Macintosh Technical Notes

#202: Resetting the Event Mask

See also: *Inside Macintosh*, Volume II, The Event Manager

Written by: Chris Knepper August 1, 1988
Revised by: Chris Knepper December 1988

As of System 4.2 and Finder 6.0, applications which alter the event mask must now restore the original event mask when quitting.
Changes since August 1, 1988: Added a related MultiFinder anomaly.

In most cases, applications should **not** modify the system event mask, which means they should avoid calling `SetEventMask` and not alter the low-memory global `SysEvtMask`. Modifying the event mask is of no use to most applications, and the only situation in which an application might need to modify it is to detect key-up events. Only those developers creating applications which must detect key-up events need to know the information presented in this Technical Note. Other developers should avoid altering the system event mask at all costs.

Since the system event mask normally prevents key-up events from being posted, those applications which need to detect key-up events call `SetEventMask` during initialization to enable key-up events. This process might be as follows:

```
myMask := EveryEvent;
SetEventMask(myMask);
```

Applications which make this call during initialization, **must** save the event mask prior to calling `SetEventMask` and restore the event mask when quitting. Given the following definitions and declarations in MPW Pascal:

```
CONST  SysEvtMask = $144;
TYPE   IntPtr = ^INTEGER;
VAR    saveMask: INTEGER;
       MaskPtr: IntPtr;
```

you save the event mask as follows:

```
{ set up our event mask pointer }
MaskPtr := IntPtr(SysEvtMask);
{ save the event mask }
saveMask := MaskPtr^;
```

and restore the event mask as follows:

```
{ restore the event mask }
MaskPtr^ := saveMask;
```

Finder Anomaly

When MultiFinder is disabled, users will notice a strange behavior in the Finder (versions 6.0 and later) after quitting applications which fail to restore the event mask. If an application failed to restore the event mask when quitting and had set the event mask to mask out mouse-up events, all mouse-up events would continue to be masked out, and the user would notice that the Finder no longer recognizes double clicks.

MultiFinder Anomaly

With the current Macintosh architecture, the interrupt handlers which service hardware interrupts call `_PostEvent` to post events in the event queue, so events are inserted in the queue unless they are masked out by `SysEvtMask`. If an event is masked out by `SysEvtMask`, then `_PostEvent` returns a `evtNotEnb` error and does not insert the event in the queue.

Applications normally retrieve events which have been successfully posted to the event queue by calling `_GetNextEvent` or `_WaitNextEvent`. If the event being retrieved is not masked out by the event mask (`mask`) which is supplied to these routines, then the routines will return `TRUE`.

Under MultiFinder, `SysEvtMask` is an application-specific global variable which is switched during context switches; therefore, under MultiFinder, if an application must alter `SysEvtMask`, it must also be prepared to handle all events, even those which normally would be masked out by `SysEvtMask`.

This anomaly occurs when MultiFinder switches out `SysEvtMask` during a “minor switch” (i.e., switching from foreground to background to service background applications). Interrupt service routines operating at interrupt time call `_PostEvent` to post events, and `_PostEvent` masks out events using the mask in `SysEvtMask` to determine whether or not to post the event. This scheme means that events are posted as they occur, regardless of whether the current value of `SysEvtMask` belongs to the foreground application or a background application; therefore, it is possible for the event queue to not contain key-up events, even though the foreground application enabled them, because the background application could mask them out.

A future version of the System Software will account for this problem by using the foreground application’s event mask when events are posted (even if a background application is running when the interrupt service routine is called).