```
* Serial communication via interrupts
*
* Allows the MCU to do something useful (besides polling the
* RDRF bit).  When a byte is received from the keyboard via
* the serial communications interface, an interrupt service
* routine reads the incoming byte and immediately transmits
* it to the video display.
*
* ELE 205 Spring 2005
* Fred Vetter

* Constants
SCCR1 EQU $102C   ; Serial Communications Control Reg 1
SCCR2 EQU $102D   ; Serial Communications Control Reg 2
BAUD  EQU $102B   ; BAUD Register
SCSR  EQU $102E   ; Serial Communications Status Reg
SCDR  EQU $102F   ; Serial Communications Data Reg

      ORG $C000

* Initialize the Serial Communications Interface (SCI)
      LDAA #$FF
      STAA $4000  ; Enable P3 connector

      LDAA #$00   ; frame: 1 START, 8 DATA, 1 STOP
      STAA SCCR1  ; M=0, WAKE=0
      LDAA #$2C   ; RIE is Receive Enable Interrupt
      STAA SCCR2  ; RIE=1 TE=1 RE=1
      LDAA #$30   ; TE & RE enable Transmitter & Receiver
      STAA BAUD   ; 9600 bps

* Initialize the SCI pseudovector in the Vector Jump Table
      LDAA #$7E   ; JMP opcode (Extended mode addressing)
      LDX #REC_INTR     ; address of ISR for SCI Receive Interrupt
      STAA $00C4  ; Vector Jump Table: store JMP opcode ...
      STX  $00C5  ; ... then the address of the ISR

      CLI         ; enable interrupts

* Main program: should do something useful.  This just
*               adds numbers to the IX register.
      CLRA
      LDX #$0000
LOOP: INCA  ; increment ACCA
      TAB   ; copy ACCA to ACCB
      ABX   ; add ACCB to IX
      BRA LOOP;
* End of main program

* Interrupt Service Routine (ISR): called when a byte
* has been received by the serial communications system
REC_INTR:
      LDAB SCSR   ; Clear RDRF bit
      LDAA SCDR   ; Read byte sent from keyboard
      STAA SCDR   ; Write byte to video display
      RTI         ; return from interrupt
* End of ISR

* end of file
```