contents	
– *entryPtr* –	Long—Pointer to element to delete from queue
– *headerPtr* –	Long—Pointer to first queue element
	<—SP

Stack after call

Previous contents	
	<—SP

Errors	$0380	`notInList`	Specified element not found in queue.
	$0381	`invalidTag`	Signature value invalid in element header.

C

```
extern pascal void DeleteFromQueue(entryPtr,
          headerPtr);

Pointer   entryPtr, headerPtr;
```

DeleteFromQueue

Out-of-Memory Queue

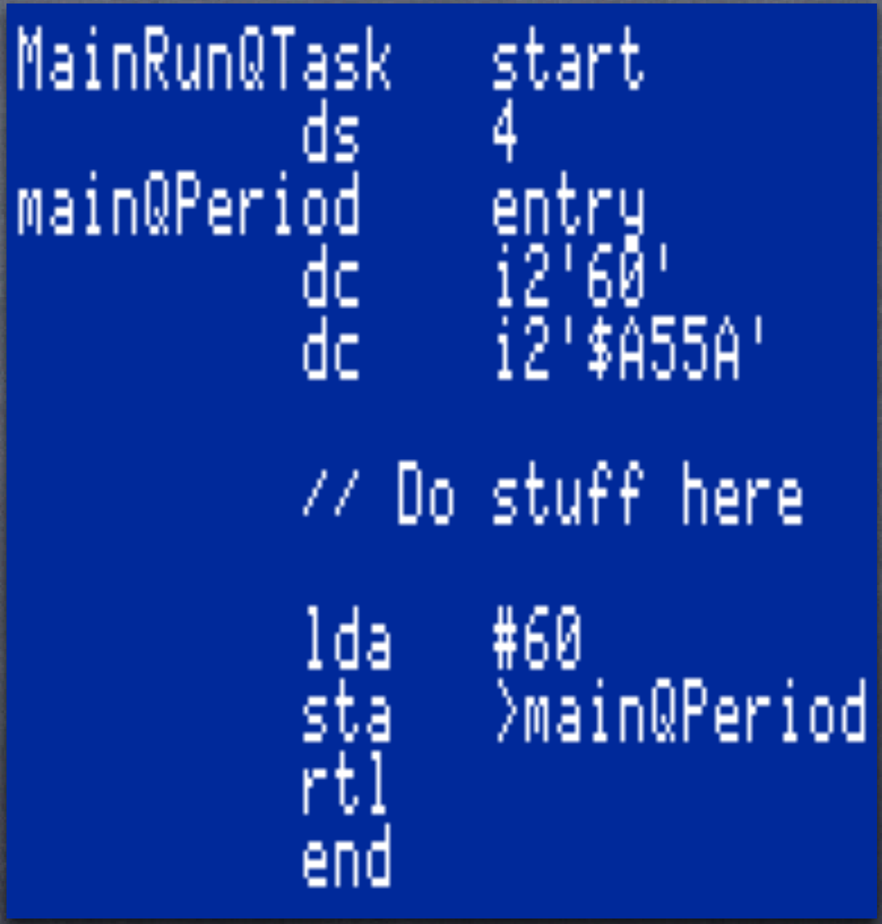
- Called when memory is low to let your code free up space.
- Tells you how much space it wants, you do your best to help.
- Called twice with memory purges in between.
- See TBR3.

Run Queues

- Background tasks that run when it’s safe.
- Each time they run, they can refresh the number of ticks until they should run again.
- Much better than the old heartbeat tasks.

Let's Take a Look

- First the header.
- Then we do our periodic stuff.
- Then we reset the period and return.



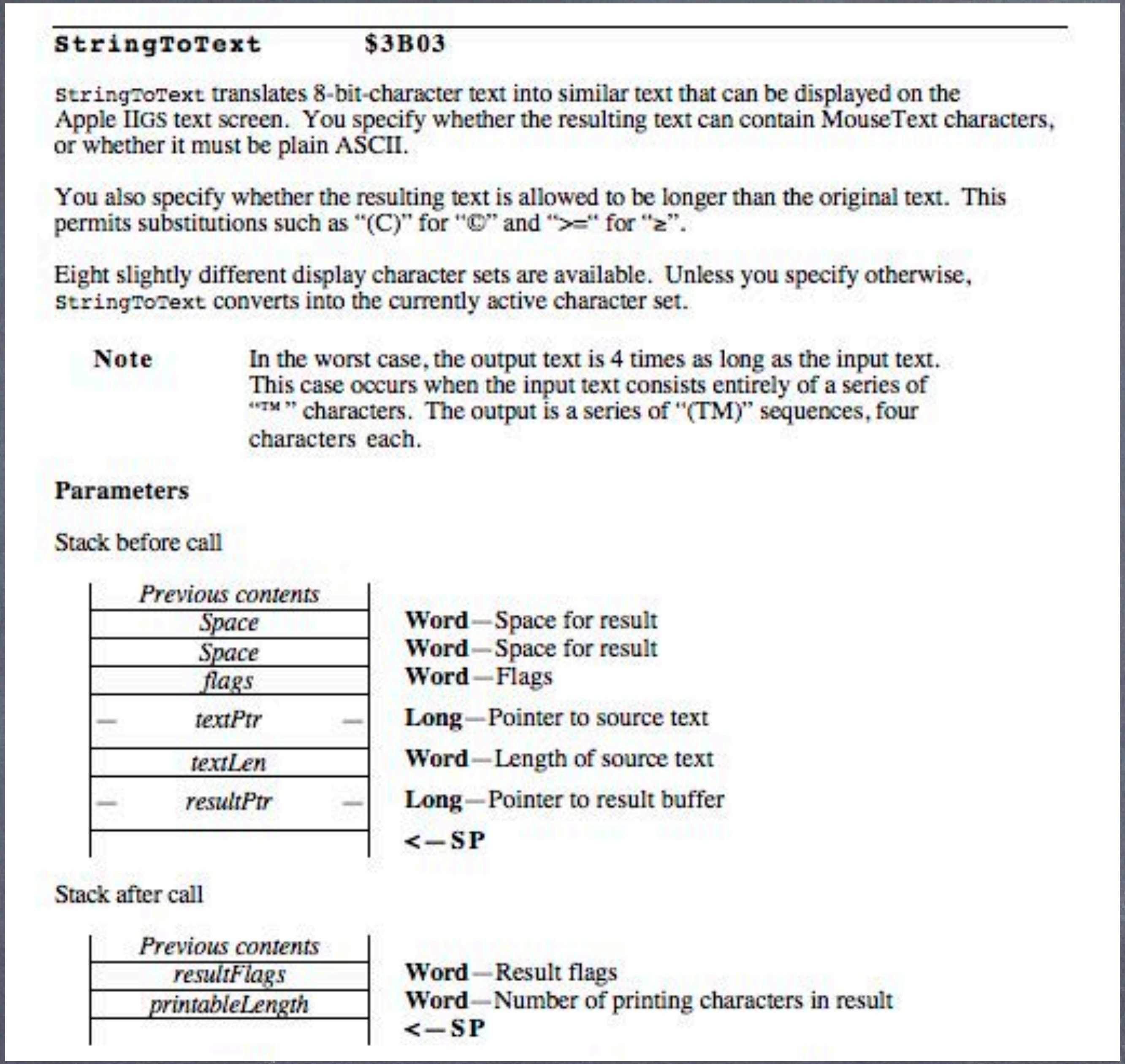
```
MainRunQTask    start
         ds     4
mainQPeriod     entry
         dc     i2'60'
         dc     i2'$A55A'

         // Do stuff here

         lda    #60
         sta    >mainQPeriod
         rtl
         end
```

Fun With Strings

- ASCII is limited to basic Roman characters.
- “Mac Roman” encoding offers a lot of special characters, such as â, π, and so forth.
- How to handle those on the text screen?



`StringToText` **$3B03**

`StringToText` translates 8-bit-character text into similar text that can be displayed on the Apple IIGS text screen. You specify whether the resulting text can contain MouseText characters, or whether it must be plain ASCII.

You also specify whether the resulting text is allowed to be longer than the original text. This permits substitutions such as "(C)" for "©" and ">=" for "≥".

Eight slightly different display character sets are available. Unless you specify otherwise, `StringToText` converts into the currently active character set.

Note In the worst case, the output text is 4 times as long as the input text. This case occurs when the input text consists entirely of a series of "™" characters. The output is a series of "(TM)" sequences, four characters each.

Parameters

Stack before call

Previous contents	
Space	**Word**—Space for result
Space	**Word**—Space for result
flags	**Word**—Flags
— *textPtr* —	**Long**—Pointer to source text
textLen	**Word**—Length of source text
— *resultPtr* —	**Long**—Pointer to result buffer
	<—SP

Stack after call

Previous contents	
resultFlags	**Word**—Result flags
printableLength	**Word**—Number of printing characters in result
	<—SP

StringToText

What Does It Do?

- Give it a string and a target language.
- Returns a string with the special characters translated into ASCII when possible.
- For example, © becomes “(c)”.
- Optionally strips out inconvertible characters.

StringToText Tips

- You provide the destination buffer.
- How big should you make it?
- Worst case string: all ™ characters. Each expands to “(TM)”.
- Make your buffer 4x the original string length.

Other Things to Check Out

- Scheduler
- Don’t forget about Programmer’s Reference for System 6 and 6.0.1!

Q&A

Stump the Geek Time!