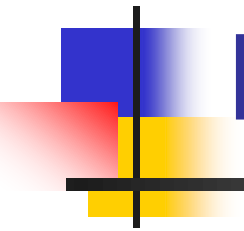
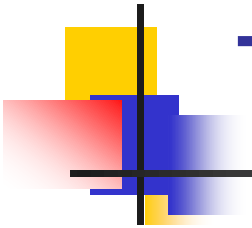


Microprocessor Systems

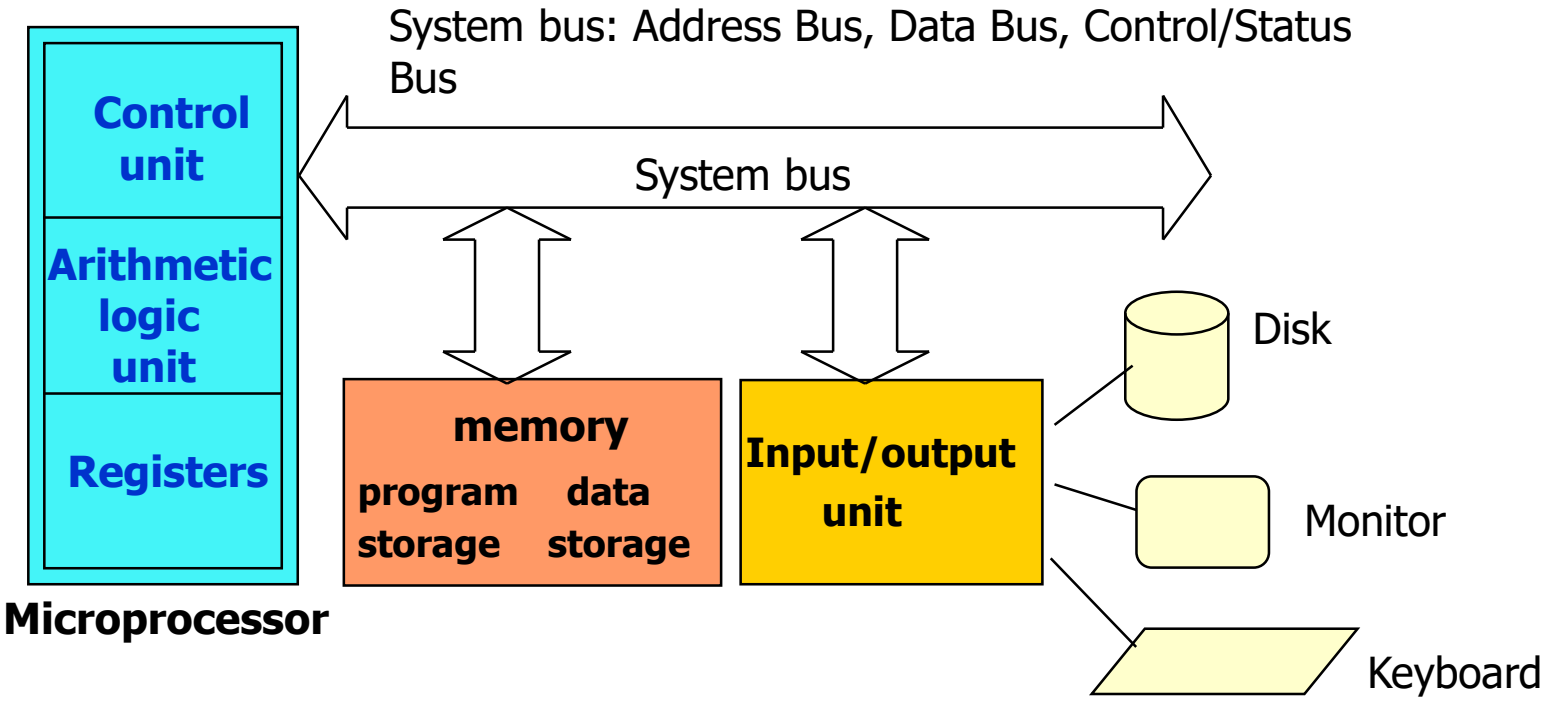
Dr. Gökhan İnce



Topics

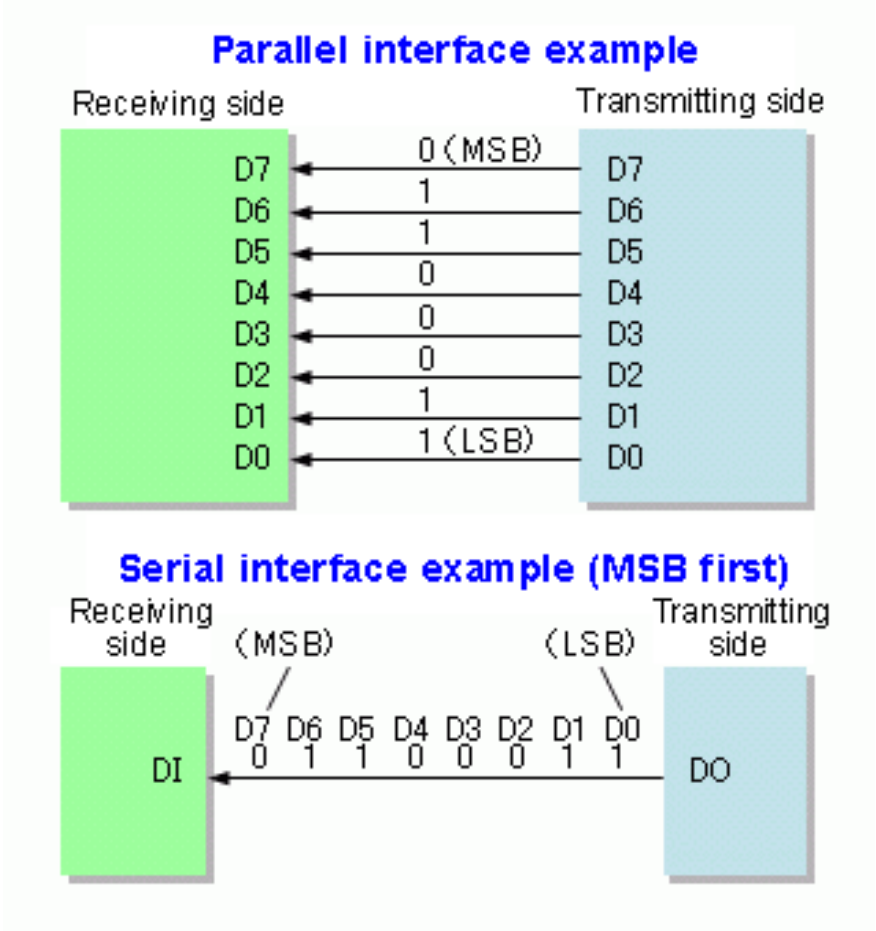
- Serial Communication
- ACIA

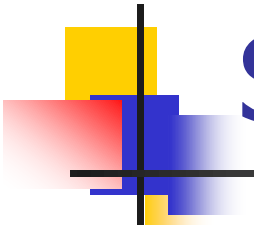
Computer Organization



I/O Interfacing

- The I/O communication can be Parallel or Serial



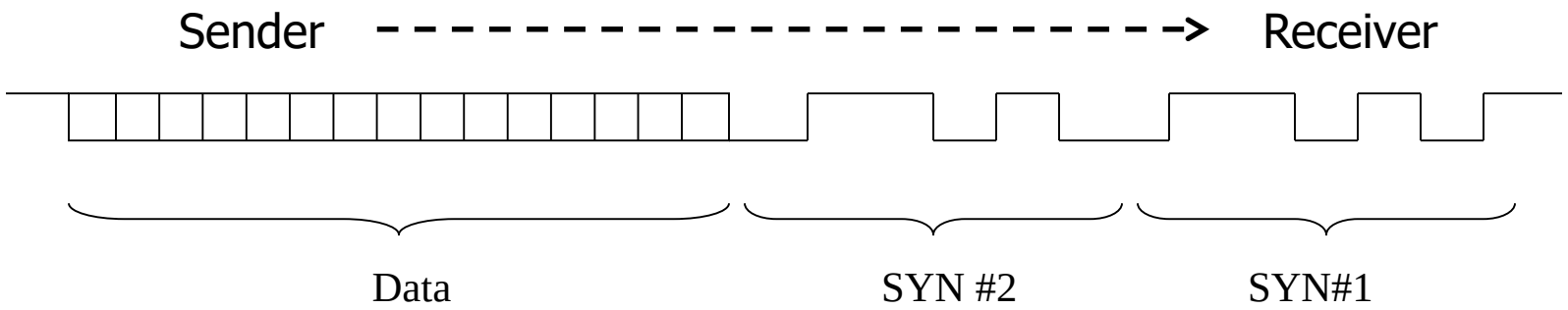


Serial Communication

- Serial communication requires **one wire**; and bits are transferred **one at a time**
  - Therefore, there needs to be an agreement on how "long" each bit stays on the line.
- There are parameters that must be agreed upon between the two computer systems. One of them is **the speed**.
  - The rate of transmission is usually measured in **bits/second** or **baud**.
- Two different types of serial transfer
  - Synchronous serial transfer
  - Asynchronous serial transfer

Synchronous Data Transmission

- The two units share a **common clock frequency**
  - The transmitter and receiver can **negotiate** the transmission rate and lock (synchronize) their clocks.
- Usually used for high speed transmission
- Message-based
  - The sender cannot transmit characters simply as they occur and consequently has to **store them until it has built up a block**
  - The system is unsuitable for applications where characters are generated at irregular intervals.
  - Synchronization occurs at the beginning of a long message.
- The SYN character is sent periodically to maintain synchronization



Asynchronous Serial Transfer

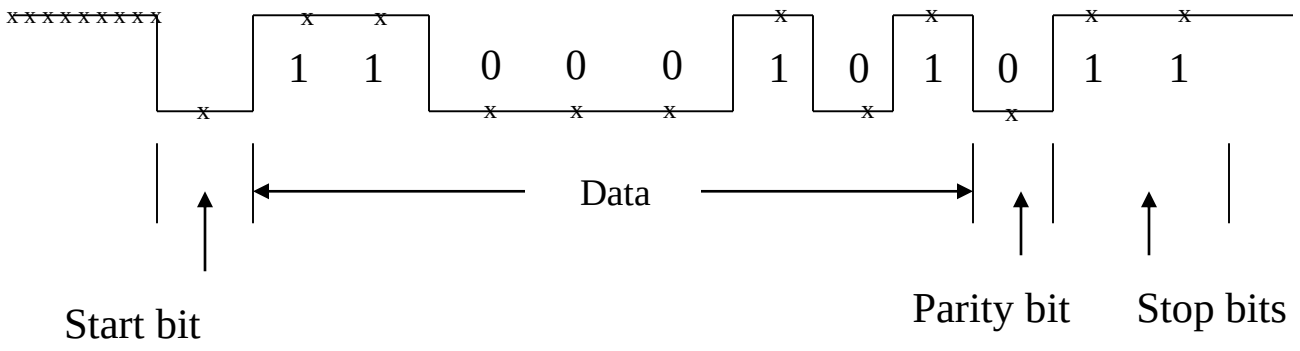
- Each character is transmitted as separate entity. The device must be able to recognize:
  - When transmission is occurring
  - When to read a bit of data
  - When the transmission ends
  - When the transmission is idle (no data being transmitted)

- **Special bits** are inserted **at both ends** of the character code

- Each character consists of three parts
  - start bit: always “0”, indicate the beginning of a character
  - information bits: data
  - stop bit: always “1”

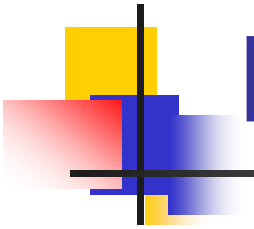
Asynchronous Serial Transfer

- Asynchronous transmission rules :
  - When a character is not being sent, the line is kept in the 1-state
  - The initiation of a character transmission is detected from the start bit, which is always "0"
  - The information bits always follow the start bit
  - Then the transmitter calculates the parity bit and transmits it
  - Finally one or two stop bits are sent by returning the line to the 1-state



Asynchronous Serial Transfer

- Follows agreed upon standards
- There is a protocol between sender and receiver specifying:
  - **Data rate:** The rate of transmission is usually measured in bits/second or baud
  - **Start bit:** The transmission begins with a start bit
  - **Data length:** The seven or eight bits representing the character are transmitted
  - **Stop bit:** The transmission is concluded with one or two stop bits.
  - **Parity bit:** This bit is set on (1) or off (0), depending on the serial communications parameters such as even/odd parity bit
    - In case of **even** parity, **the parity bit is set to 1**, if the number of ones in a given set of bits (not including the parity bit) is odd, making the number of ones in the entire set of bits (including the parity bit) even.



Rate of Transmission

- For parallel transmission, all of the bits are sent at once.
- For serial transmission, the bits are sent one at a time.
  - Therefore, there needs to be agreement on how "long" each bit stays on the line.
- The rate of transmission is usually measured in bits/second or baud.
  - Baud = bits / second.
  - Seconds / bits = 1 /baud

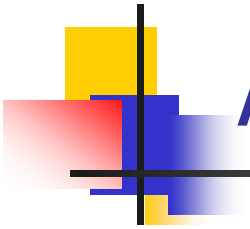
Example

- Given a certain baud rate, how long should each bit last?
- A Baud of 19200Bd for serial transmission:
  - **Bit rate:** 19200bps & Bit time is 1/19200 sec
  - **Data rate:** If there is 1 Start Bit, 1 Stop bit, 8 data bits in the frame
    → Data rate = 19200 Bd/10 = 1920 Byte/sec.

Baud	# of stop bits	Byte / s .	Bit time (ms.)
110	2	10	9.09
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

https://en.wikipedia.org/wiki/List_of_device_bit_rates

USB 2.0 is 480 Mbit/s
USB 3.0 is 5 Gbit/s

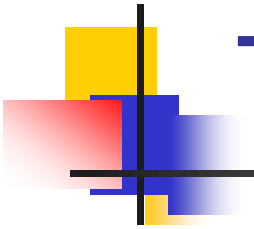


Asynchronous Serial Communication

- Transmitting numeric data is straightforward
- Transmitting characters are encoded with a binary value.
  - Most well known is American Standard Code for Information Interchange (ASCII)
  - Another one is UNICODE.

ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□



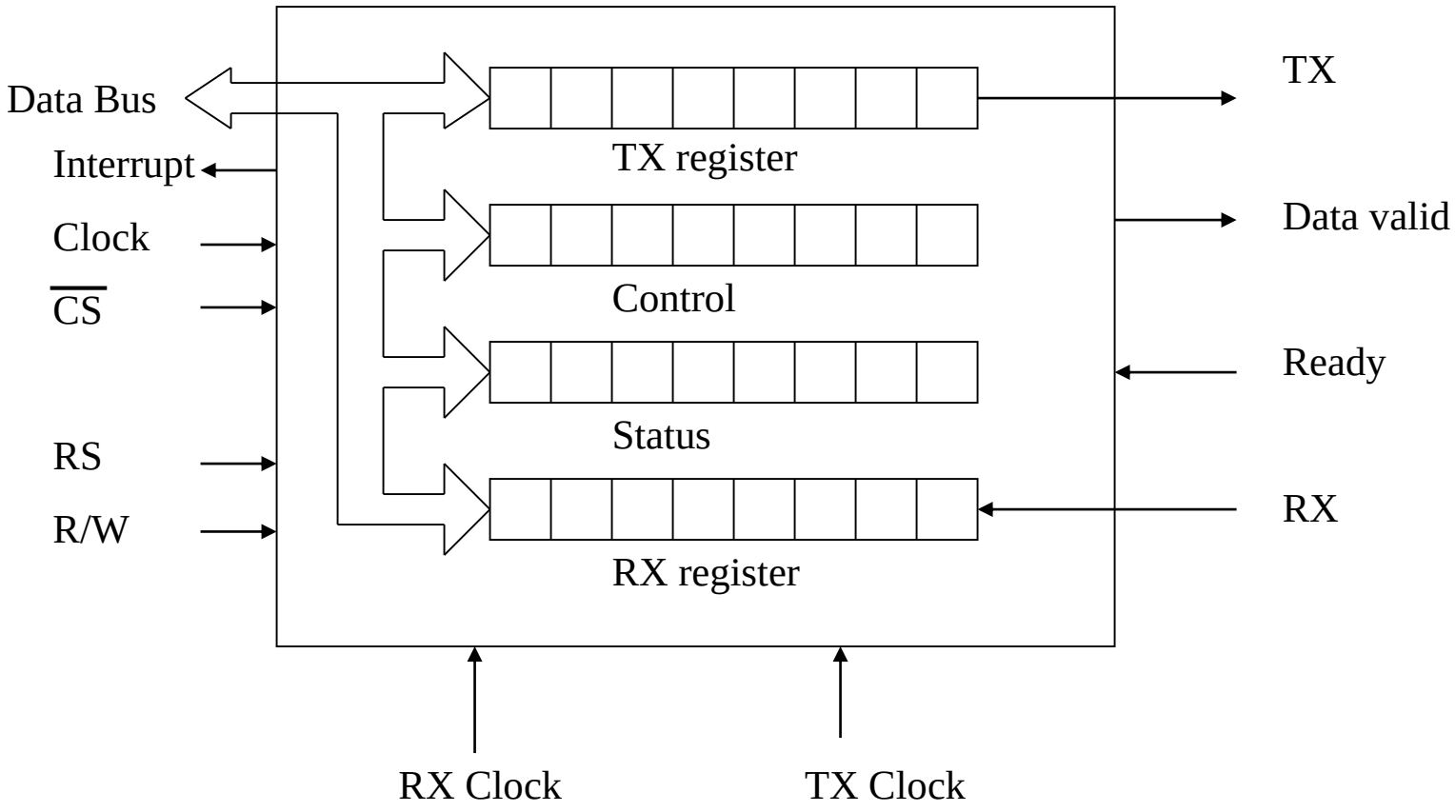
Topics

- Serial Communication
- ACIA

Asynchronous Communication Interface Adapter (EDU-ACIA)

- 4 basic units
  - **Transmitter (TX):**
    - Transmits to the peripheral.
    - Parallel input-serial output shift register.
    - Start, stop, and parity bits are appended to the data (from the CPU data bus), and transmitted serially.
    - Transmitter clock determines the bit rate.
  - **Receiver (RX):**
    - Receives 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, Transmitted Data, Parity Check, Frame Check, Peripheral Ready.
  - **Control Register:**
    - Adjusted for establishing the transmission protocol and interrupt mechanisms.

ACIA



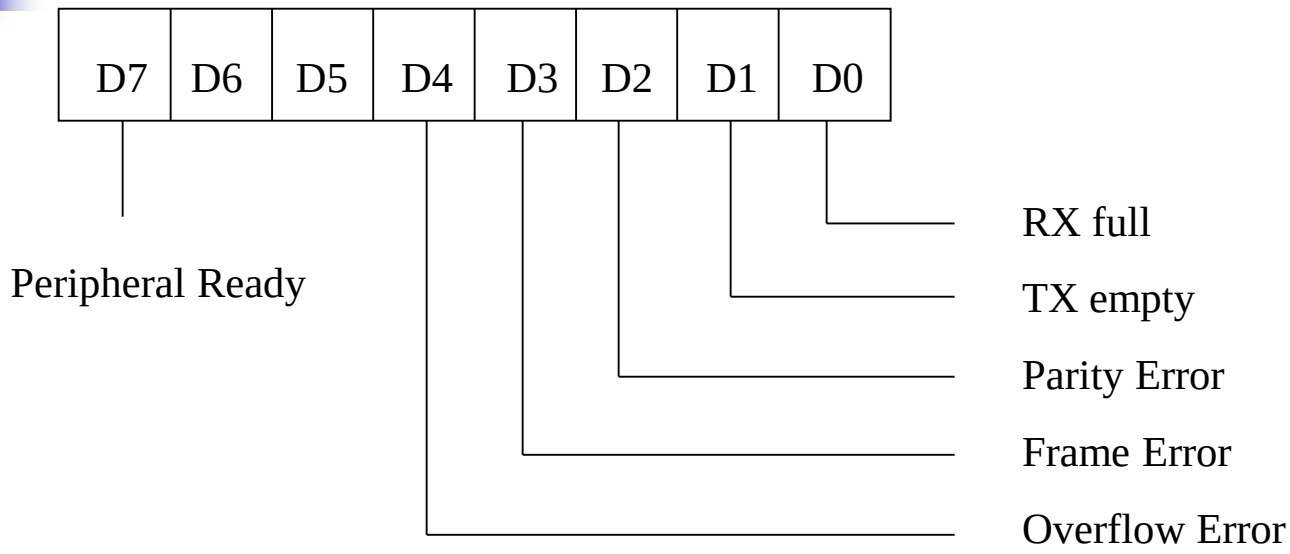
How to select each register with a single RS signal?

- Hint: Use another control signal, but which one?

ACIA Register Selection

RS	R/W	Register
0	0	TX
1	0	Control
0	1	RX
1	1	Status

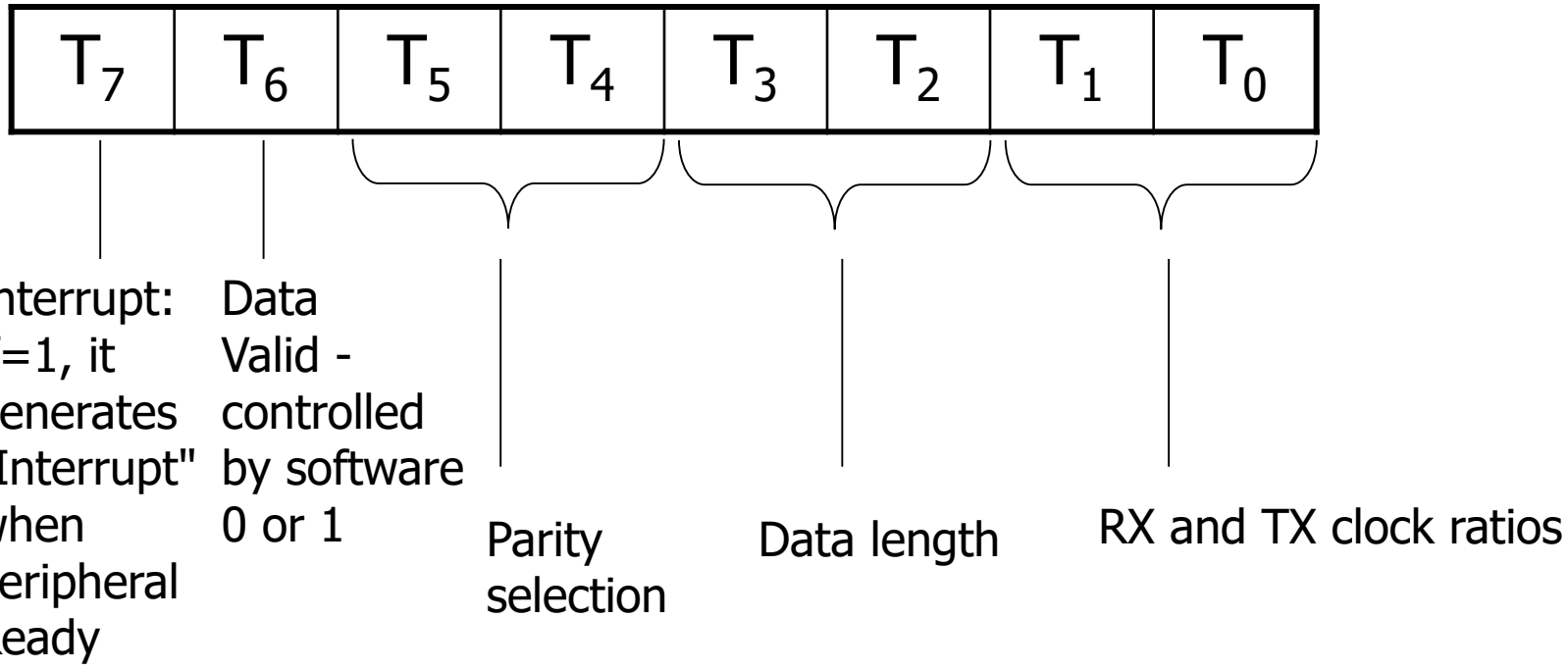
ACIA Status Register



- D0=1 Indicates new data at the RX register.
- D1=1 Indicates the data is transmitted to the peripheral.
- D2=1 Parity error.
- D3=1 Frame error: If the frame is short or long compared to the data received
- D4=1 Overflow error: New data arrives before previous one is received.
- D7=1 Indicates the peripheral is ready.

ACIA – Control Register

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



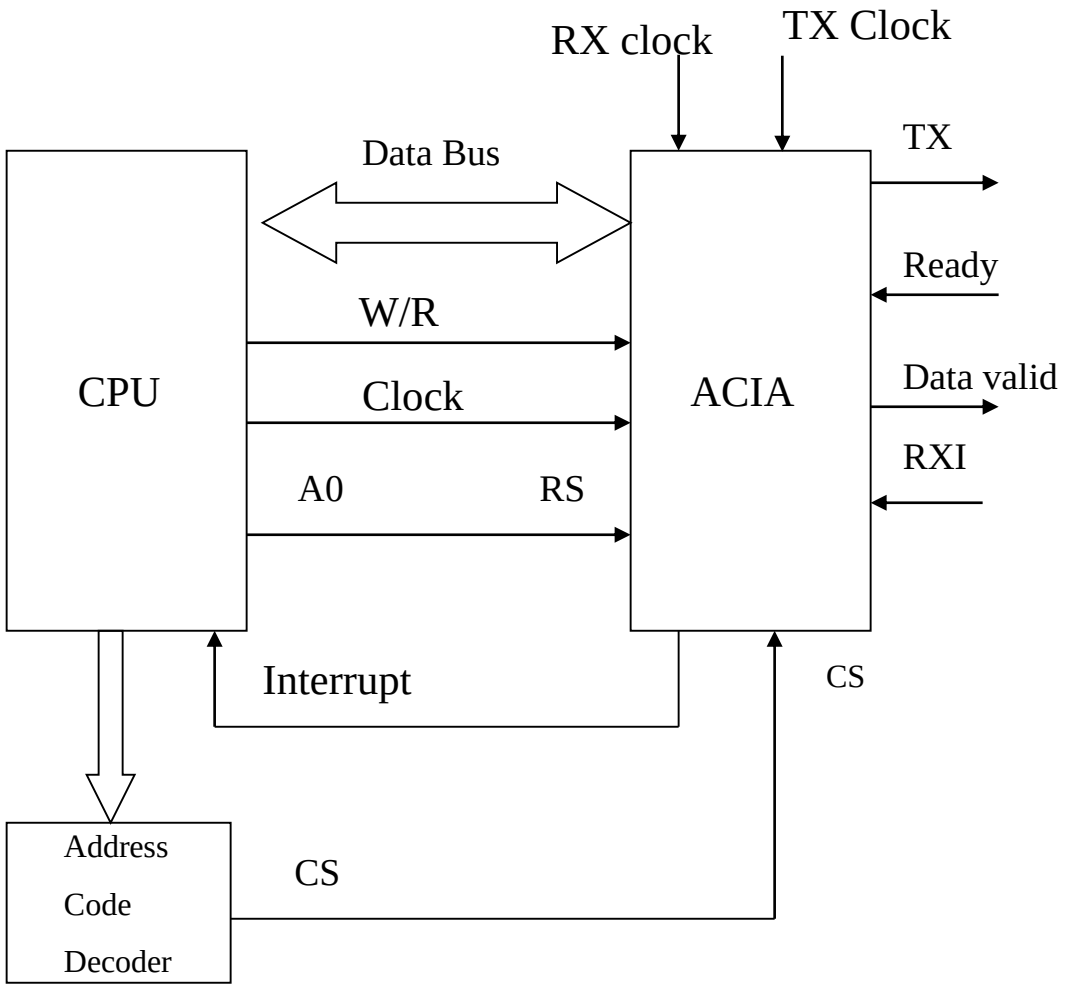
Control register

T1	T0	RX and TX clock ratios
0	0	1/1
0	1	1/2
1	0	1/4
1	1	1/8

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 dats bit + 2 stp bit

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

ACIA – Connection to the CPU



Example-1

- A computer receives signed 8-bit numbers via EDU-ACIA. If the received number is **positive or zero**, it will be sent via the EDU-PIA as it is. If the received number is **negative**, it will be **complemented** and sent via the EDU-PIA.
- ACIA Conditioning:
  - Bit rate 1200 bit/s
  - Even parity $T_5=1$ $T_4=0$
  - 8 bit data+ 1 stop bit $T_3=1$ $T_2=0$
  - RX & TX CLK ratio=1/4 $T_1=1$ $T_0=0$
  - ACIA <Control> 00101010 $2A

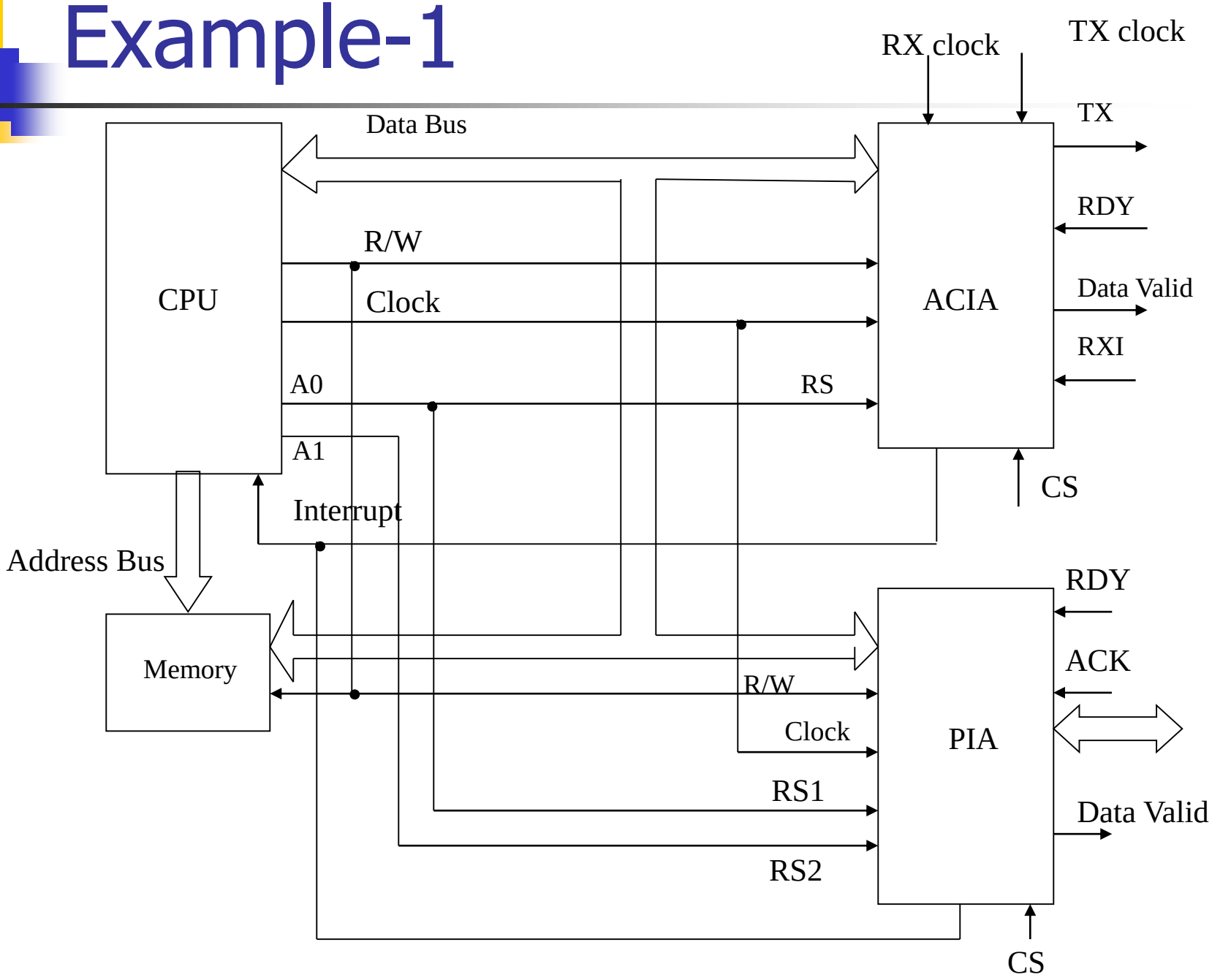
Example-1

- PIA conditioning:
  - Direction register: $FF (PIA port is all output)
  - PIA Peripheral Ready (PIA_RDY)

  Input «1→0» → D7=1,

  no interrupt generated: D1=0, D0=0
  - PIA Data Valid (PIA_DV) will be set: D5=0,D4=1
  - PIA <Status/Control> 00010000 $10

Example-1


RX clock
TX clock
Data Bus
TX
RDY
R/W
CPU
Clock
ACIA
Data Valid
RXI
A0
RS
A1
CS
Interrupt
Address Bus
RDY
Memory
ACK
R/W
Clock
PIA
RS1
Data Valid
RS2
CS

Example-1

```
START   LDA SP, $FFFF
        BSR CONA
        BSR CONP
        BSR READD
        TST A, $80
        BEQ FWD
        COM A
FWD     BSR CHK
END     SWI

READD   LDA B, <STATUS>
        AND B, $01
        BEQ READD
        LDA A, <RX>
        RTS

CONA    LDA B, $2A
        STA B, <CONTROL>
        RTS

CONP    LDA B, $FF
        STA B, <DIRECTION>
        LDA B, $10
        STA B, < STATUS/CONTROL>
        RTS

CHK     LDA B, <STATUS/CONTROL>
        AND B, $80
        BEQ CHK
        STA A, PORT
        RTS
```

Example-2

Two computers are connected via ACIA interfaces. Write a code to transfer the memory contents of Computer-1 between addresses $0000 and $0100, to the same memory addresses of Computer-2.
The ACIAs for both computers will be conditioned as follows:

RX/TX clk ratio: 1/8	T1=1	T0=1
8 bit data + 2 stop bits	T3=1	T2=1
Even parity	T5=1	T4=0

Control Register: 0010 1111 ⟹ $2F

Example-2

TX Computer:

```
START   LDA     SP, $FFFF
        BSR     COND
        LDA     IX, $0000
BACK    BSR     INSP
        LDA     A,<IX+0>
        STA     A,<TRANSMITTER>
        INC     IX
        CMP     IX,$0101
        BNEQ    BACK
        SWI

COND    LDA     A,$2F
        STA     A,<CONTROL>
        RTS

INSP    LDA     A,<STATUS>
        AND     A,$02
        BEQ     INSP
        RTS
```

RX Computer:

```
START   LDA     SP, $FFFF
        BSR     COND
        LDA     IX, $0000
BACK    BSR     INSP
        STA     A,<IX+0>
        INC     IX
        CMP     IX,$0101
        BNEQ    BACK
        SWI

COND    LDA     A,$2F
        STA     A,<CONTROL>
        RTS

INSP    LDA     A,<STATUS>
        AND     A,$01
        BEQ     INSP
        LDA     A,<RECEIVER>
        RTS
```

Example-3

- On a display “A=“ will be displayed, then the user enters a single digit decimal number.
- On the new line “B=“ will be displayed. Then, another single digit decimal number will be entered.
- On the next line “S=“ will be displayed and the sum of the entered numbers will be written.
- The characters will be transmitted with the ASCII standard.
- 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

Control Register: 0010 1100 $2C

Example-3

```
START   LDA     SP, $FFFF
        LDA     A, $2C
        STA     A,<CONTROL>
        LDA     A, $41   */ A */
        BSR     SEND
        LDA     A, $3D   */ = */
        BSR     SEND
        BSR     RECV
        BSR     SEND
        STA     A, $0010   */ 37:=7 */
        BSR     NEWLN

SEND    LDA     B,<STATUS>
        AND     B,$02
        BEQ     SEND
        STA     A,<TRANSMITTER>
        RTS

NEWLN   LDA     A, $0D
        BSR     SEND
        LDA     A, $0A
        BSR     SEND
        RTS

RECV    LDA     B,<STATUS>
        AND     B,$01
        BEQ     RECV
        LDA     A,<RECEIVER>
        RTS
        LDA     A, $42 */ B */
        BSR     SEND
        LDA     A, $3D */ = */
        BSR     SEND
        BSR     RECV
        BSR     SEND
        STA     A, $0011*/39:=9*/
        BSR     NEWLN
        LDA     A, $53 */ S */
        BSR     SEND
        LDA     A, $3D */ = */
        BSR     SEND
```

Example-3

```
        LDA     A, <$0011>
        AND     A, $OF */ 09 */
        LDA     B,<$0010>
        AND     B, $0F */ 07 */
        ADD     A,B     */ 10 */
        DAA              */ 16 */
        TAB             */A->B */
        SHR     A
        SHR     A
        SHR     A
        SHR     A       */ 01 */
        OR      A,$30 */ 31 */
        BSR     SEND
        TBA             */B->A */
        AND     A, $0F */ 06 */
        OR      A, $30 */ 36 */
        BSR    SEND
END     SWI
```