Pascal III: Accessing the extra memory (3 of 5)

```
        .MACRO SetBank
        ; This macro saves %1 (an extended address bank pointer) in Save%1
        ; and pops the new value. The code follows a convention that if the new
        ;
        ; 1. check the address currently pointed to by corresponding word
        ;    in Zero page, and
        ; 2. modify it and %1 (bank register) to make sure the address does
        ;    not point to zero page of bank pair to avoid holes in the memory
        ;    map.
        ;
        ; Zero page wraparound during execution taken care of by main loop.

        LDA     %1
        STA     Save%1          ; Save old one

        PLA                     ; Get low order byte of new bank
        CMP     #0FF            ; Don't use if = to -1
        BEQ     $1
        ORA     #80             ; Set extended addressing on
        STA     %1              ; And save bank

$1      LDA     %1-1601+1  ; Check for zero page reference of
                           ; starting pointers (rest of
                           ; algorithm guarantees it will never
        BNE     $3         ; happen again)
        LDA     #80        ; If was nn:00xx , change to
                           ; nn-1:80xx
        STA     %1-1601+1
        DEC     %1

$3      PLA
        .ENDM

        ; The following macro guarantees that the base pointer %1 will not
        ; wrap into zero page during next 256 increments of the pointer.
        .MACRO TestWrap
        LDA     %1+1         ; Before moving each page,
                             ; check for wraparound
        CMP     #0FF
        BCC     $1
        SBC     #80
        STA     %1+1
```

```
        INC     %1+1601
$1      .ENDM

;===============================================================
        ; Main procedure
    .PROC FetchBytes,6

Source  .EQU    0E0             ; Zero page pointer to read
                                ; bytes through
SrcBank .EQU    1601+Source
Dest    .EQU    0E2             ; Zero page pointer to write
                                ; bytes through
DstBank .EQU    1601+Dest

; temps
Count       .EQU 030  ; Three byte counter of # of bytes to move
SaveDstBank .EQU 033  ; Allow us to restore Pascal bank at end
SaveSrcBank .EQU 034  ; of subroutine

;===============================================================
        ; Initializations

        PULL    RetAddr         ; Get information off stack

        ; Convert 4 bytes of page/byte count to
        ; three byte integer
        PULL    Count

        PLA                ; low order byte of page count
        CLC
        ADC     Count+1    ; add two (low order) page counts
                           ; together
        STA     Count+1
        PLA                ; now get high order byte
        ADC     #0         ; and add carry to high order count
        STA     Count+2

        ; now pull destination and source off stack
        PULL    Dest
        SetBank DstBank
        PULL    Source
        SetBank SrcBank

        ; END   Initializations
;===============================================================
        ; test to see if we need to move another whole page.
MovePg  TestWrap Dest       ; guarantee copy loop works for next
        TestWrap Source     ; 256 bytes

        LDA     Count+1         ; Y pages to copy?
        BNE     PgLoop          ; yes, go copy them
        LDA     Count+2         ; X chunks of 256 pages?
        BEQ     Partial         ; no, just copy last fragment
```

```
        DEC     Count+2         ; yes,go copy it (count+1 now
                                ;  contains $100)

        ; now copy one page
PgLoop  LDY     #0              ; move one page. This is key loop.
$1      LDA     @Source,Y       ; get data
        STA     @Dest,Y         ; store it
        INY
        BNE     $1

        ; change to next page
        INC     Source+1
        INC     Dest+1
        DEC     Count+1
        JMP     MovePg
Apple Tech Notes


Keywords:  <None>

==========================================================================

This information is from the Apple Technical Information Library.

19960215 11:05:19.00
```
Tech Info Library Article Number: 641