

ARHITECTURA CALCULATOARELOR



**Deprtamentul:
Informatică și Ingineria Sistemelor**

conf. univ., dr. Victor ABABIL

Tematica disciplinei Arhitectura Calculatoarelor:

Tema 1. Introducere. Bazele fundamentale ale Arhitecturii Calculatoarelor;

Tema 2. Microprocesoare și Microcontrolere. Limbajul de programare Assembler;

Tema 3. Dispozitive pentru achiziția datelor;

Tema 4. Dispozitive pentru afișarea și imprimarea datelor;

Tema 5. Dispozitive pentru stocarea datelor.

Tema 3. Dispozitive pentru achiziția datelor:

1. Clasificarea dispozitivelor pentru achiziția datelor.
2. Tastatura. Programarea tastaturii. Funcții BIOS și DOS.
3. Mouse-ul. Programarea mouse-ului. Funcții DOS.
4. Camera video.
5. Scanner-ul.
6. Touch Screen-ul (Ecranul tactil)
7. Achiziția datelor discrete.
8. Achiziția semnalelor analogice.
9. Achiziția vorbirii.

Clasificarea dispozitivelor:

- Tastatura (KBD);
- Mouse-ul;
- Camera video;
- Scanner-ul;
- Touch Screen (Ecran tactil);
- Interfețe pentru achiziția datelor digitale;
- Interfețe pentru achiziția semnalelor analogice;
- Interfața pentru achiziția vorbirii.



Tastatura (KBD) este un dispozitiv periferic destinat pentru achiziția datelor alfa-numerice și pseudografice. Achiziția are loc prin modificarea stării matricei de senzori la efectuarea unei acțiuni fizice (tastarea unei taste). După modul de funcționare și tipul de senzori utilizați pentru implementarea tastaturii pot fi specificate următoarele caracteristici și parametri funcționali:

- Tipul de senzori: contact direct, rezistivi, capacitativi, inductivi, efect de câmp, optici, contacte ermetice în tub de sticlă (Reed switch);
- Modul de scanare a matricei de taste: Statice și Dinamice;
- Interfața de comunicare cu PC: COM, PS2, USB, Wireless, IrDA;
- Numărul de taste și funcționalitatea acestora.

Resursele rezervate de arhitectura sistemului de calcul pentru gestiunea tastaturii:

- Întreruperi Hardware: IRQ01 (INT 09h);
- Întreruperi Software: BIOS (INT 16h) și DOS (INT 21h);
- Porturi I/O: 060h – 06Fh;
- RAM BIOS: 0000:0416h ...

Structura tastaturii dinamice:

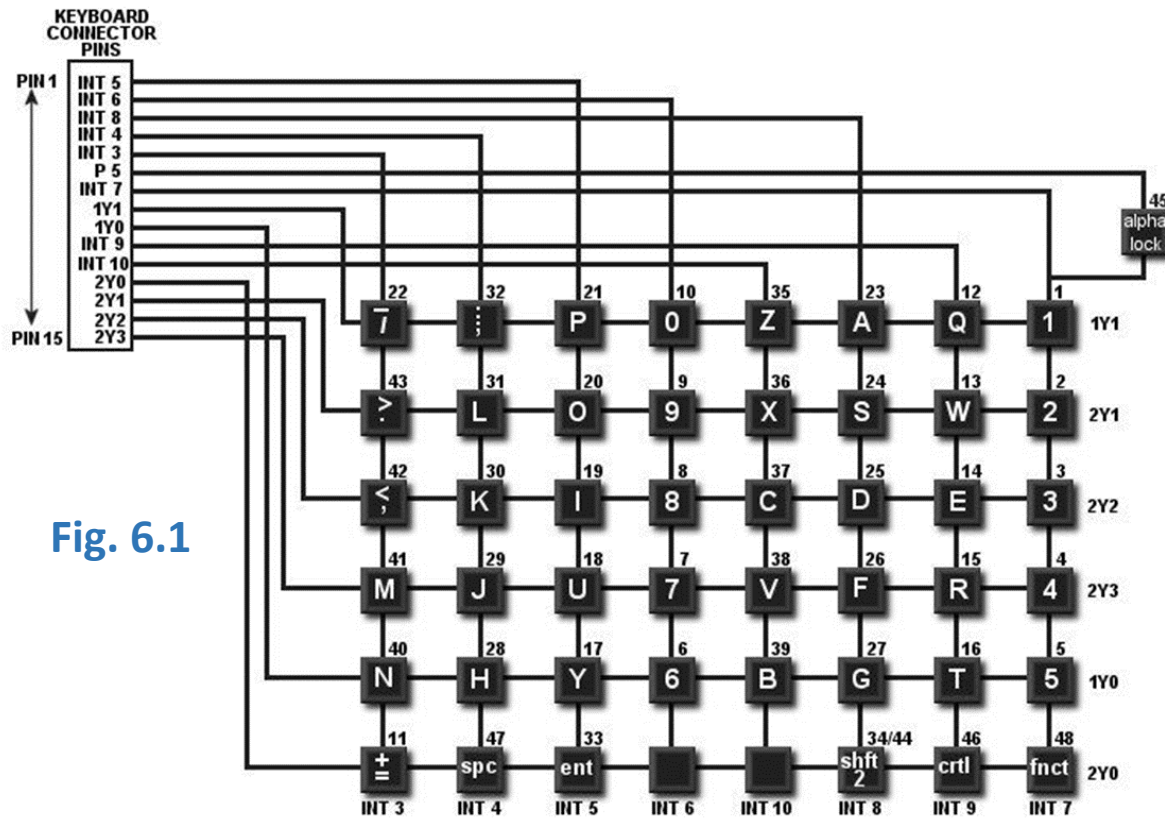


Fig. 6.1



Fig. 6.2

Structura tastaturii dinamice. Taste funcționale:

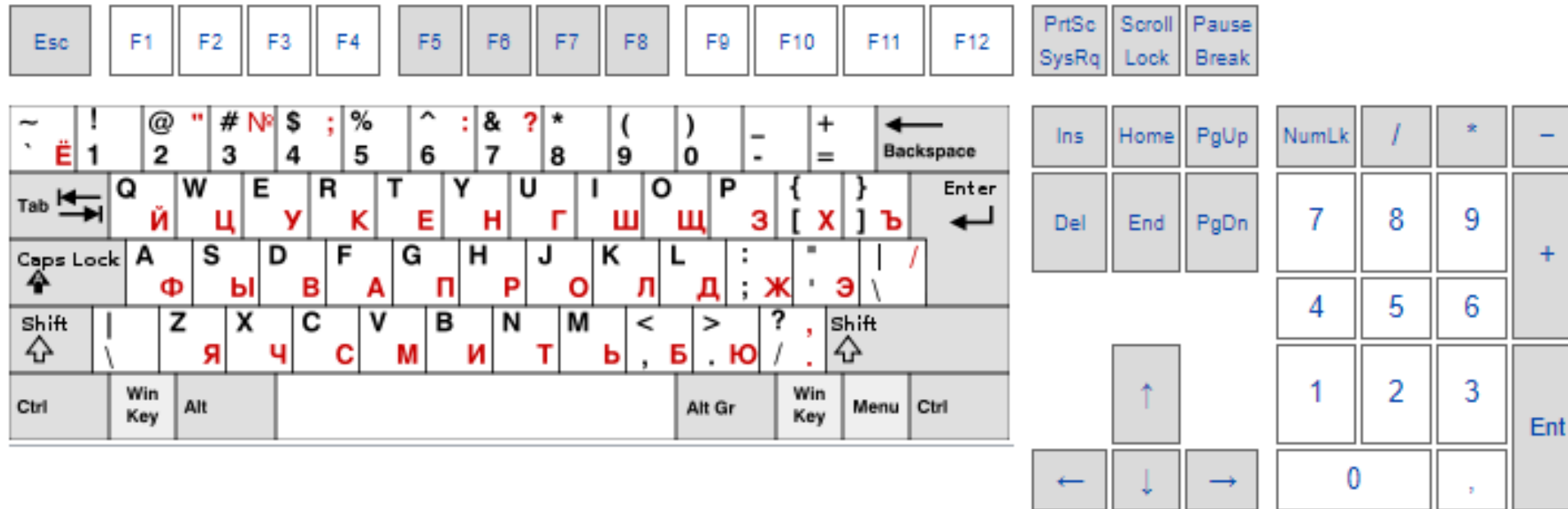


Fig. 7.1

Structura tastaturii dinamice. KBD Scan Codes:

01	Esc	12	E	23	H	34	. >	45	NumLock
02	1 !	13	R	24	J	35	/ ?	46	ScrollLock
03	2 @	14	T	25	K	36	Shft(нрав)	47	Home [7]
04	3 #	15	Y	26	L	37	* PrtSc	48	Up [8]
05	4 \$	16	U	27	; :	38	Alt	49	PgUp [9]
06	5 %	17	I	28	" '	39	Пробел	4a	K -
07	6 ^	18	O	29	~ `	3a	CapsLock	4b	<- [4]
08	7 &	19	P	2a	Shft(лев)	3b	F1	4c	[5]
09	8 *	1a	[{	2b	\	3c	F2	4d	-> [6]
0a	9 (1b] }	2c	Z	3d	F3	4e	K +
0b	0)	1c	Enter	2d	X	3e	F4	4f	End [1]
0c	- _	1d	Ctrl	2e	C	3f	F5	50	Dn [2]
0d	+ =	1e	A	2f	V	40	F6	51	PgDn [3]
0e	Bksp	1f	S	30	B	41	F7	52	Ins [0]
0f	Tab	20	D	31	N	42	F8	53	Del [.]
10	Q	21	F	32	M	43	F9		
11	W	22	G	33	, <	44	F10		

Fig. 8.1

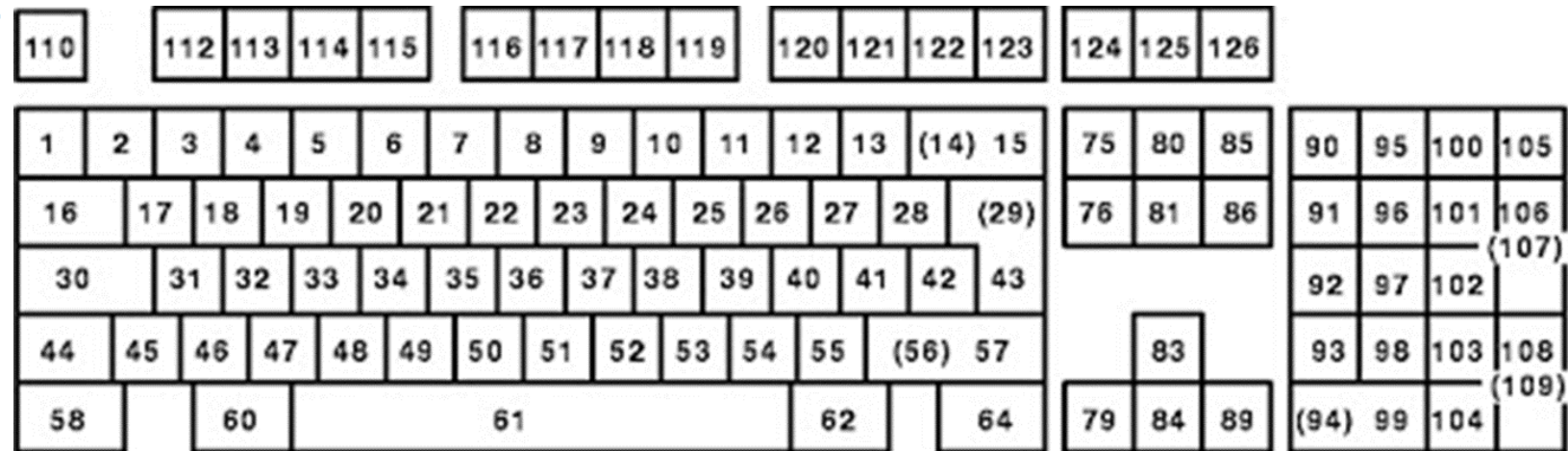


Fig. 8.2

Iteracțiunea Tastatură – PC (Aplicație):

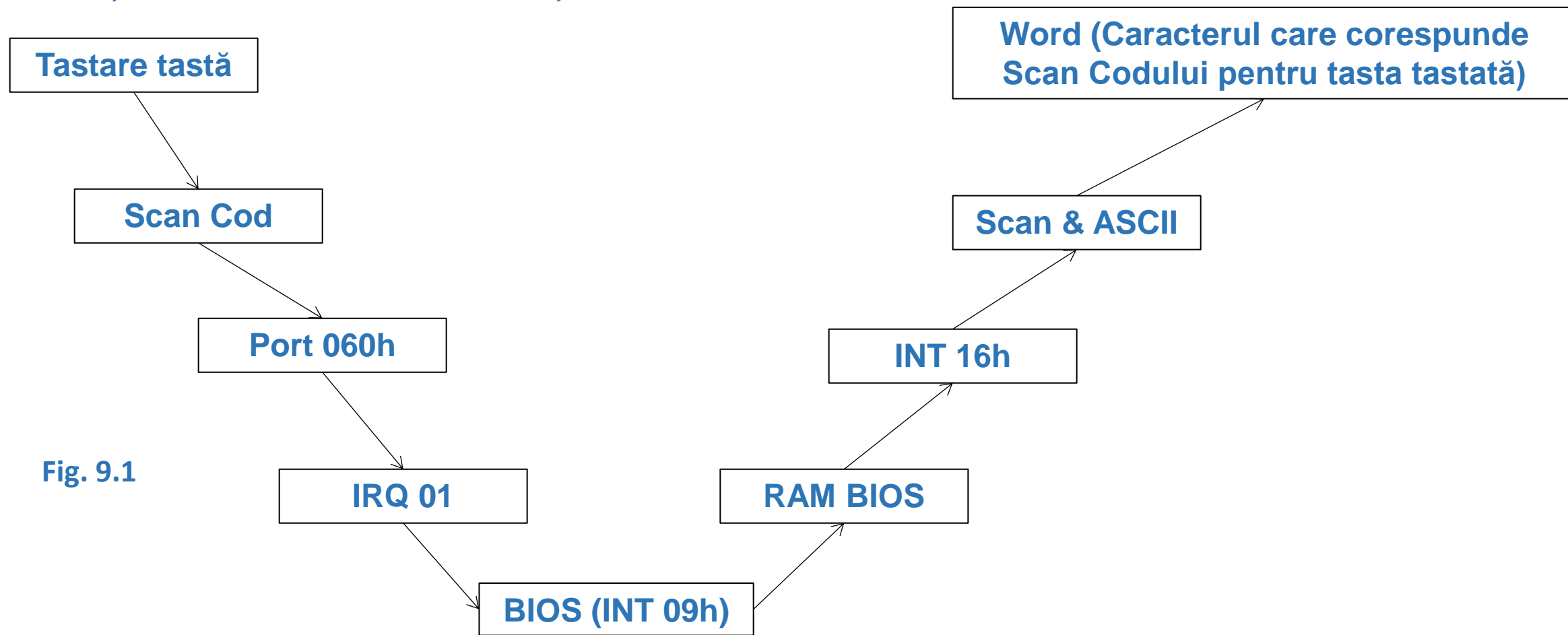


Fig. 9.1

Programarea tastaturii:

Funcții BIOS:

INT 16h / AH = 00h - get keystroke from keyboard (no echo).

return:

AH = BIOS scan code.

AL = ASCII character.

Exemplu: `mov ah, 0`

Int 16h

INT 16h / AH = 01h - check for keystroke in the keyboard buffer.

return:

AH = BIOS scan code.

AL = ASCII character.

Exemplu: `mov ah, 1`

Int 16h

Programarea tastaturii:

Funcții BIOS:

INT 16h / AH = 02h - check keyboard status.

return:

AL = status of shift and toggle keys.

Exemplu: `mov ah, 2`

`int 16h`

Funcționalitatea biților din cuvântul de stare KBD:

Bit#	Key assignment
0	Right SHIFT down
1	Left SHIFT down
2	CONTROL down
3	ALT down
4	SCROLL LOCK down
5	NUMBER LOCK down
6	CAPS LOCK down
7	INS LOCK down

Programarea tastaturii:

Funcții DOS INT 21h:

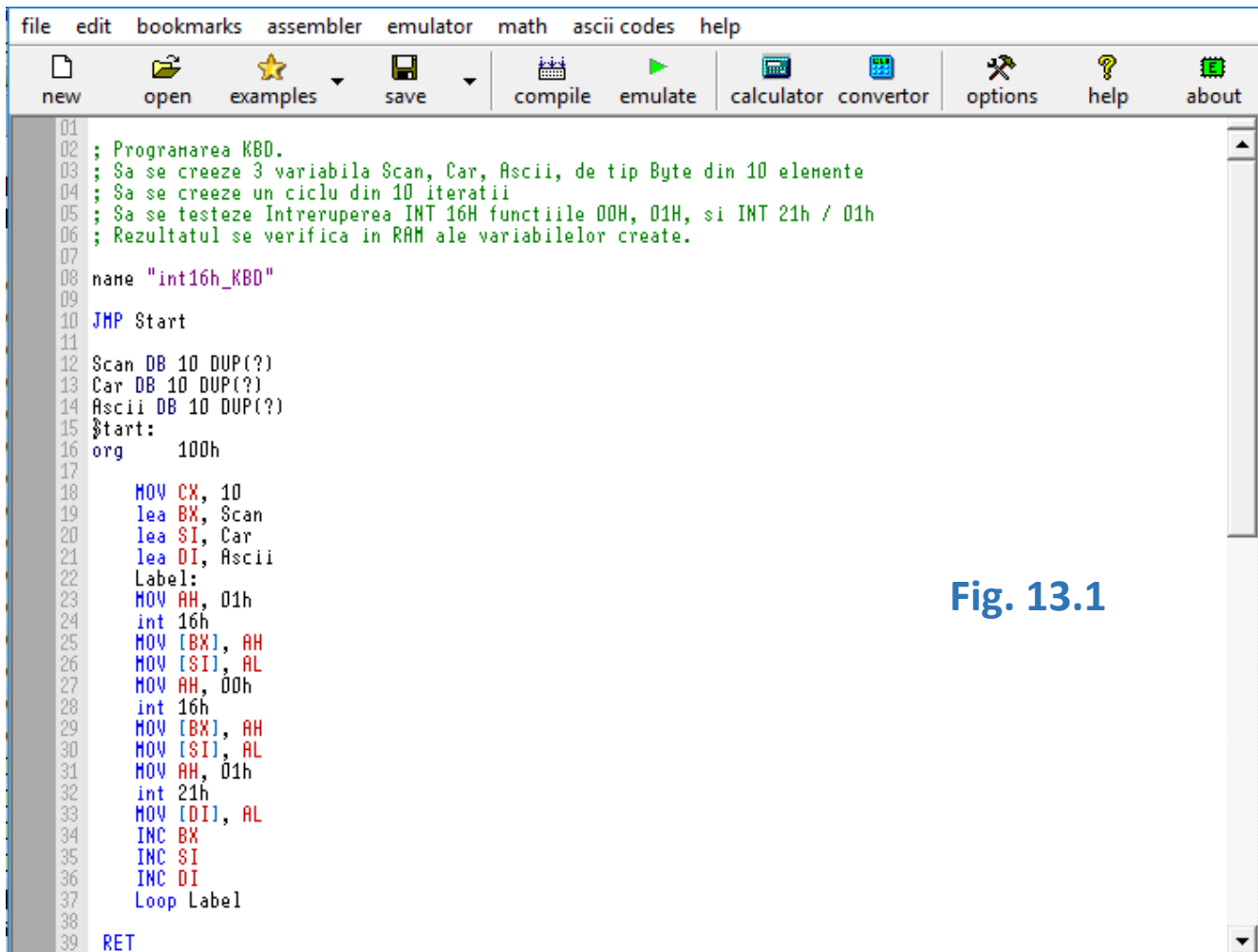
INT 21h / AH=1 - read character from standard input, with echo, result is stored in **AL**.
if there is no character in the keyboard buffer, the function waits until any key is pressed.

Exemplu:

```
mov ah, 1
```

```
int 21h
```

Exemplu de programare KBD. EMU8086:



```
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
01
02 ; Programarea KBD.
03 ; Sa se creeze 3 variabila Scan, Car, Ascii, de tip Byte din 10 elemente
04 ; Sa se creeze un ciclu din 10 iteratii
05 ; Sa se testeze Intreruperea INT 16H functiile 00H, 01H, si INT 21h / 01h
06 ; Rezultatul se verifica in RAM ale variabilelor create.
07
08 name "int16h_KBD"
09
10 JMP Start
11
12 Scan DB 10 DUP(?)
13 Car DB 10 DUP(?)
14 Ascii DB 10 DUP(?)
15 Start:
16 org 100h
17
18 MOV CX, 10
19 lea BX, Scan
20 lea SI, Car
21 lea DI, Ascii
22 Label:
23 MOV AH, 01h
24 int 16h
25 MOV [BX], AH
26 MOV [SI], AL
27 MOV AH, 00h
28 int 16h
29 MOV [BX], AH
30 MOV [SI], AL
31 MOV AH, 01h
32 int 21h
33 MOV [DI], AL
34 INC BX
35 INC SI
36 INC DI
37 Loop Label
38
39 RET
```

Fig. 13.1

Exemplu de programare KBD. EMU8086:

The image shows a screenshot of an 8086 emulator interface. The window title is "emulator: int16h_K.com_". The interface includes a menu bar (file, math, debug, view, external, virtual devices, virtual drive, help), a toolbar with buttons for Load, reload, step back, single step, run, and a step delay slider set to 0 ms. On the left, a registers window shows the state of various registers: AX (00 00), BX (00 00), CX (00 48), DX (00 00), CS (07 00), IP (01 00), SS (07 00), SP (FF FE), BP (00 00), SI (00 00), DI (00 00), DS (07 00), and ES (07 00). The main window is split into two panes. The left pane shows memory addresses from 07100 to 07114, with values like EB 235 and 1E 030. The right pane shows assembly code for "int16h_KBD". The code includes comments in Romanian and assembly instructions. Line 18, "MOV CX, 10", is highlighted in yellow. The code ends with "MOV AH, 01h".

Fig. 14.1

Fig. 14.2

```

01
02 ; Programarea KBD.
03 ; Sa se creeze 3 variabila Scan, Car, Ascii, de tip Byte din 10 elemente
04 ; Sa se creeze un ciclu din 10 iteratii
05 ; Sa se testeze Intreruperea INT 16H functiile 00H, 01H, si INT 21h / 01h
06 ; Rezultatul se verifica in RAM ale variabilelor create.
07
08 name "int16h_KBD"
09
10 JMP Start
11
12 Scan DB 10 DUP(?)
13 Car DB 10 DUP(?)
14 Ascii DB 10 DUP(?)
15 Start:
16 org 100h
17
18 MOV CX, 10
19 lea BX, Scan
20 lea SI, Car
21 lea DI, Ascii
22 Label:
23 MOV AH, 01h
24 int 16h
25 MOV [BX], AH
26 MOV [SI], AL
27 MOV AH, 00h
28 int 16h
29 MOV [BX], AH
30 MOV [SI], AL
31 MOV AH, 01h
  
```

Exemplu de programare KBD. EMU8086:

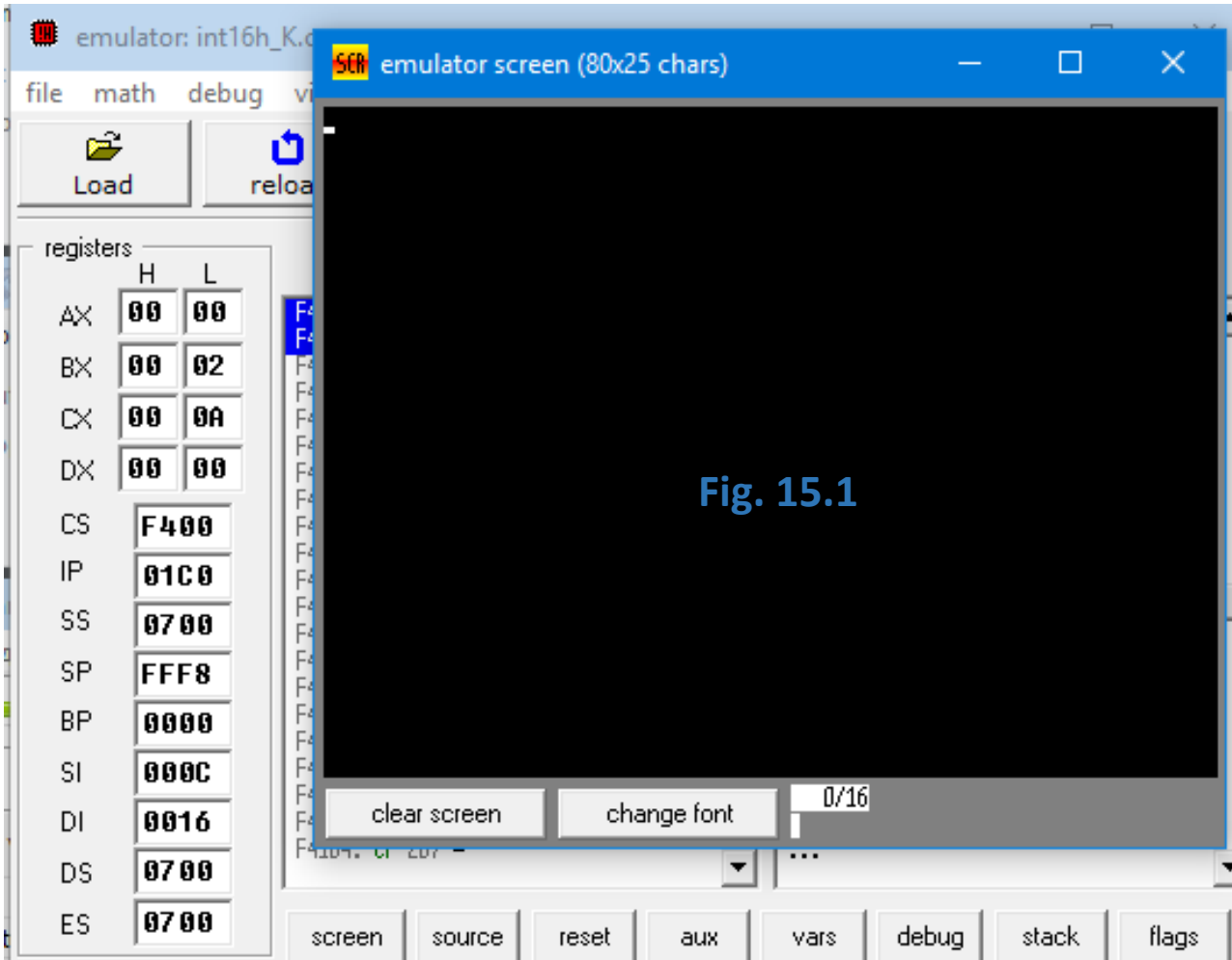


Fig. 15.1

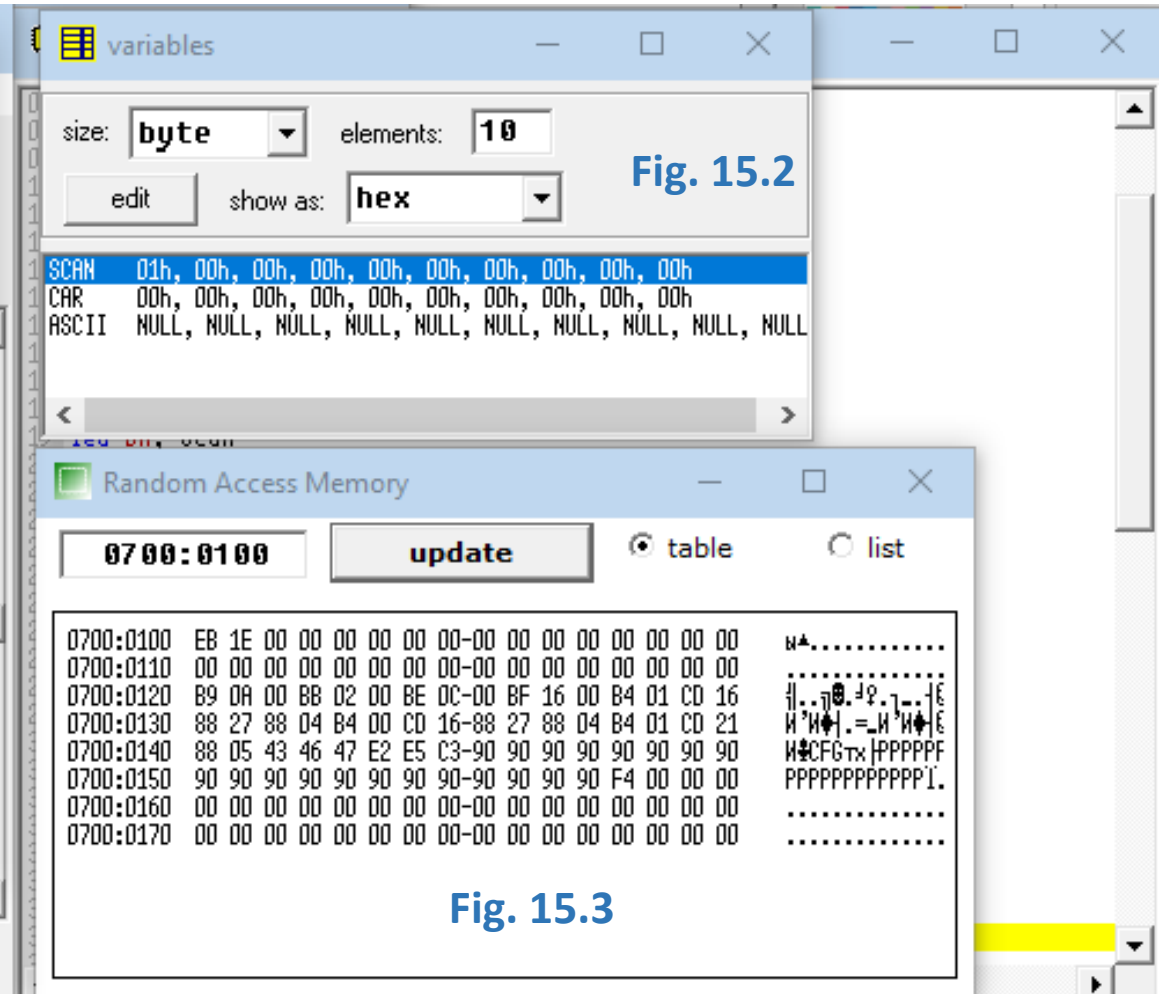


Fig. 15.2

Fig. 15.3

Exemplu de programare KBD. EMU8086:

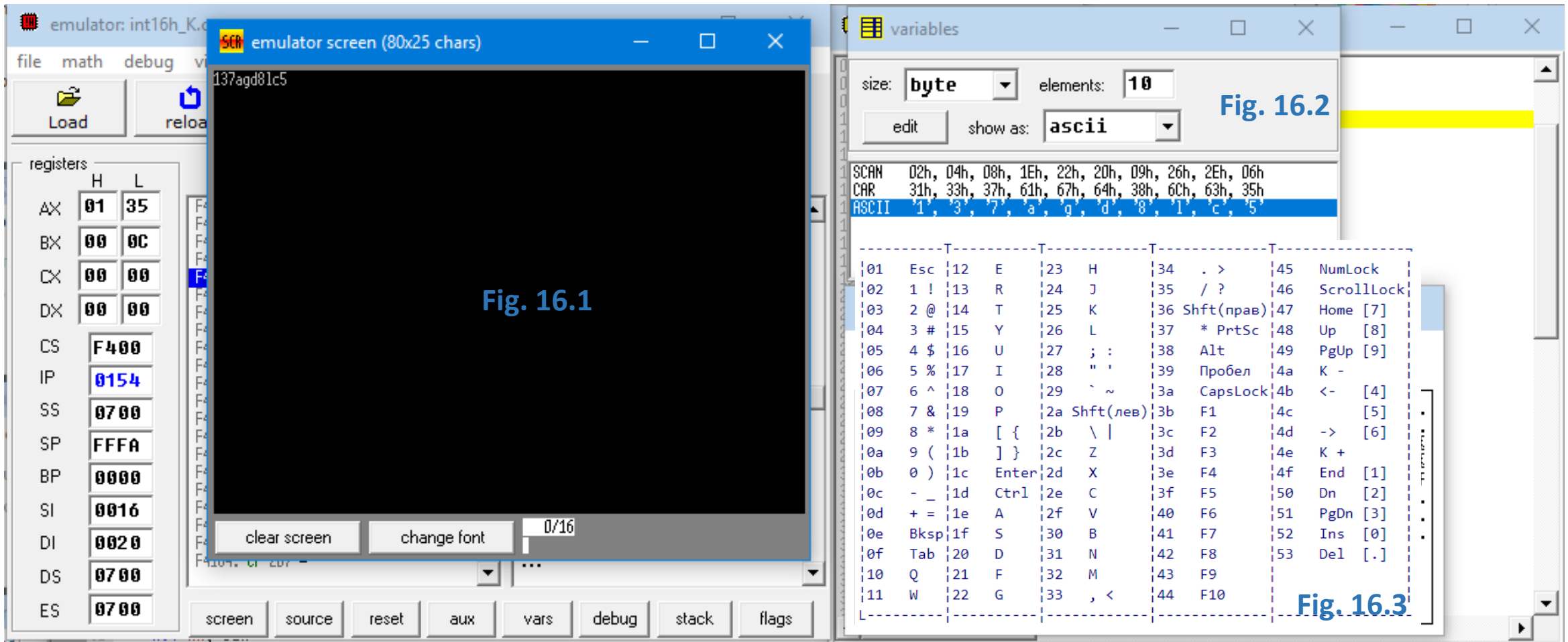


Fig. 16.1

Fig. 16.2

Fig. 16.3

SCAN	02h, 04h, 08h, 1Eh, 22h, 20h, 09h, 26h, 2Eh, 06h								
CAR	31h, 33h, 37h, 61h, 67h, 64h, 38h, 6Ch, 63h, 35h								
ASCII	'1', '3', '7', 'a', 'q', 'd', '8', 'l', 'c', '5'								
{01	Esc	{12	E	{23	H	{34	. >	{45	NumLock
{02	1 !	{13	R	{24	J	{35	/ ?	{46	ScrollLock
{03	2 @	{14	T	{25	K	{36	Shft(прав)	{47	Home [7]
{04	3 #	{15	Y	{26	L	{37	* PrtSc	{48	Up [8]
{05	4 \$	{16	U	{27	; :	{38	Alt	{49	PgUp [9]
{06	5 %	{17	I	{28	" '	{39	Пробел	{4a	K -
{07	6 ^	{18	O	{29	~	{3a	CapsLock	{4b	<- [4]
{08	7 &	{19	P	{2a	Shft(лев)	{3b	F1	{4c	[5]
{09	8 *	{1a	[{	{2b	\	{3c	F2	{4d	-> [6]
{0a	9 ({1b] }	{2c	Z	{3d	F3	{4e	K +
{0b	0)	{1c	Enter	{2d	X	{3e	F4	{4f	End [1]
{0c	- _	{1d	Ctrl	{2e	C	{3f	F5	{50	Dn [2]
{0d	+ =	{1e	A	{2f	V	{40	F6	{51	PgDn [3]
{0e	Bksp	{1f	S	{30	B	{41	F7	{52	Ins [0]
{0f	Tab	{20	D	{31	N	{42	F8	{53	Del [.]
{10	Q	{21	F	{32	M	{43	F9		
{11	W	{22	G	{33	, <	{44	F10		

Mouse-ul este destinat pentru gestiunea poziției cursorului grafic al Sistemului de Operare și efectuarea unor operații de comandă în raport cu coordonatele ecranului. Numărul de comenzi este determinat de complexitatea acestuia și numărul de butoane sau elemente funcționale prezente.

Clasificarea și caracteristici de bază ale Mouse-ului:

- Tipul de senzori: mecanici sau optici;
- Interfața de comunicare cu PC: COM, PS2, USB, Wireless, IrDA;
- Numărul de taste funcționale (L, R, Scroll).

Resursele rezervate de arhitectura sistemului de calcul pentru gestiunea mouse-ului:

- Întreruperi Hardware: IRQ12 (INT 74h);
- Întreruperi Software: BIOS / DOS (INT 33h);
- Porturi I/O: 060h – 06Fh;
- RAM gestionată de SO.



Fig. 17.1

Mouse-ul mecanic:

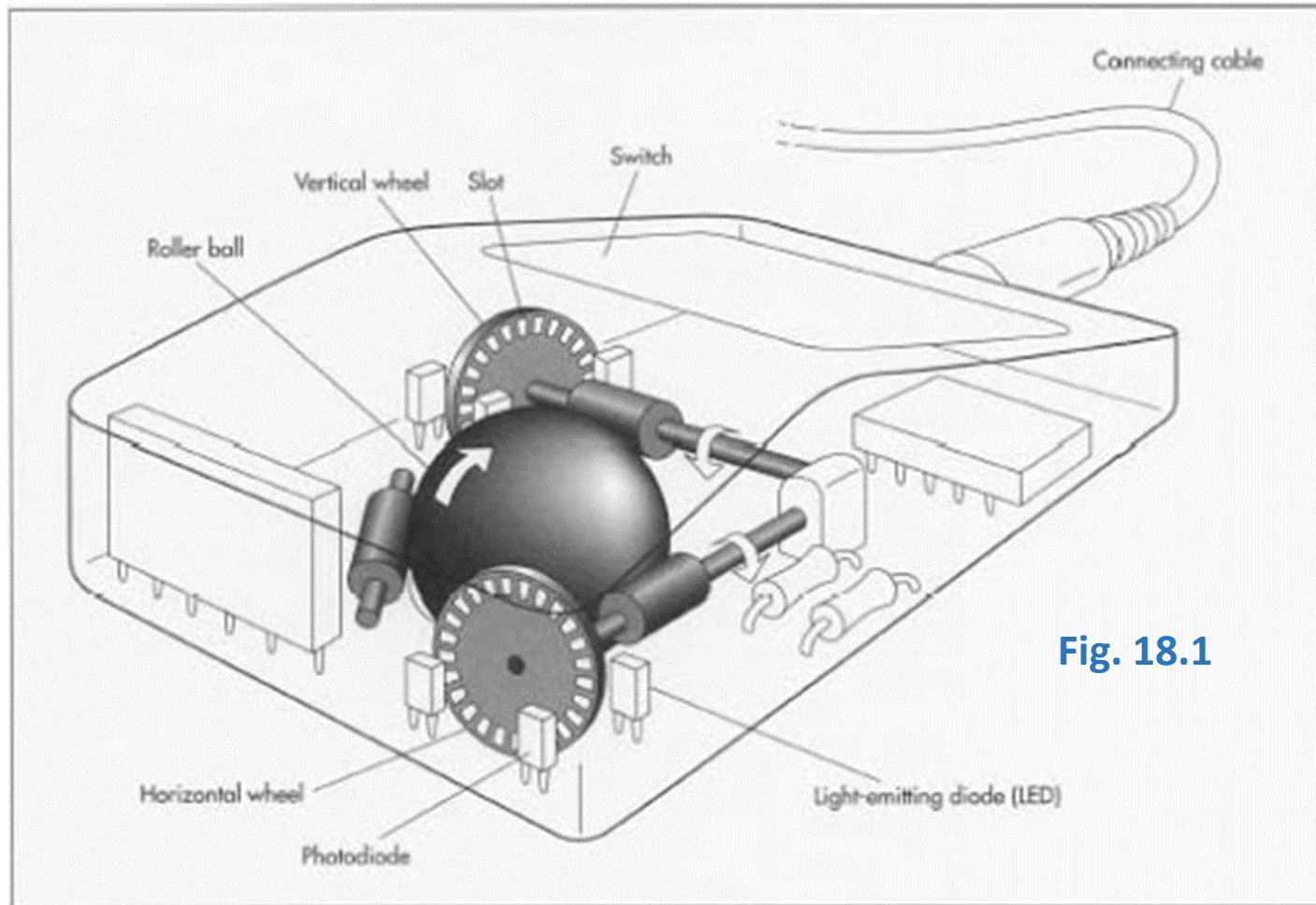


Fig. 18.1

Mouse-ul optic:

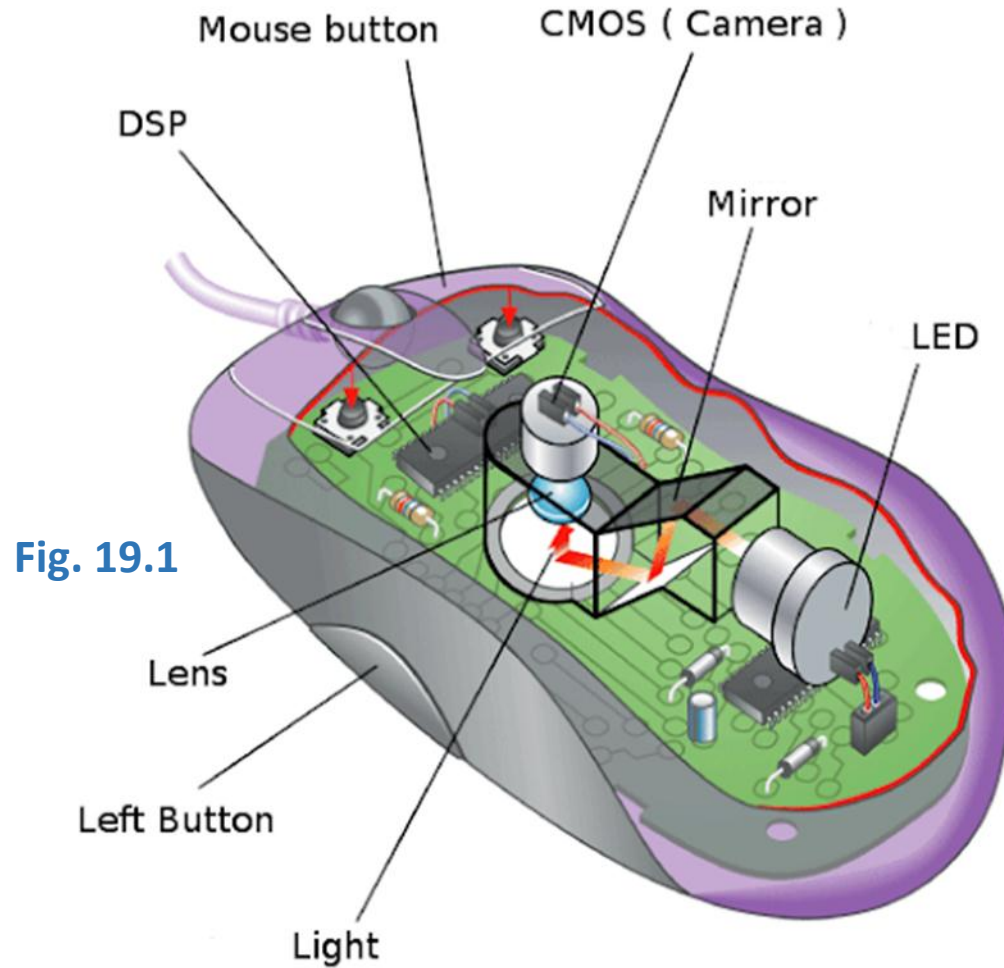
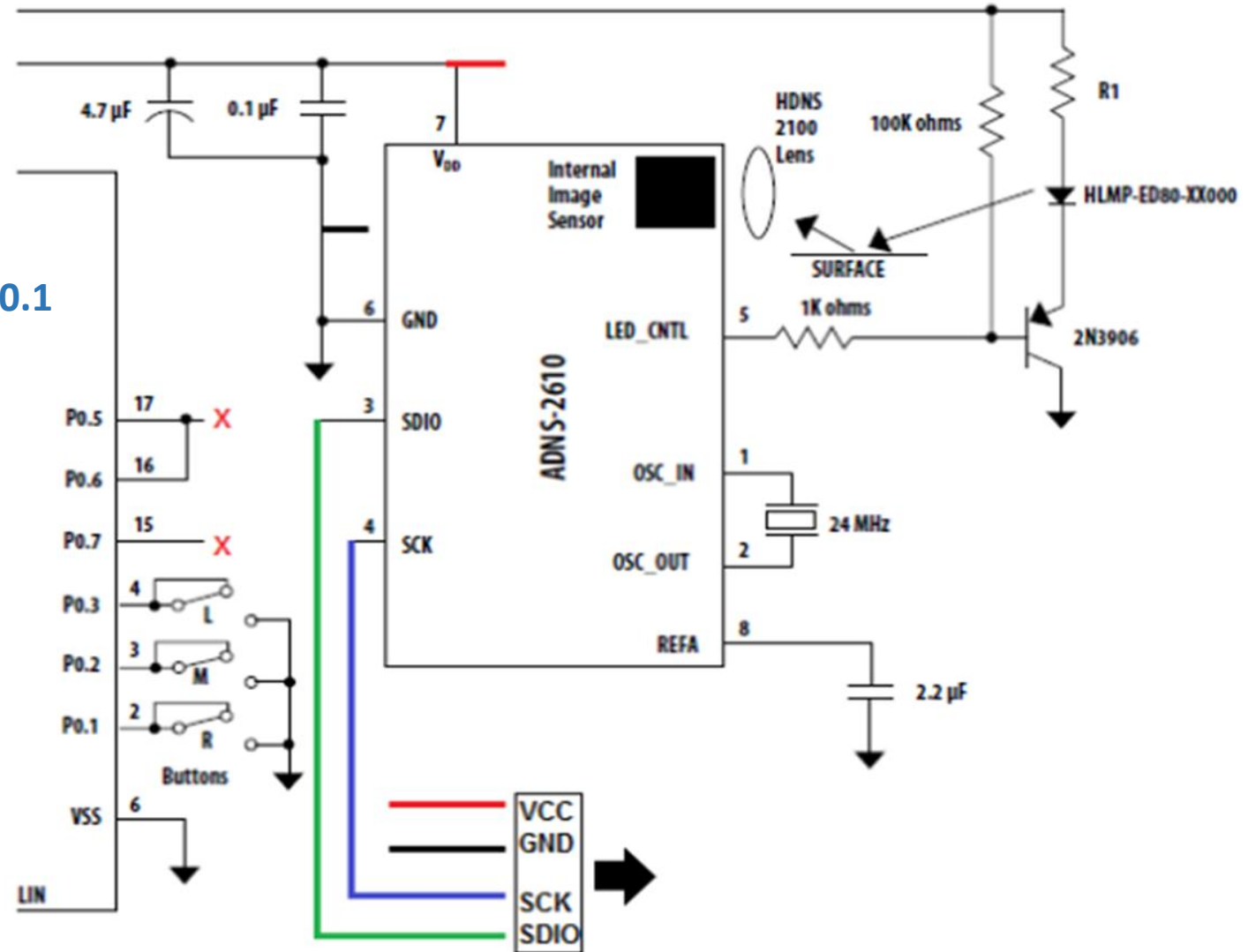


Fig. 19.1

Mouse-ul optic. Structura:

Fig. 20.1



Interacțiunea Mouse - PC:

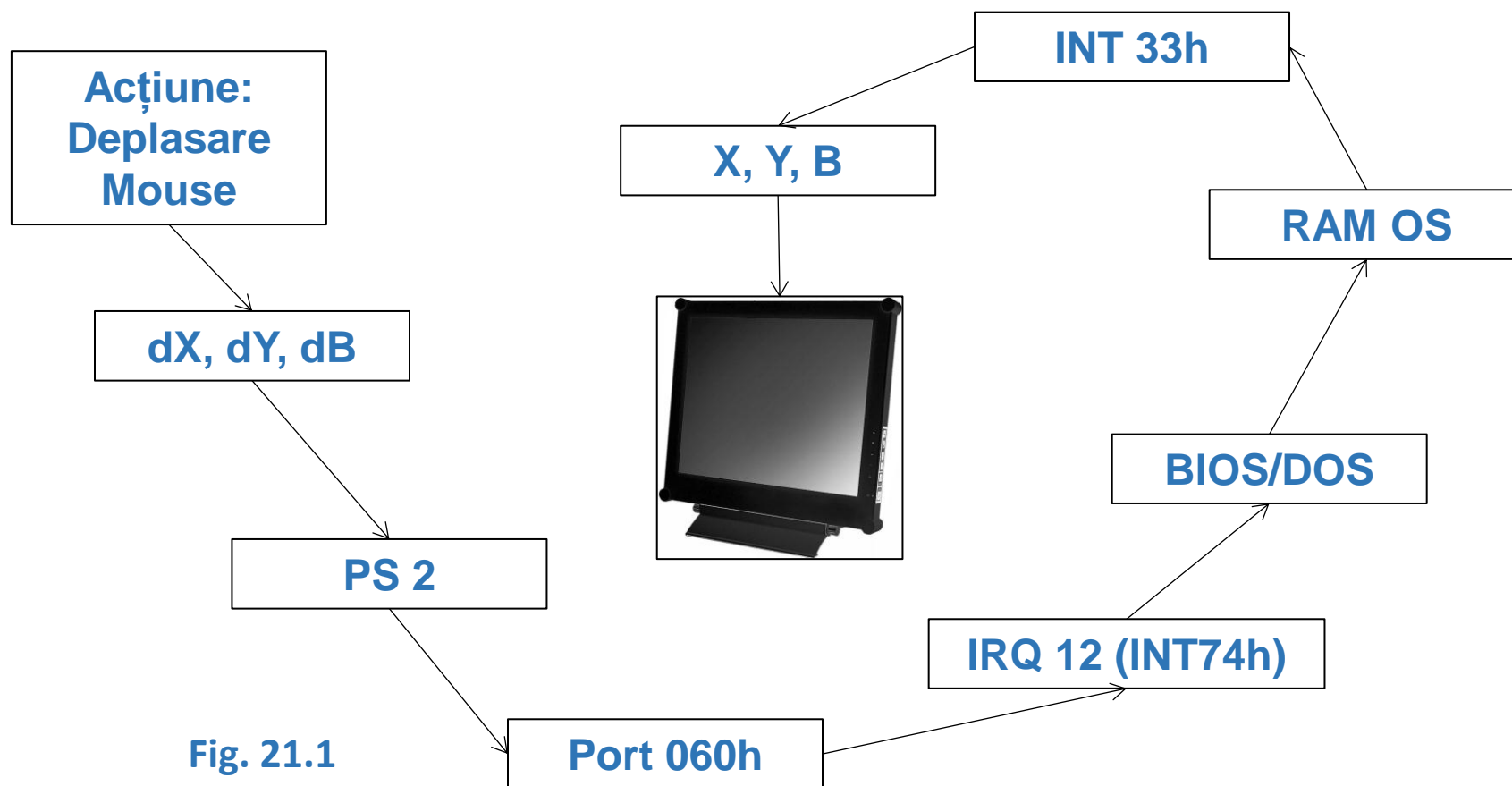


Fig. 21.1

Programarea Mouse-ului:

Pentru programarea Mouse-ului este rezervată întreruperea DOS INT 33h.

Această întrerupere oferă următoarele funcții:

INT 33h / AX=0000 - mouse initialization. any previous mouse pointer is hidden.

Returns: if successful: **AX**=0FFFFh and **BX**=number of mouse buttons.

if failed: **AX**=0

Example:

```
mov ax, 0
```

```
int 33h
```

INT 33h / AX=0001 - show mouse pointer.

Example:

```
mov ax, 1
```

```
int 33h
```

Programarea Mouse-ului:

Pentru programarea Mouse-ului este rezervată întreruperea DOS INT 33h.

INT 33h / AX=0002 - hide visible mouse pointer.

Example:

```
mov ax, 2  
int 33h
```

INT 33h / AX=0003 - get mouse position and status of its buttons.

Returns:

if left button is down: **BX=1**

if right button is down: **BX=2**

if both buttons are down: **BX=3**

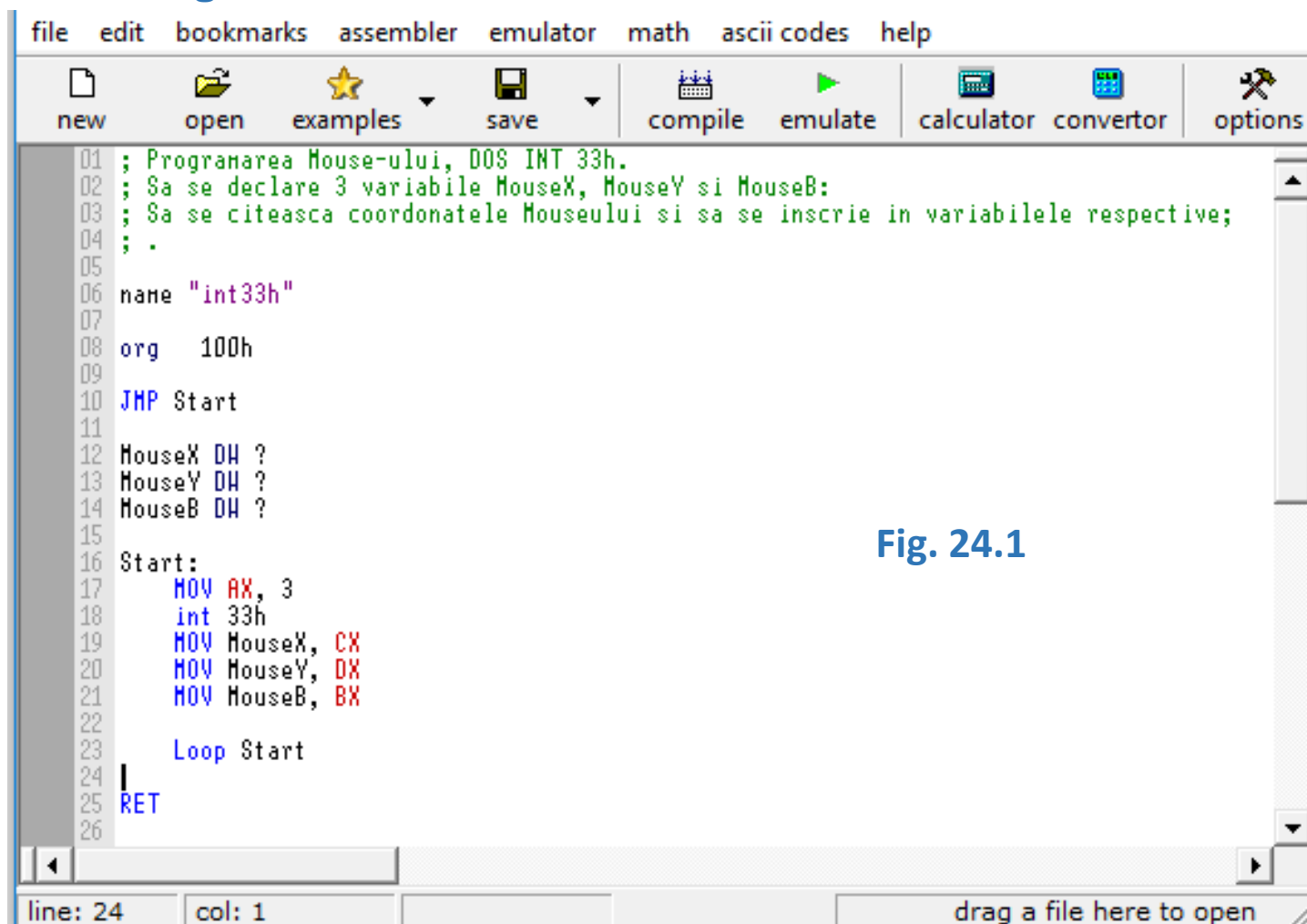
CX = x

DX = y

example:

```
mov ax, 3  
int 33h
```

Programarea Mouse-ului:



```
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options
01 ; Programarea Mouse-ului, DOS INT 33h.
02 ; Sa se declare 3 variabile MouseX, MouseY si MouseB:
03 ; Sa se citeasca coordonatele Mouseului si sa se inscrie in variabilele respective;
04 ; .
05
06 name "int33h"
07
08 org 100h
09
10 JMP Start
11
12 MouseX DH ?
13 MouseY DH ?
14 MouseB DH ?
15
16 Start:
17     MOV AX, 3
18     int 33h
19     MOV MouseX, CX
20     MOV MouseY, DX
21     MOV MouseB, BX
22
23     Loop Start
24
25 RET
26
```

line: 24 col: 1 drag a file here to open

Fig. 24.1

Programarea Mouse-ului:

emulator: int33h.com_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

	H	L
AX	00	00
BX	00	00
CX	00	1C
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0100 0700:0100

Address	Disassembly	Comment
07100:	EB 235	M
07101:	06 006	↓
07102:	00 000	NULL
07103:	00 000	NULL
07104:	00 000	NULL
07105:	00 000	NULL
07106:	00 000	NULL
07107:	00 000	NULL
07108:	88 184	↑
07109:	03 003	↓
0710A:	00 000	NULL
0710B:	CD 205	=
0710C:	33 051	3
0710D:	89 137	R
0710E:	0E 014	↓
0710F:	02 002	0
07110:	01 001	0
07111:	89 137	R
07112:	16 022	-
07113:	04 004	↓
07114:	01 001	0

JMP 0108h
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
MOV AX, 00003h
INT 033h
MOV [00102h], CX
MOV [00104h], DX
MOV [00106h], BX
LOOP 0108h
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...

screen source reset aux vars debug stack flags

Fig. 25.1

original source code

```

01 ; Programarea Mouse-ului, DOS INT 33h.
02 ; Sa se declare 3 variabile MouseX, MouseY si MouseB:
03 ; Sa se citeasca coordonatele Mouseului si sa se inscrie in variabilele res
04 ; .
05
06 name "int33h"
07
08 org 100h
09
10 JMP Start
11
12 MouseX DH ?
13 MouseY DH ?
14 MouseB DH ?
15
16 Start:
17 MOV AX, 3
18 int 33h
19 MOV MouseX, CX
20 MOV MouseY, DX
21 MOV MouseB, BX
22
23 Loop Start
24
25 RET
26
27
28

```

Fig. 25.2

Programarea Mouse-ului:

The screenshot shows an emulator window titled "emulator: int33h.com_". The interface includes a menu bar (file, math, debug, view, external, virtual devices, virtual drive, help) and a toolbar with buttons for Load, reload, step back, single step, run, and a step delay slider set to 0 ms. On the left, a registers window displays the state of various registers: AX (00 00), BX (00 00), CX (00 1C), DX (00 00), CS (07 00), IP (01 00), SS (07 00), SP (FF FE), BP (00 00), SI (00 00), DI (00 00), DS (07 00), and ES (07 00). The main window is split into two panes. The left pane shows memory addresses from 07100 to 07114 with their corresponding hex values and comments. The right pane shows assembly instructions starting with "JMP 0108h" and a series of "ADD [BX + SI], AL" instructions, followed by "MOV AX, 00003h", "INT 033h", "MOV [00102h], CX", "MOV [00104h], DX", "MOV [00106h], BX", "LOOP 0108h", and "RET". At the bottom, there are buttons for screen, source, reset, aux, vars, debug, stack, and flags.

Fig. 26.1

This block contains two overlapping windows. The top window is titled "variables" and shows a list of variables: MOUSEX (0000h), MOUSEY (0000h), and MOUSEB (0000h). The "MOUSEB" variable is highlighted in blue. The window also has settings for "size: word", "elements: 1", and "show as: hex". The bottom window is titled "Random Access Memory" and shows a memory dump for the address range 0700:0100 to 0700:0170. The dump displays hex values and their ASCII representations. For example, at 0700:0100, the hex is EB 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00, and the ASCII is "w.....=3f".

Fig. 26.2

Fig. 26.3

Programarea Mouse-ului:

The screenshot shows the main interface of the emulator. On the left, the registers window displays the following values:

Register	H	L
AX	00	03
BX	00	00
CX	02	53
DX	00	01
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The assembly code window shows the following instructions:

```

MOV [00102h], CX
MOV [00104h], DX
MOV [00106h], BX
LOOP 0208h
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
  
```

At the bottom, there are buttons for screen, source, reset, aux, vars, debug, stack, and flags.

Fig. 27.1

The screenshot shows two windows from the emulator. The 'variables' window displays the following data:

Variable	Value
MOUSEX	01E2h
MOUSEY	0051h
MOUSEB	0003h

The 'Random Access Memory' window shows a memory dump for the address range 0700:0100 to 0700:0170:

```

0700:0100 EB 06 E2 01 51 00 03 00-B8 03 00 CD 33 89 DE 02
0700:0110 01 89 16 04 01 89 1E 06-01 E2 ED C3 90 90 90
0700:0120 90 90 90 90 90 90 90 90-90 90 90 90 90 90
0700:0130 F4 00 00 00 00 00 00 00-00 00 00 00 00 00
0700:0140 00 00 00 00 00 00 00 00-00 00 00 00 00 00
0700:0150 00 00 00 00 00 00 00 00-00 00 00 00 00 00
0700:0160 00 00 00 00 00 00 00 00-00 00 00 00 00 00
0700:0170 00 00 00 00 00 00 00 00-00 00 00 00 00 00
  
```

Fig. 27.2

Fig. 27.3

Camera Video este destinată pentru achiziția informației grafice prin metoda optică. Informația video este achiziționată de un foto-receptor care transformă fluxul de lumină care cade pe suprafața acestuia în semnale electrice care ulterior și formează structura imaginii video.

Clasificarea și caracteristicile de bază ale camerelor video:

- **Tehnologie: Analogice și Digitale (CMOS, CMOS BSI, CCD);**
- **Rezoluție: numărul de puncte discrete pe unitate de măsură (pixel/inci);**
- **Marimea imaginii: numărul total de pixeli pe imagine;**
- **Gradul de cuantizare a culorilor: 8, 10, 12 biți;**
- **Gradația culorilor: a/n, color (RGB: 8, 16, 24, 32)**
- **Numărul de cadre pe secundă: frecvența de scanare;**
- **Formatul prezentării informației: MPEG-1/2/4, H.264, ...**
- **Interfața de comunicare (Modul de stocare a informației video):**
USB, Ethernet, WiFi, SCSI, ...;
- **Sensibilitatea;**

Camera Video Analogică:

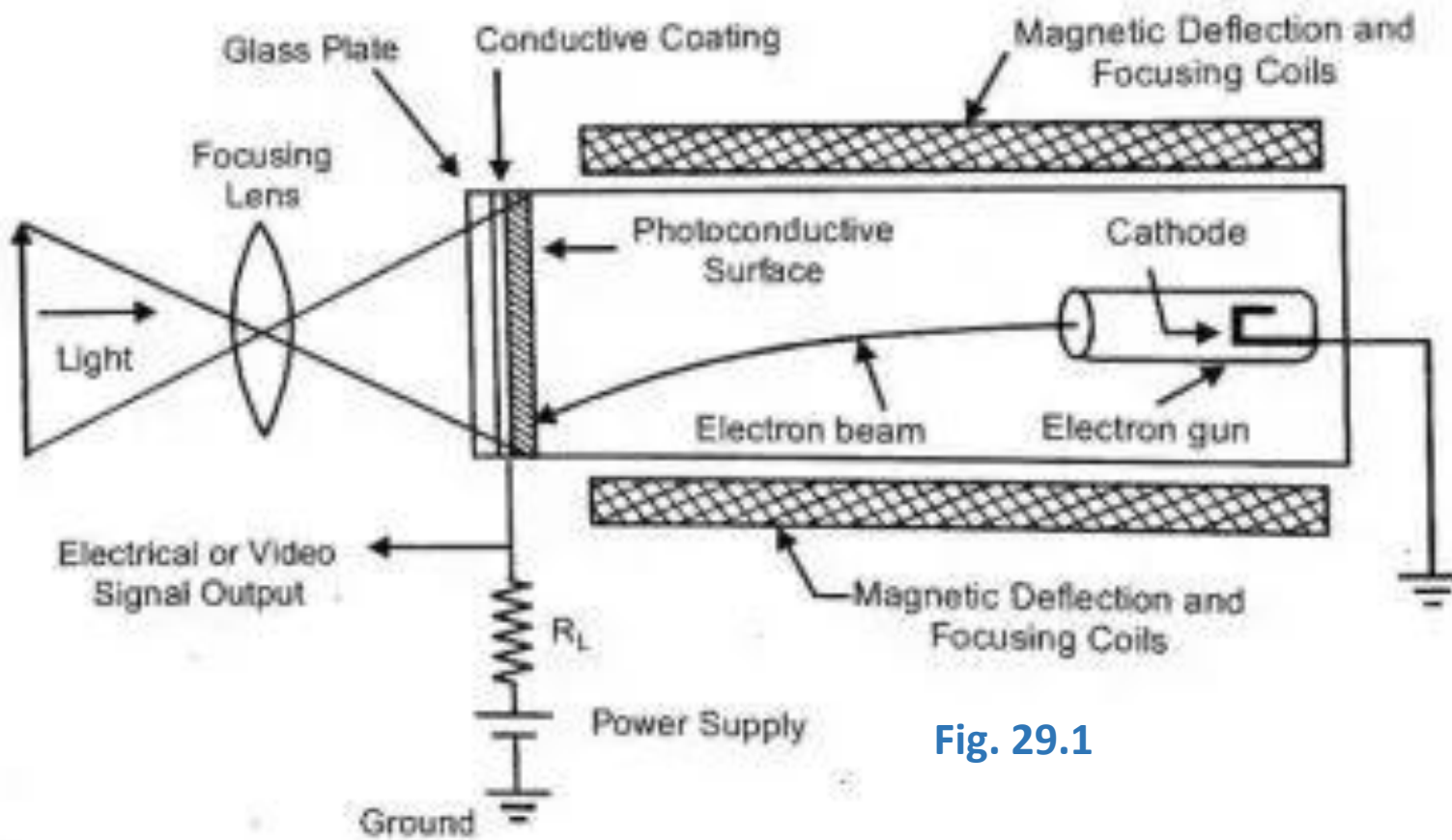


Fig. 29.1

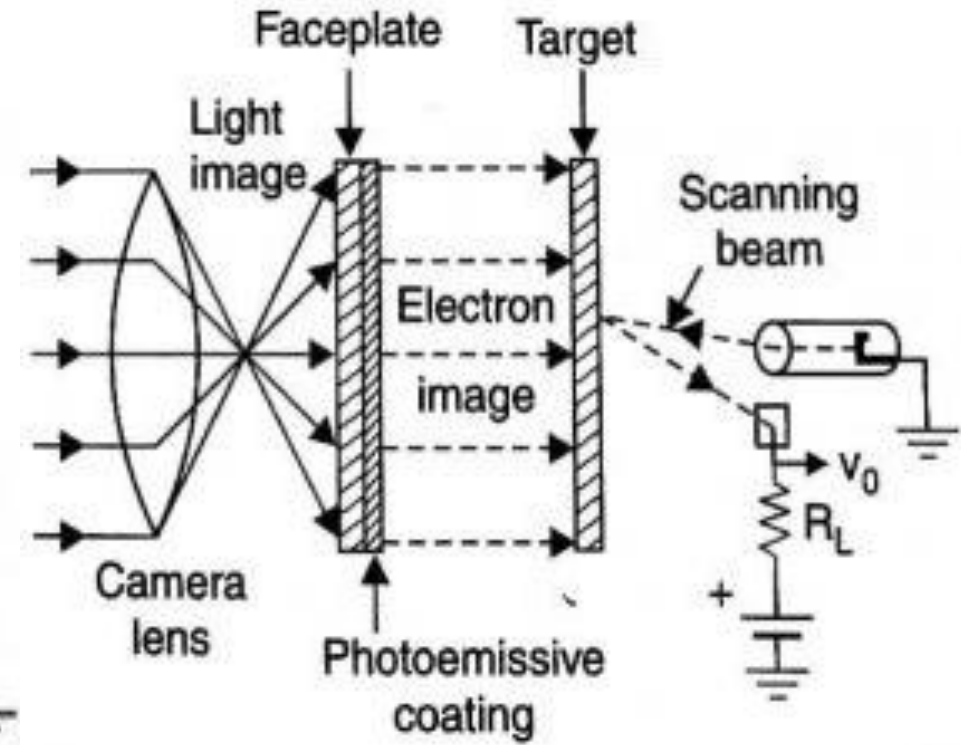


Fig. 29.2

Camera Video Digitală:

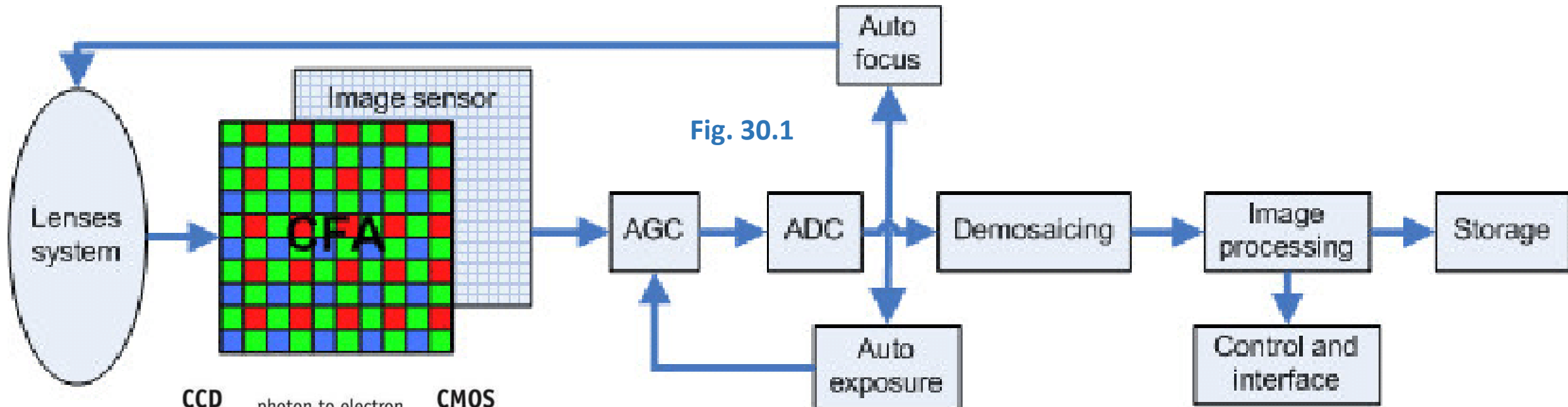


Fig. 30.1

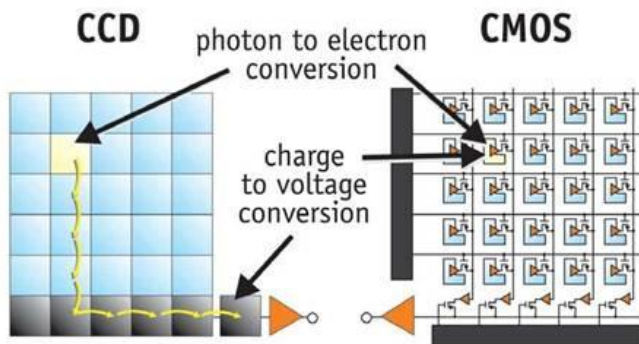
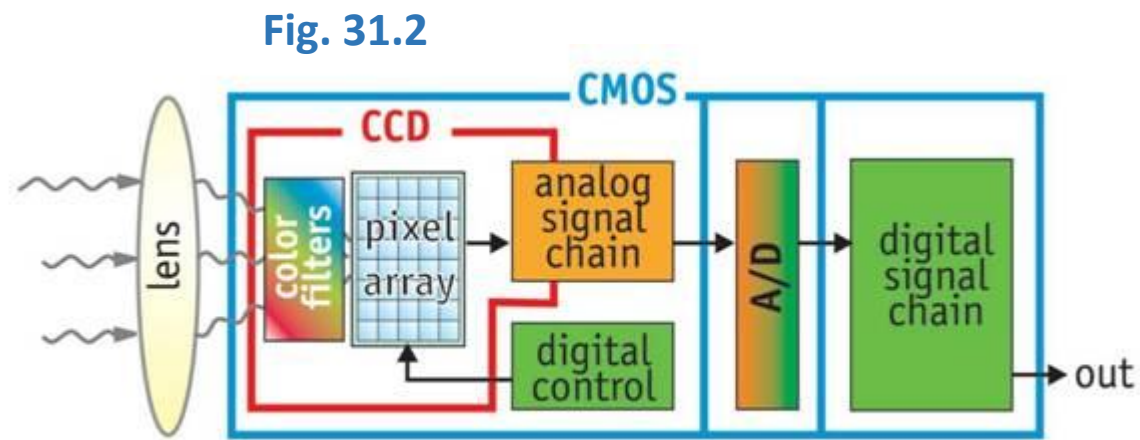
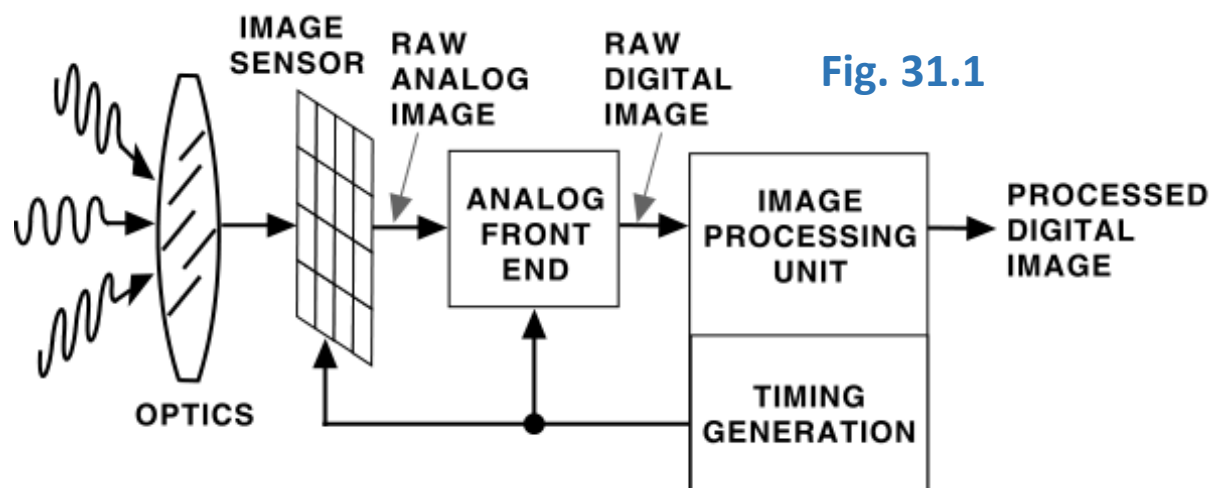


Fig. 30.2

Camera Video Digitală:



Două tehnologii pentru capturarea imaginii:

- **Dispozitivul cu cuplaj de sarcină** (Dispozitivul Cuplat cu Încărcătură) (**DCS**) / **Charge-Coupled Device**, abreviat **CCD**). Un senzor CCD este format dintr-un tablou de diode fotosensibile care captează datele imaginii și un tablou de memorare care le preia, atunci când este cuplat la tabloul de diode.
- **CMOS** (Complementary Metal-Oxide Semiconductor).

Scannerul este destinat pentru achiziția informației de pe suport fizic (hârtie) prin metoda scanării optice. Această operație permite digitalizarea imaginilor și introducerea acestora în calculator. Modul de funcționare se bazează pe aplicarea unui flux de lumină asupra obiectului scanat, reflectarea acestui flux de lumină și achiziția cu ajutorul dispozitivelor fotosensibile.

După modul de funcționare și tehnologia aplicată Scanerele sunt clasificate:

- Modul de antrenare: manual și automat;
- Modul de poziționare a suportului: manual, automat;
- Dimensiune suport: A4, ...;
- Rezoluția optică: numărul de puncte discrete pe unitate de măsură (pixel/inci);
- Tehnologia matricei de senzori: CMOS, CCD;
- Gradul de cuantizare a culorilor: 8, 10, 12 biți;
- Gradația culorilor: a/n, color (RGB: 8, 16, 24, 32)
- Interfața de comunicare cu PC-ul: SCSI, USB, Ethernet, etc.

Scannerul plat.



Fig. 33.1

Scannerul cu alimentare de hârtie



Fig. 33.2



Fig. 33.3

Scannerul. Modul de scanare cu deplasarea blocului de scanare.

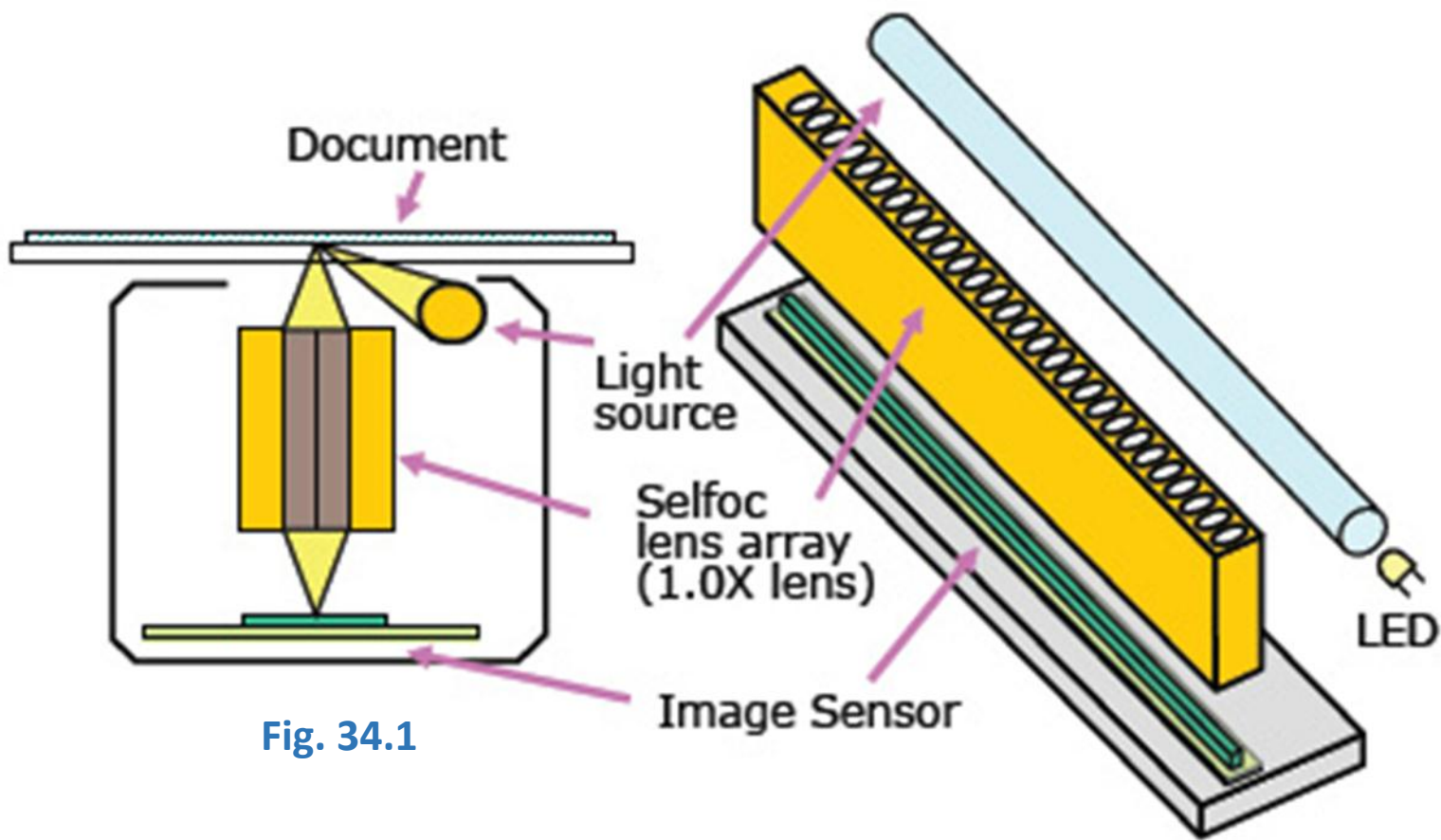


Fig. 34.1

Scannerul. Modul de scanare cu deplasarea suportului informațional.

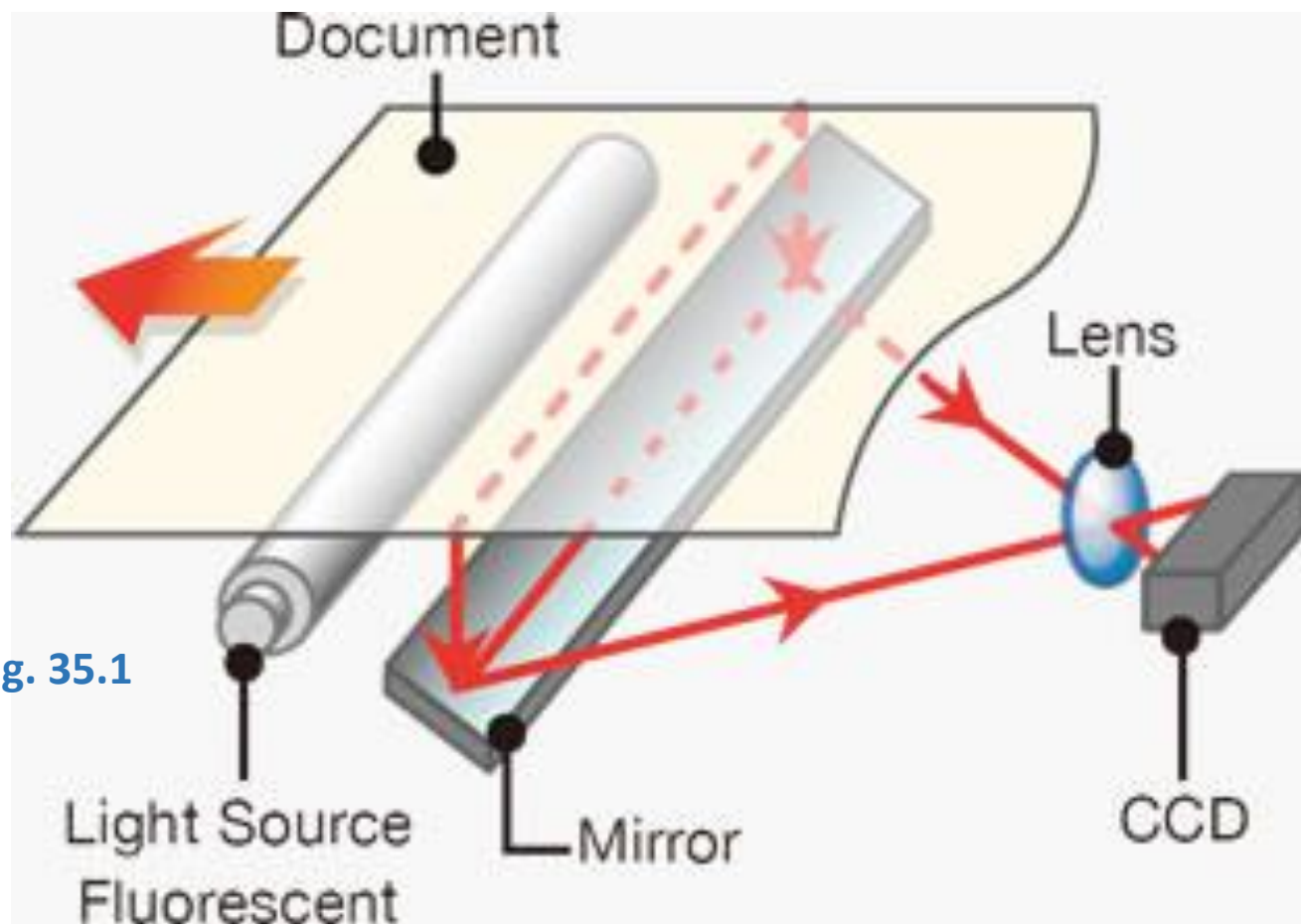


Fig. 35.1

Scannerul. Modul de scanare cu deplasarea oglinzilor de reflecție.

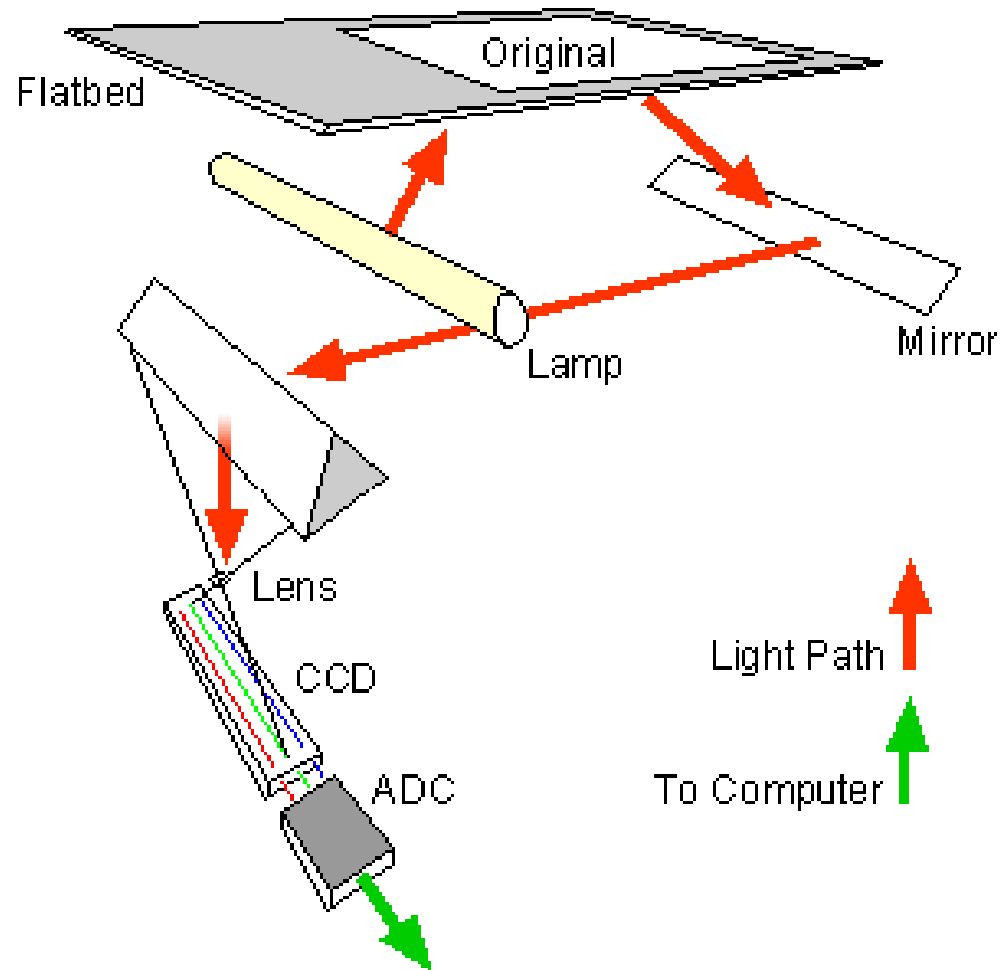
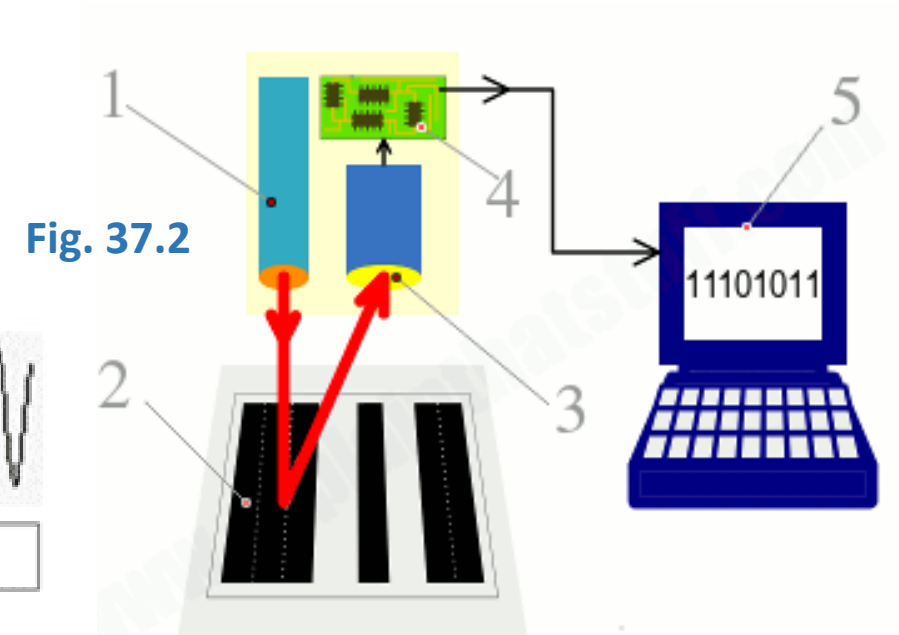
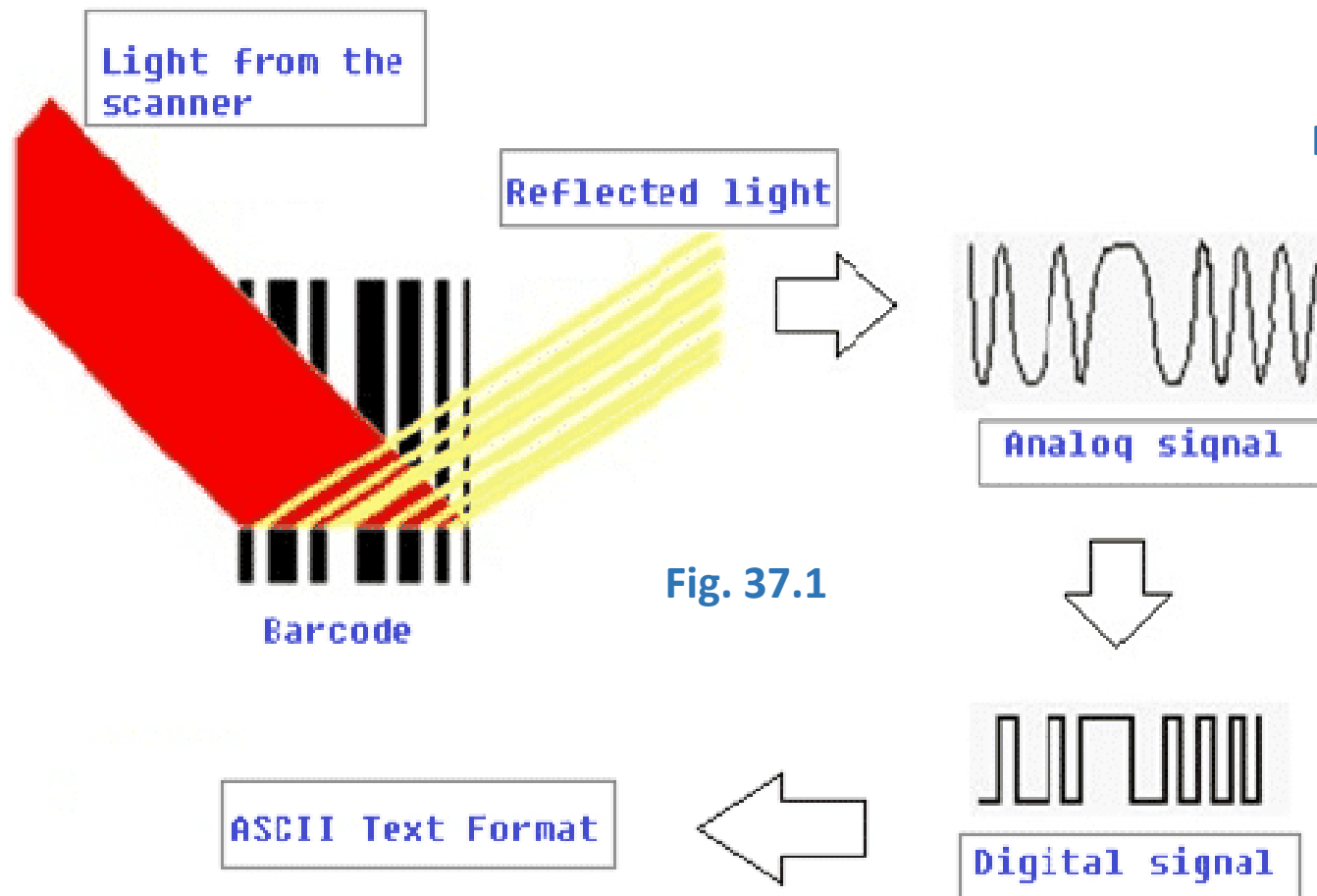


Fig. 36.1

Scannerul. Bar Code Scanner/Reader:



Touch Screen-ul este destinat pentru achiziția datelor și poate înlocui tastatura sau mouse-ul. Ca de obicei este plasat pe suprafața unui ecran LCD. Modul de funcționare se bazează pe componenta sensibilă la atingere sau deformarea acestuia. Aceste dispozitive sunt în telefoanele mobile, tablete și altele. Aceste dispozitive sunt clasificate după următorii parametri:

- Tehnologia funcțională: capacitive, rezistive, optice, acustice, inductive;
- Dimensiunea senzorului tactil;
- Sensibilitatea senzorului;
- Exactitatea detectării poziției tastate.

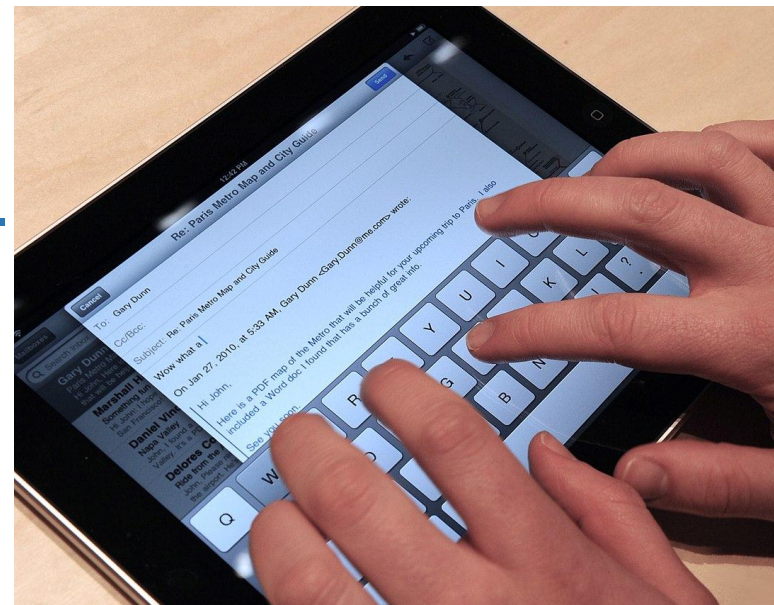
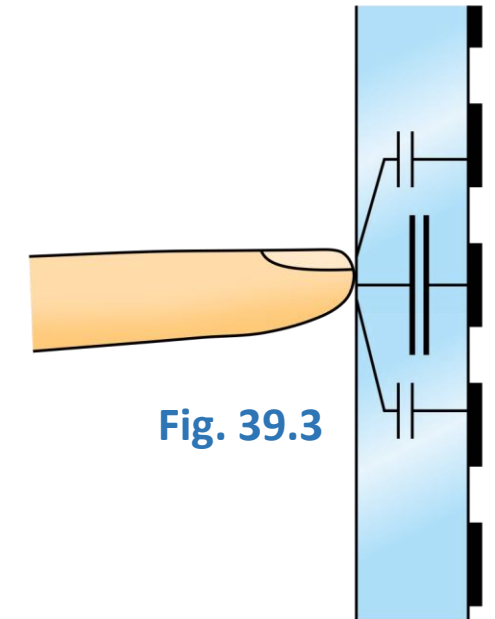
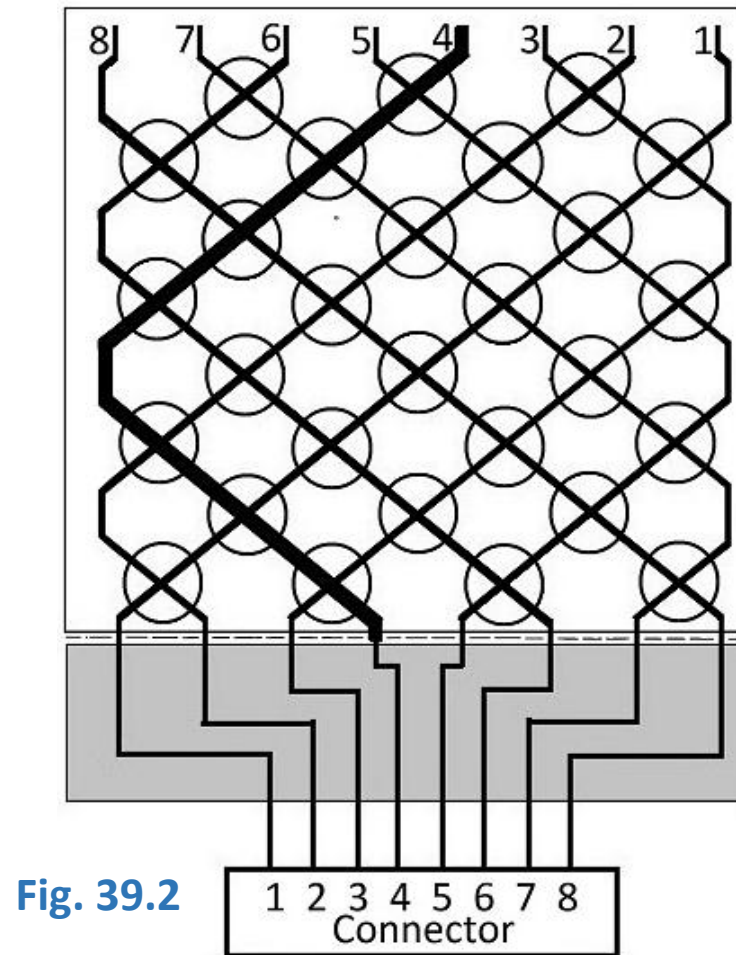
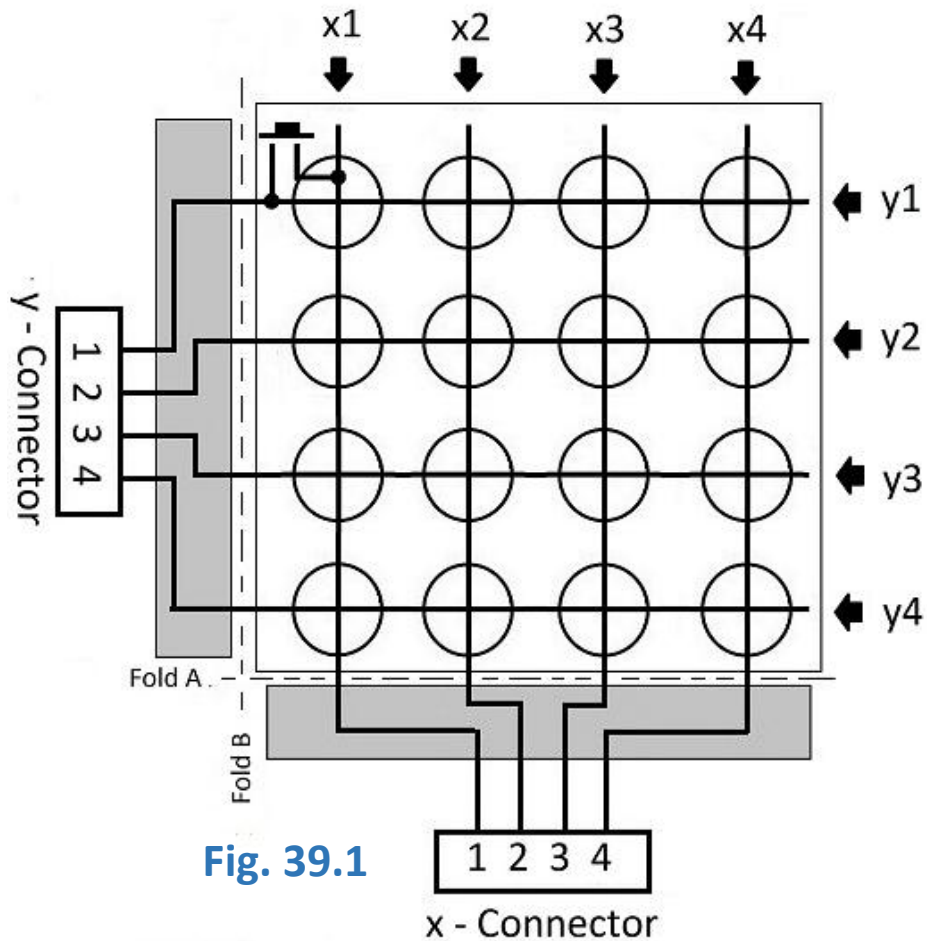
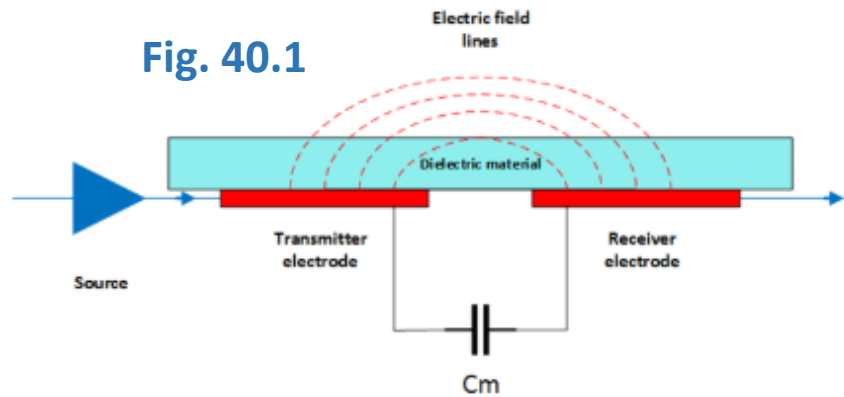


Fig. 38.1

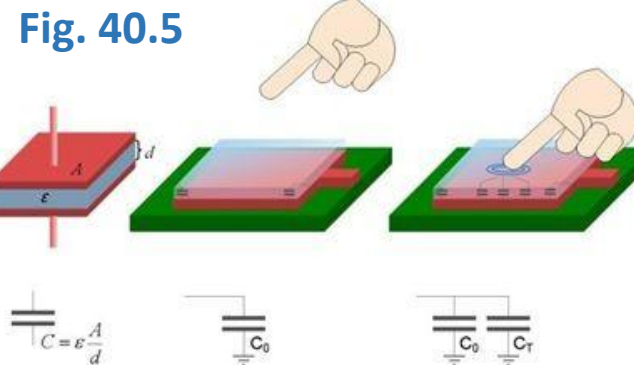
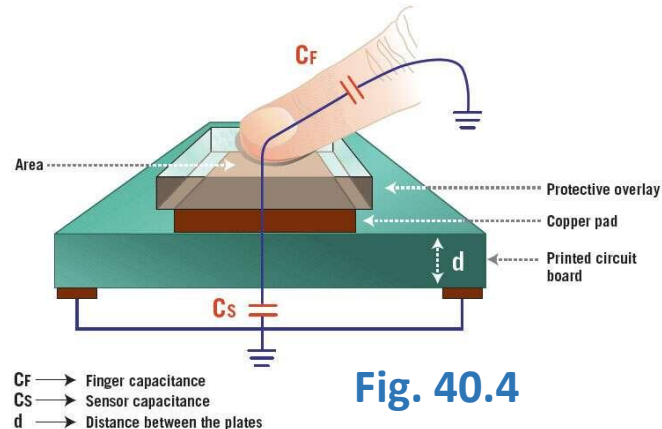
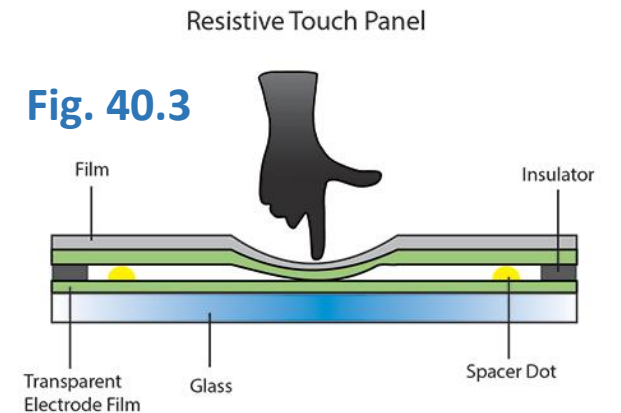
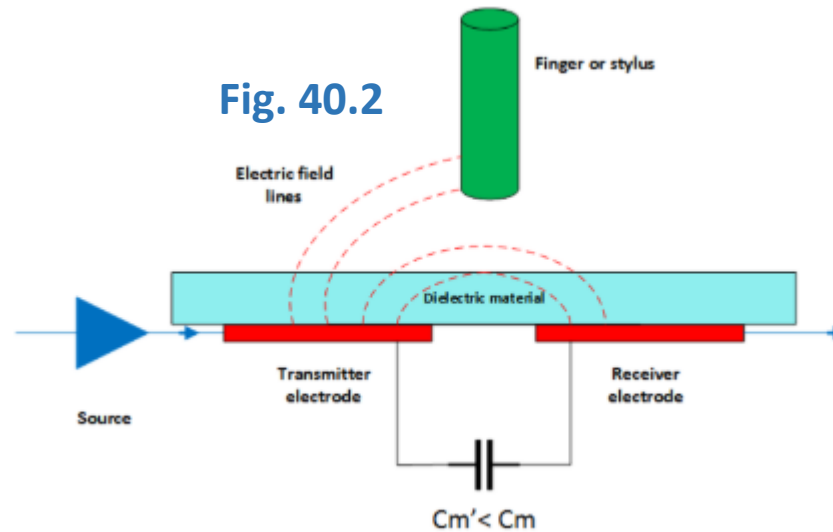
Touch Screen-ul. Structura și modul de funcționare:



Touch Screen-ul. Structura și modul de funcționare. Sensor Capacitativ:



The principles of capacitive touch sensing.



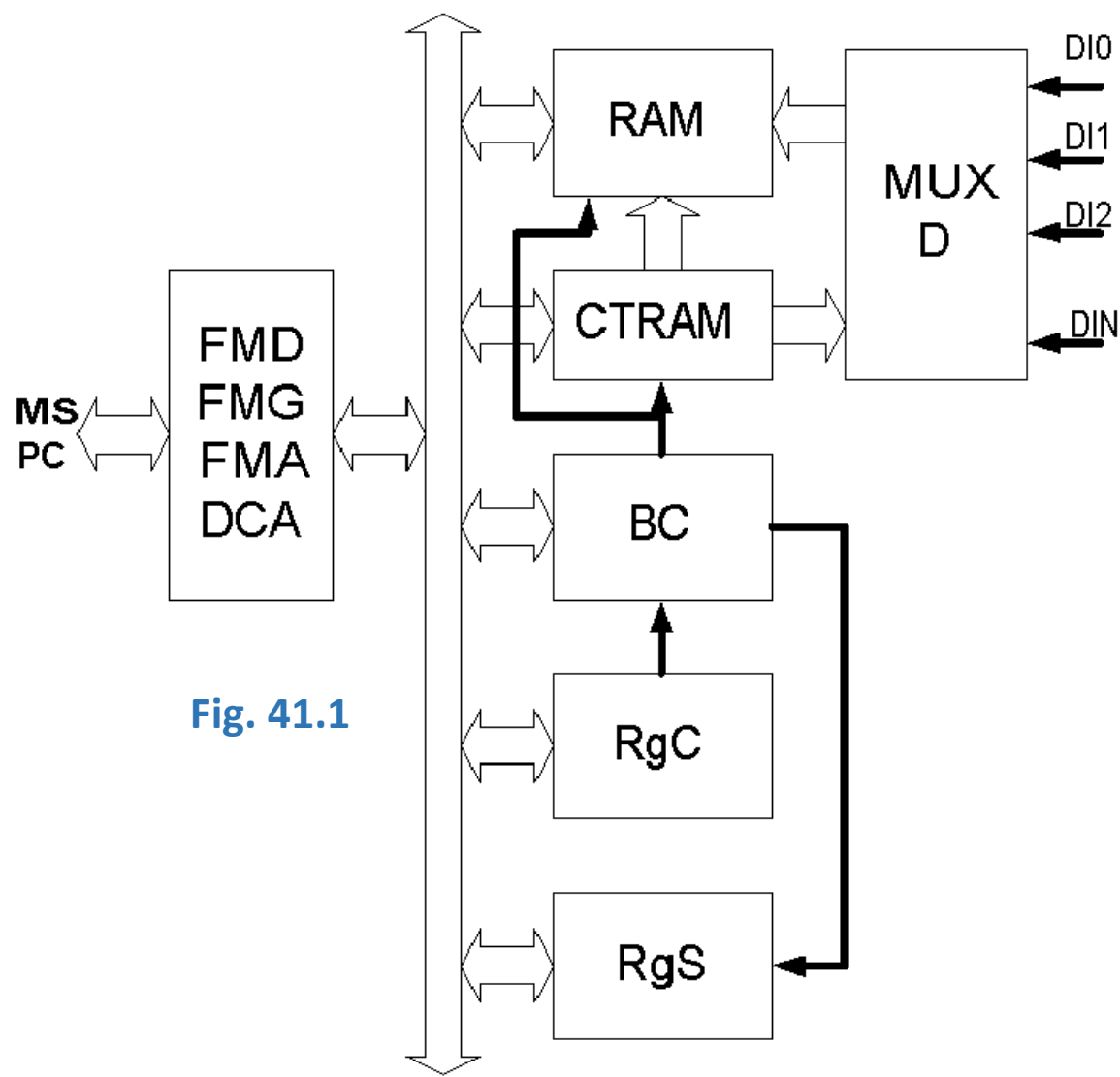


Fig. 41.1

Procesul de sinteză a interfeței pentru achiziția datelor discrete include următoarele etape:

- Proiectarea blocului de comandă (BC) este proiectat în baza circuitelor logice combinaționale și cu memorie sau în baza unui Microcontrolor universal AVR ATmega, Intel 8051, PIC...;
- Proiectarea RAM statică sau dinamică;
- Proiectarea multiplexorului digital (MUX D);
- Proiectarea blocului de comunicare cu PC. Este proiectat cu funcții de întrerupere Hardware, canale DMA și Porturi I/O.

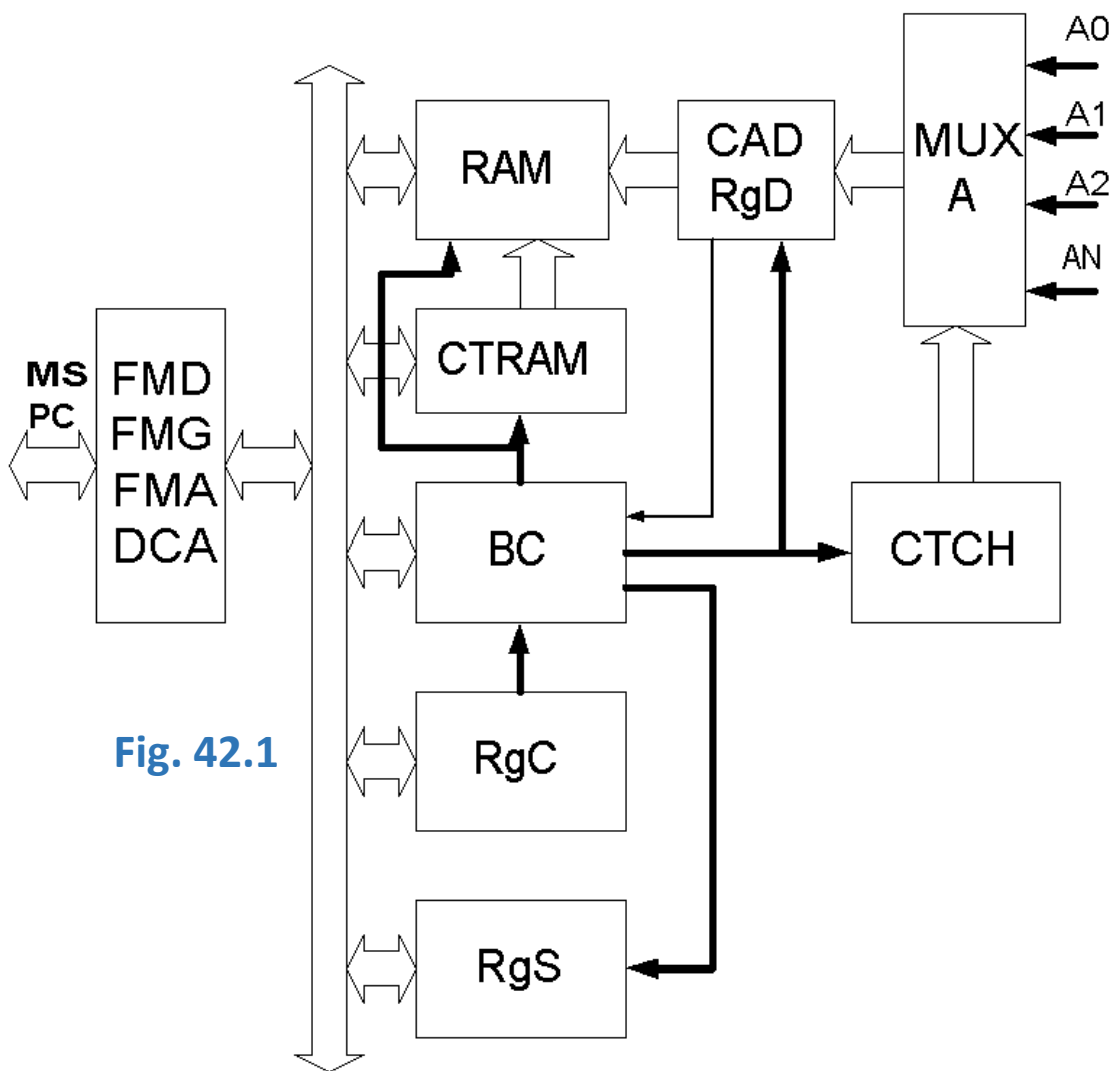


Fig. 42.1

Proiectarea interfeței pentru achiziția semnalelor analogice include următoarele etape de sinteză:

- Proiectarea blocului de comanda (BC) în baza MCU sau scheme combinaționale;
- Proiectarea RAM statică sau dinamică;
- Proiectarea multiplexorului analogic (MUX A);
- Proiectarea Convertorului Analog-Digital (CAD): integrare dublă, aproximare secvențială, paralel;
- Proiectarea blocului de comunicare cu PC: canale DMA, Int, Porturi I/O.

Convertorul Analog-Digital (CAD) este un dispozitiv funcțional care acceptă la intrare o mărime analogică (curent sau tensiune) și furnizează la ieșire un cod de aproximare a acesteia. Principalele caracteristici ale Convertoarelor Analog-Digitale sunt:

- **Tehnologia de conversie Analog-Digitală: Integrare dublă, aproximare secvențială, paralelă;**
- **Frecvența de cuantificare: 1KHz, 100 KHz, 1 MHz, 100 MHz;**
- **Rezoluția: numărul de stări posibile pentru aproximare: 8, 10, 12 biți;**
- **Eroarea de cuantificare.**

Convertorul Analog-Digital (CAD). Structura:

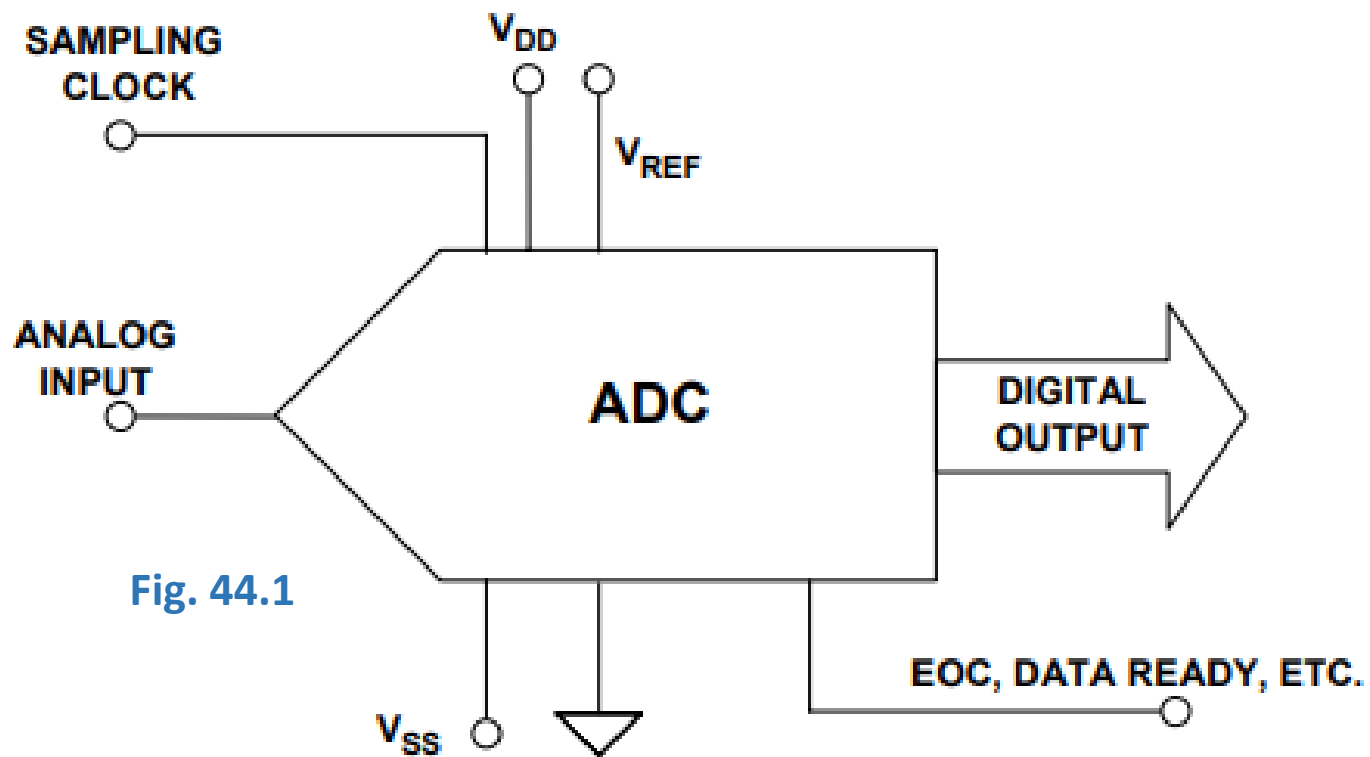


Fig. 44.1

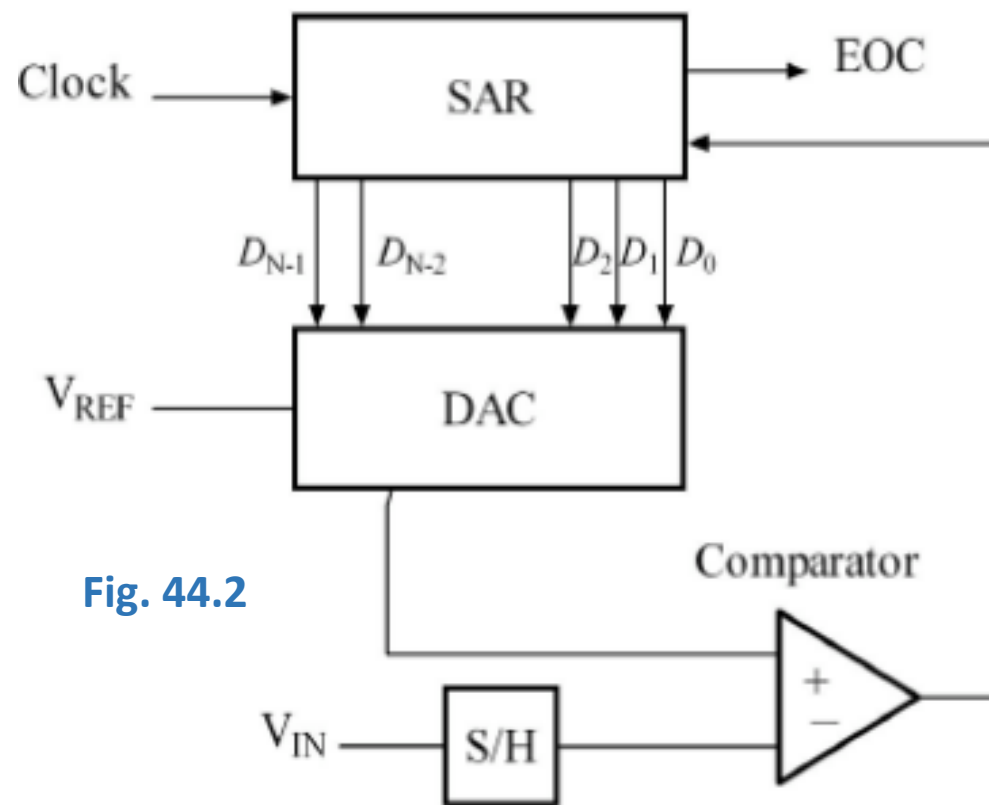


Fig. 44.2

Convertorul Analog-Digital (CAD) paralel. Structura:

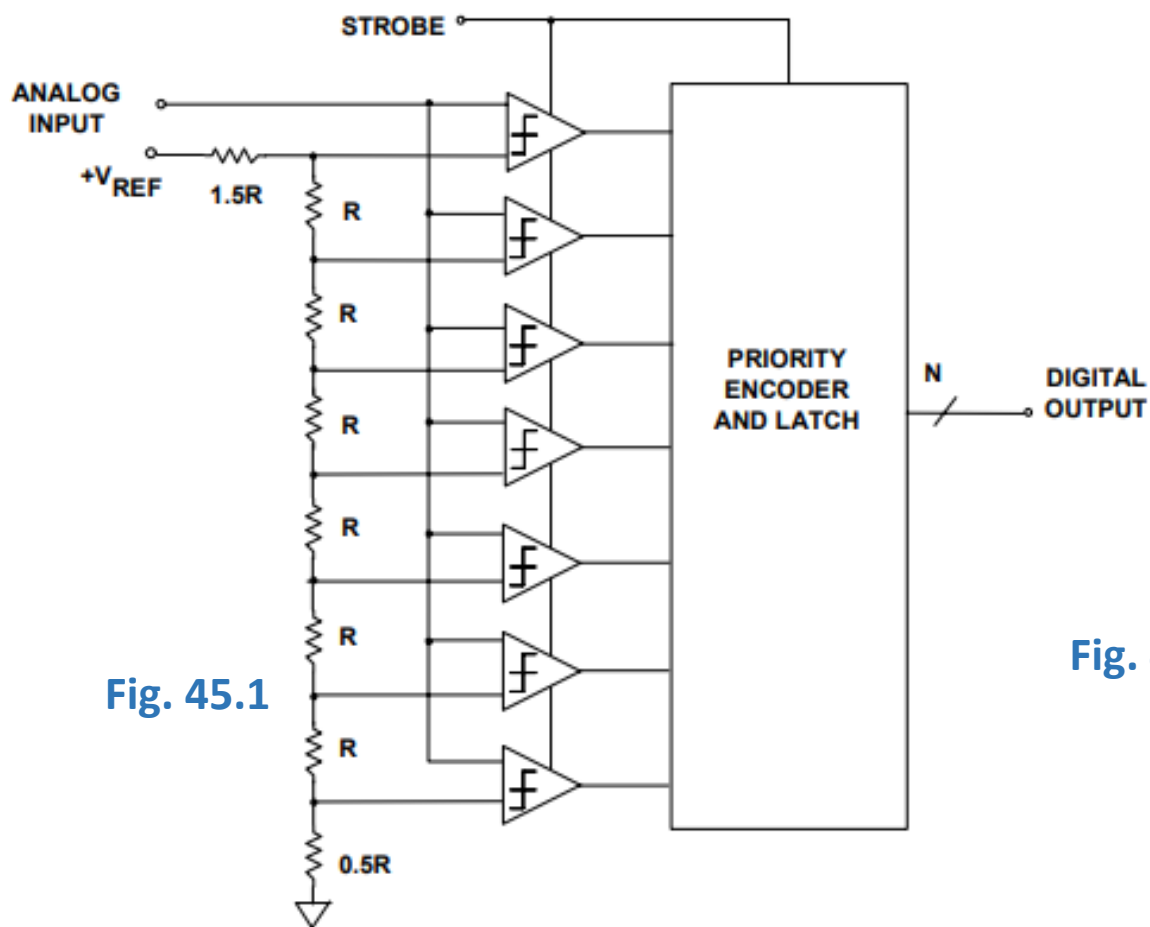


Fig. 45.1

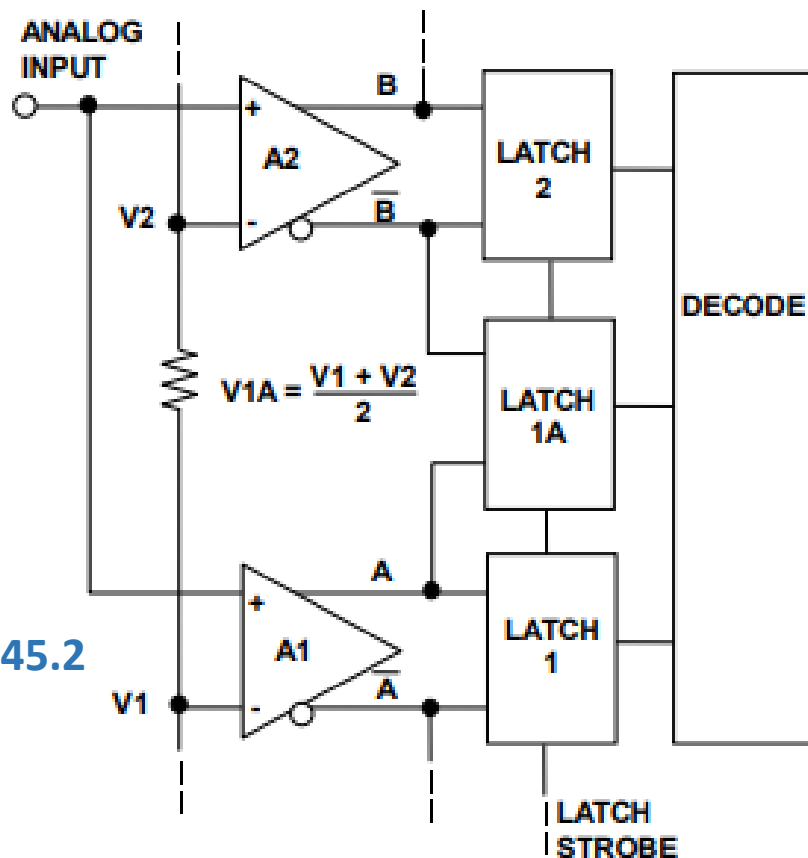


Fig. 45.2

AD9410: 10-Bits, 210MSPS

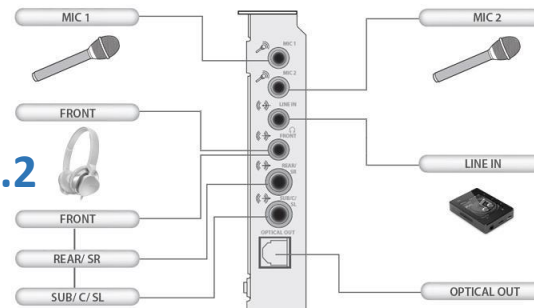
Placa de sunet este un dispozitiv hardware care facilitează intrarea și ieșirea semnalelor audio generate de un calculator. Acest dispozitiv poate fi integrată pe placa de bază sau este un component instalat pe Magistrala de Sistem (ISA, PCI). În raport cu funcționalitățile acesteia pot fi specificate următoarele caracteristici:

- Rata de cuantificare: 44 KHz;
- Gradul de cuantizare a sunetului: 8, 10, 12 biți;
- Intreruperea Hardware IRQ10 (INT72h);
- Memorie RAM pentru stocarea blocurilor de date;
- Canale acces direct la memorie: DMA0 sau DMA1 (cu cea mai înaltă prioritate).
- Numărul de canale de intrare (sunet stereo);
- Coeficientul de amplificare.

Fig. 46.1



Fig. 46.2



Dispozitive pentru achiziția sunetului. Amplificatoare de sunet:

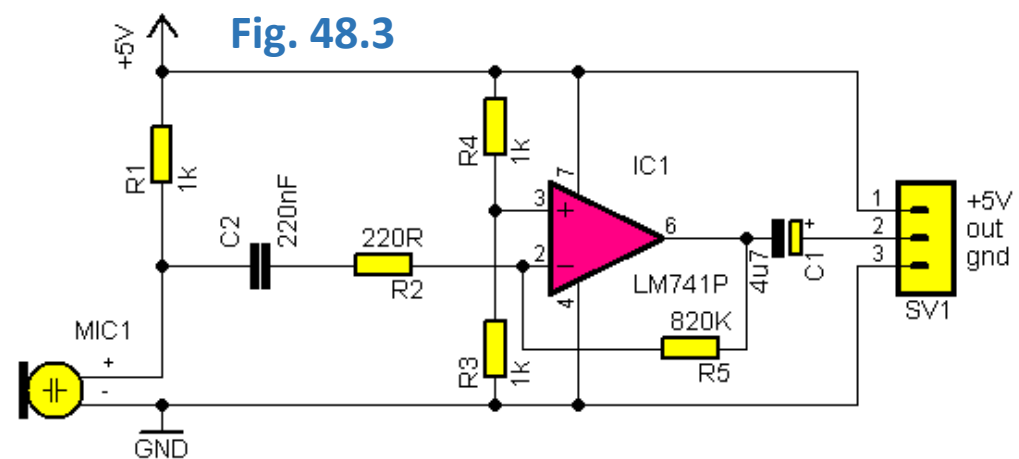
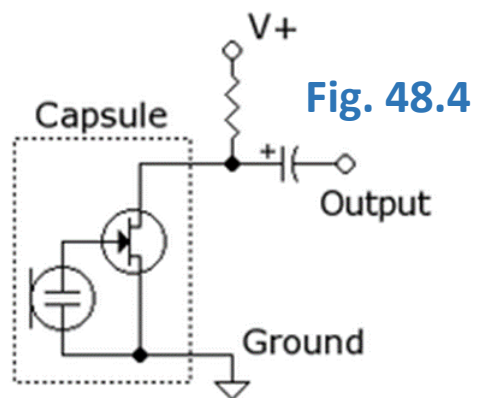
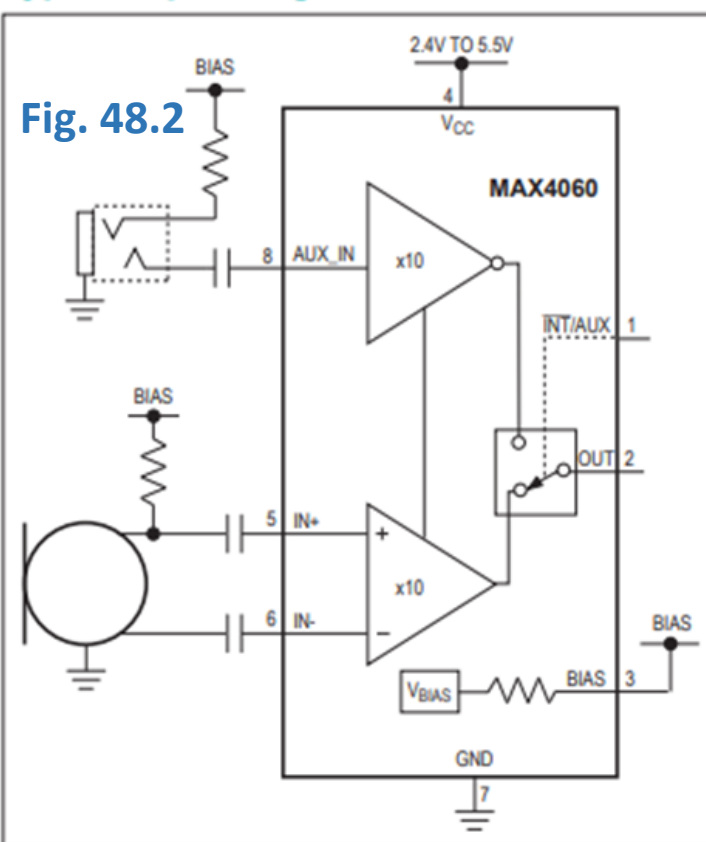
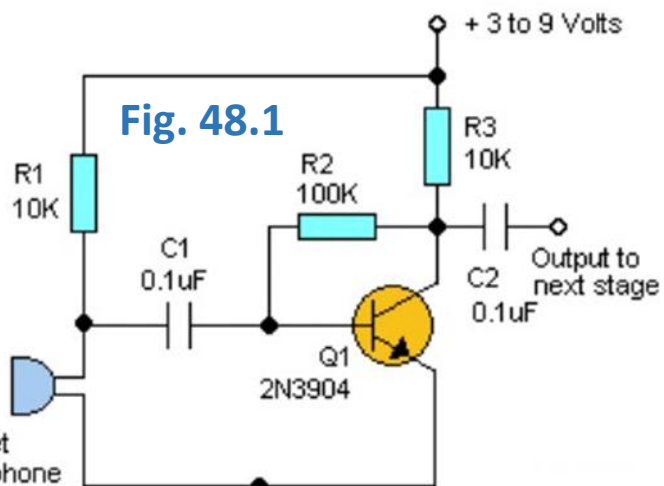
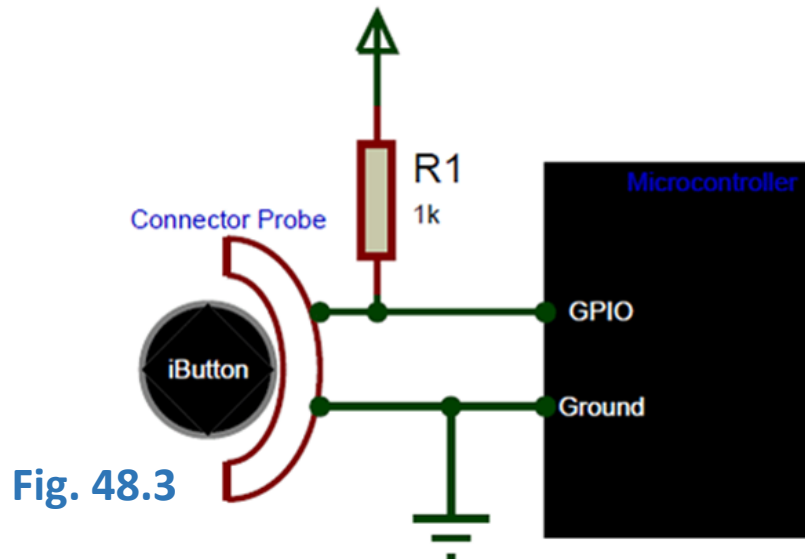
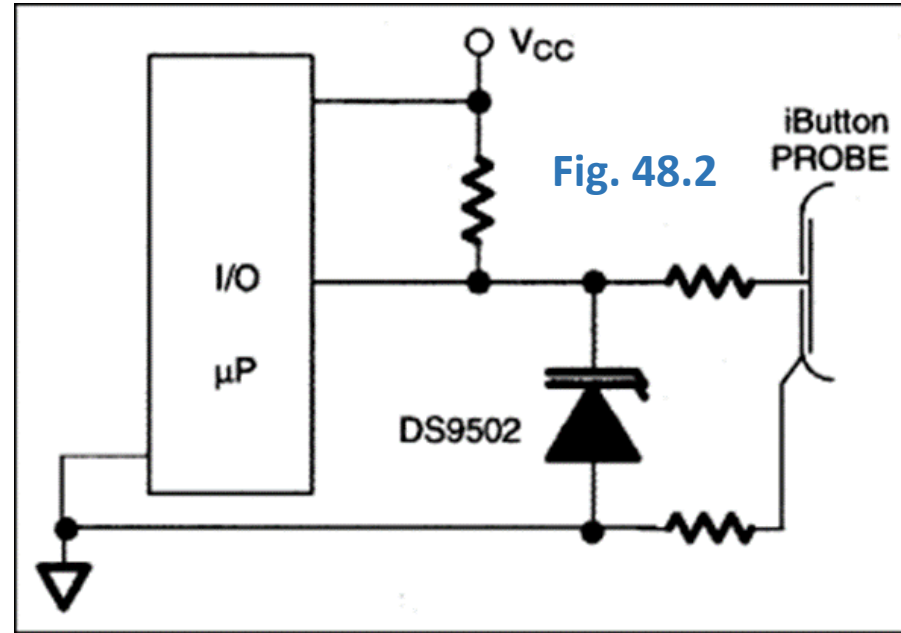


Fig. 48.5



Tema 3. Dispozitive pentru achiziția datelor:

1. Clasificarea dispozitivelor pentru achiziția datelor.
2. Tastatura. Programarea tastaturii. Funcții BIOS și DOS.
3. Mouse-ul. Programarea mouse-ului. Funcții DOS.
4. Camera video.
5. Scanner-ul.
6. Touch Screen-ul (Ecranul tactil)
7. Achiziția datelor discrete.
8. Achiziția semnalelor analogice.
9. Achiziția vorbirii.

Tematica disciplinei Arhitectura Calculatoarelor:

Tema 1. Introducere. Bazele fundamentale ale Arhitecturii Calculatoarelor;

Tema 2. Microprocesoare și Microcontrolere. Limbajul de programare Assembler;

Tema 3. Dispozitive pentru achiziția datelor;

Tema 4. Dispozitive pentru afișarea și imprimarea datelor;

Tema 5. Dispozitive pentru stocarea datelor.

Mulțumesc pentru atenție