MICROPROCESSOR SYSTEMS

Lecture 9

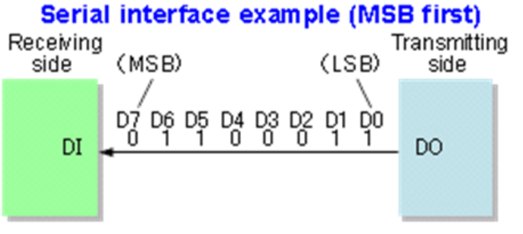
Serial Communication

Topics

- Serial Communication
- ACIA

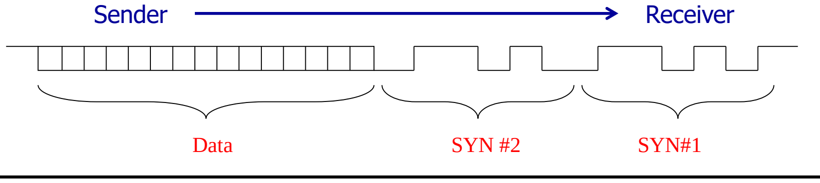
Serial I/O Interfacing

- Serial communication requires **one wire.**
- Bits are transferred **one at a time.**
- There are two types of serial transfer:
  - Synchronous serial transfer
  - Asynchronous serial transfer
- Example : Two **serial** interfaces are connected.



Synchronous Serial Transfer

- Two units **share a common clock frequency**
  - The transmitter and the receiver can **setup** the transmission rate and synchronize their clocks.
- Transmission is **MESSAGE-BASED**.
  - The sender doesn't transmit characters simply as they occur.
  - It stores them in a **buffer.**
  - **Synchronization** occurs at the **beginning** of a long message.
  - A message contains multiple characters (bytes).
- Example protocols using synchronous serial transfer method:
  - I^2C (Inter-Integrated Circuit)
  - SPI (Serial Peripheral Interface)



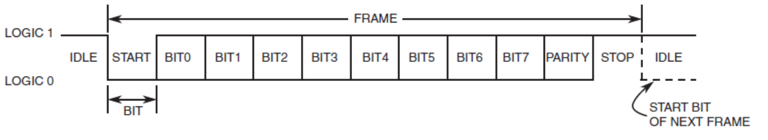
Asynchronous Serial Transfer

- Transmission is **FRAME-BASED**.
- Each character (8-bit data) is transmitted as separate entity (frame).
- The recevier device must be able to recognize when transmission starts, and when transmission ends.

- Example protocols using asynchronous serial transfer method:
  - ACIA (Asynchronous Communication Interface Adapter)
  - RS-232 (Recommended Standard 232)
  - UART (Universal Asynchronous Receiver-Transmitter)
  - USB (Universal Serial Bus) (also supports Synchronous transfer)

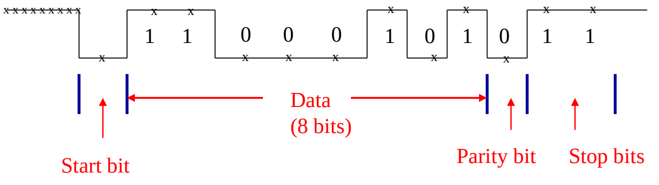
Frame Bits

- **Special bits** are added **at both ends** of the character code frame.
- Each character code frame consists of following parts:
  - **Start bit:** always "0", indicates beginning of a character
  - **Information bits:** data (8 bits)
  - **Parity bit:** 1 or 0, depending on the Parity method (Odd/Even).
    - Parity (equality) bit is a checking (verification) bit appended to data bits to make the sum of all the data bits, including the checking bit itself.
    - Parity bit can be either odd (1) or even (0).
  - **Stop bit:** always "1"



Frame Transfer Rules

- Asynchronous transmission rules :
  - When a character is not being sent, the transmisson line is kept in the logical 1 state
  - Character transmission is detected from the start bit (0)
  - Information bits follow the start bit
  - Transmitter calculates the parity bit (0 or 1) and transmits it
  - One or two stop bits are sent



Rate of Transmission

- There needs to be an agreement on how long each bit stays on the line.
- **Speed parameter:** The rate of transmission is usually measured in bits per second (baud)
- **Baud Rate = Bits/Second**
  (bps : bits per second)
- Baud Rate is the standard transmission speed unit in serial communication.

Rate of Transmission

- The followings are some of the commonly used standard transmission rates (Baud speeds) in serial communication.

Baud (bps)	# of stop bits	Byte / second	Bit time (mili seconds)
110	2	10	9.09 $= (1 / 110) * 10^3$
150	1	15	6.67
300	1	30	3.33
1200	1	120	0.83
2400	1	240	0.42
4800	1	480	0.21
9600	1	960	0.10
19200	1	1920	0.05

Example: Calculating Bit Rate and Data Rate

- Suppose the **Baud Rate** is 19200 bps.
- **QUESTIONS**
  1) Calculate how many mili seconds **one bit** should stay on the line. (**Bit Rate**)
  2) Calculate how many mili seconds **one frame** should stay on the line. (**Data Rate**)
  3) Calculate how many **frames** can be transferred in **one second**.
- **ANSWERS**

  1) Bit Rate calculation:

  Time for one bit is $= 1/19200$ seconds
  $= 0.00005$ seconds
  $= 0.00005 * 10^3$ mili seconds
  $= \boxed{0.05 \text{ mili seconds}}$

  2) Data Rate calculation :

  Assume there is 1 Start Bit, 1 Stop bit, 8 data bits, and no parity bits in the frame (Total length of frame is 10 bits).

  Time for one frame is = Bit rate * Frame length
  $= 0.05 * 10$ bits $= \boxed{0.5 \text{ mili seconds}}$

  3) Calculation of frames per second :

  $1 / (0.5 * 10^3) = 1920$ frames per second
  $\approx \boxed{2K \text{ frames per second}}$

Topics

- Serial Communication
- ACIA

Asynchronous Communication Interface Adapter (ACIA)

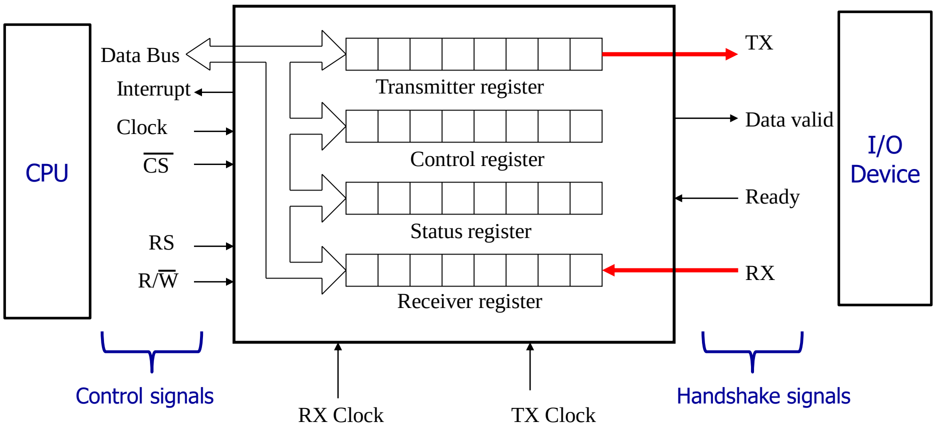
- ACIA is a serial interface chip which contains 4 basic registers (each is 8-bit)
  - **Transmitter (TX) Register:**
    - Transmits bits to the peripheral.
    - Parallel input - Serial output shift register.
    - Start, stop, and parity bits are appended to data (CPU data bus), and transmitted serially.
    - Transmitter clock determines the bit rate.

- **Receiver (RX) Register :**
  - Receives bits from the peripheral.
  - Serial input - Parallel output shift register.
  - Start, stop, and parity bits are removed from the transmission and transferred to CPU data bus.

- **Status Register:**
  - Status flags for Received Data and Transmitted Data.

- **Control Register:**
  - Used for establishing the transmission protocol and interrupt.

ACIA



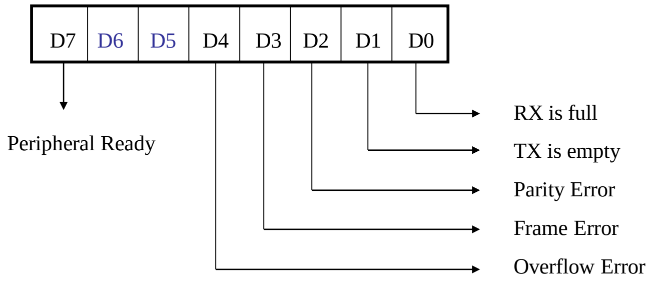
ACIA Register Selection

- In order to select a register in ACIA, the following two control signals are used (like an address decoder).
  - RS (Register Select) signal : A0 line of Address bus.
  - R/W’ signal : Read or Write signal.

RS (A0)	R/W'	Selected Register
0	0	Transmitter
1	0	Control
0	1	Receiver
1	1	Status

15

ACIA - Status Register



D0=1 Indicates new data at RX register

D1=1 Indicates data is transmitted to peripheral

D2=1 Parity error

D3=1 Frame error: If frame is short or long compared to data received

D4=1 Overflow error: New data arrives before previous one is received

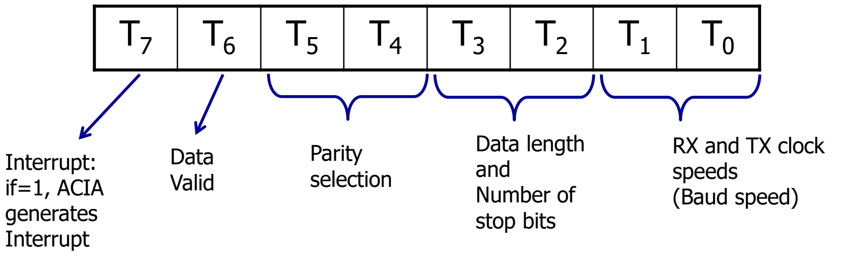
D5-D6 : Not used

D7=1 Indicates peripheral is ready

16

ACIA - Control Register

- Determines the Data Valid output, interrupt, and communication protocol.



17

ACIA - Control Register

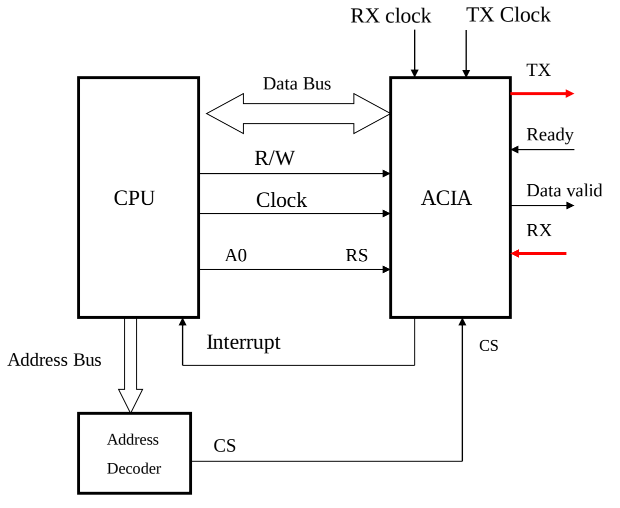
T1	T0	Transmission Clock speed (Baud speed)
0	0	1/1 (maximum)
0	1	1/2 (half of maximum)
1	0	1/4 (quarter of maximum)
1	1	1/8 (1/8 of maximum)

T3	T2	Data Length and the Number of Stop Bits
0	0	7 data bit + 1 stop bit
0	1	7 data bit + 2 stop bit
1	0	8 data bit + 1 stop bit
1	1	8 data bit + 2 stop bit

T5	T4	Parity Bit Settings
0	0	no parity check
0	1	odd parity
1	0	even parity
1	1	-

18

ACIA – Connection to CPU



Handshaking signals:

Ready: Indicates peripheral device is ready.

Data Valid: Indicates valid data is at the ACIA port.

Example1: Reading 8-bit data from ACIA, and sending it to PIA

- **APPLICATION PROGRAM:**
- A computer receives 8-bit numbers from **ACIA** (input).
  - ACIA gets serial data thru its RX line, from a 4x4 Keypad Module.
- The number will be sent to **PIA** (output).
  - A 10-LED-Bar Module is connected to the PIA's port.
- **ACIA Conditioning:**
  - Even parity (T_5=1 T_4=0)
  - 8 bit data + 1 stop bit (T_3=1 T_2=0)
  - Transmission Clock speed = 1/4 (T_1=1 T_0=0)
  - ACIA Control register (binary value) :

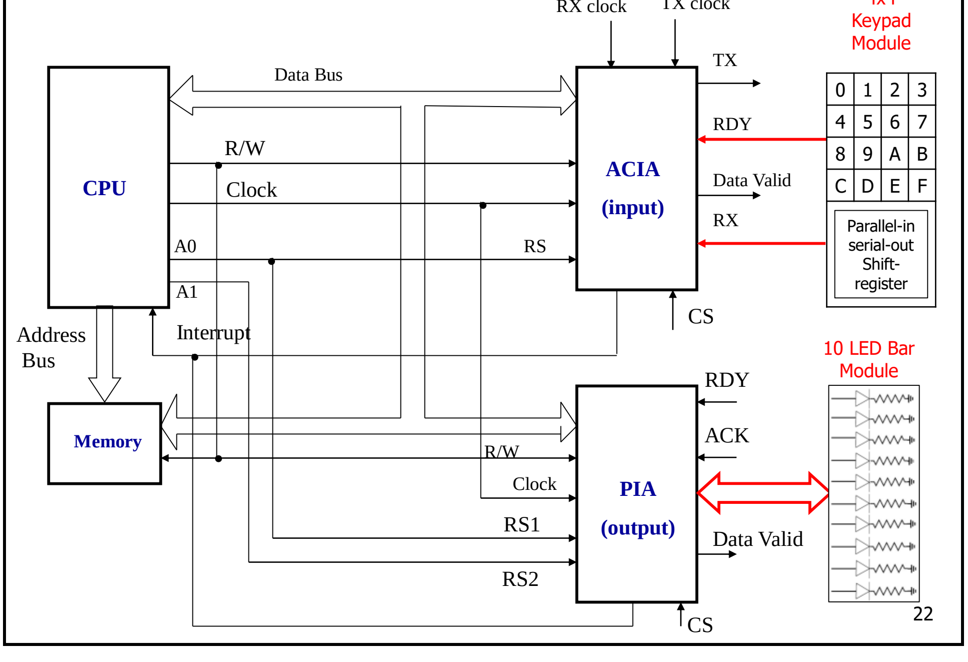
($T_7\,T_6\,T_5\,T_4\,T_3\,T_2\,T_1\,T_0$)

(00101010 → $2A)

- **PIA conditioning:**
  - Direction register: $FF (All bits of PIA port are outputs)
  - No interrupt will be generated: D_1=0, D_0=0
  - PIA Data Valid will be set: D_5=0, D_4=1
  - PIA Control register (binary value) :

(D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0)
(00010000 → $10)

Block Diagram of Connections



Main Program

Note: ACIA is not implemented in EDU-CPU.

```
* Register addresses

ACIA_STATCON    EQU    $8010
ACIA_RX_TX      EQU    $8011

PIA_PORT        EQU    $8020
PIA_DIRECTION   EQU    $8021
PIA_STATCON     EQU    $8022
```

- In EDU-CPU, Receiver and Transmitter are the same register, named as ACIA_RX_TX.
- Status and Control are the same register, named as ACIA_STATCON.

```
START   BSR  CONA    ; subroutine for ACIA conditioning
        BSR  CONP    ; subroutine for PIA conditioning

LOOP    BSR  READ    ; subroutine for reading data from ACIA
        BSR  WRITE   ; subroutine for writing data to PIA
        BRA  LOOP    ; endless loop
```


Subroutines for ACIA and PIA conditionings

```
* Subroutine Conditioning of ACIA (input)
CONA        LDA B, $2A
*Bit rate, parity, etc.
            STA B, ACIA_STATCON
            RTS
```

```
* Subroutine Conditioning of PIA (output)
CONP        LDA B, $FF ;output
            STA B, PIA_DIRECTION

            LDA B, $10 ; Data Valid
            STA B, PIA_STATCON
            RTS
```

Subroutines for Reading and Writing

```
* Subroutine Reading data from ACIA
READ    LDA B, <ACIA_STATCON>
        AND B, $01 ;Filter rightmost bit (RX is full)
        BEQ READ  ; Status not ready yet

        LDA A, <ACIA_RX_TX> ; Load data from receiver register
        RTS
```

```
* Subroutine Writing data to PIA
WRITE   LDA B, <PIA_STATCON>
        AND B, $80 ;Filter leftmost bit
        BEQ WRITE  ; Status not ready yet

        STA A, PIA_PORT ; Store data to port register
        RTS
```

25

Example2 : Two computers communicating via their ACIA interfaces

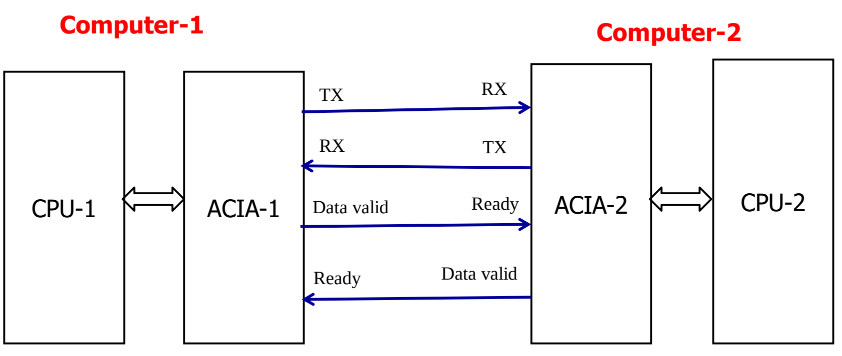
- Two computers are connected via their own ACIA interfaces.
- **APPLICATION PROGRAMS:** Write two programs to transfer memory contents in Computer-1 between addresses $0000 and $0035, to the same memory addresses in Computer-2.
- The ACIAs for both computers will be conditioned as follows:

Transmission Clock speeds : 1/8	(T_1=1 T_0=1)
8 bit data + 2 stop bits	(T_3=1 T_2=1)
Even parity	(T_5=1 T_4=0)

ACIA Control Register (binary value) : **(0010 1111 → $2F)**

26

Connection Diagram



27

Two Main Programs

TRANSMITTER Computer:

```
CONTROL     EQU $8010
STATUS      EQU $8010
TRANSMITTER EQU $8011

START  BSR    COND   ;conditioning
       LDA    SK,  $0000

BACK   BSR    INSP   ;inspection

       ;Load data from array and transmit it
       LDA    A,<SK+0>
       STA    A,<TRANSMITTER>

       INC    SK
       CMP    SK,$0035
       BLT    BACK

       INT
```

RECEIVER Computer:

```
CONTROL  EQU $8010
STATUS   EQU $8010
RECEIVER EQU $8011

START  BSR    COND  ;conditioning
       LDA    SK,  $0000

BACK   BSR    INSP   ;inspection

       ;Receive data and store it in array
       LDA    A,<RECEIVER>
       STA    A,<SK+0>

       INC    SK
       CMP    SK,$0035
       BLT    BACK

       INT
```

28

Two Subroutines

TRANSMITTER Computer:

```
* Conditioning of ACIA

COND     STA     $2F,<CONTROL>
         RTS
```

```
* Inspection of ACIA ready status
* (loop)

INSP     LDA     B,<STATUS>
         AND     B,$02  ;Apply filter to get D1 bit
         CMP     B,$02  ;Compare
         BNE     INSP   ; D1 bit is not 1 yet
         RTS
```

Transmitter status flags

D1 (TX is empty)	D0 (RX is full)
1	0

RECEIVER Computer:

```
* Conditioning of ACIA

COND     STA     $2F,<CONTROL>
         RTS
```

```
* Inspection of ACIA ready status
* (loop)

INSP     LDA     B,<STATUS>
         AND     B,$01  ;Apply filter to get D0 bit
         CMP     B,$01  ;Compare
         BNE     INSP   ; D0 bit is not 1 yet
         RTS
```

Receiver status flags

D1 (TX is empty)	D0 (RX is full)
0	1

29

Example3 : Using I/O Devices (Keyboard & Screen) via ACIA

- **APPLICATION PROGRAM: Program asks (prompts) user to enter two numbers from keyboard, calculates the Sum, and displays the result on screen.**

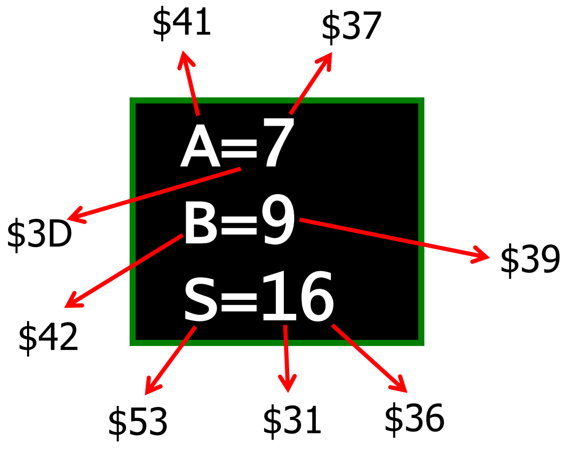
- Suppose an I/O workstation terminal is connected to computer via ACIA.
- Workstation has two component devices: Keyboard and Screen (monitor).
- Both devices are using ASCII data encoding, therefore the program will receive and transmit all characters as ASCII values.

- **PHASE 1:**
  - Firstly, "A=" text will be displayed by program on screen.
  - Then, the user enters a single digit decimal number from the keyboard, without hitting the ENTER key.
  - Program writes back the number entered by user, to the same line on screen.
  - In order to go to the beginning of next line on screen, program writes **NEWLINE** characters on screen (Carriage Return character, followed by Line Feed character).
- **PHASE 2:**
  - On the next line "B=" text will be displayed.
  - Then, the user enters another single digit decimal number from the keyboard, without hitting the ENTER key.
  - Program writes back the second number entered by user, to the same line on screen.
- **PHASE 3:**
  - On the next line "S=" text will be displayed.
  - Program calculates the sum of the two numbers.
  - The sum (as two digits) will be displayed to the same line on screen.

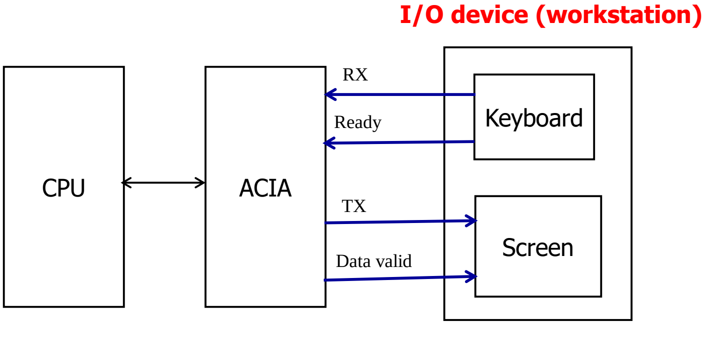
30

Example Screen Output

- The followings are characters displayed by program on the screen. (ASCII values are drawn for explanation purpose.)
- The prompt letters (A, B, S), equal signs (=), the numbers entered by user from keyboard, and the calculated sum (as two digits) are displayed on screen as ASCII characters (each 1 byte).
- There will be no screen cursor symbol (blinking caret) on screen output.



Connection Diagram



Conditioning of ACIA Control Register

- ACIA will be conditioned as follows:

RX / TX Clock frequency ratio: 1/1	(T1=0	T0=0)
8-bit data + 2 stop bits	(T3=1	T2=1)
Even Parity	(T5=1	T4=0)

ACIA Control Register: **(0010 1100 → $2C)**

Main Program

Screen and keyboard not implemented in Mikbil.

```
ACIA_STATCON   EQU    $8010
ACIA_RX_TX     EQU    $8011

START  LDA     YG, $FFFF
       LDA     A, $2C
       STA     A, ACIA_STATCON

* The first data will be read from keyboard and displayed on screen.
       LDA     A, $41  ; A letter
       BSR     SEND
       LDA     A, $3D  ; = symbol
       BSR     SEND
       BSR     RECV
       BSR     SEND
       STA     A, $0010  ;Example: ascii $37 is stored for 7
*              Memory storage address is $0010
       BSR     NEWLN    ;Goes to next line on screen
```

Main Program (continued)

```
* The second data will be read from keyboard and displayed on screen.
        LDA     A, $42 ; B letter
        BSR     SEND
        LDA     A, $3D ; = symbol
        BSR     SEND
        BSR     RECV
        BSR     SEND
        STA     A, $0011 ; EXAMPLE: ascii $39 → for 9
*                Memory storage address is $0011
        BSR     NEWLN

* The sum (as two digits) will be calculated and displayed on screen.
        LDA     A, $53 ; S letter
        BSR     SEND
        LDA     A, $3D ; = symbol
        BSR     SEND
```

35

Main Program (continued)

```
* Calculate the sum.
* ASCII numbers should not be added directly.
* They have to be converted from ASCII to normal decimals firstly.
* $0F is used as a filter in AND instructions.

        LDA     A, <$0011> ; contains ascii $39
        AND     A, $0F     ; converted to $09

        LDA     B,<$0010>  ; contains ascii $37
        AND     B, $0F     ; converted to $07

        ADD     A, B       ; Result is $10 (decimal 16)
```

36

Main Program (continued)

```
* The results will be converted to ASCII for displaying.

        DAA   A   ; Decimal adjust accumulator A
                  ; Result $10 converted to  BCD  16
        MOV    B, A     ; Copy A to B
        LSR     A       ; Logical shift right
        LSR     A
        LSR     A
        LSR     A   ; contains $01 (first digit of sum)
        OR      A, $30  ; converted to ascii $31
        BSR     SEND

        MOV     A, B     ; Copy B to A
        AND     A, $0F   ; contains $06 (second digit of sum)
        OR      A, $30   ; converted to ascii $36
        BSR   SEND

END     INT
```

Obtaining two ASCII values from BCD 16

ASCII $36

ASCII $31

37

Subroutines for Sending and Receiving One Character

```
*Write to screen
SEND   LDA     B, <ACIA_STATCON>
       AND     B,$02
       BEQ     SEND       ;Status not ready yet
       STA     A,  ACIA_RX_TX
       RTS
```

```
*Read from keyboard
RECV   LDA     B, <ACIA_STATCON>
       AND     B,$01
       BEQ     RECV       ;Status not ready yet
       LDA     A,  ACIA_RX_TX
       RTS
```

38

Subroutine for Sending the NewLine Characters

```
*Write New Line to screen
NEWLN  LDA   A, $0D    ;ascii Carriage Return character
       BSR   SEND
       LDA   A, $0A    ;ascii Line Feed character
       BSR   SEND
       RTS
```

39