APPLE /// COMPUTER INFORMATION

Apple /// Business BASIC Peek/Poke Invokable Module Information

Source

Bank Switch Razzle-Dazzle: Peeking and Poking the Apple ///
Dr. John Jeppson
SOFTALK magazine -- August 1982 -- pages 38-48

Compiled By
David T Craig -- December 4, 2008

INTRODUCTION

This document contains information about the Peek/Poke invokable module that was created for the Apple /// computer in 1982 for use by the Apple ///'s BASIC language interpreter, Apple /// Business BASIC.

This invokable module allows Apple /// BASIC programmers to use PEEK and POKE commands which this BASIC's command set does not support. This information consists of the programmer usage instructions, the peek/poke invokable module source listing from its parent article, and the source listing courtesy of the Washington Apple Pi (WAP) Apple /// disk archive.

This source is written in 6502 assembly language for assembly by the Apple /// Pascal System 6502 assembler.

PEEK/POKE INVOKABLE USAGE INSTRUCTIONS

Down to Business. The accompanying assembly language program contains the function peek and the procedure poke. It depends primarily on extended addressing but, regrettably, uses less legal methods as well. After assembly, the resulting code can either be linked to a Pascal program or invoked from Basic as an invokable module. It works the same way in both languages.

Peek is a function and returns an integer value, the contents of the memory location at which you've peeked. The function requires two parameters. You must supply the address (as viewed by the 6502) and the Xbyte. Both are passed as integers. In Pascal you declare peek an external function:

```
function peek (addr, xbyte : integer) : integer;
external;
```

You may then make an assignment statement to an integer variable:

```
int := peek (addr, xbyte);
```

In Basic the process looks like this:

```
10 INVOKE "peek.poke.code": REM the pathname of the codefile.

100 int = EXFN%.peek(%addr,%xbyte)
```

Poke is similar, but since it doesn't return anything (except, occasionally, disaster), it is a procedure. It has a third parameter, the value to be poked. *Value* must also be an integer.

In Pascal:

```
procedure poke (addr, xbyte, value : integer);
external;
```

then one can use:

```
value := 128;
poke (addr, xbyte, value);
```

In Basic:

```
10 INVOKE "peek.poke.code"
100 value = 128
110 PERFORM poke(%addr,%xbyte,%value)
```

Don't forget that the variables are all decimal integers. You may want to enter them and display them as hexadecimal strings, but you will have to convert. Basic has handy built-ins: HEX\$(integer) and TEN(hexstring). In Pascal you will have to write your own.

The address can be any legal, ordinary integer. Value and Xbyte must be integers in the range 0 to 255. If they are greater, the integer MOD 256 is used. Only certain Xbyte values have meaning; all the rest are treated as 0. Table 2 has some useful Xbytes and some comments. There are a couple of peculiarities that you should know about:

- 1. Nothing terrible happens if you give the Xbyte of a bank pair that doesn't exist (yet)—for example, (\$8C = 140). Peek will either return \$FF, signifying nothing, or some value from one of the existing banks—also of little use.
- 2. The artificial Xbyte \$FF (decimal 255) isn't actually used as an Xbyte. It is merely a signal to the function to do all sorts of illegal things to the environment register, zero-page register, and interrupts in order to get at areas normally inaccessible. With this "Xbyte" you get a block that looks like ordinary (system) addressing but with "true" zero-page and "true" (\$01) stack-page. Also, the area \$C000 to \$CFFF is "I/O space," and \$F000 to \$FFFF is the read-only memory used in the boot process.

```
Hex
     Decimal
                'ordinary" system bank. User bank at $2000..$9FFF
$00
          0
$80
       128
               bank pair 0,1
               bank pair 2,3 — bank 3 nonexistent in 128K machine
$82
       130
               bank pair 6,7 — bank 7 nonexistent in 256K machine
$86
       134
               like system bank. Bank 0 to $2000..$9FFF. ALL RAM!
S8F
       143
$FF
       255
                "artificial" — gives a system type bank with
                      1. "true" zero-page and stack-page
                      2. $C000 to $CFFF = I/O space
                       3. $F000 to $FFFF = ROM
                      Table 2. Xbyte values.
```

Note: There are locations on \$C100 page of I/O space that will cause the computer to "hang" just by reading them. It really isn't dangerous, but you'll have to reboot.

A Program by Any Other Name. Boot up Pascal, enter the editor, and type in the program. Capital letters are not required. Neither are the comments, but it would be a shame if you left out all of them. You can save a lot of typing by just typing in peek and duplicating it with the copy buffer. Then go through and make the necessary changes to convert one of them to poke. Save the program on disk. Use a path name of ten characters or less and permit the editor to add the suffix TEXT to your path name (for example, peek.poke.text).

Next, enter the assembler and assemble the program. The assembler will want to add the suffix .code. Let it. Otherwise the resulting file will not be type named code file and will not invoke properly. Later you can change the name (for example, peek.poke.inv) and the type name won't be affected.

The output of the assembler is the invokable module. Move it is your Basic disk and invoke it by its path name. You can then use either peek or poke at will in your program. Details of the required Basic program syntax may be found starting on page 160 of the Apple III Basic Manual.

Pascal is even simpler. You just declare peek and poke as external and use the linker to add them to your program.

PEEK/POKE INVOKABLE SOURCE CODE LISTING (SOFTALK MAGAZINE)

		ADDRESS BANKSW ZEROPG ENVRMT RETURN XBYTE RESULT	.EQU .EQU .EQU .FUNC .FUNC .WORD .WORD	OE8 OFFEF OFFDO OFFDF PEEK,2 BEGIN O O	;zeropage "pseudo" register
		OLD_XBT	.BYTE	0	
		OLDZPG ENV	.BYTE	0	
		BEGIN	.BYTE POP	0 RETURN	
		DEGIN	PLA	RETORIA	;"dummy" bytes for function
			PLA		, dominy bytes for function
			PLA		
			PLA		
			POP POP	XBYTE	parameters come off in reverse order
			LDA	ADDRESS + 1601	
			STA	OLDXBT	;save original x-byte value
			•		;which bank is desired
			LDA	XBYTE	,
			CMP	#OFF	;FF = ROM #1, C0-CF = I/O,
			BEQ	SPECIAL	; "true" 00 and 01 pages
			CMP	#80	
			BMI CMP	SYSTEM	;80-8F=extended addressing
	PEEK.POKE.TEXT — Source Code		BMI	#90 EXTEND	; else system bank (ordinary 6502)
.MACRO	POP		D/111	EXIEND	;handle system bank
PLA		SYSTEM	LDY	#0	manate system bank
STA	% 1	0.0.2	STY	ADDRESS + 1601	
PLA			LDA	@ADDRESS,Y	xbyte = 0 so get ordinary 6502 indirect indexed addressing
STA	% 1+1		STA	RESULT	; indirect indexed addressing
.ENDM			JMP	DONE	
.MACRO	PUSH				;handle extended addressing to a
LDA	%1+1				bankpair or \$8F
PHA	0/1	EXTEND	STA	ADDRESS+1601	place extend byte
LDA PHA	% 1		LDY	#0	•
.ENDM			LDA	@ADDRESS,Y	;"extended" addressing to desired
			STA	RESULT	; bank pair

					CMP	#ODF	
	JMP	DONE	;handle artificial bank 'FF'		BEQ	DONE	;if you really want to crash, just start
SPECIAL	LDA	ADDRESS+1	manale armicial bank FF		CMP	#OEF	POKing into SOS
	BEQ	TRUEPGS	;true \$00, \$01 desired?				(RAM \$8800 = FFFF)
	CMP	#1			BEQ	DONE	; soon he will get very sick
	BEQ	TRUEPGS	BOW#1 > 50000 FFFF	\$2	LDA	XBYTE	;detect artificial bank 'FF'
			;ROM#1 —> F0000-FFFF, C000-CFFF—>I/O		CMP	#OFF	;FF = ROM #1, C0-CF = I/O
	PHP		;save status, then disable interrupts		BEQ	SPECIAL	; "true" 00 and 01 pages
	SEI		;(an "illegal" move)	CVCTEAL	LDV	"	;handle system bank
	LDA	ENVRMT	;save environment	SYSTEM	LDY STY	#0 ADDRESS+1601	;xbyte = 0 so get ordinary 6502
	STA	ENV			LDA	VALUE	; indirect indexed addressing
	LDA	#73	;#% 0111 0011 — new environment reg		STA	@ADDRESS,Y	,
	STA	ENVRMT	;(an "illegal" move)		JMP	DONE	
	LDY	#0	,				;handle extended addressing to a
	STY	ADDRESS+1601	;system bank xbyte = 00	EXTEND	STA	ADDRESS 1401	bankpair or \$8F
	LDA STA	@ADDRESS,Y		EXTEND	LDY	ADDRESS + 1601 #0	;place extend byte
	LDA	RESULT ENV	restore ENVRMT		LDA	VALUE	
	STA	ENVRMT	, estore Elevelet		STA	@ADDRESS,Y	;"extended" addressing to desired
	PLP		restore status (including interrupts)				; bank pair
	JMP	DONE			JMP	DONE	beedle carried to the
			;desired address on true 00 or	SPECIAL	LDA	ADDRESS+1	;handle artificial bank `FF'
TRUEPGS	PHP		01 page	0. 20	BEQ	TRUEPGS	true zp or \$01 desired?
INOLI OS	SEI		;save status, then disable interrupts ;(an "illegal" move)		CMP	#1	·
	LDX	ADDRESS	;load BEFORE leaving old z-page		BEQ	TRUEPGS	
	LDY	ADDRESS+1					;ROM#1 —> F000-FFFF,C000-CFFF
	LDA	ZEROPG	;save old zpg		PHP		—>I/O
	STA LDA	OLDZPG #0	abanana anna 4 0 atanb ta 1		SEI		;save status, then disable interrupt: ;(an "illegal" move)
	STA	ZEROPG	;changes zero-page to 0, stack to 1 ;(an "illegal" move)		LDA	ENVRMT	;save environment
	TYA	22.00.0	is high byte 00 or 01		STA	ENV	,
	BEQ	\$1			LDA	#73	;#% 0111 0011 = new environment
	LDA	0100,X	;indexed addressing (x = addr)				reg
••	JMP	\$2			STA	ENVRMT	;(an "illegal" move)
\$1 \$2	LDA STA	0000,X RESULT			LDY STY	#0 ADDRESS+1601	;system bank xbyte = 00
	LDA	OLDZPG	;restore ZEROPG (and stack page)		LDA	VALUE	,system bank xbyte — oo
	STA	ZEROPG	,		STA	(a) ADDRESS, Y	
	PLP		;restore interrupts (status)		LDA	ENV	restore ENVRMT
DONE	LDA	OLDXBT	restore Pascal's xbyte		STA	ENVRMT	
	STA PUSH	ADDRESS+1601 RESULT			PLP JMP	DONE	restore status (including interro:
	PUSH	RETURN			37411	DOINE	;desired address on true 00 or 01
	RTS						page
	.PROC	POKE,3		TRUEPGS	PHP		;save status, then disable interrupts
	JMP	BEGIN			SEI		;(an "illegal" move)
RETURN XBYTE	.WORD .WORD	0			LDX	ADDRESS	;load BEFORE leaving old z-page
VALUE	.WORD	Ö			LDY LDA	ADDRESS+1 ZEROPG	serve ald see
OLD_XBT	BYTE	ŏ			STA	OLDZPG	;save old zpg
OLD_ZPG	.BYTE	0			LDA	#0	;changes zero page to 0, stack to 1
OLDENV	.BYTE	0			STA	ZEROPG	;(an "illegal" move)
ENV BEGIN	.BYTE POP	O BETURNI			LDA	VALUE	
BEGIN	POP	RETURN VALUE	parameters come off in reverse order		CPY BEQ	#0 \$ 1	is high byte 00 or 01
	POP	XBYTE			STA	0100,X	;indexed addressing $(x = addr)$
	POP	ADDRESS			JMP	\$2	,,
	LDA		;save original x-byte value	\$1	STA	• 0000,X	
	STA LDA	OLDXBT	ENN/BAAT	\$2	LDA	OLDZPG	;restore ZEROPG (and stack page)
	STA	ENVRMT OLDENV	;save ENVRMT		STA PLP	ZEROPG	restore interrupts (status)
	AND	#0F7	for POKE, enable write C000 to FFFF	DONE	LDA	OLDXBT	restore Pascal's xbyte
	STA	ENVRMT			STA	ADDRESS+1601	• · · · · · · · · · · · · · · · · · · ·
	104	VAVTE	;which bank is desired		LDA	OLD_ENV	restore C0-CF read/write status
	LDA CMP	XBYTE #80			STA	ENVRMT	
	BMI	#80 \$1	;80-8F=extended addressing		PUSH RTS	RETURN	
	CMP	#90	; else system bank (ordinary 6502)		.END)I
	BMI	EXTEND	(3.2, 3.3)				
			;disallow certain addresses				
\$1	LDA	ADDRESS+1	;POKE disallowed at (system bank):				
	CMP BNE	#0FF \$2	; BANKSW = FFEF ; ENVRMT = FFDF				
	PINE	4 4	; ZEROPG = FFD0				
	LDA	ADDRESS	,				
	CMP	#000	; in this program — suicide certain				
	BEQ	DONE	; in your program — suicide probable				

Apple /// Computer Information

Page 6 of 11

PEEK/POKE INVOKABLE SOURCE CODE LISTING (WAP DISK ARCHIVE)

```
The procedure PEEK and the function POKE
written by John Jeppson
published in SOFTALK magazine: AUG 82 in the article
; BANK SWITCH RAZZLE DAZZLE: Peeking and Poking the Apple ///
; MACROS
.MACRO POP
    PLA
    STA
        %1
    PLA
    STA
        %1+1
    .ENDM
    .MACRO PUSH
    LDA
        %1+1
    PHA
    LDA
        %1
    PHA
    . ENDM
EQUATES
ADDRESS .EQU
        0E8
                     ;zeropage "pseudo" register
BANKSW .EQU
        0FFEF
ZEROPG .EQU
        0FFD0
ENVRMT
   . EQU
        0FFDF
PEEK, 2
    . FUNC
\mathsf{JMP}
        BEGIN
RETURN .WORD
XBYTE
    .WORD
        0
RESULT .WORD
        0
   .BYTE
OLDXBT
        0
OLDZPG .BYTE
        0
ENV
    .BYTE
        0
BEGIN
    POP
        RETURN
    PLA
                     ; "dummy" bytes for function
    PLA
    PLA
    PLA
    POP
        XBYTE
                     ;parameters come off in reverse order
    POP
        ADDRESS
```

```
LDA
                ADDRESS+1601
                                       ;save original x-byte value
        STA
               OLDXBT
             ;which bank is desired
       LDA
               XBYTF
       CMP
                #0FF
                                        ; FF = ROM #1, CO-CF = I/O,
                SPECIAL
                                             "true" 00 and 01 pages
        BEQ
       CMP
                #80
       BMI
                SYSTEM
                                        ;80-8F=extended addressing
                #90
       CMP
                                        ; else system bank (ordinary 6502)
       BMI
               EXTEND
             ;handle system bank
SYSTEM LDY
                #0
                ADDRESS+1601
        STY
                                      ;xbyte = 0 so get ordinary 6502
       LDA
                @ADDRESS,Y
                                       ; indirect indexed addressing
        STA
                RESULT
        JMP
                DONE
             ; handle extended addressing to a bankpair or $8F
EXTEND STA
               ADDRESS+1601
                                       ;place extend byte
       LDY
                #0
       LDA
                @ADDRESS,Y
                                        ; "extended" addressing to desired
                RESULT
       STA
                                               bank pair
        JMP
                DONE
             ;handle artifical bank 'FF'
SPECIAL LDA
               ADDRESS+1
       BFO
                TRUEPGS
                                       ;true $00, $01 desired?
        CMP
                #1
        BEQ
                TRUEPGS
             ;ROM#1 --> F000-FFFF, C000-CFFF --> I/O
       PHP
                                        ;save status, then disable interrupts
        SEI
                                        ;(an "illegal" move)
                ENVRMT
       LDA
                                        ;save environment
        STA
                ENV
       LDA
                #73
                                       ;#% 0111 0011 - new environment reg
        STA
                ENVRMT
                                       ;(an "illegal" move)
       LDY
                #O
        STY
               ADDRESS+1601
                                       ;system bank xbyte = 00
        LDA
                @ADDRESS, Y
        STA
               RESULT
       LDA
                ENV
                                        ;restore ENVRMT
               ENVRMT
        STA
       PI P
                                        ;restore status (including interrupts)
                DONE
        JMP
            ;desired address on true 00 or 01 page
TRUEPGS PHP
                                        ;save status, then disable interrupts
        SFT
                                        ;(an "illegal" move)
       LDX
               ADDRESS
                                       ;load BEFORE leaving old z-page
       LDY
                ADDRESS+1
                ZEROPG
        LDA
                                       ;save old zpg
        STA
                OLDZPG
        LDA
                #0
                                        ;changes zero-page to 0, stack to 1
               ZEROPG
                                        ; (an "illegal" move)
        STA
```

Apple /// Computer Information

Page 8 of 11

Apple /// Business BASIC Peek/Poke Invokable Module Information Bank Switch Razzle-Dazzle: Peeking and Poking the Apple /// Dr. John Jeppson -- SOFTALK magazine, August 1982, pages 38-48

```
TYA
                                   ; is high byte 00 or 01
       BEQ
              $1
       LDA
              0100,X
                                   ; indexed addressing (x = addr)
       JMP
              $2
$1
       LDA
              0000,X
       STA
$2
              RESULT
       LDA
              OLDZPG
                                   ;restore ZEROPG (and stack page)
       STA
              ZEROPG
       PLP
                                   ;restore interrupts (status)
DONE
       LDA
              OLDXBT
                                   ;restore Pascal's xbyte
       STA
              ADDRESS+1601
       PUSH
              RESULT
       PUSH
              RETURN
       RTS
.PROC
              POKE, 3
JMP
              BEGIN
RETURN .WORD
              0
      .WORD
XBYTE
              0
       .WORD
VALUE
              0
OLDXBT
       . BYTE
              0
OLDZPG
      .BYTE
              0
OLDENV
      .BYTE
              0
ENV
       .BYTE
              0
BEGIN
       POP
              RETURN
                                   ;parameters come off in reverse order
       POP
              VALUE
       POP
              XBYTE
       POP
              ADDRESS
       LDA
              ADDRESS+1601
                                   ;save original x-byte value
       STA
              OLDXBT
       LDA
              ENVRMT
                                   ;save ENVRMT
       STA
              OLDENV
       AND
                                   ;for POKE, enable write C000 to FFFF
              #0F7
       STA
              ENVRMT
           ;which bank is desired
       LDA
              XBYTE
       CMP
              #80
              $1
                                   ;80-8F=extended addressing
       BMI
       CMP
              #90
                                        else system bank (ordinary 6502)
       BMT
              EXTEND
           ;disallow certain addresses
$1
       LDA
              ADDRESS+1
                                    ; POKE disallowed at (system bank):
                                               BANKSW = FFEF
       CMP
              #0FF
                                               ENVRMT = FFDF
       BNE
              $2
```

Apple /// Computer Information

Page 9 of 11

Apple /// Business BASIC Peek/Poke Invokable Module Information Bank Switch Razzle-Dazzle: Peeking and Poking the Apple /// Dr. John Jeppson -- SOFTALK magazine, August 1982, pages 38-48

```
ZEROPG = FFD0
        LDA
                ADDRESS
        CMP
                #0D0
                                          ; in this program - suicide certain
                DONE
        BE0
                                          ; in your program - suicide probable
                #0DF
        CMP
        BEQ
                DONE
                                          ; if you really want to crash, just star
        CMP
                #0EF
                                              POKing into SOS (RAM $B800 - FFFF)
        BEQ
                DONE
                                               soon he will get very sick
             :detect artificial bank 'FF'
$2
        LDA
                XBYTE
        CMP
                #0FF
                                          ; FF = ROM #1, CO-CF = I/O
        BE0
                SPECIAL
                                                "true" 00 and 01 pages
             ;handle system bank
SYSTEM LDY
                #0
                ADDRESS+1601
        STY
                                         ;xbyte = 0 so get ordinary 6502
        LDA
                VALUE
                                         ; indirect indexed addressing
        STA
                @ADDRESS, Y
        IMP
                DONE
             ;handle extended addressing to a bankpair or $8F
EXTEND STA
                ADDRESS+1601
                                         ;place extend byte
        LDY
                #0
        LDA
                VALUE
                                          ; "extended" addressing to desired
        STA
                @ADDRESS, Y
                                                 bank pair
        JMP
                DONE
             ;handle artifical bank 'FF'
SPECIAL LDA
                ADDRESS+1
        BEQ
                TRUEPGS
                                          ;true zp or $01 desired?
        CMP
                #1
                TRUEPGS
        BEQ
             ;ROM#1 --> F000-FFFF, C000-CFFF --> I/O
        PHP
                                         ;save status, then disable interrupts
                                          ;(an "illegal" move)
        SEI
        LDA
                ENVRMT
                                          ;save environment
        STA
                ENV
        LDA
                #73
                                         ;#% 0111 0011 - new environment reg
        STA
                ENVRMT
                                         ;(an "illegal" move)
        LDY
                #0
                ADDRESS+1601
                                         ;system bank xbyte = 00
        STY
        LDA
                VALUE
        STA
                @ADDRESS.Y
        LDA
                ENV
                                          ;restore ENVRMT
        STA
                ENVRMT
        PLP
                                          ;restore status (including interrupts)
        \mathsf{JMP}
                DONE
            ;desired address on true 00 or 01 page
                                          ;save status, then disable interrupts ;(an "illegal" move)
TRUEPGS PHP
        SEI
                ADDRESS
                                          ;load BEFORE leaving old z-page
        LDX
        LDY
                ADDRESS+1
```

Apple /// Computer Information

Page 10 of 11

Apple /// Business BASIC Peek/Poke Invokable Module Information Bank Switch Razzle-Dazzle: Peeking and Poking the Apple /// Dr. John Jeppson -- SOFTALK magazine, August 1982, pages 38-48

	LDA STA	ZEROPG OLDZPG	;save old zpg
	LDA STA LDA	#0 ZEROPG VALUE	;changes zero page to 0, stack to 1; (an "illegal" move)
	CPY BEQ	#0 \$1	;is high byte 00 or 01
	STA JMP	0100,X \$2	;indexed addressing (x = addr)
\$1	STA	0000,X	
\$2	LDA STA PLP	OLDZPG ZEROPG	;restore ZEROPG (and stack page)
			;restore interrupts (status)
DONE	LDA STA	OLDXBT ADDRESS+1601	;restore Pascal's xbyte
	LDA STA	OLDENV ENVRMT	;restore C0-CF read/write status
	PUSH RTS	RETURN	
	. END		

END OF INFORMATION