ProDOS 8
#9: Buffer Management Using BASIC.SYSTEM

Revised by: Matt Deatherage November 1988
Revised by: Pete McDonald October 1985

This Technical Note discusses methods for allocating buffers which will not be arbitrarily deallocated in BASIC.SYSTEM.

Section A.2.1 of the *ProDOS 8 Technical Reference Manual* describes in detail how an application may obtain a buffer from BASIC.SYSTEM for its own use. The buffer will be respected by BASIC.SYSTEM, so if you choose to put a program or other executable code in there, it will be safe.

However, BASIC.SYSTEM does not provide a way to selectively deallocate the buffers it has allocated. Although it is quite easy to allocate space by calling `GETBUFR` ($BEF5) and also quite easy to deallocate by calling `FREEBUFR` ($BEF8), it is not so easy to use `FREEBUFR` to deallocate a particular buffer.

In fact, `FREEBUFR` always deallocates all buffers allocated by `GETBUFR`. This is fine for transient applications, but a method is needed to protect a static code buffer from being deallocated by `FREEBUFR` for a static application.

Location `RSHIMEM` ($BEFB) contains the high byte of the highest available memory location for buffers, normally $96. `FREEBUFR` uses it to determine the beginning page of the highest (or first) buffer. By lowering the value of `RSHIMEM` immediately after the first call to `GETBUFR`, and before any call to `FREEBUFR`, we can fool `FREEBUFR` into not reclaiming all the space. So although it is not possible to selectively deallocate buffers, it is still possible to reserve space that `FREEBUFR` will not reclaim.

Physically, we place the code buffer between BASIC.SYSTEM and its buffers, in the space from $99FF down.

After creating the protected static code buffer, we can call `GETBUFR` and `FREEBUFR` to maintain temporary buffers as needed by our protected module. `FREEBUFR` will not reclaim the protected buffer until after `RSHIMEM` is restored to its original value.

The following is a skeleton example which allocates a two-page buffer for a static code module, protects it from FREEBUFR, then deprotects it and restores it to its original state.

```
START       LDA #$02                ;get 2 pages
            JSR GETBUFR
            LDA RSHIMEM             ;get current RSHIMEM
            SEC                     ;ready for sub
            SBC #$02                ;minus 2 pages
            STA RSHIMEM             ;save new val to fool FREEBUFR
            JSR FREEBUFR            ;CALL FREEBUFR to deallocate.
```

At this point, the value of RSHIMEM is the page number of the beginning of our protected buffer. The static code may now use GETBUFR and FREEBUFR for transient file buffers without fear of freeing its own space from RSHIMEM to $99FF.

To release the protected space, simply restore RSHIMEM to its original value and perform a JSR FREEBUFR.

```
END         LDA RSHIMEM             ;get current val
            CLC                     ;ready for ADD
            ADC #2                  ;give back 2 pages
            STA RSHIMEM             ;tell FREEBUFR about it
            JSR FREEBUFR            ;DO FREEBUFR
            RTS
```

You can reserve any number of pages using this method, as long as the amount you reserve is within available memory limits.

Further Reference

- *ProDOS 8 Technical Reference Manual*