



Tech Info Library

Pascal III: Accessing the extra memory (4 of 5)

Revised: 11/30/84
Security: Everyone

Pascal III: Accessing the extra memory (4 of 5)

```

=====
; At this point, there is an assumption that
; 1. move partial page and
; 2. Testwrap on source and dest
; have been done so we can move at least 256 bytes without
; thinking about it again.
Partial LDA    Count          ; all done yet?
        BEQ    Return
        LDY    #0             ; now move (low byte of Count)
                                ; # of bytes
$1      LDA    @Source,Y
        STA    @Dest,Y
        INY
        CPY    Count
        BNE    $1
;=====
Return  LDA    SaveSrcBank    ; and go home
        STA    SrcBank
        LDA    SaveDstBank
        STA    DstBank
        PUSH   RetAddr
        RTS

Retaddr .WORD

        .FUNC   Atsign,1
; x := Atsign(Y) causes X to point to Y
        PULL   RetAddr
        PLA
        PLA
        PLA
        PLA
        PUSH   RetAddr
        RTS

RetAddr .WORD
Loc     .WORD

```

.END

```

=====
{$SETC Debug := TRUE}
UNIT StringSpace;

INTERFACE
  TYPE
    STRING1   = STRING[1];
    STRING255 = STRING[255];
    STRPTR    = INTEGER;

    FUNCTION  InitStringSpace(ColdStart:BOOLEAN): INTEGER;
    PROCEDURE FreeStringSpace;
    FUNCTION  PutString(VAR S:STRING1; VAR WHERE:STRPTR):
                BOOLEAN;
    PROCEDURE GetString(Who:STRPTR; VAR S:STRING255);
                {S had better be 256 bytes long!}
IMPLEMENTATION

  VAR
    SegNum,      {segment number of memory chunk}
    Bank,        {what bank is chunk in}
    Base,        {start of chunk received (byte address)}
    Tos,         {next word to allocate}
                {(base rel word address)}
    Limit: INTEGER; {last word allocatable}
                {(base rel word address)}

    PROCEDURE Allocate(VAR NumPages, Segnum, Bank, SegBase:
                        INTEGER); EXTERNAL;
    {allocates a chunk of SOS memory:
    Input:
      NumPages: Maximum number of pages to try for.
    Output:
      NumPages: Number of pages actually allocated.
      SegNum:   SOS Segment number (for deallocate)}
    { Bank:    Starting address bank number
      SegBase: Starting address byte address ($0200..$9E00)}

    PROCEDURE DeAllocate(SegNum:INTEGER); EXTERNAL;

    PROCEDURE FetchBytes(SrcBank:INTEGER; Source:INTEGER;
                        DstBank:INTEGER; Dest:INTEGER;
                        PageCount:INTEGER; Count:INTEGER
                        ); EXTERNAL;

    FUNCTION Atsign( VAR x:INTEGER):INTEGER; EXTERNAL;

    FUNCTION InitStringSpace{(ColdStart:BOOLEAN): INTEGER};
  VAR
    NumPages:INTEGER;
    TempBase:INTEGER;

```

```
BEGIN
  IF ColdStart THEN BEGIN
    FreeStringSpace;

    {$IFC Debug}
    WRITELN('How many pages to allocate?');
    READLN(NumPages);
    IF NumPages > 512 THEN NumPages := 512;
    {$ELSEC Debug}
    NumPages := 512;
    {$ENDC Debug}

    Allocate(NumPages, SegNum, Bank, TempBase);
    Base := TempBase - {$2000}8192;    {shift into}
                                         {extended address}
    Limit := NumPages*128 - 32767-1;  {128 = words/page}

    {$IFC Debug}
    WRITELN('Assembly results:');
    WRITELN('Segment ', Segnum, ' Allocated ', NumPages,
            ' pages', ' in bank ', Bank, ' at real address ',
            TempBase);
    IF Base MOD 256 <> 0 THEN
      WRITELN(CHR(7), 'bad base address');
    {$ENDC Debug}

    END;
    Tos := -32767-1;          {compiler doesn't like -32768}
    InitStringSpace := NumPages;
  END;
```

Apple Tech Notes

Tech Info Library Article Number:642