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 \textbf{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.

\textbf{Parameters}

\textbf{Stack before call}

\begin{tabular}{|c|c|}
\hline
\textit{previous contents} & \\
\hline
\textit{changedFlag} & \textbf{Word}—Indicating that device's position and/or button state has changed \\
\textit{modLatch} & \textbf{Byte}—Keyboard modifiers latch \textbf{1 byte}—Set to 0 \\
\textit{padding} & \textbf{Word}—Device's clamped absolute X position \\
\textit{xPosition} & \textbf{Word}—Device's clamped absolute Y position \\
\textit{yPosition} & \textbf{Word}—Device's button status \\
\textit{buttonStatus} & \hline
\end{tabular}

\textbf{Stack after call}

\begin{tabular}{|c|c|}
\hline
\textit{previous contents} & \hline
\textit{SP} & \hline
\end{tabular}

\textbf{Errors}

None

\textbf{C}

Call cannot be made from C.
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

<table>
<thead>
<tr>
<th>previous contents</th>
<th>changedFlag</th>
<th>modLatch</th>
<th>padding</th>
<th>xPosition</th>
<th>yPosition</th>
<th>buttonStatus</th>
</tr>
</thead>
</table>

Stack after call

<table>
<thead>
<tr>
<th>previous contents</th>
</tr>
</thead>
</table>

Errors

None.

C

Call cannot be made from C.

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

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

<table>
<thead>
<tr>
<th>previous contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>changedFlag</td>
</tr>
<tr>
<td>modLatch</td>
</tr>
<tr>
<td>xPosition</td>
</tr>
<tr>
<td>yPosition</td>
</tr>
<tr>
<td>buttonStatus</td>
</tr>
<tr>
<td>Word—Indicating that device's position and/or button state has changed</td>
</tr>
<tr>
<td>Byte—Keyboard modifiers latch</td>
</tr>
<tr>
<td>Word—Device's clamped absolute X position</td>
</tr>
<tr>
<td>Word—Device's clamped absolute Y position</td>
</tr>
<tr>
<td>Word—Device's button status</td>
</tr>
<tr>
<td>← SP</td>
</tr>
</tbody>
</table>

Stack after call

<table>
<thead>
<tr>
<th>previous contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>← SP</td>
</tr>
</tbody>
</table>

Errors

None

C

Call cannot be made from C.
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

<table>
<thead>
<tr>
<th>previous contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>changedFlag</td>
</tr>
<tr>
<td>modLatch</td>
</tr>
<tr>
<td>xPosition</td>
</tr>
<tr>
<td>yPosition</td>
</tr>
<tr>
<td>buttonStatus</td>
</tr>
</tbody>
</table>

Word—indicating that device’s position and/or button state has changed

Byte—Keystroke

Word—Button mask

Word—Button mask

Stack after call

<table>
<thead>
<tr>
<th>previous contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟷ SP</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>previous contents</th>
<th>Word—Indicating that device's position and/or button state has changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>changedFlag</td>
<td>Byte—Keyboard modifiers latch 1 Byte—Set to 0</td>
</tr>
<tr>
<td>modLatch</td>
<td>paddingle</td>
</tr>
<tr>
<td>xPosition</td>
<td>Word—Device's clamped absolute X position</td>
</tr>
<tr>
<td>yPosition</td>
<td>Word—Device's clamped absolute Y position</td>
</tr>
<tr>
<td>buttonStatus</td>
<td>Word—Device's button status</td>
</tr>
<tr>
<td>← SP</td>
<td></td>
</tr>
</tbody>
</table>

Stack after call

<table>
<thead>
<tr>
<th>previous contents</th>
<th>← SP</th>
</tr>
</thead>
</table>

Errors

None

C

Call cannot be made from C.
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(6, 0, pt.h, pt.v, 0x4000);
```

Button and mouse position both changed
Click! Click, I say!

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!

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

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

<table>
<thead>
<tr>
<th>Previous contents</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>= newEntryPtr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>= headerPtr</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Long—Pointer to element to add to queue
Long—Pointer to first queue element

<—SP

Stack after call

<table>
<thead>
<tr>
<th>Previous contents</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>&lt;—SP</td>
</tr>
</tbody>
</table>

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 $2403$

Deletes a specified element from a queue.

**Parameters**

Stack before call

- **Previous contents**
  - `entryPtr`
  - `headerPtr`

  Long—Pointer to element to delete from queue
  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**

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

Pointer entryPtr, headerPtr;
```
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.
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

string-to-text 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

<table>
<thead>
<tr>
<th>Previous contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
</tr>
<tr>
<td>Space</td>
</tr>
<tr>
<td>flags</td>
</tr>
<tr>
<td>textPtr</td>
</tr>
<tr>
<td>textLen</td>
</tr>
<tr>
<td>resultPtr</td>
</tr>
</tbody>
</table>

Word — Space for result
Word — Space for result
Word — Flags
Long — Pointer to source text
Word — Length of source text
Long — Pointer to result buffer
<— SP

Stack after call

<table>
<thead>
<tr>
<th>Previous contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>resultFlags</td>
</tr>
<tr>
<td>printableLength</td>
</tr>
</tbody>
</table>

Word — Result flags
Word — Number of printing characters in result
<— SP
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!