



Tech Info Library

Super Serial Card: Using with Machine Language (12/96)

Article Created: 22 September 89

Article Reviewed/Updated: 16 December 1996

TOPIC -----

This article describes assembly language addressing methods for the 6502 and 6551 microprocessors through the Super Serial Card.

DISCUSSION -----

The 6502 does a false read to the current page. This is inherent in the 6502 design. A false read occurs during a read to memory. The 6502 will hold the target address + 1 line open after it accesses the target address. This does not alter the contents of the address but can affect a memory-mapped I/O device that is toggled by the address line.

The false read does not affect the Super Serial Card as none of the card's functions are set when the address line is held open by the false read. However, for good programming to an I/O device, where false reads could toggle a function, you should use the indirect indexed-addressing mode with the address for your indirect accesses in the zero page.

The following example is available in the Tech Info Library and uses the absolute, indirect-addressing method; it has been modified here as an example of indirect, indexed-addressing. The program uses zero-page addresses \$FA and \$FB, because these are generally unused by both DOS and BASIC. See pages 74 and 75 of the "Apple II Reference Manual" for a map of the zero-page locations.

Super Serial Card: Accessing It Through Machine Language

Although Apple's Super Serial Card can be used from Applesoft BASIC, it is often desirable to use machine language to increase the speed with which characters are sent and received. The assembler program below illustrates a method of communicating with another computer through the Super Serial Card. You may use this routine as a starting point for your own program.

On page 291 of the "Apple IIe Reference Manual" and on pages 261 to 265 of the Apple IIc Reference Manual, there are lists of the registers and entry points used by routines resident in the Super Serial Card. The equates in the program below use these locations, as well as input/output hooks found in the Apple II family of computers.

The initialization routine (INIT) stores the address of the Super Serial Card's initialization routine in CSW (the Apple II monitor character output hook). This activates the card for output by jumping to COUT. Following this, DOS or ProDOS hooks are reinstalled.

The OUTput routine checks the 6551 status port bit 4. If this is equal to zero, the previous character has not yet been sent, so we must check the status byte again until that register is clear. When the value in bit 4 becomes one, the 6551 is ready to send another character. To do this, store the data in the transmit register (TDREG) of the chip.

Bit 3 of the status port is checked by the INput routine. If this bit is zero, the program either loops continuously or returns to the calling program, depending on the state of the return flag found in location \$FF. If bit 3 is one, a character is waiting at the input port, and the character is then read from the read register (RDREG) of the 6551.

The DEMO portion of this program calls the INIT routine, and sends each letter of the alphabet to the connected device. After each character is sent, the program waits to see if a response has been received from the external device. If a character is waiting, the program ends.

Assembly Language Source Code Demo

Here is a demo of accessing the Super Serial Card with Assembly Language.

```

        ORG      $2000
COUT     EQU     $FDED      ; CHARACTER OUT IN MONITOR
CSWL     EQU     $36       ; OUTPUT HOOK
CSWH     EQU     $37
WAIT     EQU     $FCA8     ; MONITOR ROUTINE TO WAIT
BASELO   EQU     $FA       ; ZERO PAGE INDEX ADDRESS FOR INDIRECT ADDRESSING
BASEHI   EQU     $FB       ; THE TARGET ADDRESS IS STORED IN FA AND FB
IO       EQU     $C0       ; IO PAGE HIBYTE ADDRESS THIS GOES IN BASEHI
;
;   SSC EQUATES
;
DIPSW1   EQU     $81       ; +N0  DIPSWITCH BLOCK 1
DIPSW2   EQU     $82       ; +N0  DIPSWITCH BLOCK 2
TDREG    EQU     $88       ; +N0  6551 DATA REGISTER
RDREG    EQU     $88       ; +N0  6551 DATA REGISTER
STATUS   EQU     $89       ; +N0  6551 STATUS REGISTER
RESET    EQU     $89       ; +N0  6551 SOFTWARE RESET
COMMAND  EQU     $8A       ; +N0  6551 COMMAND REG
CONTROL  EQU     $8B       ; +N0  6551 CONTROL REG
;
START    JMP     DEMO      ; SKIP AROUND ALL THE SUBROUTINES
;
; USE THE SSC FIRMWARE TO INITIALIZE THE 6551.
```

```

;
INIT   LDA    CSWL      ; STORE THE CURRENT CSW
      PHA                ; SO THAT WE DO NOT DISCONNECT
      LDA    CSWH      ; DOS OR ProDOS
      PHA
      LDA    #$00      ; STORE $Cs00 IN CSW
      STA    CSWL
      STX    CSWH      ; THIS ALREADY CONTAINS $Cs
      LDA    #$00
      JSR    COUT      ; JUMP TO COUT TO INIT THE CARD
      PLA
      STA    CSWH      ; RESTORE THE DOS OR ProDOS
      PLA                ; HOOKS AND THEN RETURN
      STA    CSWL
      RTS

;
; OUTPUT A CHARACTER TO 6551
;
OUT    PHA                ; STORE DATA ON STACK
      LDA    #STATUS    ; GET THE STATUS ADDRESS
      STA    BASELO     ; SET UP THE INDIRECT INDEXED ACCESS
OLP    LDA    (BASELO),Y ; CHECK BIT 4 OF STATUS BYTE
      AND    #$10      ; TO SEE IF IT'S OK TO SEND
      BEQ    OLP        ; CHARACTER WAITING TO GO OUT
      LDA    #TDREG     ; ADDRESS FOR TRANSMIT
      STA    BASELO     ; SET UP THE INDIRECT INDEXED ACCESS
      PLA                ; GET DATA BACK FROM STACK
      STA    (BASELO),Y ; AND OUTPUT THE CHARACTER
      RTS

;
; INPUT A CHARACTER FROM 6551
;
IN     LDA    #STATUS    ; GET THE STATUS ADDRESS
      STA    BASELO     ; SET UP THE INDIRECT INDEXED ACCESS
      LDA    (BASELO),Y ; CHECK STATUS
      AND    #$08      ; BIT 3 OF STATUS
      BEQ    INTST     ; NO CHAR WAITING TO BE RECEIVED
      LDA    #RDREG     ; GET THE READ ADDRESS
      STA    BASELO     ; SET UP THE INDIRECT INDEXED ACCESS
      LDA    (BASELO),Y ; GET THE INPUT FROM 6551
      RTS
INTST  LDA    $FF       ; CHECK RETURN FLAG
      BNE    IN         ; IF NOT 0 THEN WAIT FOR INPUT

      RTS                ; IF ZERO, DON'T WAIT

;
; BEGIN THE DEMO PROGRAM
;
DEMO   LDY    #$10      ; Y CONTAINS $s0 - DEMO USES SLOT 1
      LDX    #$C1      ; LOAD X WITH $Cs
      JSR    INIT      ; INIT THE CARD
      LDA    #IO        ; HIBYTE ADDRESS C0 FOR IO ACCESS
      STA    BASEHI    ; STORE IT IN ZERO PAGE AS HIBYTE OF ADDRESS

```

```

LDA    #$FF      ; SET RETURN FLAG FOR INPUT
STA    $FF      ; FF MEANS WAIT FOR CHAR
JSR    IN       ; INPUT A CHARACTER - SEE ABOVE
OLOOP  LDA    #$41  ; OUTPUT THE ASCII CODES
OLP1   TXA          ; FROM A-Z TO THE SSC.  IT WILL STOP
JSR    OUT      ; WHEN THE SSC RECEIVES A CHAR.
LDA    #$80     ; DELAY BETWEEN CHARACTERS
JSR    WAIT    ; TO ALLOW TIME FOR INPUT.
LDA    #$00
STA    $FF     ; RETURN IF NO CHARS WAITING
JSR    IN      ; CHECK FOR A CHARACTER
BNE    ALLDONE ; THEY SENT SOMETHING - WE END
INX
CPX    #$5B    ; THE LETTER 'Z'
BNE    OLP1
LDA    #$0D
JSR    OUT     ; SEND A CARRIAGE RETURN
JMP    OLOOP  ; BEGIN THE ALPHABET AGAIN
ALLDONE RTS    ; END ROUTINE

```

Article Change History:

16 Dec 1996 - Reviewed for technical accuracy, revised formatting.

Copyright 1989-96, Apple Computer, Inc.

Keywords: <None>

=====

This information is from the Apple Technical Information Library.

19961218 09:46:46.00

Tech Info Library Article Number: 4435