Program to setup & test the 68HC12 SCI system
* Inputs a fixed number of characters, saves them
* and echos them to the screen. Uses 9600 Baud.

* DESCRIPTION: *
* Prompt user, read & echo a 9-char string *

* Define the internal registers used in the SCI *

RAM EQU $0900 ; Start of Ram
STK EQU $0900 ; User Stack
RAMPROG EQU $0A00 ; Program space in RAM

SCOBDRH EQU $00C0 ; SCI Baud Rate Control Register High
SCOBDRDL EQU $00C1 ; SCI Baud Rate Control Register Low
SCOCR1 EQU $00C2 ; SCI Control Register 1
SCOCR2 EQU $00C3 ; SCI Control Register 2
SCORSR1 EQU $00C4 ; SCI Status Register 1
SCORSR2 EQU $00C5 ; SCI Status Register 2
SCODRH EQU $00C6 ; SCI Data Register High
SCODRL EQU $00C7 ; SCI Data Register Low
DDRS EQU $00D7 ; Data Direction Port S

* MASKS
BIT1 EQU %00000010
BIT0 EQU %00000001
BIT5 EQU %00100000
BIT7 EQU %10000000
BIT32 EQU %00001100

* Constants & Data Definitions

ORG RAM
STRING DS.B 9
STRLEN DC.B *-STRING
PROMPT DC.B "?"

* MAIN PROGRAM
* CALLS: INIT_SCI, IN_CHAR, OUT_CHAR

ORG RAMPROG ; Program is stored in RAM
LDS #STK ; Initialize stack pointer
LDY #STRING ; Y points to the input string
LDAB STRLEN ; B the number of chars to be rcvd
JSR INIT_SCI ; Initialize SCI

LDAA PROMPT ; Load the prompt char in Reg A
JSR OUT_CHAR ; Prompt the user

GET JSR IN_CHAR ; Take input from keyboard in Reg A
STAA 0,Y ; Save the input character
JSR OUT_CHAR ; Output character to the screen
INY ; Point to the next char received
DECB ; Count down
BNE GET ; Get another character
SWI ; Exit back to monitor program

* Subroutine to setup the 68HC12 SCI system *
* Baud Rate 9600, 8 data bits, 1 stop bit *

* INPUT: None.
* OUTPUT: Initializes SCI.
* DESTROYS: None.
* REGS USED: DDRS, SC0BDH, SC0BDL, SC0CR1, SC0CR0
* CALLS: None.

```
* *************************************************
* INIT_SCI PSHA ; Store A register
BSET DDRS Bit1 ; Set data direction to output for
BCLR DDRS Bit0 ; Port S bit 1 (the Tx pin)
* ; and input for Port S bit 0 (Rx)
LDAA #0 ; Set baud rate to 9600
STAA SC0BDH ; 2MHz / 16 / 13 = 9600Hz
LDAA #13 ;
STAA SC0BDL ;
LDAA #0 ; 1 start/stop bit, 8 data bits ;
STAA SC0CR1 ; no wakeup
LDAA #BIT32 ; Enable Tx and Rx;
STAA SC0CR2 ; all interrupts disabled
PULA ; Restore A register
RTS
*
* *************************************************
* SUBROUTINE: OUT_CHAR
* FUNCTION: Outputs the character in register A to the screen after
* checking if the Transmitter Data Register is Empty.
* INPUT: Data to be transmitted is in register A.
* OUTPUT: Transmit the data.
* DESTROYS: None.
* REGS USED: SC0DRL, SC0SR1
* CALLS: None.

```
```
* *************************************************
* OUT_CHAR PSHB                    ; save the B register
LOOP1
LDAB SC0SR1 ; Check status reg (TDRE in bit 7)
ANDB #BIT7 ; Check if transmit buffer is empty
BEQ LOOP1 ; Wait until empty
* WAIT_TX BRCLR SC0SR1 BIT7, WAIT_TX ; One line polling alternative
STAA SC0DRL ; Output character
PULB                    ; Restore B register
RTS                     ; Return from subroutine
*
* *************************************************
* SUBROUTINE: IN_CHAR
* FUNCTION: Receives typed character into register A.
* INPUT: None.
* OUTPUT: Register A = input from SCI
* DESTROYS: A register.
* REGS USED: SC0DRL, SC0SR1
* CALLS: None.

```
```
* *************************************************
* IN_CHAR LDAA SC0SR1 ; Check status reg (RDRF in bit 5)
ANDA #BIT5 ; Check if receive buffer full
BEQ IN_CHAR ; Wait until data present
* WAIT_RX BRCLR SC0SR1 BIT5, WAIT_RX ; One line polling alternative
LDAA SC0DRL ; Data -> A register
RTS ; Return from subroutine
```