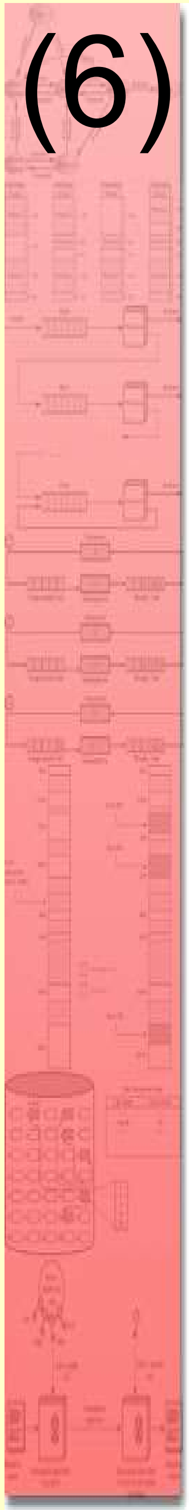


(6)

Escuela Politécnica Superior

UAM

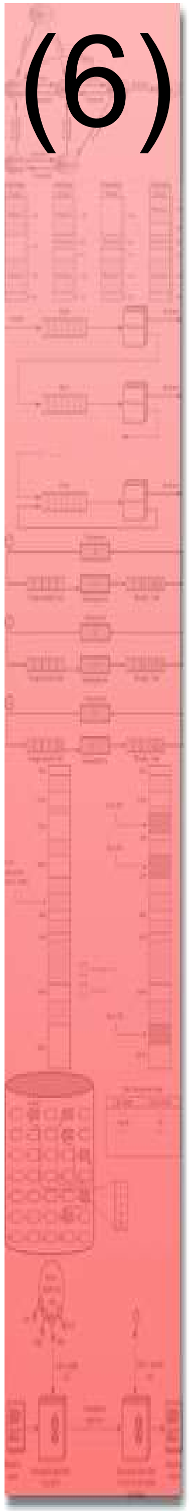
Universidad Autónoma
de Madrid

Unit 6
Programming the Basic Hardware Resources of the PC

MICROPROCESSOR-BASED SYSTEMS

Degree in Computer Science Engineering
Double Degree in Computer Engineering and Mathematics

EPS - UAM



(6)

Índice

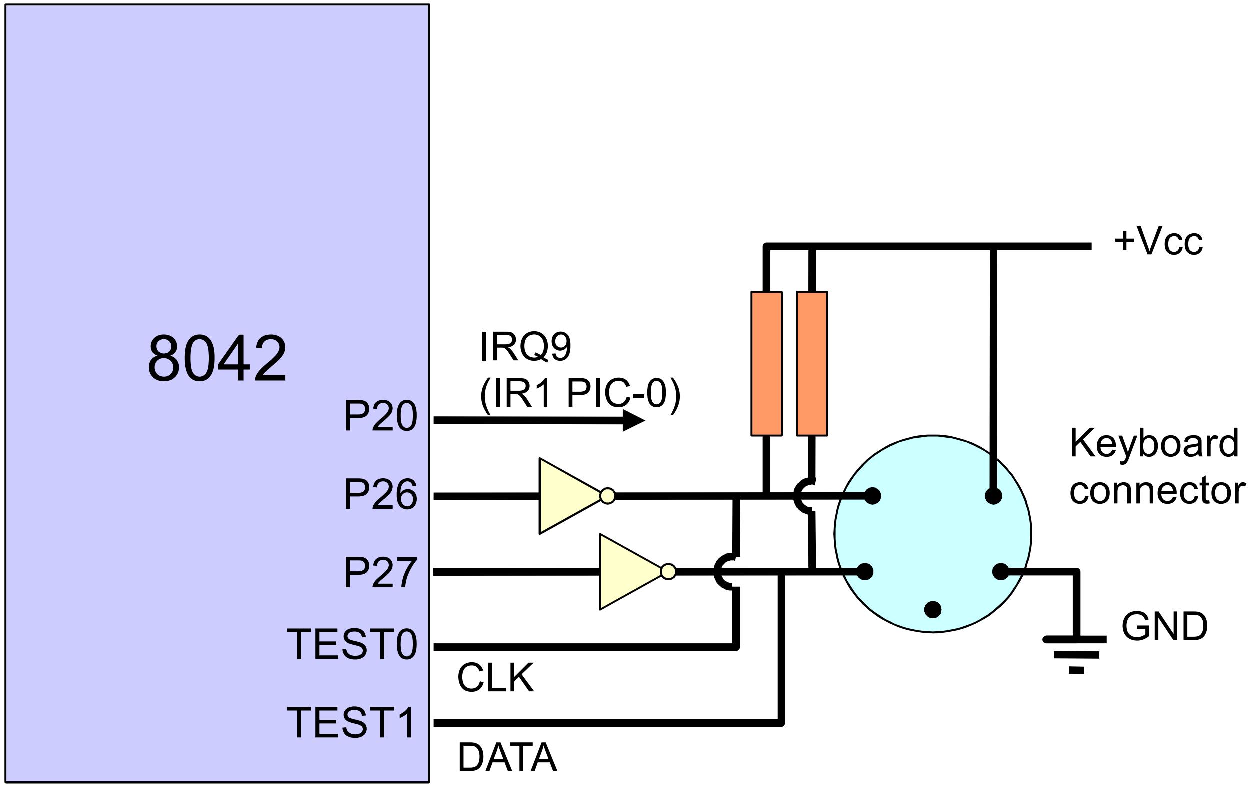
6. Programming the basic hardware resources of the PC.

6.1. Keyboard.
6.2. Timer.
6.3. Real time clock (RTC).
6.4. Video and screen controller.
6.5. Parallel port. Printer.
6.6. Asynchronous serial port (UART 8250).

(6)

6.1. Keyboard (I)

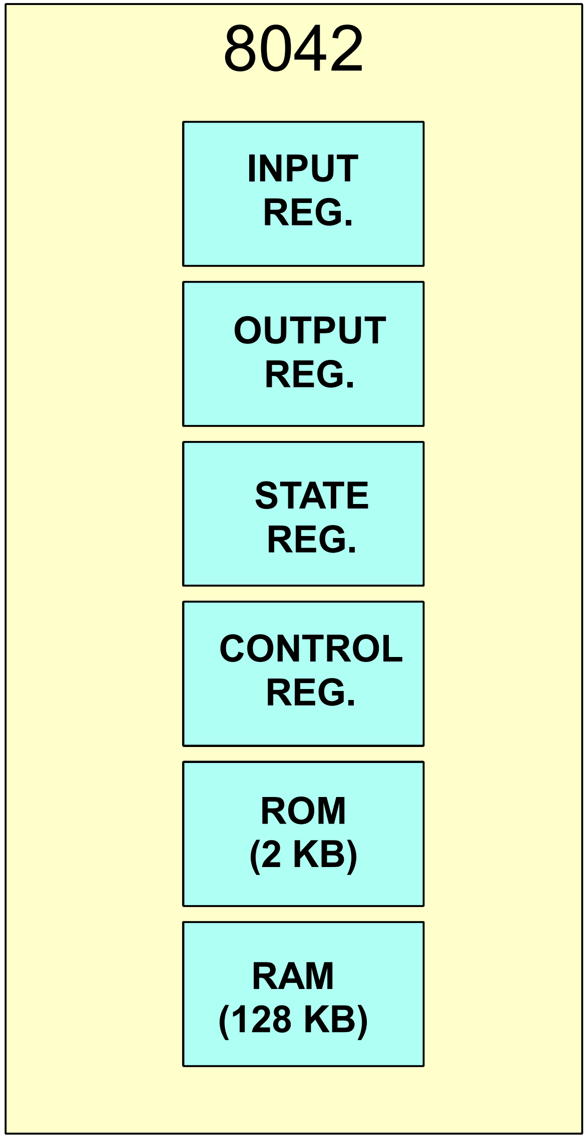
- Based on an Intel’s integrated circuit (8042)
  - The keyboard controller 8042 has 2 programmable I/O parallel ports (8 bits) and 2 programmable serial inputs (TEST0, TEST1).
  - Pins P26 and P27 used for sending CLK and DATA to keyboard.
  - Serial inputs TEST0 and TEST1 used for receiving CLK and DATA sent by the keyboard.



(6)

6.1. Keyboard (II)

- The 8042 is a microcontroller compatible with Intel's families MCS-48 (8048) and MCS-51 (8051).
- It behaves as a PPI (Programmable Peripheral Interface) that allows the implementation of interfaces tailored to different peripherals.
- It has 2 KB of ROM (8042) or EPROM (8742), 128 KB of RAM, 2 programmable 8-bit I/O ports, 2 programmable serial inputs, 8-bit internal counter, 12 MHz clock, and other control and special signals
- Addresses: 60h (INPUT and OUTPUT REG.), 64h (CONTROL and STATE REG.)



8042

INPUT REG.

OUTPUT REG.

STATE REG.

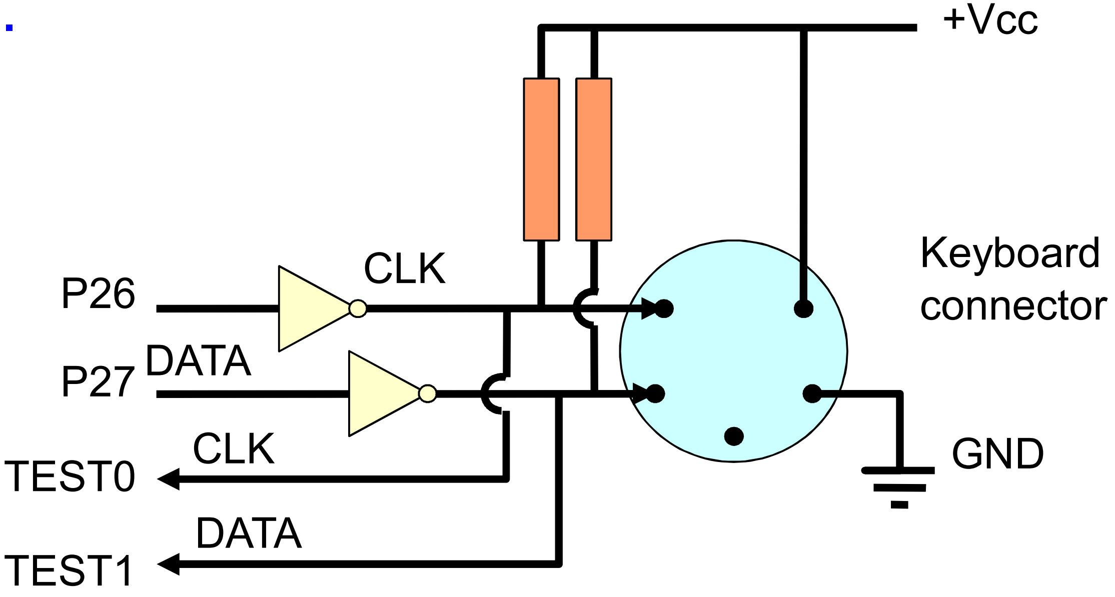
CONTROL REG.

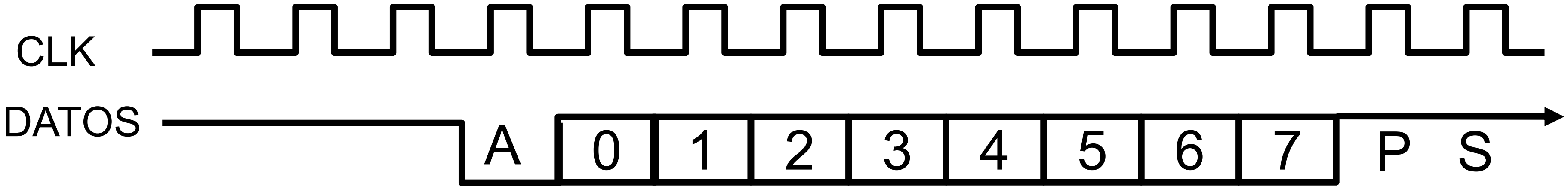
ROM (2 KB)

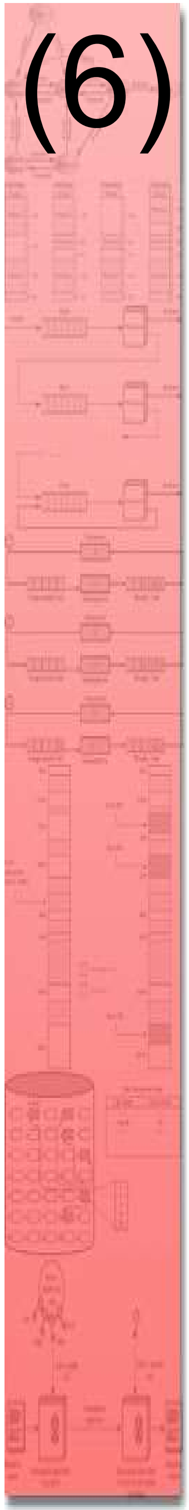
RAM (128 KB)

6.1. Keyboard (III)

- Not necessary a serial line controller (UART) since the clock signal (CLK) is transmitted independently of the data.
- No clock transmission problems due to the small length of the line.

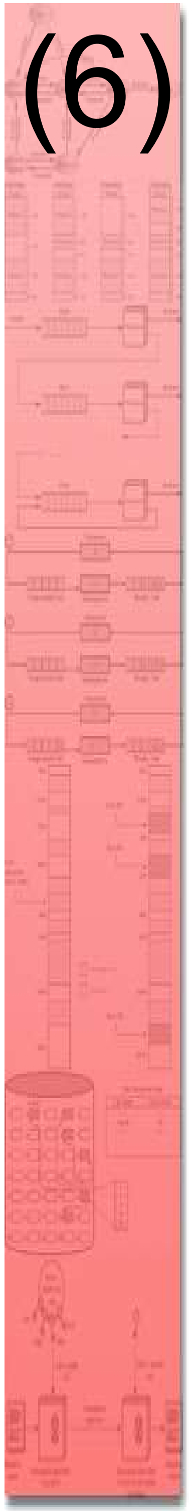






6.1. Keyboard (IV)

- PC keyboard has 83 keys organized into 3 groups:
  - Function keys.
  - Alphanumeric keyboard.
  - Numeric keyboard.
- Keyboard communicates with BIOS through INT 09h (PIC-0, IR1).
  - **INT 09h** generated every time a key is pressed/released.
  - Service routine obtains key code (SCAN CODE) by reading port 60h of the keyboard controller.
  - Code of key release is the same as code of key press but with most significant bit to 1.
  - Every time a key is pressed, the service routine stores two bytes into a BIOS memory buffer:
    - Corresponding ASCII code or 00h in case of special key (F1,…,F12, shift, arrows, etc.).
    - Identifier of pressed key (SCAN CODE).



6.1. Keyboard (V)

- Possible to press **combinations of keys**. Usually modifier keys (ALT, CAPITAL, CONTROL) along with other keys.

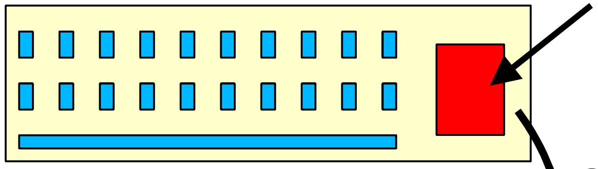
- Some combinations are not recognized by the BIOS routines and do not generate any code.

- Keys with special meaning:
  - **Alt-Ctrl-Del** (system’s boot).
  - **Ctrl-Pause** (call BIOS INT 1Bh).
  - **Pause** (halt program until key is pressed).
  - **Print Scrn** (call INT 5h for printing screen)

6.1. Keyboard (VI)

Keyboard

- When a key is pressed or released, KSCAN code (KEYBOARD SCAN CODE) is sent, which codifies physical position of key into keyboard.



83 keys

µC (8048, 8051)
It explores the keyboard matrix and sends KSCAN to the PC

TEST0,1

Keyboard controller (8042)

- ROM program translates KSCAN to SCAN CODE.
- Keys F1 to F12, arrows, shift, etc. do not generate ASCII code.

µC (8042)

SCAN CODE

PIC-0
(INT 09h)

Keyboard buffer (DOS)

BIOS (Int 09h)

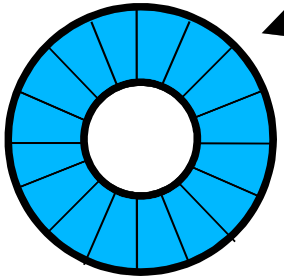
ASCII USA

SCAN CODE

BIOS
(INT 09h)

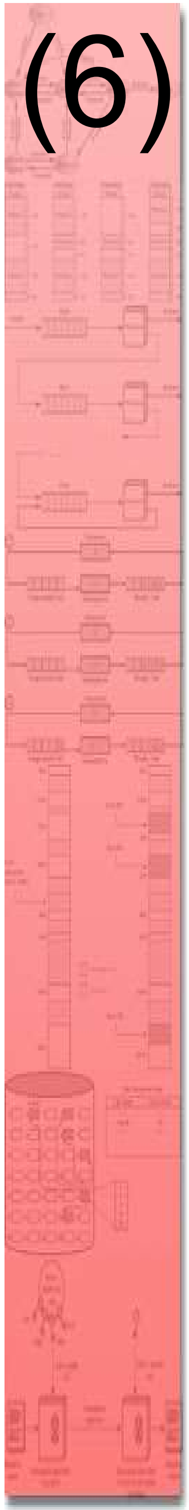
Word format

ASCII-COUNTRY	SCAN CODE



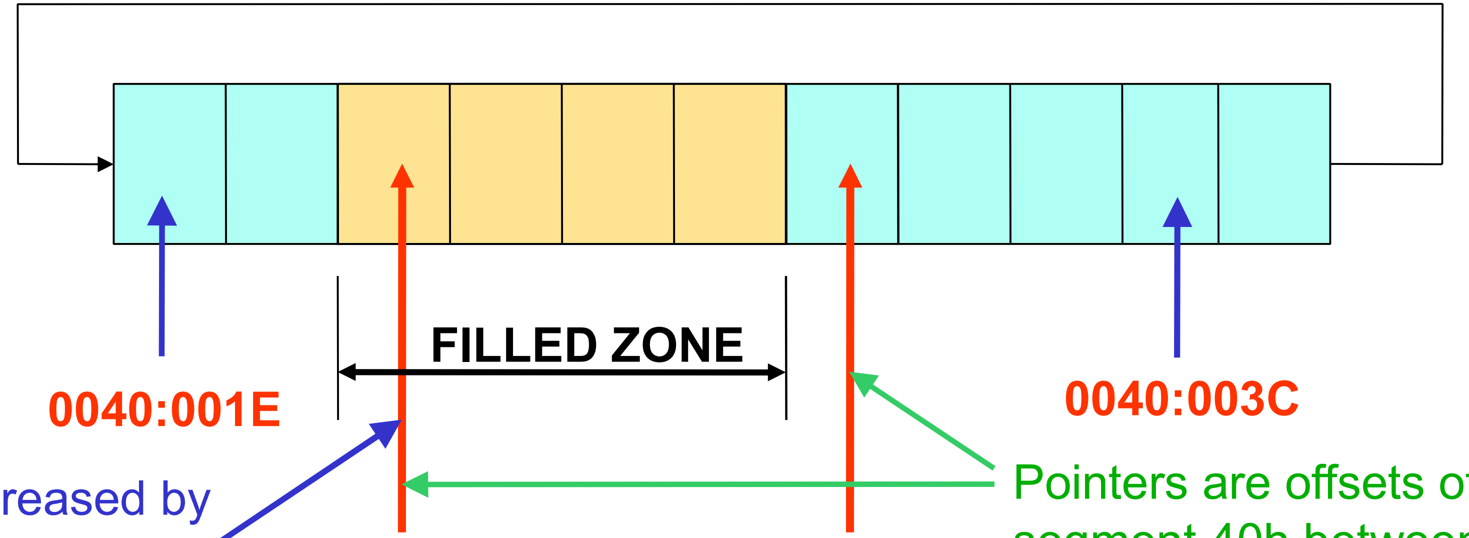
16-word buffer

KEYB.COM


(6)

6.1. Keyboard (VII)

Cyclic buffer


0040:001E ... 0040:003D IN AREA OF BIOS VARIABLES

FILLED ZONE

0040:001E

0040:003C

Pointer increased by 2 after reading

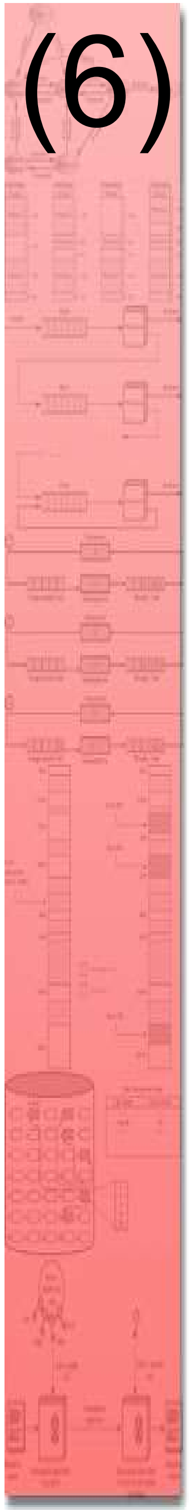
READ POINTER
(Position of next character to be read)

WRITE POINTER
(Position of next character to be written)

Pointers are offsets of segment 40h between 1E and 3C

A position (word) is left free. Possible to store up to 15 characters (words) into the 16-word buffer.

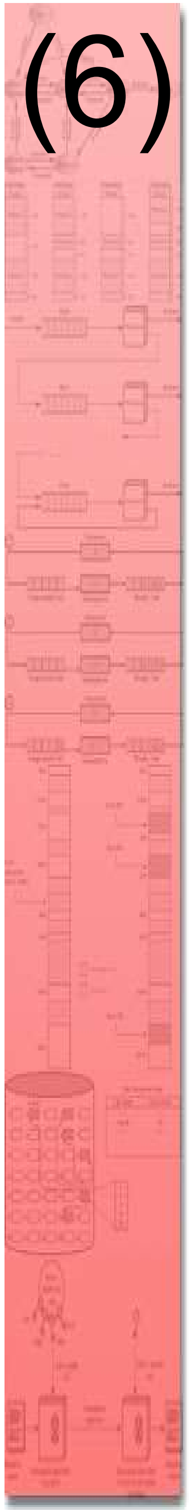
ASCII COUNTRY (1st byte)	SCAN CODE (2nd byte)



6.1. Keyboard (VIII)

INT 16h (BIOS)

AH	Function
00h and 10h	**Read the codes associated with the key or combination of keys from the keyboard buffer and advance the buffer pointer to the next character. Wait for key press if buffer is empty.**
	Output: **AH** = Key identifier (SCAN CODE) **AL** = ASCII code of character
01h and 11h	**Return state of the keyboard buffer**
	Output: **ZF** = 1 if empty buffer **ZF** = 0 if non-empty buffer **AH** = Key identifier **AL** = ASCII code of character
02h and 12h	**Press state of different keys**
	Output: Return in **AL** the 1-byte press state of several keys: 7: Ins, 6: Caps Lock, 5: Num Lock, 4: Scroll Lock, 3: Alt, 2: Ctrl, 1: Left Capital, 0: Right Capital

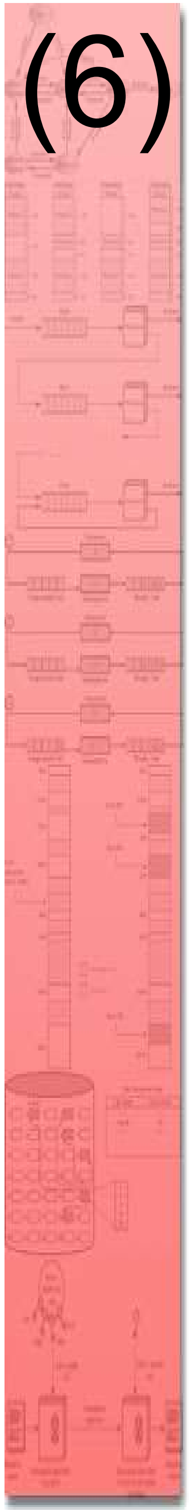


(6)

6.1. Keyboard (IX)

INT 16h (BIOS)

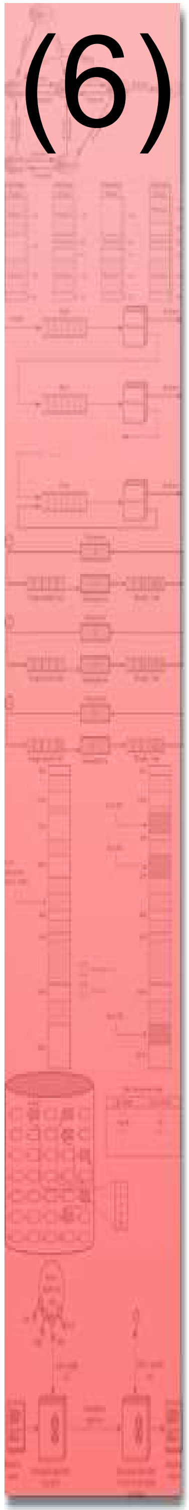
AH	Function
12h	**Press state of different keys.**
	Output: Return in **AH** the 1-byte press state of several keys 7: Print Scrn, 6: Caps Lock, 5: Num Lock, 4: Scroll Lock, 3: Right Alt, 2: Right Ctrl, 1: Left Alt, 0: Left Ctrl.
05h	**Write the given character into the keyboard buffer.**
	Input: **CH** = Key identifier (SCAN CODE) **CL** = ASCII code of character. Output: **AL** = 1 if buffer is full.



6.1. Keyboard (X)

INT 21h (DOS)

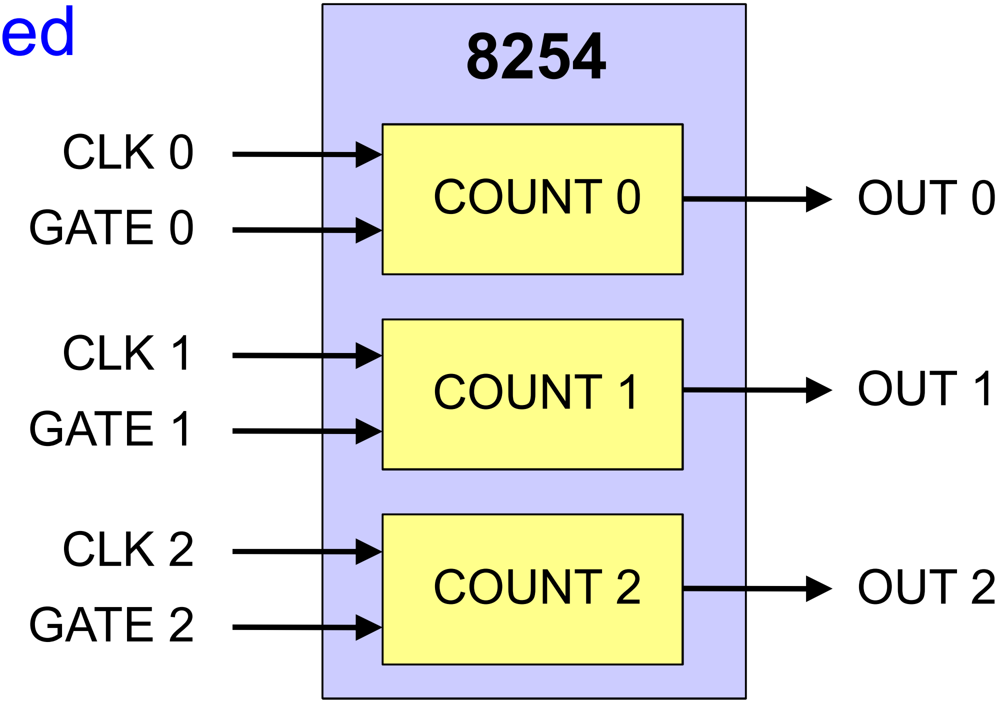
AH	Function
01h	**Wait for a character from keyboard and print it on screen**
	Output: **AL** = character
06h	**Read a character from keyboard without printing it (DL = FFh)**
	Output: **AL** = character (if available), **AL** = 0 if no character
07h	**Wait for a character from keyboard without printing it on screen**
	Output: **AL** = character (if available)
0Ah	**Read characters from keyboard without printing them on screen**
	Input: **DS:DX** = address of memory buffer First byte in buffer must indicate the maximum number of characters to be read (including carriage return) Output: Second byte in buffer indicates number of read characters excluding the carriage return. ASCII codes of pressed characters stored starting from the third byte.
0Bh	**Read state of keyboard**
	Output: **AL** = FFh if character is available, **AL** = 0 if no character.

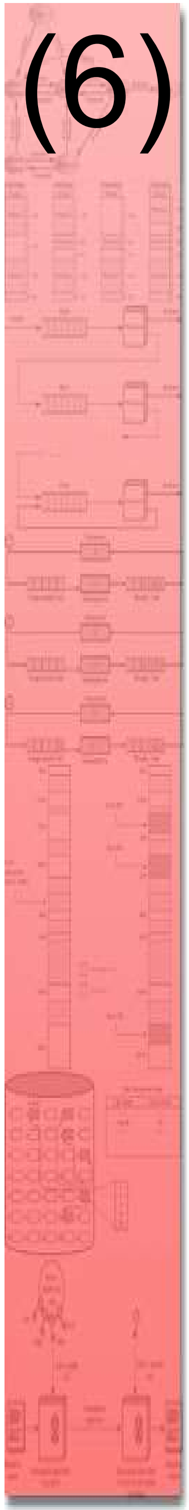


(6)

6.2. Timer (I)

- Intel's programmable interval timer (8254).
- It contains 3 independent 16-bit counters.
- Every counter decrements by one at every clock's falling edge (CLK input) if its enable signal is activated (GATE input).
- The output of every counter (OUT output) is activated according to the mode in which the counter has previously been programmed.
- 6 operation modes.

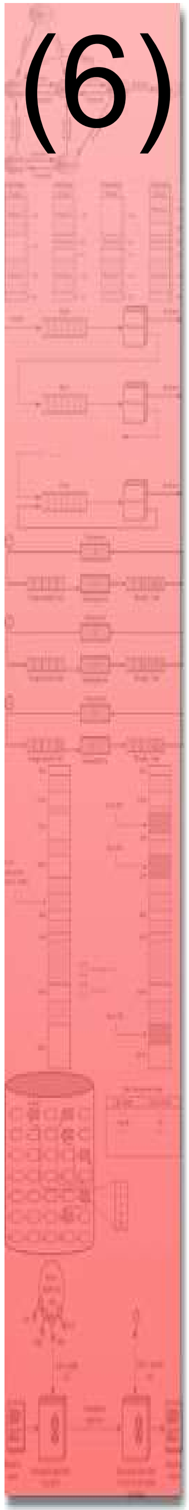




(6)

6.2. Timer (II)

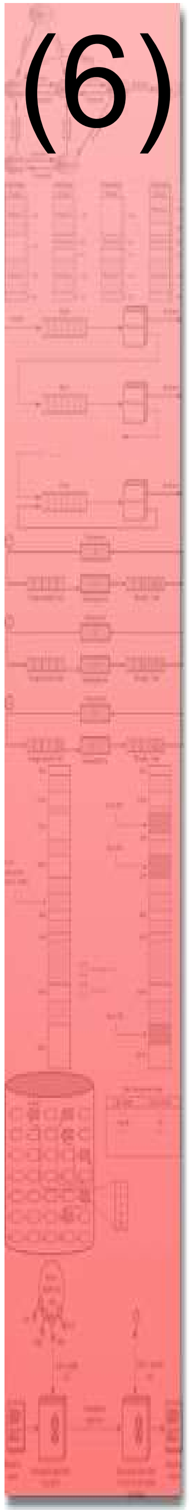
- The timer has 4 I/O ports.
- Port 43h (write only)
  - Timer's control register.
  - Define initial configuration of a timer and send memorization command of the current count value.
- Port 40h (COUNT 0)
  - (out) Modify initial count value of COUNT 0.
    - Counter decrements from the initial value.
    - When zero is reached, counter can be reloaded with the same initial value depending on the operation mode.
  - (in) Return stored value of COUNT 0.
- Ports 41h (COUNT 1) and 42h (COUNT 2)
  - Same behavior as 40h for the other counters.



6.2. Timer (III)

- Control word
  - The next value is written to port 43h:

SC1	SC0	RW1	RW0	M2	M1	M0	BCD



6.2. Timer (IV)

Control word

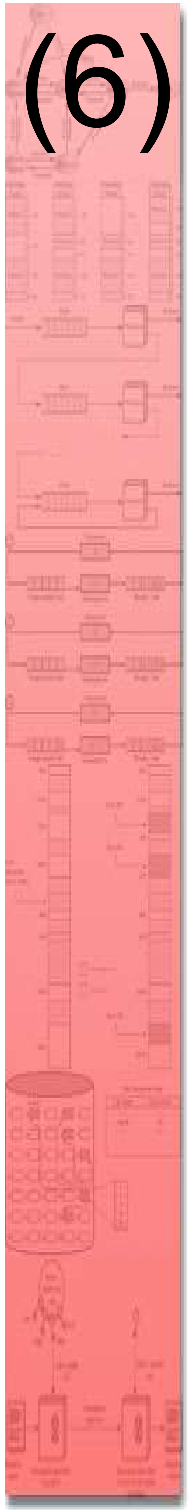
- The next value is written to port 43h:

SC1	SC0	RW1	RW0	M2	M1	M0	BCD

- Counter selection

SC1	SC0	Action
0	0	Select COUNT 0
0	1	Select COUNT 1
1	0	Select COUNT 2

- The rest of the control word configurations only applied to selected counter.



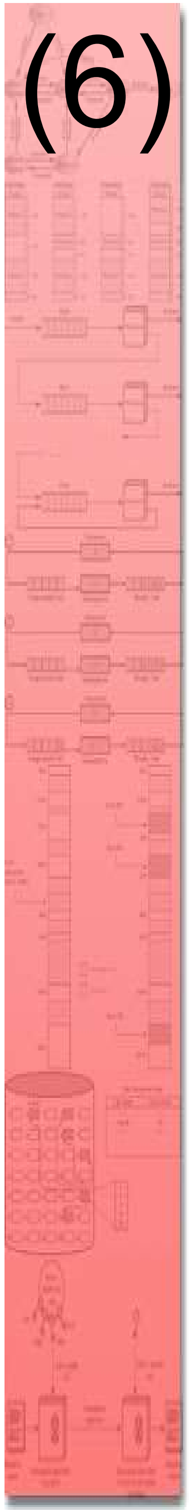
6.2. Timer (V)

- Control word
  - The next value is written to port 43h:

SC1	SC0	RW1	RW0	M2	M1	M0	BCD

  - Memorization command of the counter’s current value + Configuration of counter’s read/write

RW1	RW0	Action
0	0	Memorization command of current value (fields M and BCD ignored).
0	1	Read/Write to counter’s port only affects low byte.
1	0	Read/Write to counter’s port only affects high byte.
1	1	First Read/Write to counter’s port affects low byte and second to high byte.



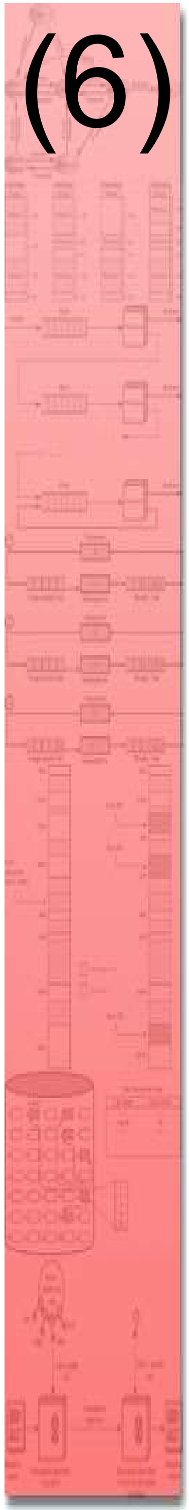
6.2. Timer (VI)

- Control word
  - The next value is written to port 43h:

SC1	SC0	RW1	RW0	M2	M1	M0	BCD

  - Configuration of operation mode

M2	M1	M0	Action
0	0	0	Mode 0: End-of-count interrupt.
0	0	1	Mode 1: Programmable monostable.
X	1	0	Mode 2: Frequency generator.
X	1	1	**Mode 3: Square wave generator.**
1	0	0	Mode 4: Strobe pulse started by software.
1	0	1	Mode 5: Strobe pulse started by hardware.



6.2. Timer (VII)

- Control word
  - The next value is written to port 43h:

SC1	SC0	RW1	RW0	M2	M1	M0	BCD

  - Configuration of counter's coding

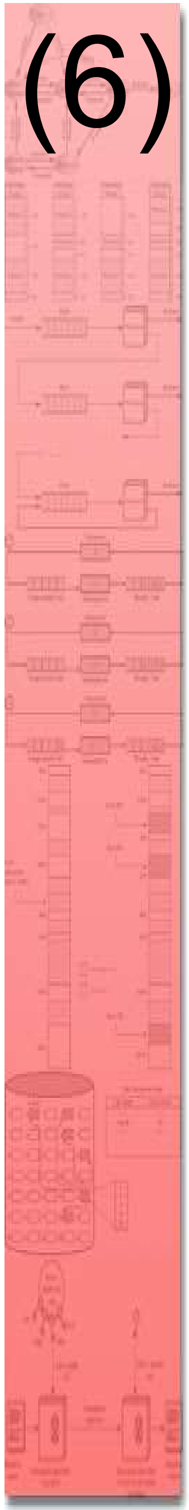
BCD	Action
0	16-bit binary counter.
1	4-digit BCD counter.

  - Binary-Coded Decimal (BCD): Every four bits codify a digit between 0 (0000b) and 9 (1001b).

6.2. Timer (VIII)

- Initial configuration: Control word sent to port 43h indicating counter and initial configuration (RW ≠ 00b).
- Initial count value of a counter:
  - Sent to the counter's port (40h, 41h or 42h).
  - Send low byte, high byte or low byte followed by high byte according to field RW of the control word.
- Example: Configure counter 2 (SC = 10b) into mode 3 (M = 011b), with binary coding (BCD = 0), and initialize it to 1234h (RW = 11b).

  -
    ```
    mov al, 10110110b      ; Control word: SC | RW | M | BCD
    out 43h, al            ; Send control word
    mov al, 34h
    out 42h, al            ; Send low byte of initial count value.
    mov al, 12h
    out 42h, al            ; Send high byte of initial count value.
    ```



6.2. Timer (IX)

- Read the current value of a counter:

  (1) Send memorization command of the current count value (RW=00b) to port 43h, specifying the counter.

  (2) Read stored count value from the port of the specified counter (40h, 41h or 42h):

  Read low byte, high byte or low byte followed by high byte according to field RW of initial configuration.

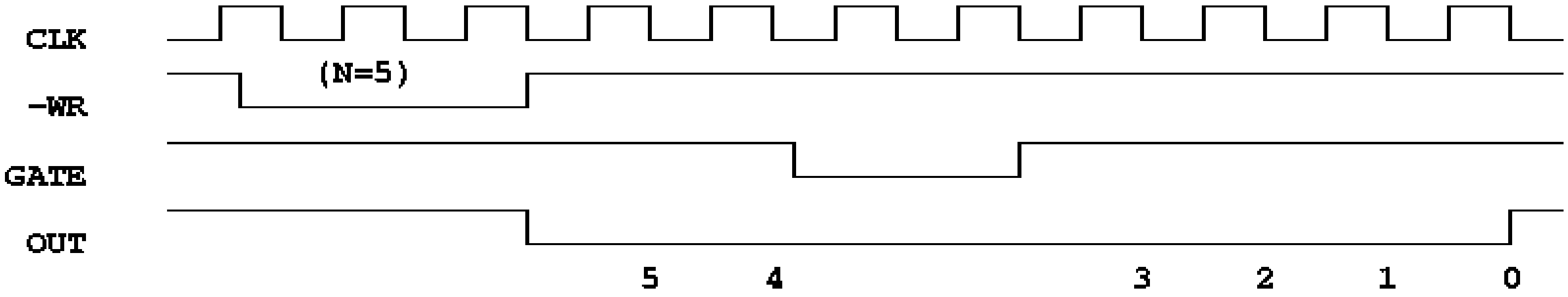
- Example: Read current count value of counter 2 (SC=10b), which was initialized to RW=11b.

```
mov al, 10000000b        ; SC | RW | M | BCD
out 43h, al              ; Store current count value
in al, 42h               ; Read low byte of current count value.
mov bl, al
in al, 42h               ; Read high byte of current count value.
mov bh, al               ; BX with stored count value.
```

The stored value may not coincide with the current count value: the counter keeps decrementing after having been memorized.

6.2. Timer (X)

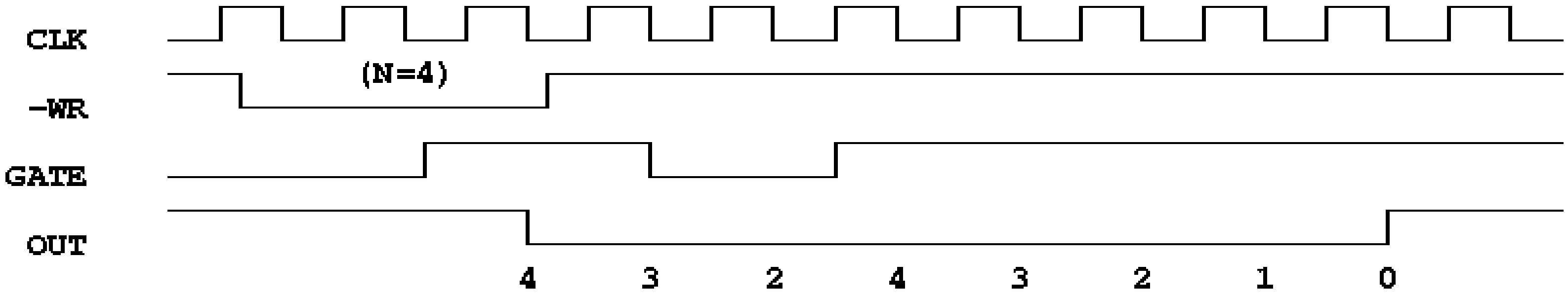
- Mode 0: End-of-count interrupt.
  - OUT goes from 0 to 1 when counter reaches 0.
  - Counter decrements while GATE=1.
  - Counter restarts count when a new initial count value or a new control word are sent.



(6)

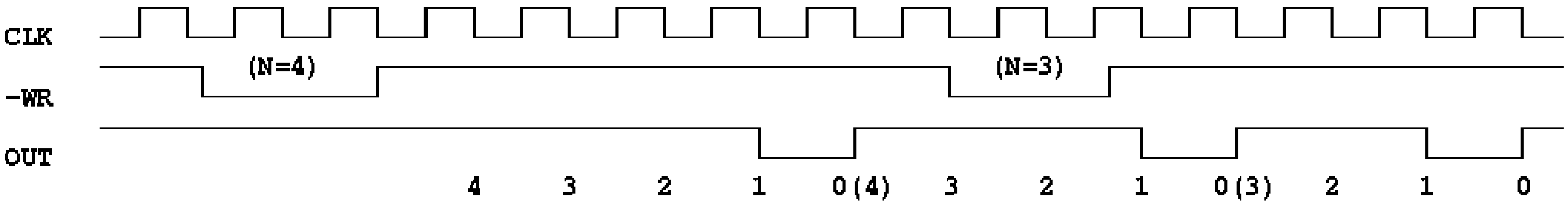
6.2. Timer (XI)

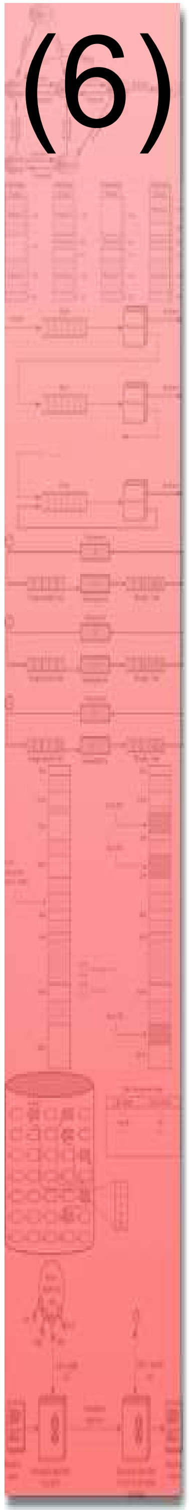
- Mode 1: Programmable monostable.
  - OUT goes from 0 to 1 when counter reaches 0.
  - Counter restarts count with initial value when GATE goes from 0 to 1.



6.2. Timer (XII)

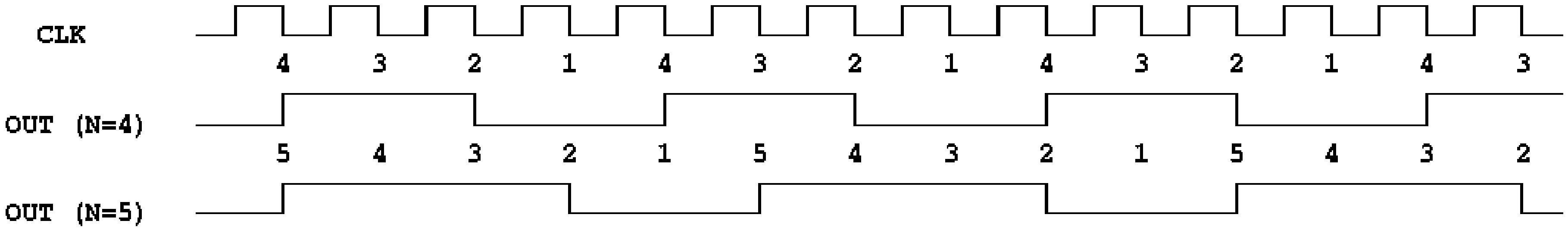
- Mode 2: Frequency generator.
  - OUT goes from 1 to 0 when counter reaches 1 and from 0 to 1 when it reaches 0.
  - Counter decrements while GATE=1.
  - Counter restarts count to initial value when GATE goes from 0 to 1 and when counter reaches 0.





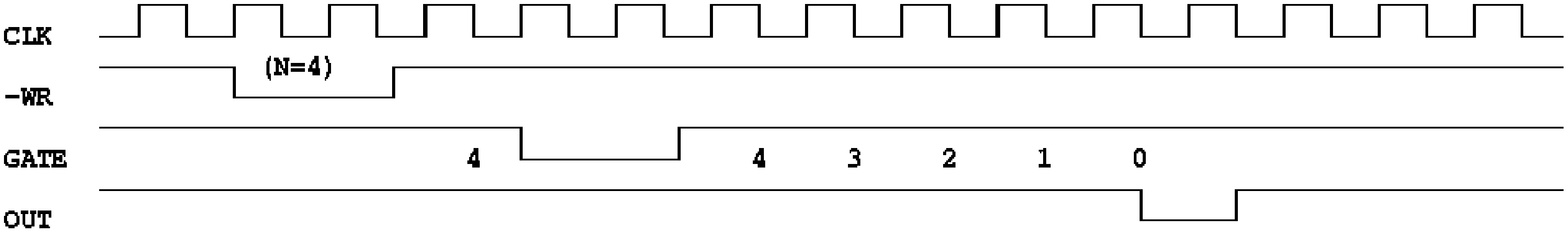
6.2. Timer (XIII)

- Mode 3: Square wave generator.
  - Usual mode.
  - OUT is 1 during the first half of the count and 0 during the second half.
  - Counter decrements while GATE=1.
  - Counter restarts count to initial value when GATE goes from 0 to 1 and when counter reaches 0.



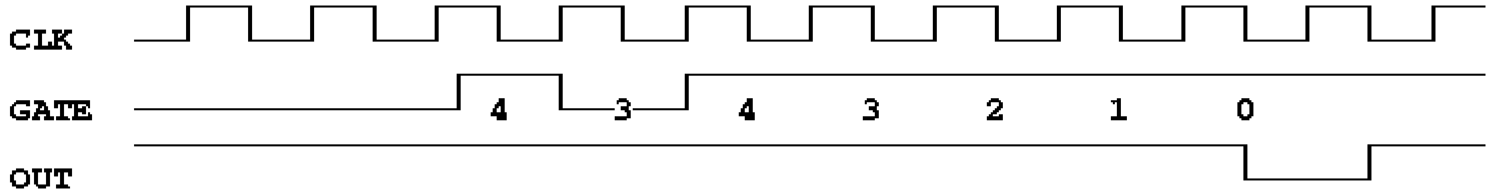
6.2. Timer (XIV)

- Mode 4: Strobe pulsed started by software.
  - OUT is 0 during a clock cycle when counter reaches 0.
  - Counter decrements while GATE=1.
  - Counter restarts count with initial value when GATE goes from 0 to 1.
  - Counter restarts count when a new initial count value or a new control word are sent (software synchronization)



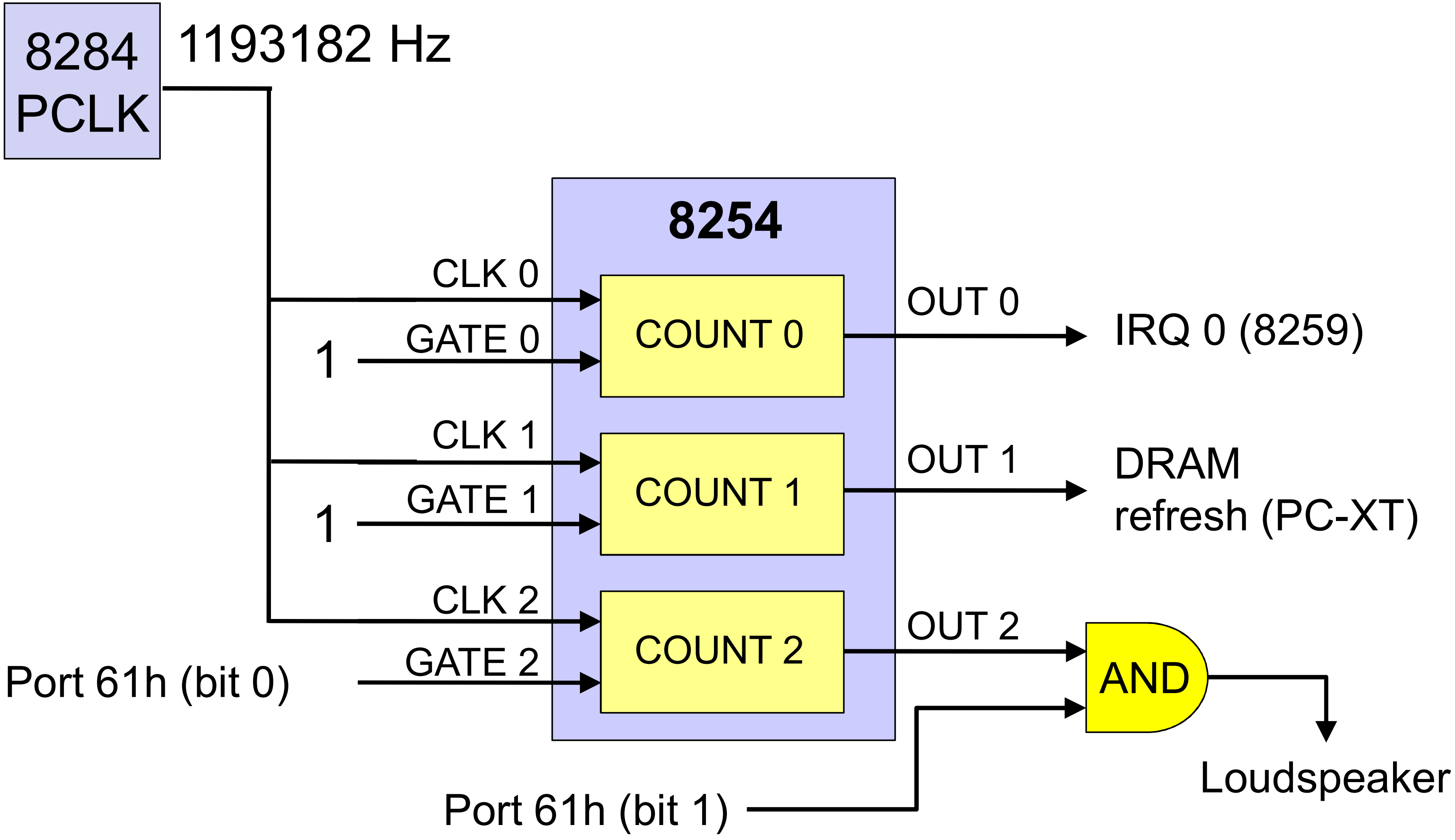
6.2. Timer (XV)

- Mode 5: Strobe pulse started by hardware.
  - OUT is 0 during a clock cycle when counter reaches 0.
  - Counter restarts initial count value when GATE goes from 0 to 1.
  - Counter restarts count after reaching 0 when a new initial count value or a new control word are sent.



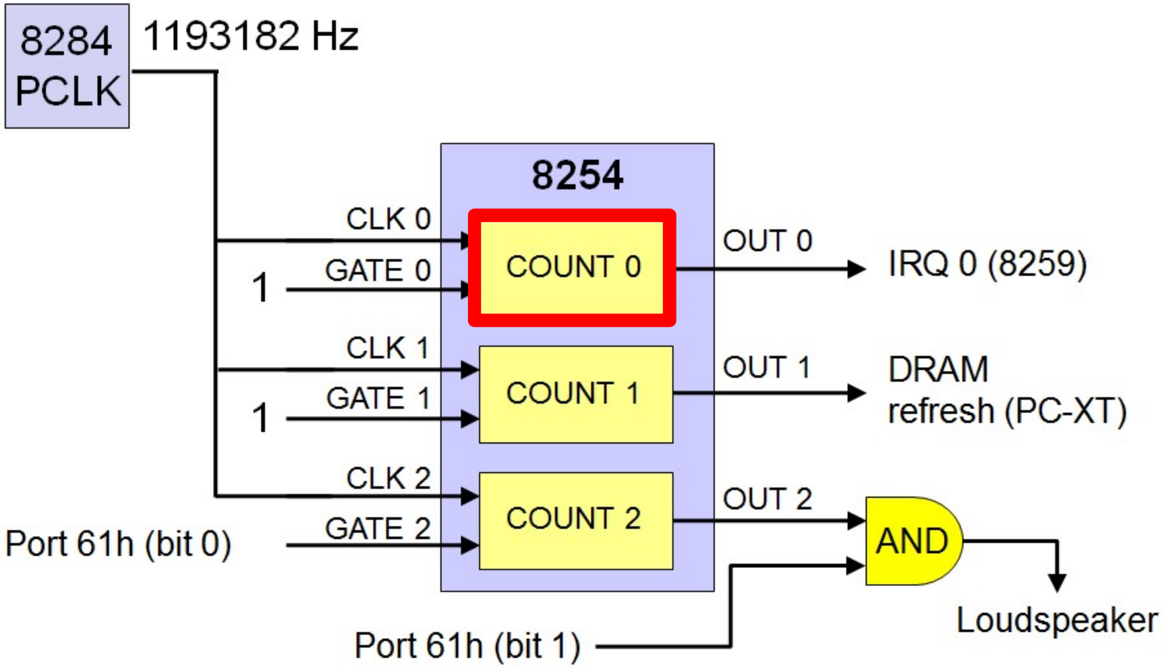
6.2. Timer (XVI)

- Timer connection on the PC



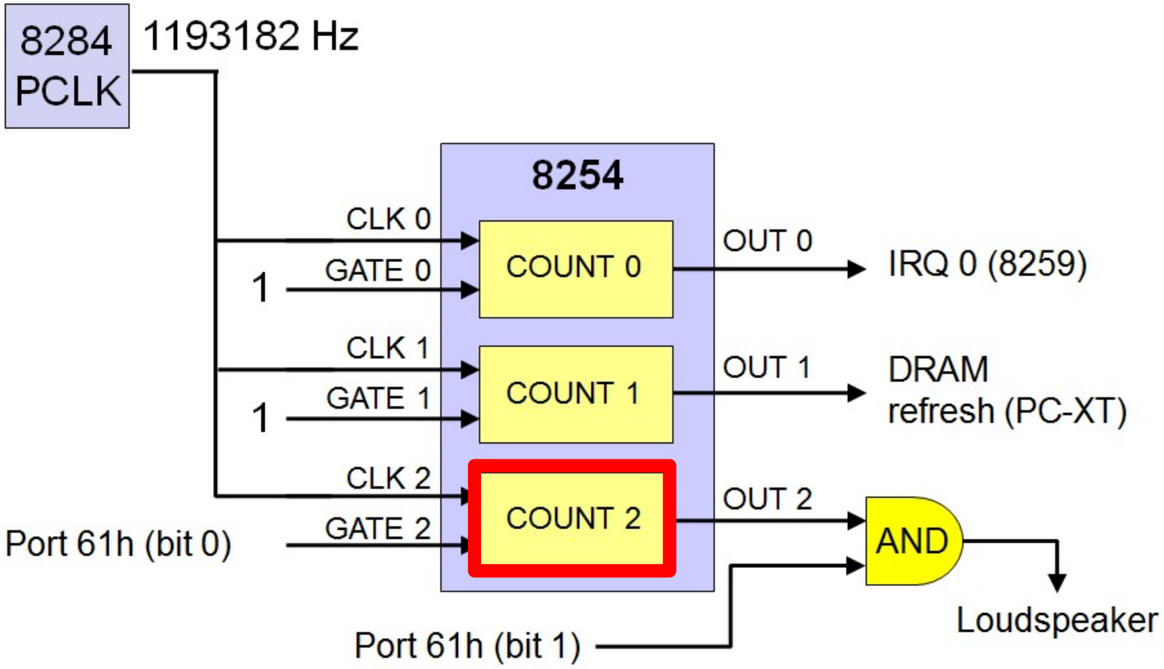
6.2. Timer (XVII)

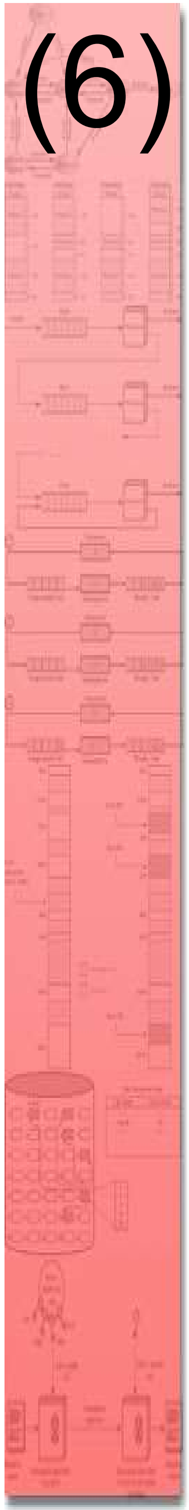
- Timer connection on the PC
  - COUNT 0 generates maskable interrupt INT 8.
  - Configured in mode 3 (square wave)
  - Initial count value = 0 (≡ 65536)
  - Wave frequency = 1193182 / 65536 = 18.2 (Hz)



6.2. Timer (XVIII)

- Timer connection on the PC
  - COUNT 2 free for user programs.
  - Output OUT connected to PC’s loudspeaker if bit 1 of port 61h is 1.
  - Input GATE connected to bit 0 of port 61h.
  - Usual configuration in mode 3 (square wave)
  - Initial value = 1193182 / Desired frequency (Hz)

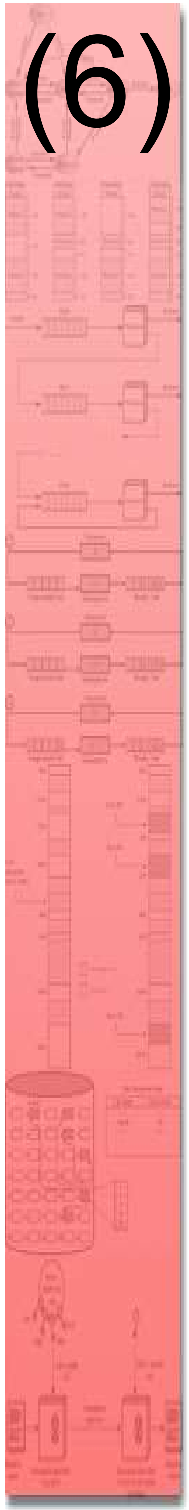




(6)

6.3. Real Time Clock (RTC) (I)

- IBM mounted the battery-fed RTC Real Time Clock (MC146818 de Motorola) on the PC-AT.
- Low-consumption CMOS technology (ideal for batteries).
- It includes a 64-byte RAM memory for system configuration:
  - 6 bytes for time (Seconds, Minutes, Hours) of clock and alarm.
  - 4 bytes for date (Day of week, Day of month, Month, Year).
  - 4 bytes for control registers (A, B, C, D)
  - 50 bytes free for user programs.
- It can generate periodic interrupts, alarms and hardware signals.



(6)

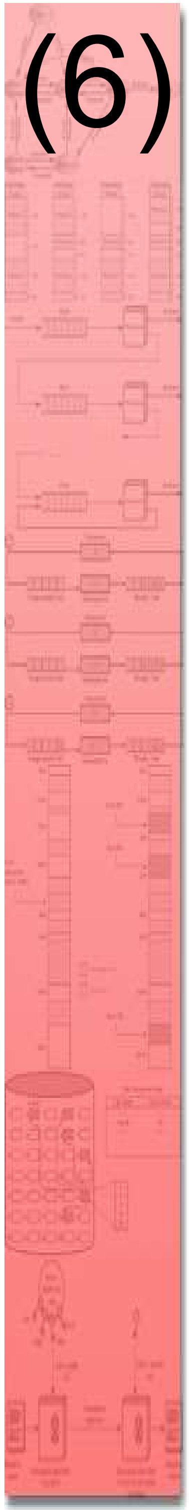
6.3. Real Time Clock (RTC) (II)

- RTC reading:
  - Write to port 70h (OUT) the address of the position to be read.
  - Read port 71h (IN).
- RTC writing:
  - Write to port 70h (OUT) the address of the position to be written.
  - Write the value to port 71h (OUT).

6.3. Real Time Clock (RTC) (III)

- Register A (send value 0Ah to port 70h)
  - Read or write to port 71h the value given by:

UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0



6.3. Real Time Clock (RTC) (IV)

- Register A (send value 0Ah to port 70h)
  - Read or write to port 71h the value given by:

UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0

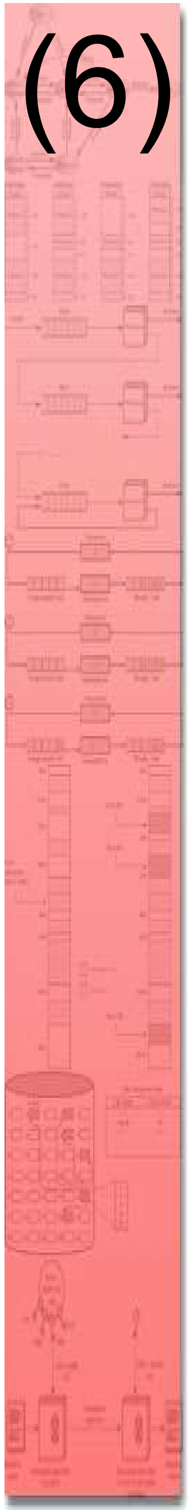
  - **Update_In_Progress** (read only): When 0, it is safe to read/write the clock’s ports without interfering with internal updates.

6.3. Real Time Clock (RTC) (V)

- Register A (send value 0Ah to port 70h)
  - Read or write to port 71h the value given by:

UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0

DV2...DV0	Oscillator frequency
000	4.193404 MHz
001	1.048576 Mhz
010	32.768 kHz



(6)

6.3. Real Time Clock (RTC) (VI)

- Register A (send value 0Ah to port 70h)
  - Read or write to port 71h the value given by:

UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0

  - Computation of RS from the desired frequency of the clock’s periodic interrupts:

$$\text{RS} = 1 + \log_2 \frac{32768\ (\text{Hz})}{\text{Frequency}\ (\text{Hz})}$$

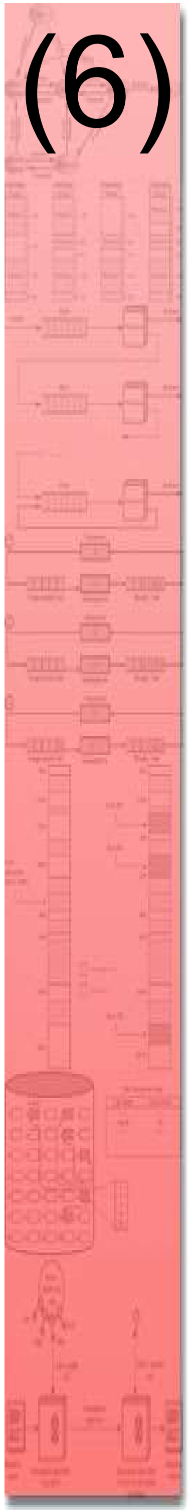
  - Example for a frequency of 512 Hz
    - RS = 7 = 0111b

6.3. Real Time Clock (RTC) (VII)

- Register B (send value 0Bh to port 70h)
  - Read or write to port 71h the value given by:

SET	PIE	AIE	UIE	SQWE	DM	12/24	DSE

- DSE: (Daylight Saving Enable) Activated with 1.
- 12/24: Activated to 24 hours with 1.
- DM: (Date Mode) 0 is BCD, 1 is Decimal.
- SQWE: (Square Wave Enable) Enable clock's output wave.
- UIE: Enable interrupt after change of time or date (clock update).
- AIE: Enable interrupt at the alarm's expiration time.
- PIE: Enable periodic interrupts.
- SET: The clock is active when SET is 0. When SET is 1, the clock is halted, being safe to change the 14 configuration bytes.



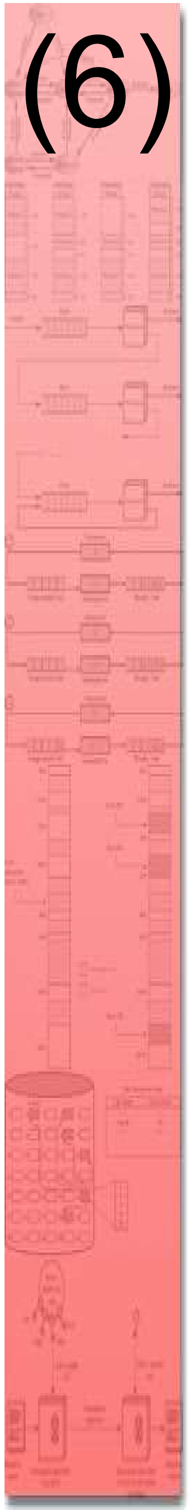
(6)

6.3. Real Time Clock (RTC) (VIII)

- Register C (send value 0Ch to port 70h)
  - Read (read only) from port 71h the value given by:

IRQF	PF	AF	UF	----	----	----	----

  - If they are 1, they indicate the type of event that has raised the interrupt.
    - IRQF: Interrupt request
    - PF: Periodic interrupt
    - AF: Alarm
    - UF: Date/Time update



(6)

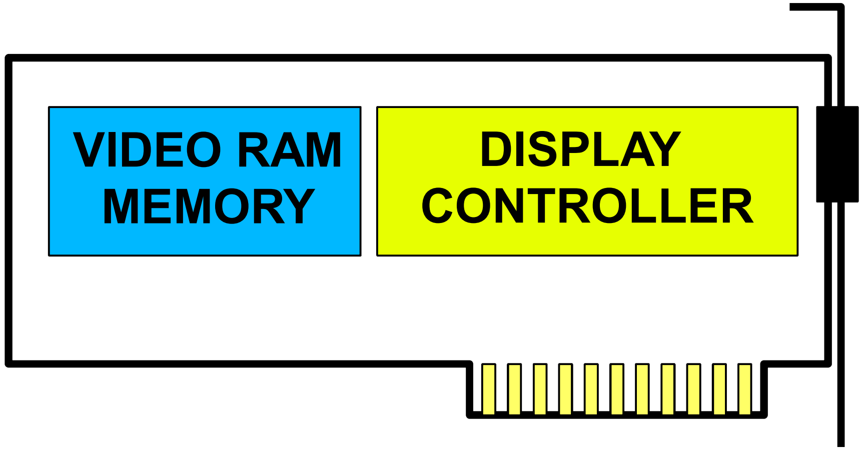
6.3. Real Time Clock (RTC) (IX)

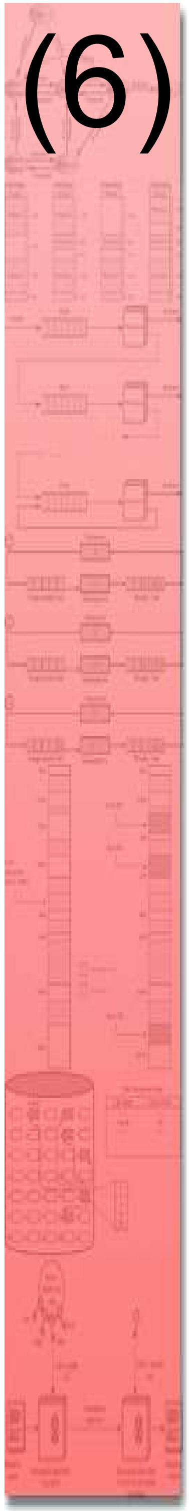
- The RTC does not generate interrupt requests by default. It is necessary to program it.
- Register B is used for determining what event has raised the interrupt (alarm, periodic interrupt or change of time/date).
- The interrupt service routine must verify if the event that has raised the interrupt is the desired one by reading register C.
- At the end of the interrupt routine, it is necessary to send the corresponding EOIs to the slave and master PICs (8259).
- It generates interrupt 70h and is connected to IRQ 0 of the slave PIC.

(6)

6.4. Video and display controller (I)

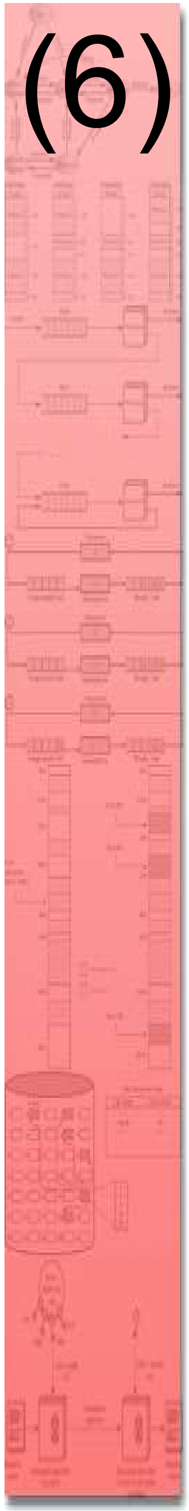
- Components of the video controller:
  - Video RAM memory:
    - It stores the information shown on the screen.
    - Mapped into the PC’s address space.
    - Organized as pages.
  - Display controller:
    - It periodically reads the contents of the active page from the video RAM, generating the display output signal.
    - The program can change the active page at any time (***double buffering*** technique).





6.4. Video and display controller (II)

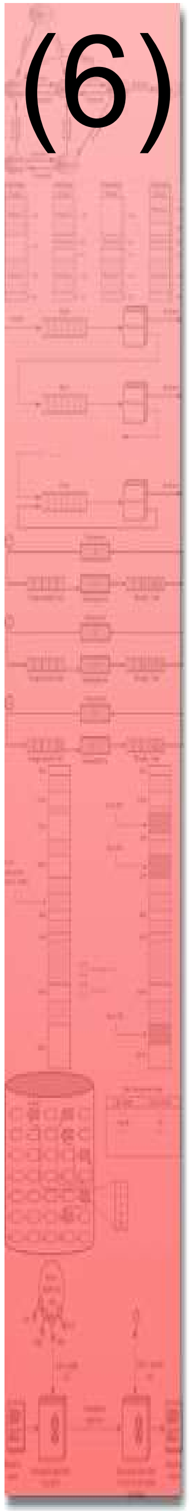
- Pixel
  - Basic unit of graphical information of the display, which is organized as a 2D matrix of pixels.
- Display resolution
  - Number vertical pixels (rows) x Number horizontal pixels (columns)
- Each pixel is constituted by 3 colors in color displays:
  - Red, Green and Blue (RGB).
- *Dot Pitch*
  - Pixel size in mm. Quality factor of display. The best displays usually have a dot pitch of 0.18 mm and the worst of 0.25 mm.



(6)

6.4. Video and display controller (III)

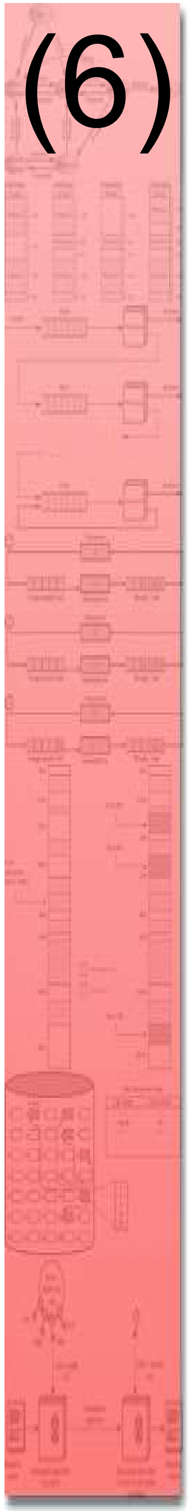
- Evolution of the first video controllers
  - MDA mode (Monochrome Display Adapter)
    - First monochrome text mode.
  - CGA mode (Color Graphics Adapter)
    - First color graphical mode. Low resolution and number of colors.
  - EGA mode (Enhanced Graphics Adapter)
    - Launched in 1985. Very costly. It had little acceptance.
  - HERCULES mode
    - Quality improvement of the CGA graphics, but it is monochrome (720 x 348 pixels). It supports MDA mode (text).
  - VGA mode (Video Graphics Adapter)
    - Launched in 1987. Substitute of EGA in being much more affordable.
  - SVGA mode (Super VGA)
    - Improvement of VGA.



(6)

6.4. Video and display controller (IV)

MODE	TYPE	RESOL.	COLOR	# COLORS	CGA	EGA	VGA	SVGA
0	T	40x25	M	16	S	S	S	S
1	T	40x25	C	16	S	S	S	S
2	T	80x25	M	16	S	S	S	S
3	T	80x25	C	16	S	S	S	S
4	G	320x200	C	4	S	S	S	S
5	G	320x200	M	4	S	S	S	S
6	G	640x200	M	2	S	S	S	S
7	T	80x25	M	2		S	S	S
13	G	320x200	C	16		S	S	S
14	G	640x200	C	16		S	S	S
15	G	640x350	M	2		S	S	S
16	G	640x350	C	16		S	S	S
17	G	640x480	M	2			S	S
18	G	640x480	C	16			S	S
19	G	320x200	C	256			S	S


(6)

6.4. Video and display controller (V)

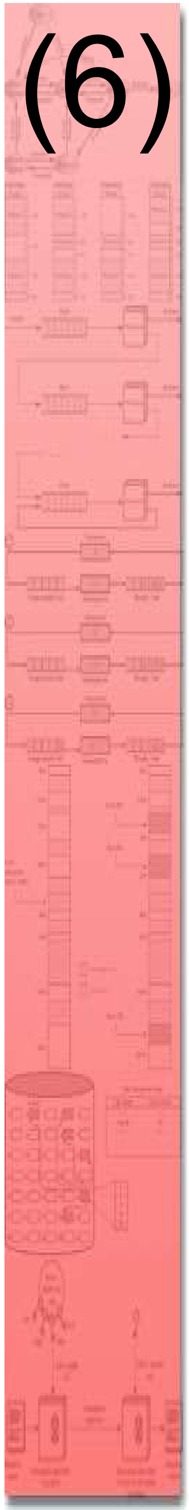
- In the PC’s memory map there are 128 KB reserved for the video controller.
- Reads/Writes from/to those memory addresses of the PC are mapped to video memory.

A000h:0000h

...

B000h:FFFFh

Video mode	Initial address	Final address	Size
Mode 7 (EGA, VGA)	B000:0000	B000:7FFF	32 KB
Modes 0 to 6 (CGA)	B800:0000	B800:7FFF	32 KB
Modes 13 ... (EGA, VGA, SVGA)	A000:0000	B000:FFFF	128 KB



6.4. Video and display controller (VI)

- Text modes (alphanumeric)
  - Necessary to define attribute and ASCII code of each character.
  - Necessary to calculate the memory address where to write a character that appears at the screen position (X, Y).
  - The cursor indicates the active position on the screen. It is physical in text mode.
  - The cursor’s position can be modified and queried through the BIOS.
  - There is no physical cursor in graphical mode ⇒ the programs must create it.

(6)

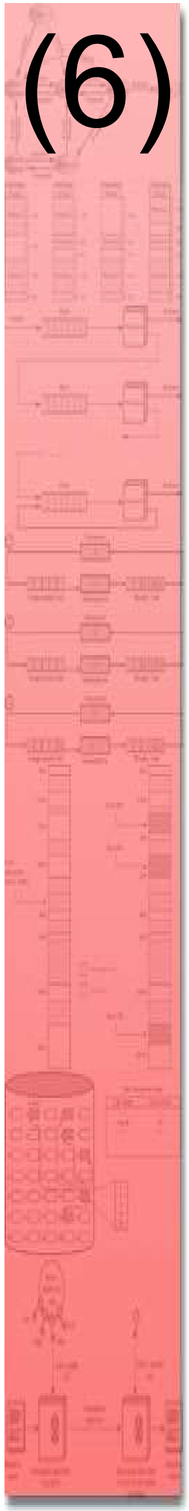
6.4. Video and display controller (VII)

- Information unit ⇒ character (2 consecutive bytes)
- Every character is defined by a matrix of pixels of fixed size (all of them occupy the same space).

7	6	5	4	3	2	1	0
ASCII code of character Even address							

7	6	5	4	3	2	1	0
Attribute of character Odd address							
K	R	G	B	I	R	G	B
Background				Text			

- **K** : Blink
- **R** : Red
- **G** : Green
- **B** : Blue
- **I** : Intensity



6.4. Video and display controller (VIII)

- Character’s address at position (X,Y):

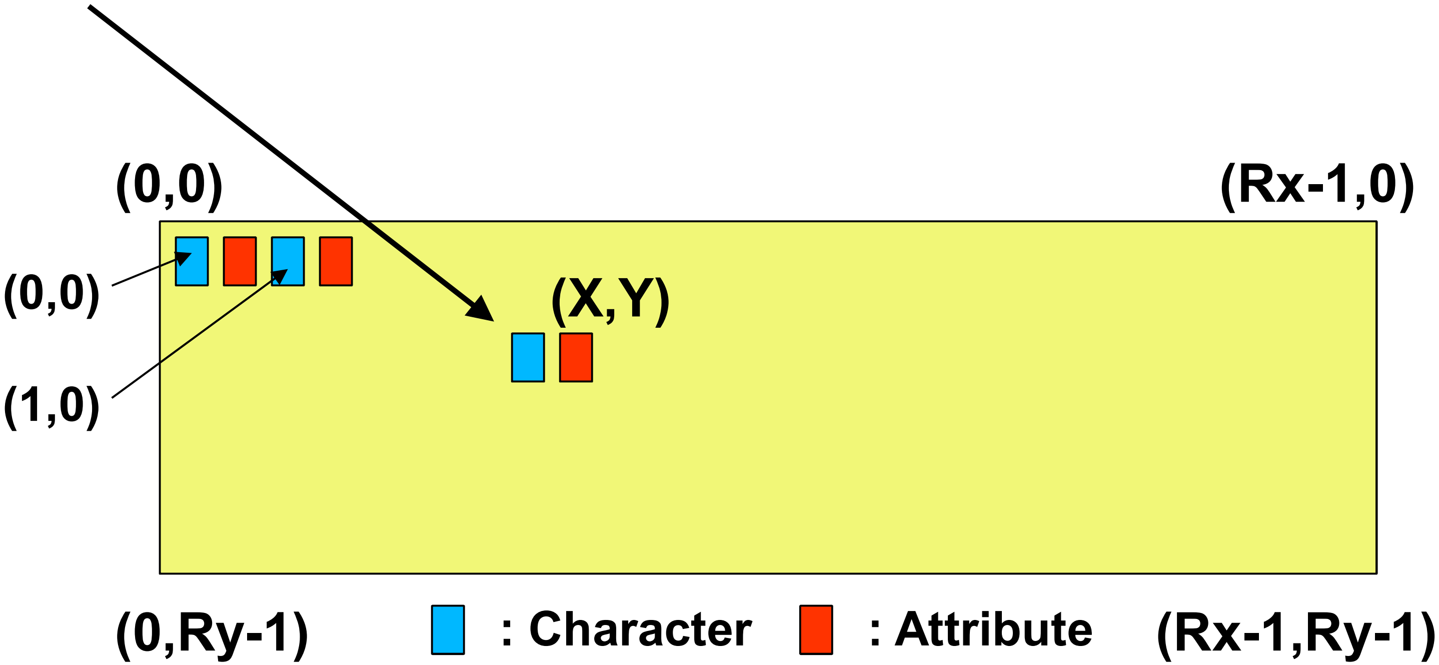
  @ = **Buff** + **Page** * **PageSize** + 2 * **Y** * **Rx** + 2 * **X** + **B**

  - Buff : Initial address on the PC where the video RAM memory is mapped in text mode.
  - Page : Page number
  - PageSize: Number of bytes of a screen page
  - B : 0 if byte is ASCII code, 1 if byte is attribute
  - X : Column
  - Y : Row
  - Rx : Column resolution (number of columns / row)

6.4. Video and display controller (IX)

- Character's address at position (X,Y):

$$@ = \mathbf{Buff} + \mathbf{Page} * \mathbf{PageSize} + 2 * \mathbf{Y} * \mathbf{Rx} + 2 * \mathbf{X} + \mathbf{B}$$



6.4. Video and display controller (X)

CGA text mode

- Character matrix: 8 x 8 píxels.
- Character: 7 x 7 pixels.
- Resolution: 80 x 25 characters.
- Colors: 16 for character and 16 for backgnd.
- Page size: 2000 char x 2 bytes/char.
- Video buffer size: 16 KB (4 pages).
- Initial segment of video buffer: B800h
- Screen:
  - 80x25x2=4000 bytes (0FA0h) positions of video memory.
  - Four different screens fit into the video buffer.

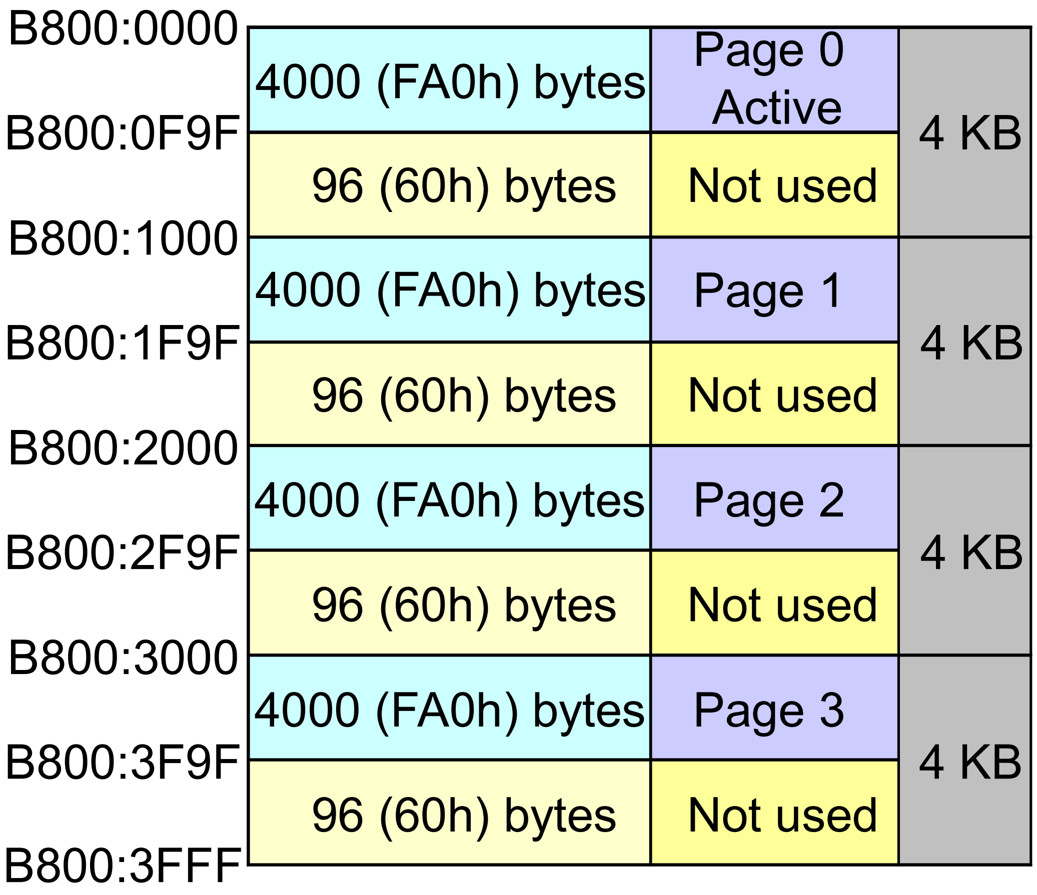
I	R	G	B	Color
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	White
1	0	0	0	Gray
1	0	0	1	Light blue
1	0	1	0	Light green
1	0	1	1	Light cyan
1	1	0	0	Light red
1	1	0	1	Light magenta
1	1	1	0	Yellow
1	1	1	1	Intense white

(6)

6.4. Video and display controller (XI)

CGA text mode

- One active page can only be active at any time and it is the one being displayed.

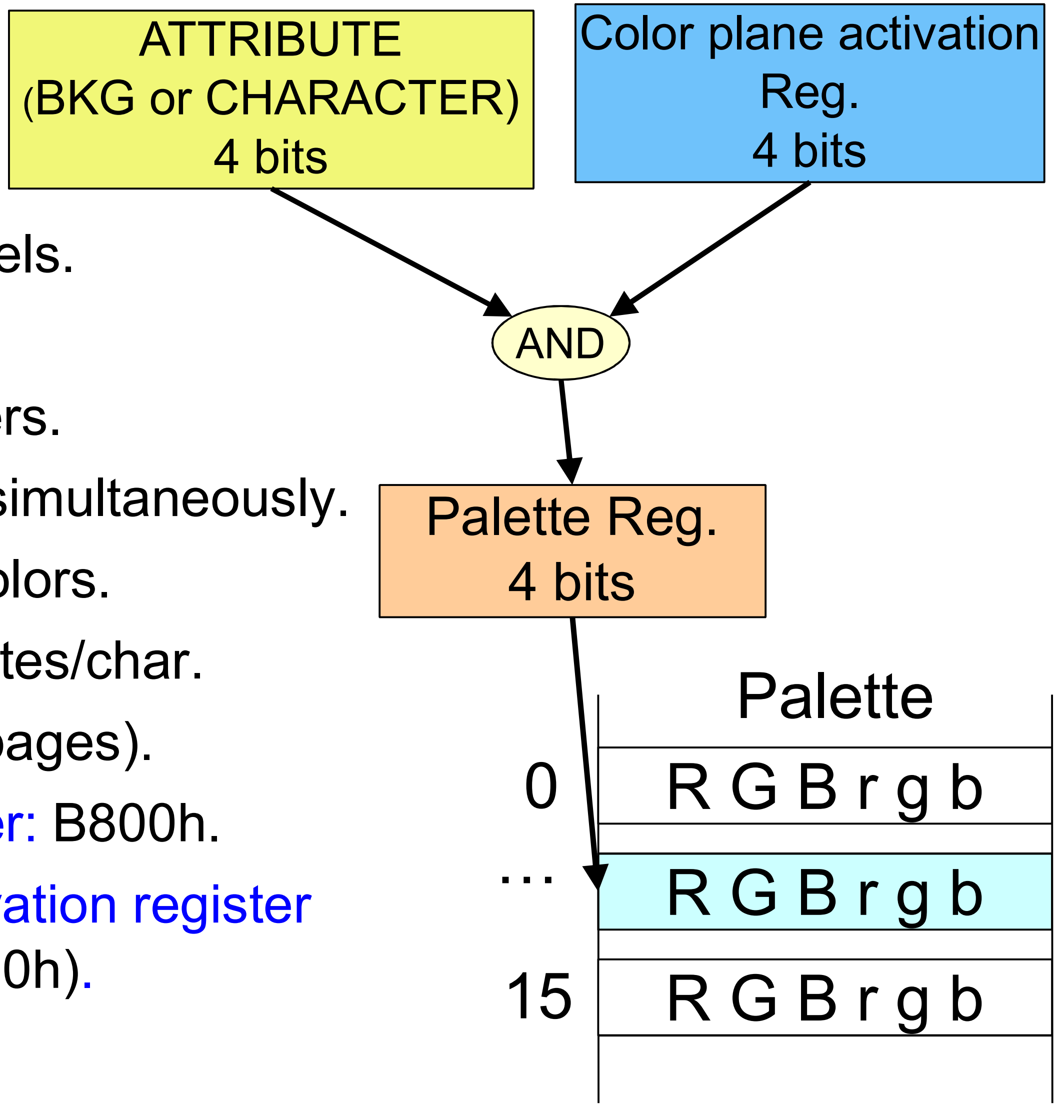


Address	Bytes	Page	Size
B800:0000	4000 (FA0h) bytes	Page 0 Active	4 KB
B800:0F9F	96 (60h) bytes	Not used	
B800:1000	4000 (FA0h) bytes	Page 1	4 KB
B800:1F9F	96 (60h) bytes	Not used	
B800:2000	4000 (FA0h) bytes	Page 2	4 KB
B800:2F9F	96 (60h) bytes	Not used	
B800:3000	4000 (FA0h) bytes	Page 3	4 KB
B800:3F9F	96 (60h) bytes	Not used	
B800:3FFF			

(6)

6.4. Video and display controller (XII)

EGA text mode



ATTRIBUTE
(BKG or CHARACTER)
4 bits

Color plane activation
Reg.
4 bits

AND

Palette Reg.
4 bits

Palette

0 R G B r g b

… R G B r g b

15 R G B r g b

- Character matrix: 8 x 14 pixels.
- Character: 7 x 7 pixels.
- Resolution: 80 x 25 characters.
- Colors: 64 (2^6) with 16 (2^4) simultaneously.
- **Palettes**: 4 palettes of 16 colors.
- Page size: 2000 char x 2 bytes/char.
- Video buffer size: 16 KB (4 pages).
- Initial segment of video buffer: B800h.
- Contents of palette and activation register defined through BIOS (INT 10h).

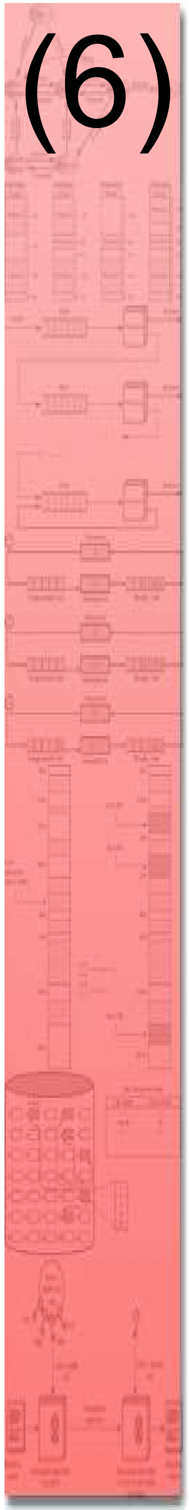
6.4. Video and display controller (XIII)

EGA text mode

- Default palette compatible with CGA mode.
- Control register of the controller can be programmed for activating blinking: 1 in most significant bit of the attribute byte yields blinking.

  The background color index is reduced to the 3 least significant bits (8 simultaneous colors).

Índex	R	G	B	r	g	b	Color
0	0	0	0	0	0	0	Black
1	0	0	0	0	0	1	Blue
2	0	0	0	0	1	0	Green
3	0	0	0	0	1	1	Cyan
4	0	0	0	1	0	0	Red
5	0	0	0	1	0	1	Magenta
6	0	0	0	1	1	0	Brown
7	0	0	0	1	1	1	White
8	1	1	1	0	0	0	Gray
9	0	0	1	0	0	1	Light blue
10	0	1	0	0	1	0	Light green
11	0	1	1	0	1	1	Light cyan
12	1	0	0	1	0	0	Light red
13	1	0	1	1	0	1	Light magenta
14	1	1	0	1	1	0	Yellow
15	1	1	1	1	1	1	Intense white

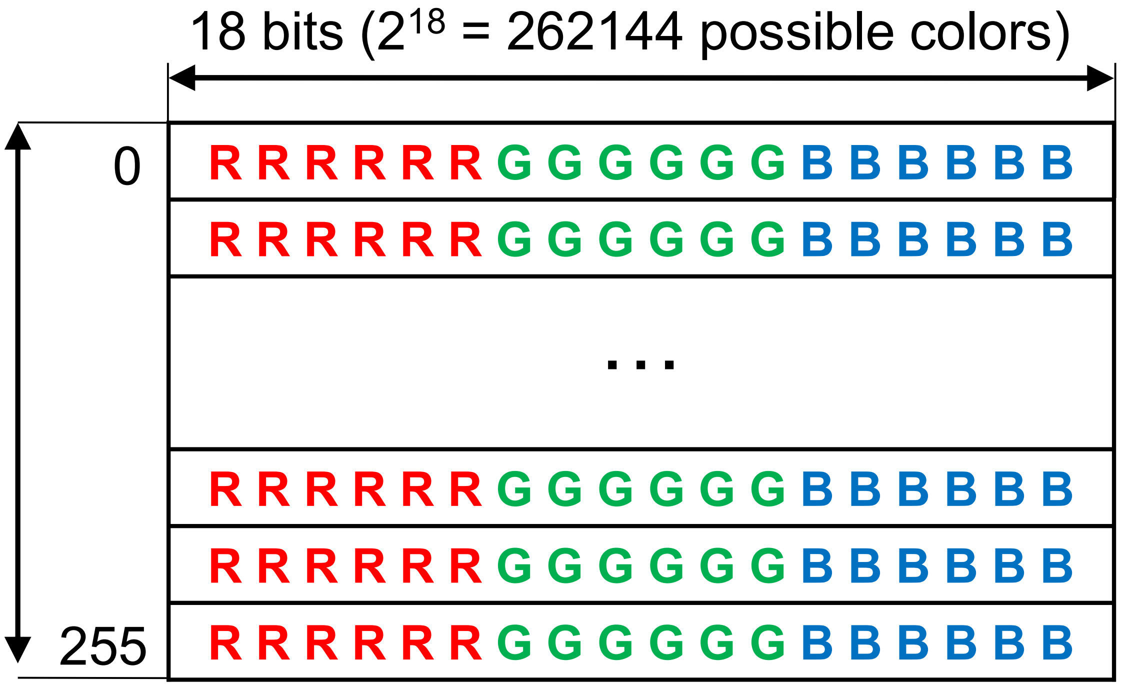

(6)

6.4. Video and display controller (XIV)

VGA text mode

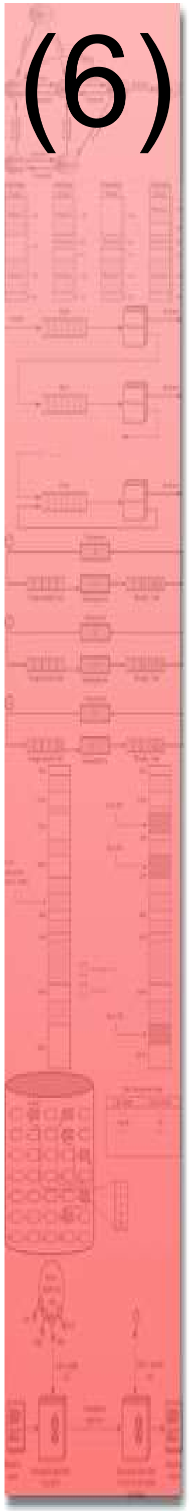
- Video memory of 256 KB.
- **Analogic signal** generated through a digital-to-analog converter (DAC) is sent to the display.
- 256 color registers of 18 bits.

256 entries =
4 blocks x
4 palettes/block x
16 colors/palette


18 bits (2^{18} = 262144 possible colors)

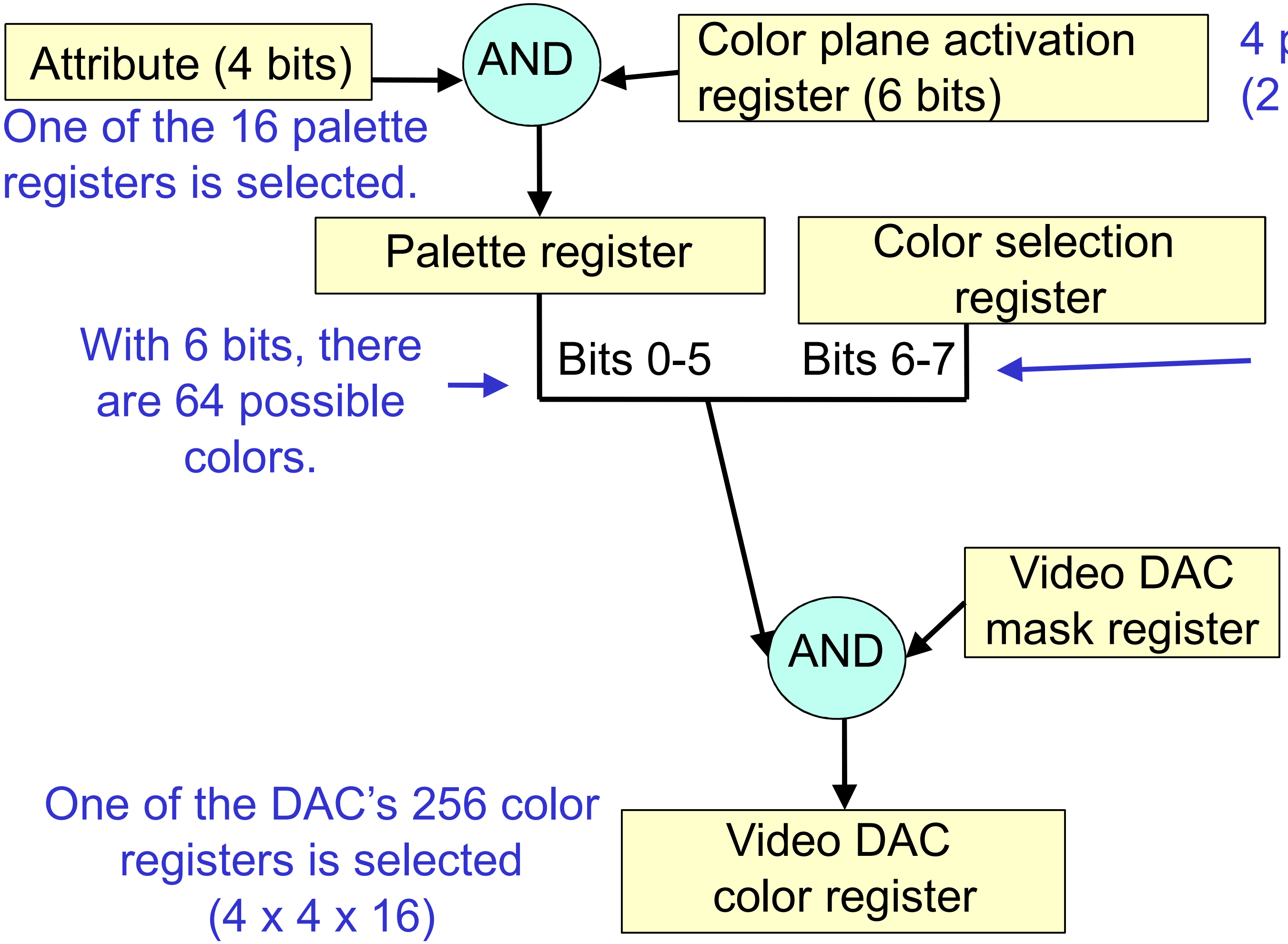
0	R R R R R R G G G G G G B B B B B B
	R R R R R R G G G G G G B B B B B B
	…
	R R R R R R G G G G G G B B B B B B
	R R R R R R G G G G G G B B B B B B
255	R R R R R R G G G G G G B B B B B B

(6)



6.4. Video and display controller (XV)

VGA text mode



4 palettes/block (2 highest bits)

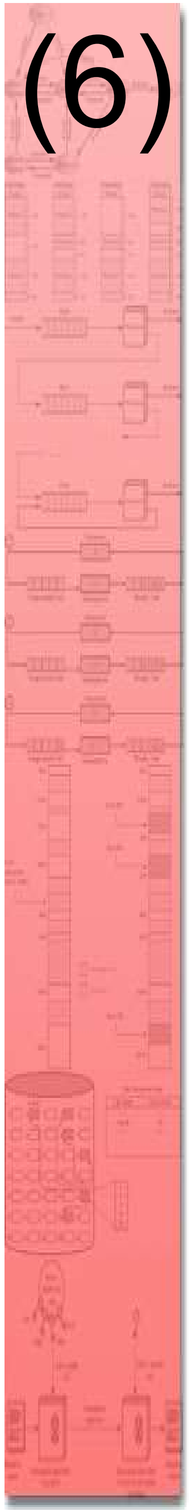
One of the 16 palette registers is selected.

With 6 bits, there are 64 possible colors.

With 2 bits, one of the DAC’s 4 blocks of 64 colors is selected (4 palettes/block)

One of the DAC’s 256 color registers is selected (4 x 4 x 16)

DAC’s color, activation, selection and mask registers defined through BIOS (INT 10h).



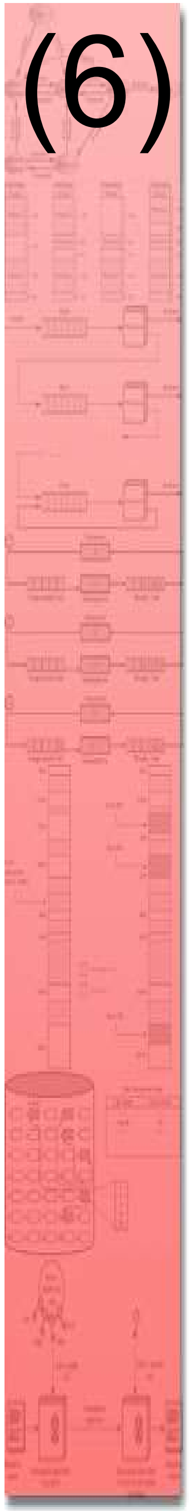
6.4. Video and display controller (XVI)

- Graphical modes
  - The addressable unit is the pixel.
  - Every pixel has an associated color.
  - Pixels do not blink.
  - The color of every pixel depends on the contents of a group of bits in a specific position within the video RAM memory.
  - The number of colors that can be simultaneously displayed depends on the number of bits associated with each pixel.
- Example: Mode 13h ⇒ VGA (320x200, 256 colors)

6.4. Video and display controller (XVII)

VGA (320x200, 256 colors)

- Every pixel is defined by an 8-bit attribute stored in a position of the video RAM.
- Each pixel can be displayed in one out of 256 different colors (8 bits).
- The attribute value of every pixel indexes one of the 256 DAC’s color registers.
- The video memory is linear, beginning at A000:0000 and ending at A000:FA00.



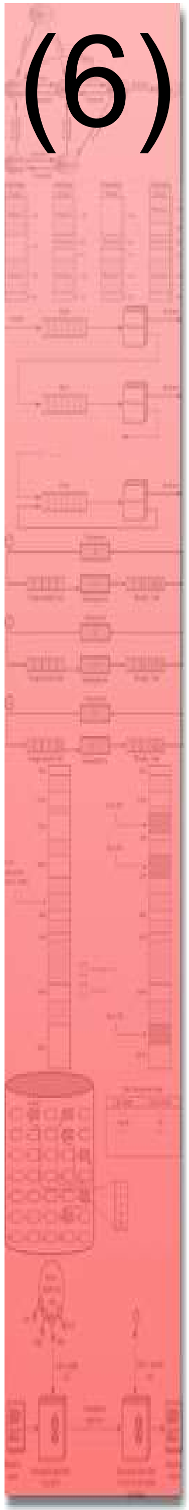
6.4. Video and display controller (XVIII)

AH	Function INT 10h (BIOS)
00h	**Set display mode**

	Input:
	AL=00h, 40 x 25 text, gray; AL=01h, 40 x 25 text, color;
	AL=02h, 80 x 25 text, gray; AL=03h, 80 x 25 text, color;
	AL=04h, 320 x 200 graphical, color; AL=05h, 320 x 200 graphical, gray;
	AL=06h, 640 x 200 graphical, B&W; AL=07h, 80 x 25 text, B&W;
	AL=10h, 640 x 350, graphical, 16 colors;
	AL=12h, 640 x 480 graphical, 16 colors;
	AL=13h, 320 x 200, graphical, 256 colors

01h	**Set cursor size**
	Input: CH = Initial line of cursor (0-15). If 0, the top line blinks. CL = Final line of cursor (0-15)
02h	**Set cursor position**

	Input:
	DH = row (0-24),
	DL = column (0-79),
	BH = page number



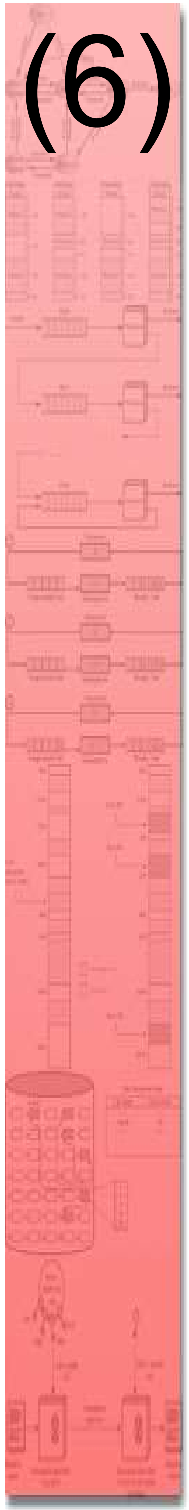
(6)

6.4. Video and display controller (XIX)

AH	Function INT 10h (BIOS)
07h	**Scroll down window in active page (the active page must be set before calling it)**

	Input:
	AL= number of lines (top lines in the window are cleared.
	If AL=0, the entire window is cleared);
	CH = top row,
	CL = left column,
	DH = bottom row,
	DL = right column,
	BH = attribute used to write blank lines at windows's top

08h	**Read character and attibute at cursor position (text mode)**
	Input: BH = page number Output: AL = read character, AH = attribute of read character
09h	**Write character and attribute at current cursor position (both modes)**
	Input: BH = page number, BL = attribute of character, CX = number of characters to be written, AL = character to be written

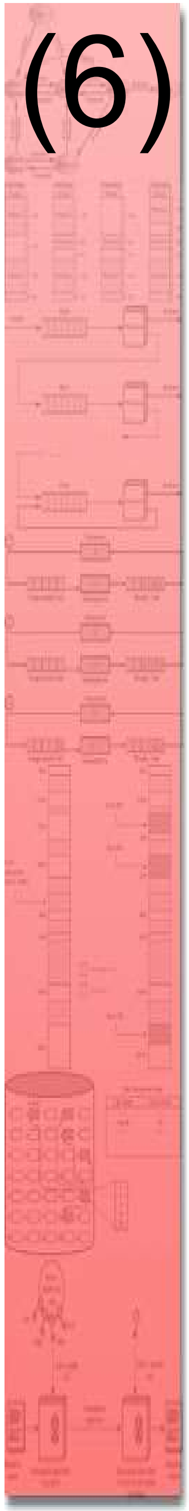


6.4. Video and display controller (XX)

AH	Function INT 10h (BIOS)
0Ah	**Write character at cursor position (both modes)** Input: BH=page number, BL=attribute, AL=character, CX= no. times

0Bh	**Set background color and palette (graphical, 320x200)**
	Input:
	BH = 0 and BL = background color;
	BH = 1 and BL = palette (0-1)

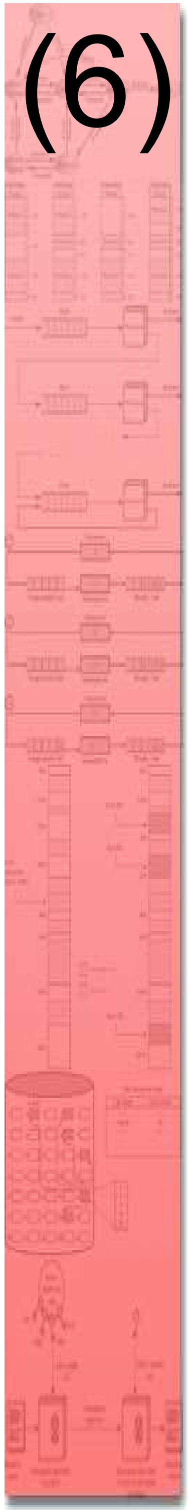
0Ch	**Write pixel (graphical mode)**
0Dh	**Read pixel (graphical mode)**
0Eh	**Write character at cursor position and advance cursor** Input: AL = character to be written BH = number of active page
0Fh	**Read current state of screen** Input: BH = page number Output: AL = number of screen mode, AH = number of columns in screen, BH = number of active page



(6)

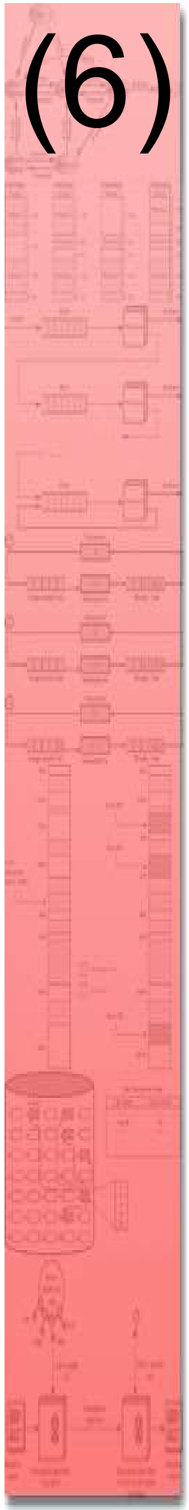
6.4. Video and display controller (XXI)

AH	AL	Function INT 10h (BIOS)
10h	10h	**Update DAC’s color register (VGA)** Input: BX = no. color register (0..255), CH = green value, CL = blue value, DH = red value (color values: 0..63)
10h	12h	**Update block of DAC’s color registers (VGA)** Input: BX=first register to be updated, CX=number of registers to be updated, ES:DX=Table of red-green-blue values
10h	13h	**Set selection space** Input: BL= 0 and BH=0 ⇒ 4 out of 64, BL=0 and BH=1 ⇒16 out of 16
10h	15h	**Read DAC’s color register (VGA)** Input: BL=number of color register (0..255) Output: CH=green value, CL=blue value, DH=red value
10h	17h	**Read a block of DAC’s color registers** Input: BX=first register to be read, CX=number of registers, ES:DX=Table of red-green-blue values Output: table
10h	18h	**Update DAC’s mask register** Input: BL=new value
10h	19h	**Read DAC’s mask register** Output: BL= value of mask register



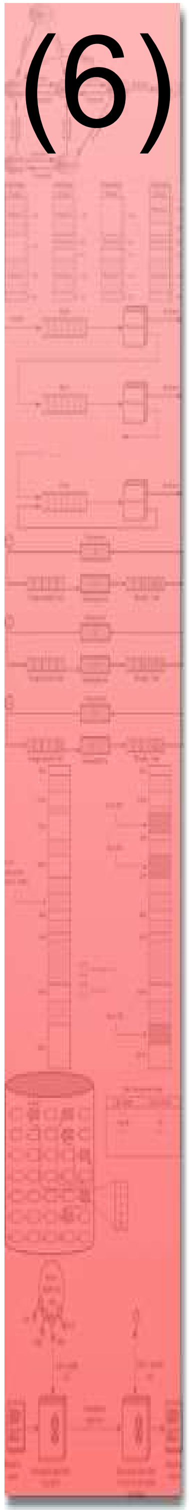
6.4. Video and display controller (XXII)

AH	Function INT 21h (DOS)
02h	**Write a character at the cursor position** Input: DL = ASCII code of character.
09h	**Write a string of characters at the cursor position.** **The string must be terminated with character '$'.** Input: DS:DX = String address.



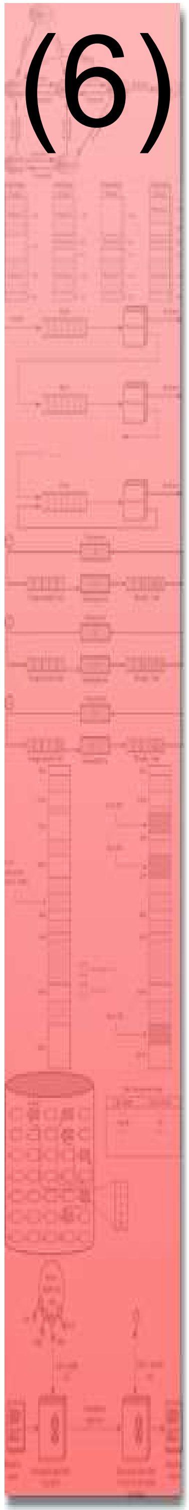
6.5. Parallel port. Printer (I)

- The PC supports up to 4 parallel ports (LPT1, LPT2, LPT3, LPT4)
- Every parallel port is handled through three registers accessible at consecutive I/O addresses:
  - **Data register**: in which the CPU sends data or reads data if the port is bidirectional.
  - **State register**: in which the CPU reads the state of the parallel port (e.g., printer off, out of paper, etc.).
  - **Control register**: in which the CPU sends control signals to the parallel port (e.g., data validation, printer start up, etc.)



6.5. Parallel port. Printer (II)

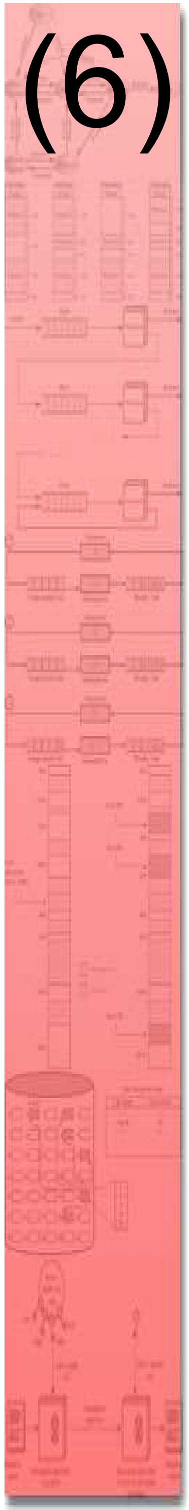
- Types of parallel ports (selected through BIOS):
  - Standard parallel port (SPP)
    - Output data register (8 bits).
    - Input state register (4 bits)
    - CPU sends each data byte and manages protocol.
  - Bidirectional port (PS/2 or extended)
    - Input or output data register (8 bits).
    - Input state register (4 bits)
    - CPU sends/**receives** each data byte and manages protocol.
  - Enhanced Parallel Port (EPP)
    - Input or output data register (8 bits).
    - CPU sends/receives each data byte.
    - **Port manages protocol.**
    - Signal configuration different to SPP and bidirectional.
  - Enhanced Capability Port (ECP)
    - Similar to EPP but **data byte transfers through DMA**.



(6)

6.5. Parallel port. Printer (III)

Type of Parallel Port	Input Mode	Output Mode	Speed
SPP (Standard)	Nibble (4 bits) (State Reg.)		50 KByte/s
SPP (Standard)		Compatible (8 bits)	150 KB/s
Bidirectional (PS/2 or Extended) (1987)	Byte (8 bits)		150 KB/s
Bidirectional (PS/2 or Extended) (1987)		Compatible (8 bits)	150 KB/s
EPP (Enhanced Parallel P.) (IEEE 1284)(1991) (Peripherals)	EPP (8 bits)		500 KB/s – 2 MB/s
EPP (Enhanced Parallel P.) (IEEE 1284)(1991) (Peripherals)		EPP (8 bits)	500 KB/s – 2 MB/s
ECP (Enhanced Capability P.) (IEEE 1284) (DMA) (1992)	ECP (8 bits)		500 KB/s – 2 MB/s
ECP (Enhanced Capability P.) (IEEE 1284) (DMA) (1992)		ECP (8 bits)	500 KB/s – 2 MB/s



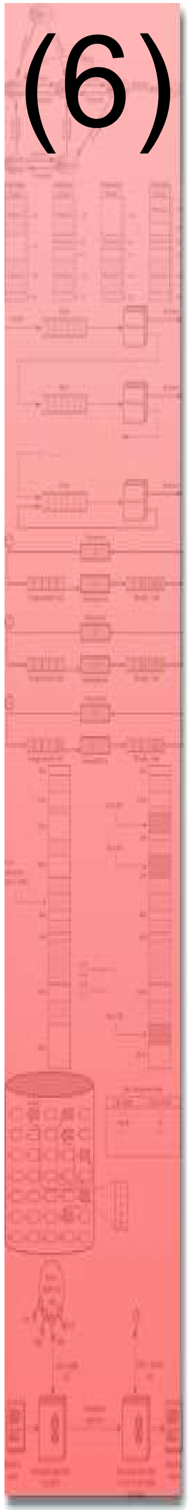
6.5. Parallel port. Printer (IV)

- During PC booting, the BIOS routines are in charge of:
  - Detecting the installed parallel ports.
  - Storing the base and time-out addresses.

0000h:0408h	0000h:040Ah	0000h:040Ch	0000h:040Eh
Base address LPT1	**Base address** LPT2	**Base address** LPT3	**Base address** LPT4

0000h:0478h	0000h:0479h	0000h:047Ah	0000h:047Bh
Time-out LPT1	**Time-out LPT2**	**Time-out LPT3**	**Time-out LPT4**

- **Time-out** : maximum number of seconds waiting for some event. In case of a printer, it is the waiting time to determine that the printer is not available (off, off-line, out of paper, ...)



6.5. Parallel port. Printer (V)

- Ports are accessed using IN or OUT, specifying the register.
- Usual register addresses (they depend on the manufacturer of the motherboard):

Port	Data register	State register	Control register
LPT1	03BCh	03BDh	03BEh
LPT2	0378h	0379h	037Ah
LPT3	0278h	0279h	027Ah

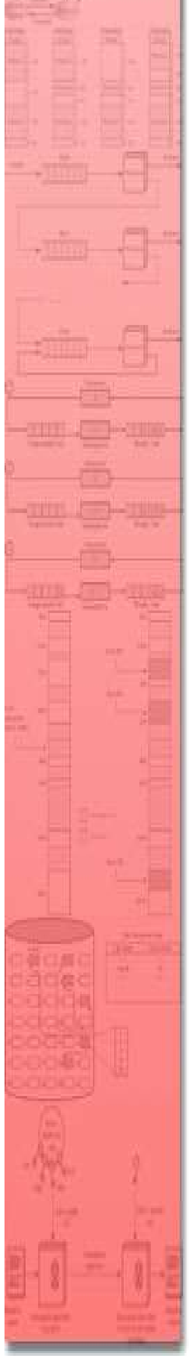
- LPT4 is only used in some cases.

6.5. Parallel port. Printer (VI)

- Data register: Base address
  - 8-bit register corresponding to pins 2 to 9 of the external connector (DB-25) of the parallel port.
  - In SPP mode, it is an output port to send data to the printer.
  - In bidirectional mode, it allows sending or receiving data (not simultaneously ⇒ *half duplex*).

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

(6)



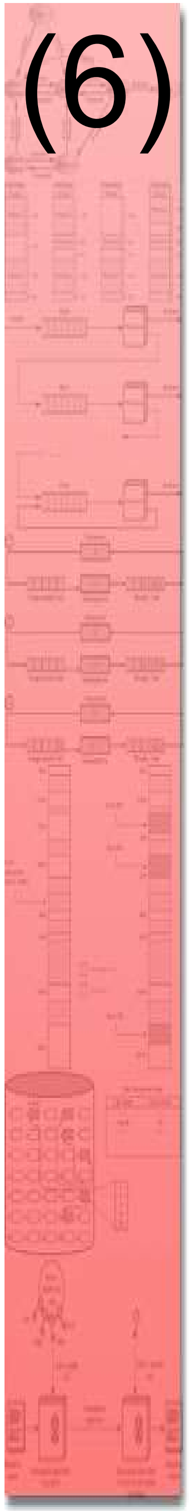
6.5. Parallel port. Printer (VII)

- State register: Base address + 1
  - Read only (5 bits are used).

7	6	5	4	3	2	1	0
BUSY (inv)	ACK	PE	SLCT	ERROR	----	----	----

- **BUSY (inv)**: 0 indicates busy printer. Also set to 0 in error situations.
- **ACK**: 0 indicates the printer has received a data byte and it is ready to receive a new one.
- **PE**: 1 indicates printer is out of paper.
- **SLCT**: 1 indicates printer is online.
- **ERROR**: 0 indicates printer error (out of paper, malfunction, etc.).

(inv) Signal inverted by hardware in the DB-25 connector.



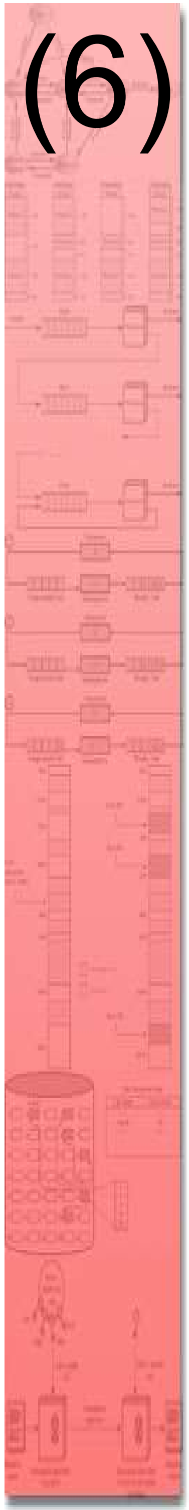
(6)

6.5. Parallel port. Printer (VIII)

- Control register: Base address + 2
  - Write only (6 bits are used).

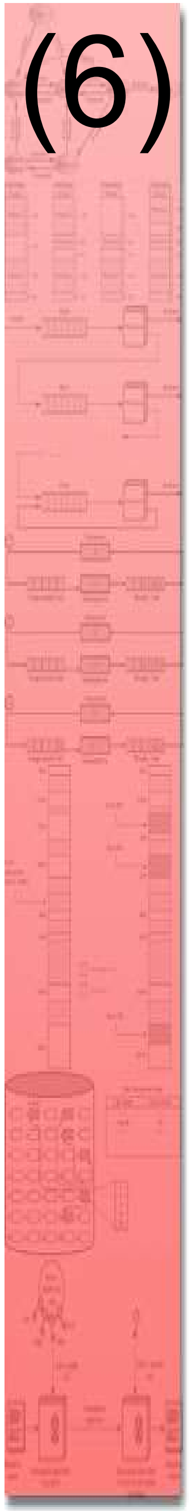
7	6	5	4	3	2	1	0
----	----	BIDIR	IRQEN	SLCT IN (inv)	$\underline{\text{INIT}}$	AUTOFD (inv)	STROBE (inv)

- **BIDIR:** 1 indicates input port. 0 (default) indicates output. It must be restored to 1 every time a data byte is received.
- **IRQEN:** 1 allows signal ACK to activate IRQ7 of the master PIC.
- **SLCT IN (inv):** 1 tells the printer that it has been selected.
- **$\underline{\text{INIT}}$:** Usually 1. When a 0 pulse is sent, the printer is reset.
- **AUTOFD (inv):** When this bit is 1, the printer executes a line feed when it receives the *carriage return* character (13).
- **STROBE (inv):** Transition 0-1 in this bit tells the printer the data byte is valid and can be read.



6.5. Parallel port. Printer (IX)

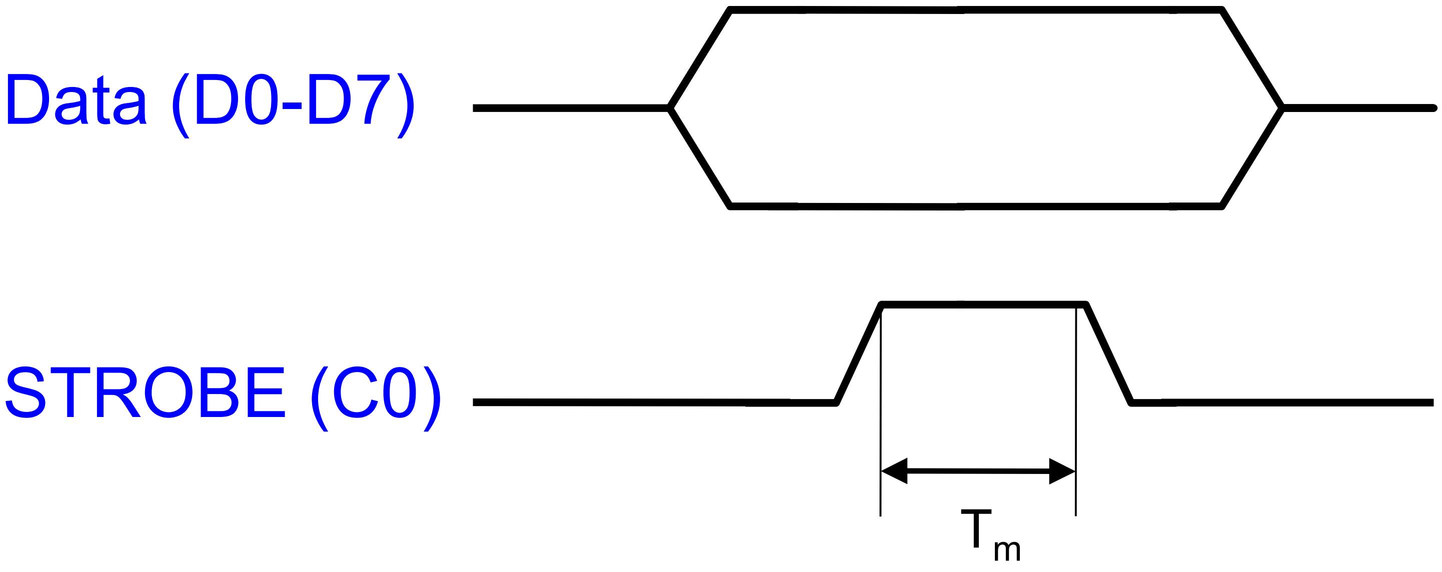
- Necessary to execute protocol to send a byte and wait for the acceptance from the printer:
  - STROBE protocol (simplest)
  - BUSY protocol (with feedback and active wait)
  - ACK protocol (based on interrupts).



(6)

6.5. Parallel port. Printer (X)

STROBE protocol



Data (D0-D7)

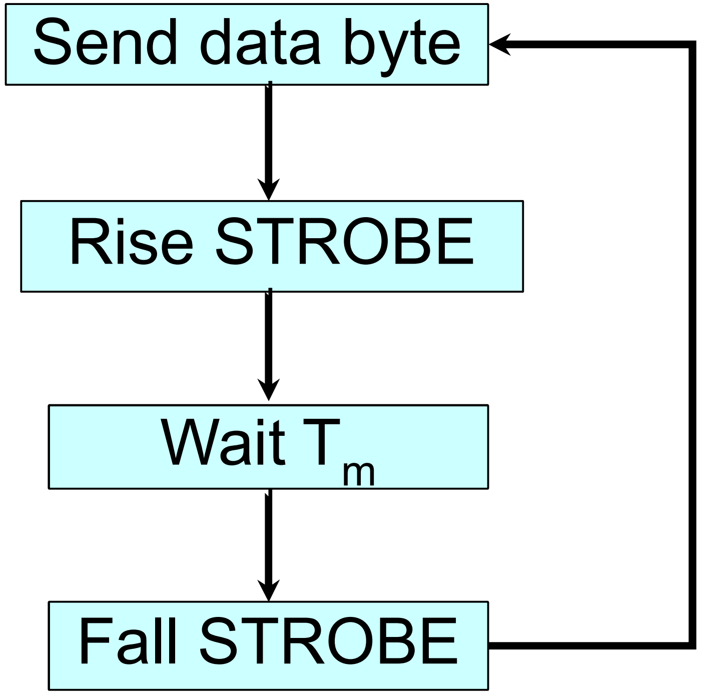
STROBE (C0)

T_m

T_m= minimum time for STROBE being effective

6.5. Parallel port. Printer (XI)

STROBE protocol

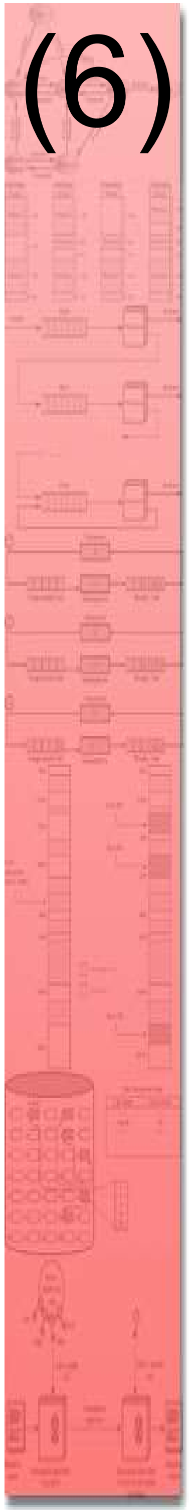


Send data byte

Rise STROBE

Wait T_m

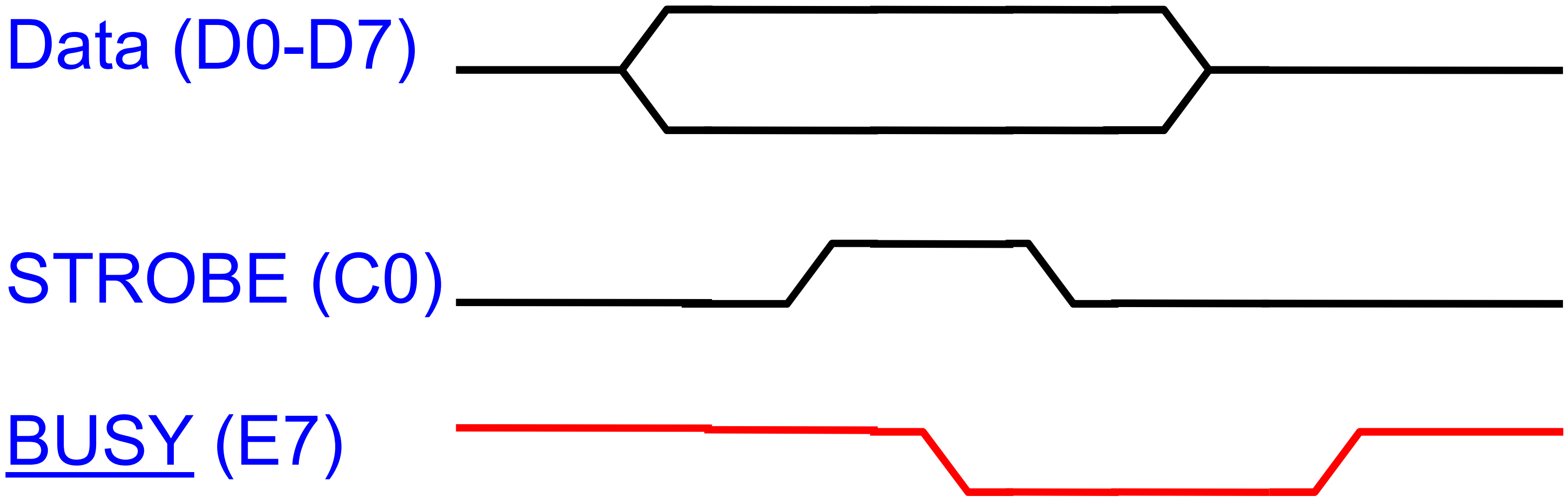
Fall STROBE



(6)

6.5. Parallel port. Printer (XII)

BUSY protocol



Data (D0-D7)

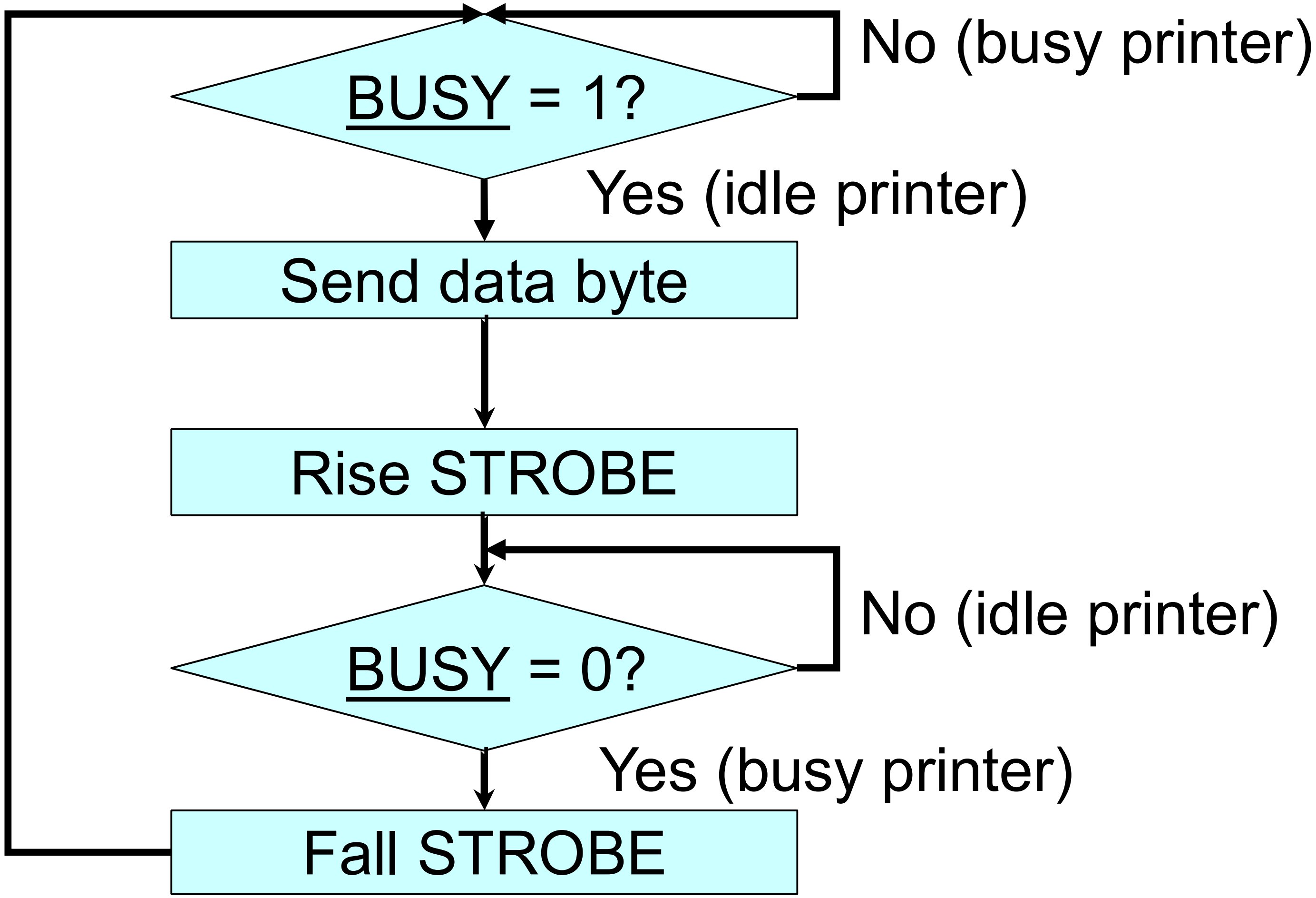
STROBE (C0)

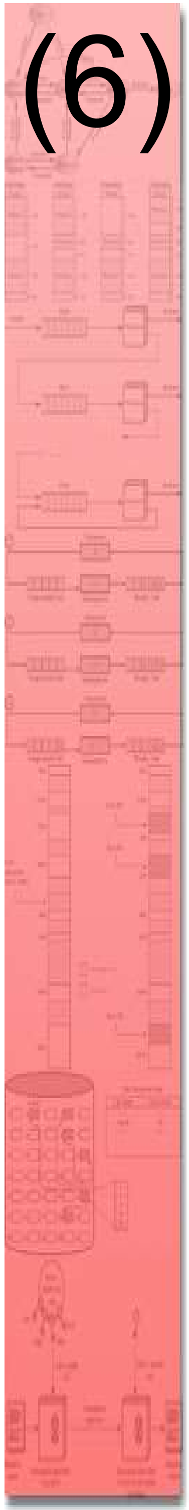
<u>BUSY</u> (E7)

(6)

6.5. Parallel port. Printer (XIII)

BUSY protocol

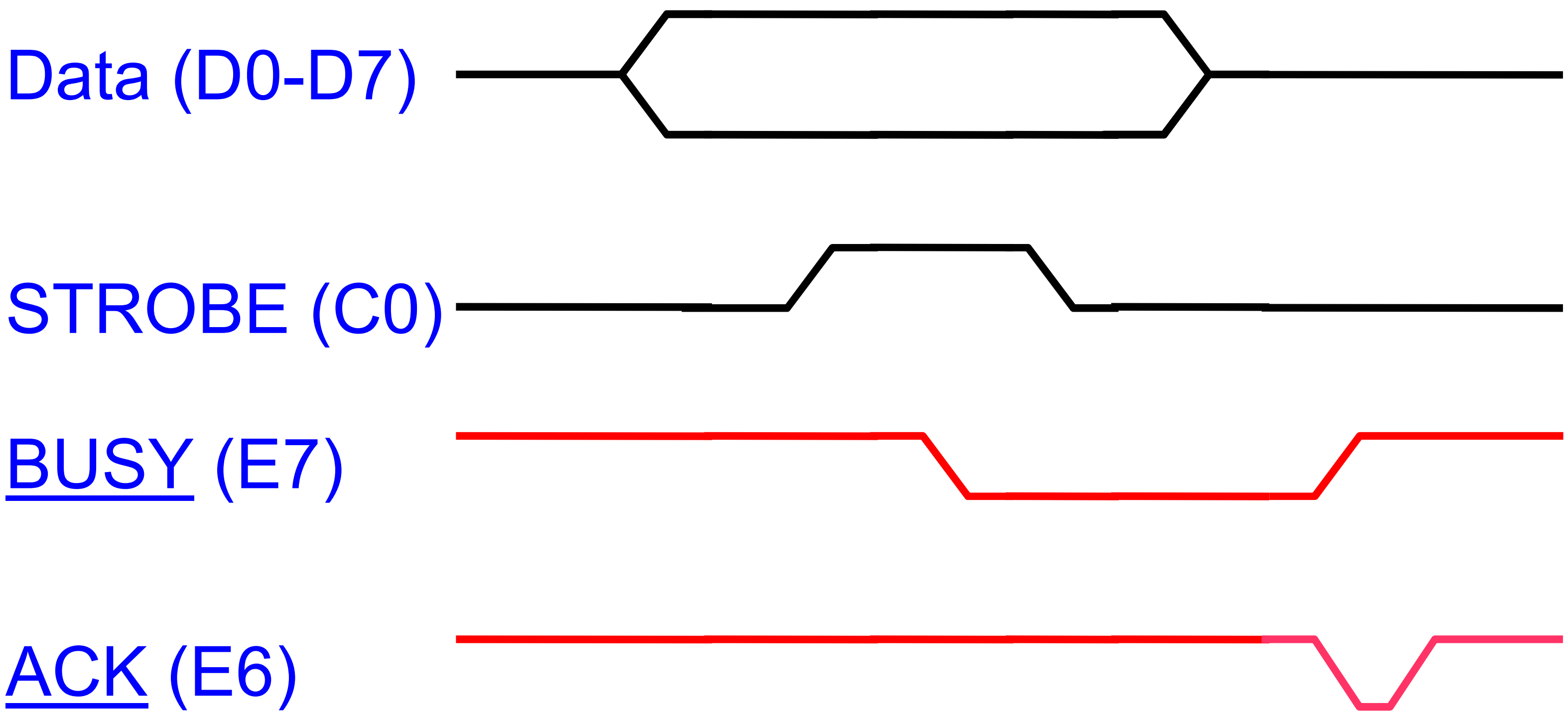




(6)

6.5. Parallel port. Printer (XIV)

ACK protocol



Data (D0-D7)

STROBE (C0)

BUSY (E7)

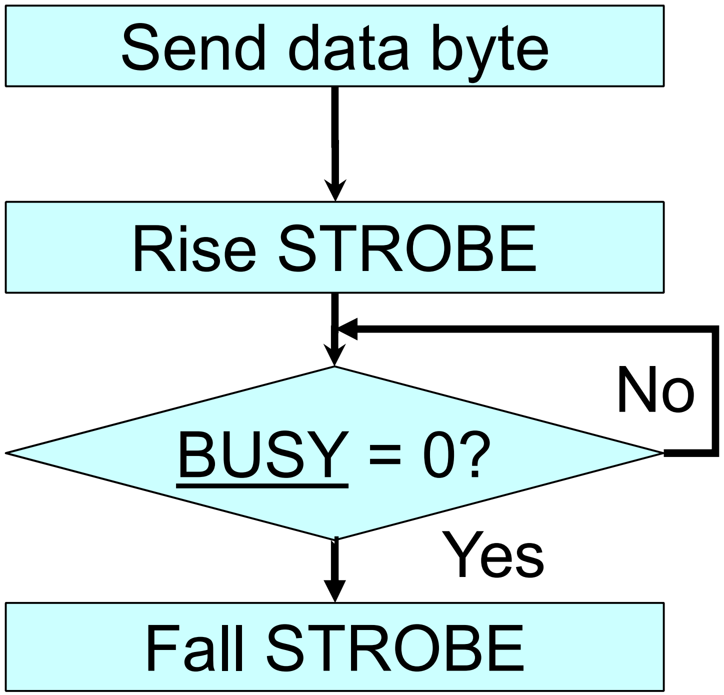
ACK (E6)

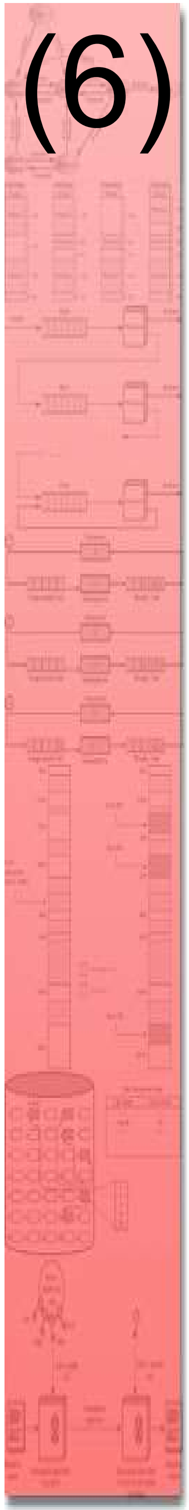
(6)

6.5. Parallel port. Printer (XV)

ACK protocol

- ACK activates INT 0Fh.
- ISR starts emission
- Not necessary to wait for rising edge of BUSY since it is 1 whenever an ACK is generated.
- First data byte sent through software interrupt.
- Remaining data bytes sent through hardware interrupt.





(6)

6.5. Parallel port. Printer (XVI)

- Relationship with pins of the DB-25 connector
  - Data register

7	6	5	4	3	2	1	0
PIN 9	PIN 8	PIN 7	PIN 6	PIN 5	PIN 4	PIN 3	PIN 2

  - Control register

7	6	5	4	3	2	1	0
				PIN 17	PIN 16	PIN 14	PIN 1

  - State register

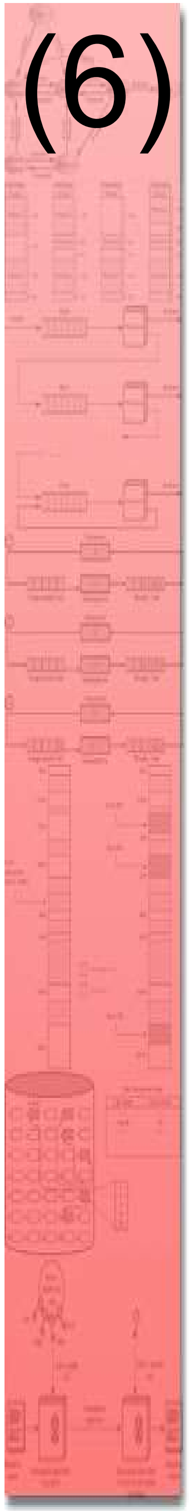
7	6	5	4	3	2	1	0
PIN 11	PIN 10	PIN 12	PIN 13	PIN 15			

PIN n indicates inverted signal in the connector through inverter gate.

6.5. Parallel port. Printer (XVII)

- Interrupts associated with the printer

INT	AH	Function	I/O	
5h		Print screen		Called from keyboard
17h	0h	Print character	Input	AL : ASCII to be printed DX : printer to be used (1,2,3)
			Output	AH : state byte
	1h	Initialize printer	Input	DX : printer to be used (1,2,3)
			Output	AH : state byte
	2h	Read printer state	Input	DX : printer to be used (1,2,3)
			Output	AH : state byte
21h	5h	Print character	Input	DL : character to be printed

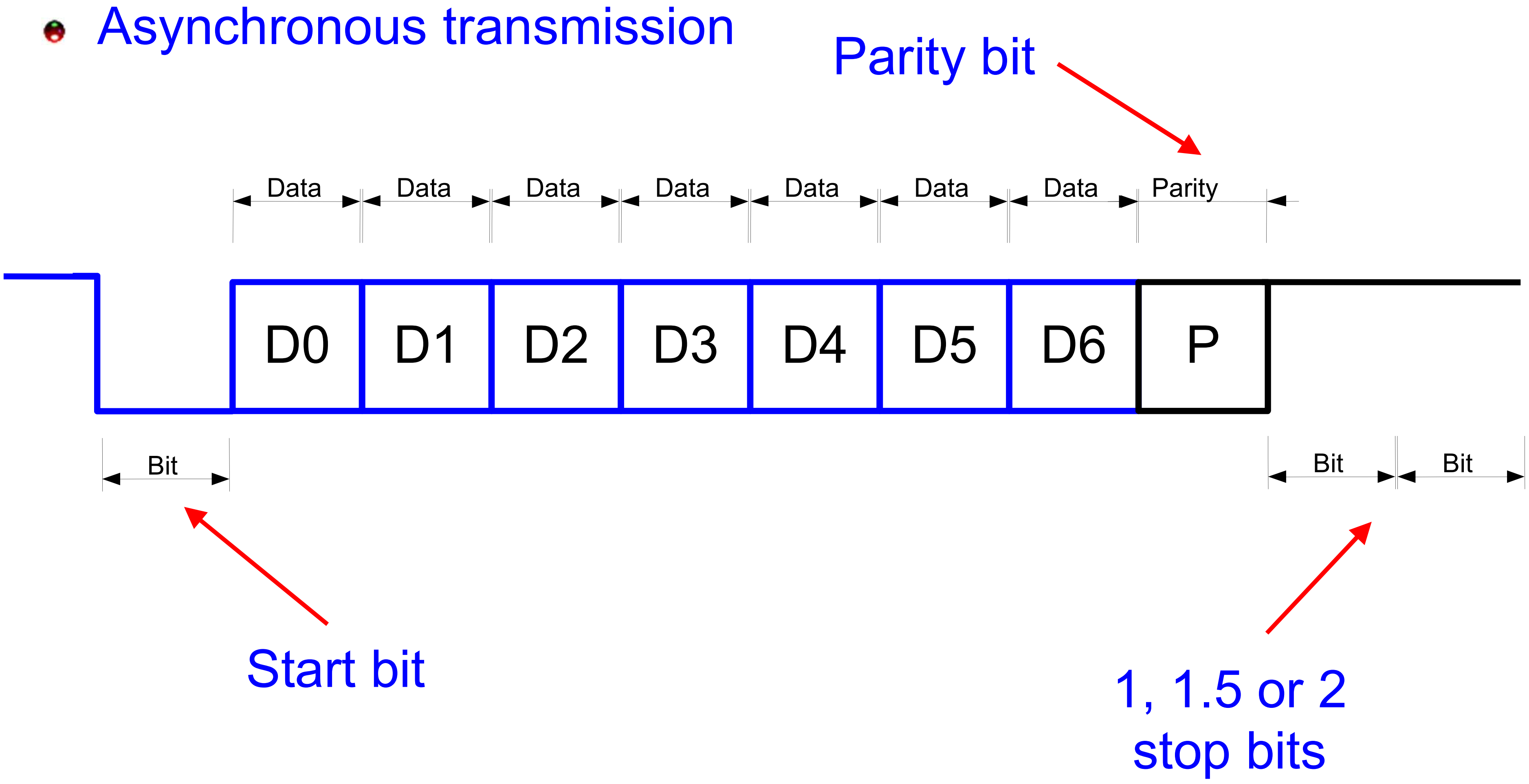


6.6. Asynchronous serial port (I)

- Based on an integrated circuit by *National Semiconductor* (UART 8250).
- It allows serial transmission of data.
- It allows longer transmission distances than in parallel (e.g., 15 meters using EIA RS-232-C codification).
- Data and control bits are sequentially transmitted through a single line.
- The time width of each bit depends on the transmission speed, which is expressed in bits per second (bps).
- Both the emitter and the receiver must be configured with the same values of data length, parity type, transmission speed and number of stop bits.

6.6. Asynchronous serial port (II)

- Asynchronous transmission



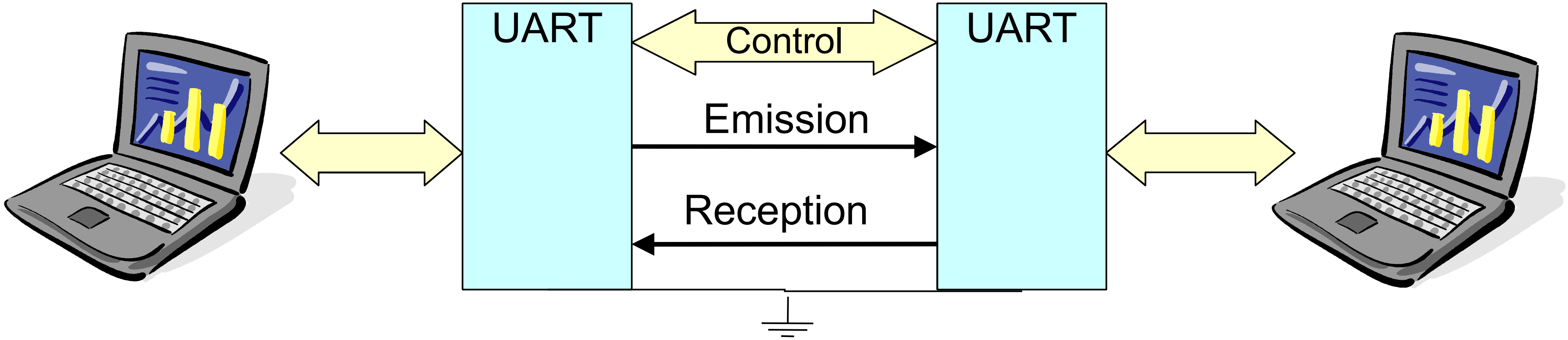
Parity bit

Start bit

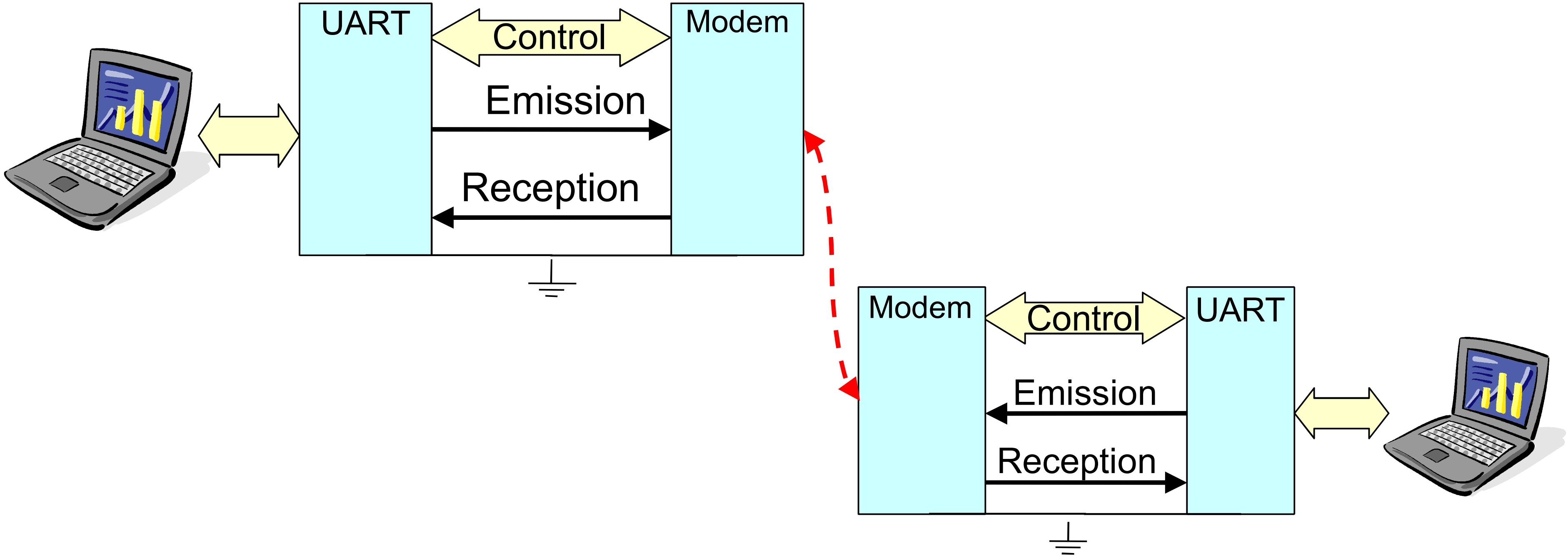
1, 1.5 or 2 stop bits

6.6. Asynchronous serial port (III)

- Direct serial transmission:

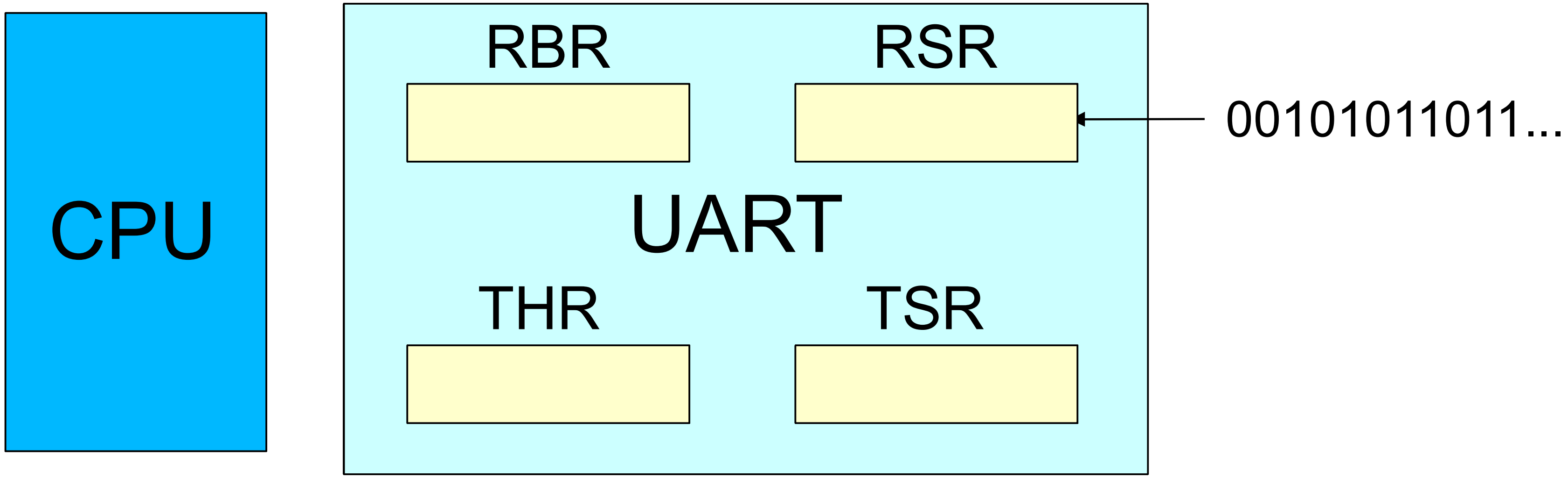


- Serial transmission using phone line:



6.6. Asynchronous serial port (IV)

- Input in direct serial transmission:



RBR (Receiver Buffer Register)

RSR (Receiver Shift Register)

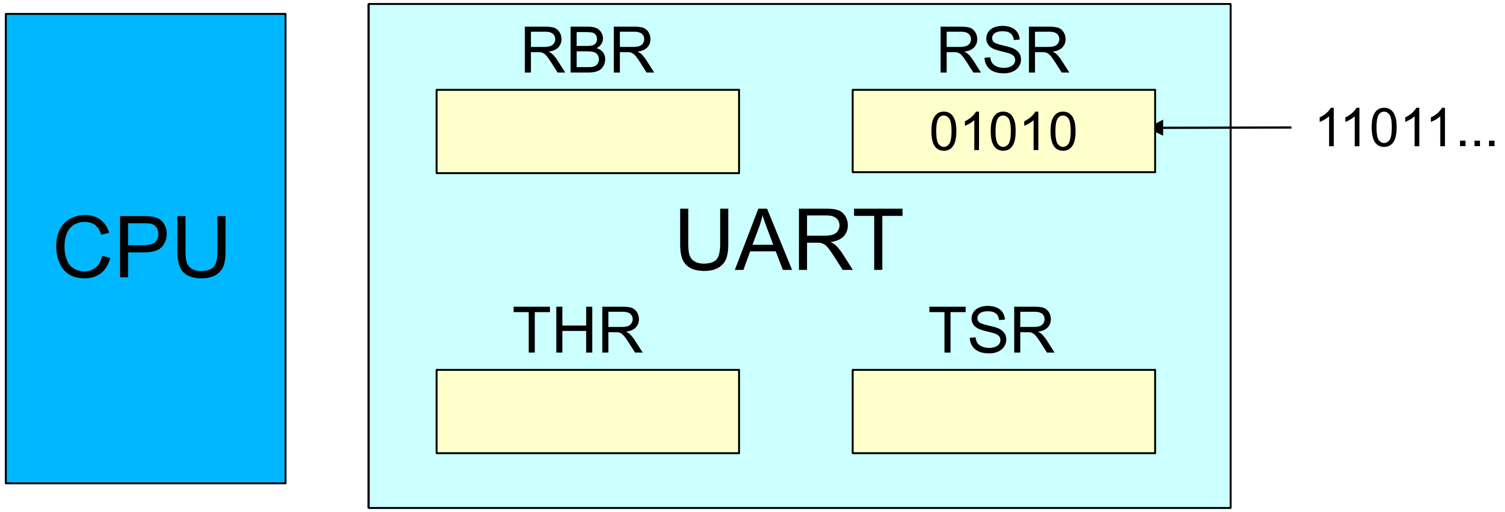
THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)

(6)

6.6. Asynchronous serial port (V)

- The start bit is removed and the remaining bits are loaded into the RSR one after another.
- The stop bits are received.



RBR (Receiver Buffer Register)

RSR (Receiver Shift Register)

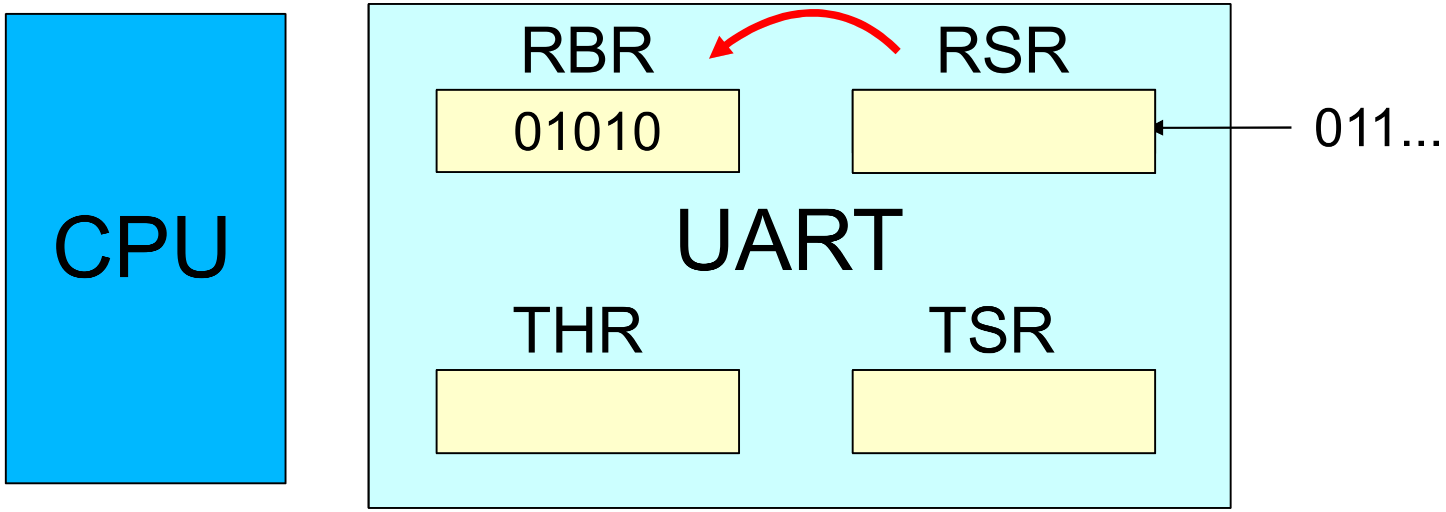
THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)

(6)

6.6. Asynchronous serial port (VI)

- When the RSR has all bits corresponding to the data byte, they are copied to the RBR and the RSR is cleared.



RBR (Receiver Buffer Register)

RSR (Receiver Shift Register)

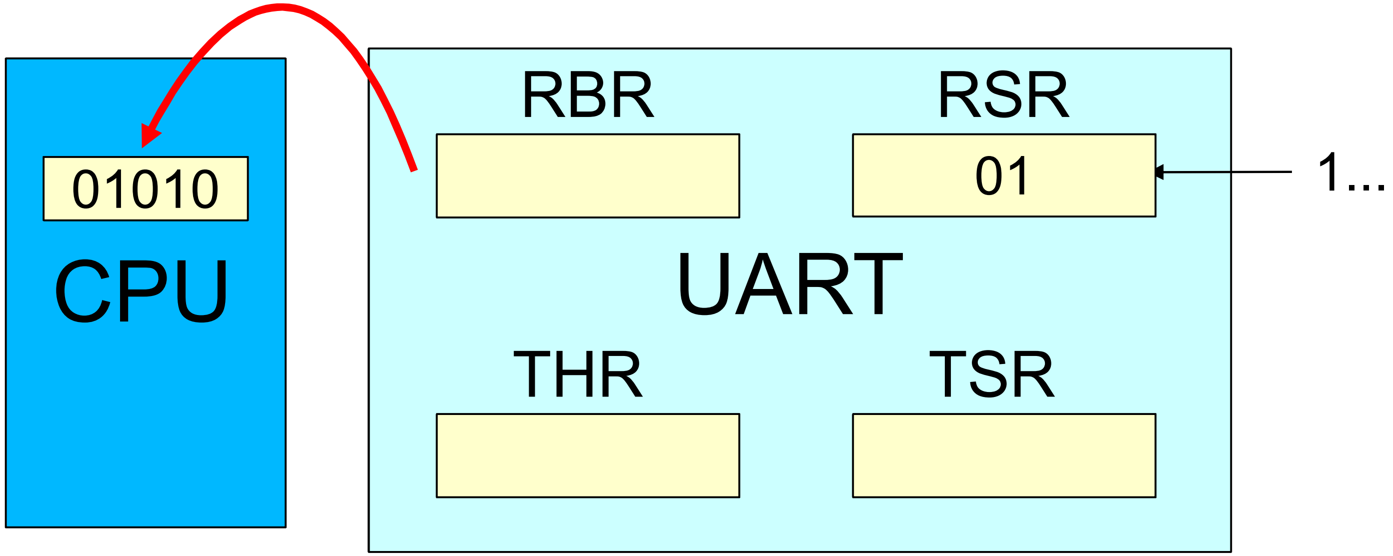
THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)

(6)

6.6. Asynchronous serial port (VII)

- The RSR is empty and keeps accepting new data.
- Data byte is transmitted to the CPU from the RBR and the latter is cleared.



RBR (Receiver Buffer Register)

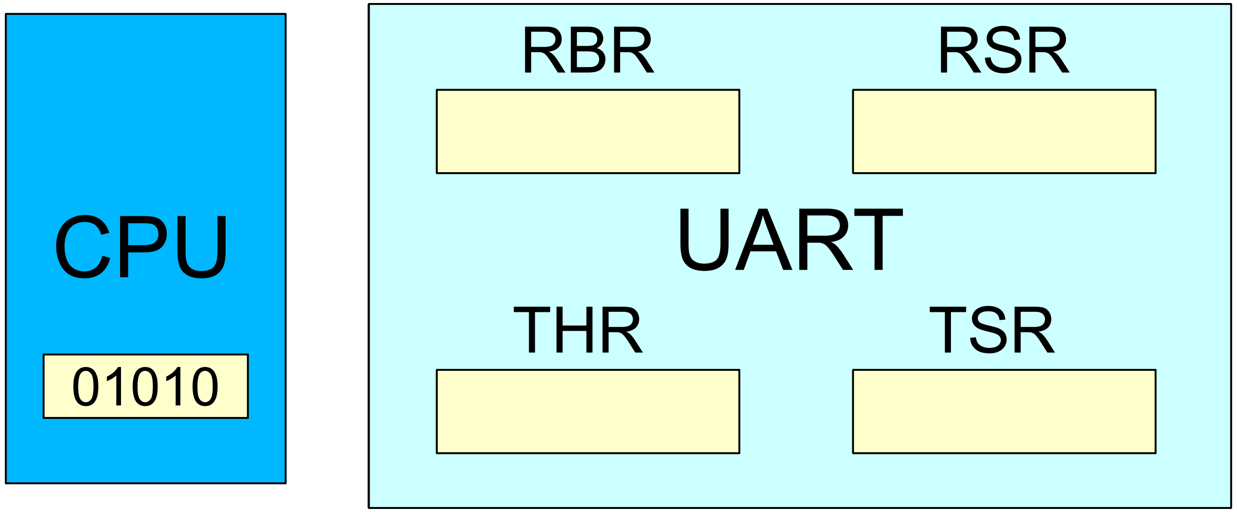
RSR (Receiver Shift Register)

THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)

6.6. Asynchronous serial port (VIII)

- Output in direct serial transmission:



CPU

01010

UART

RBR

RSR

THR

TSR

RBR (Receiver Buffer Register)

RSR (Receiver Shift Register)

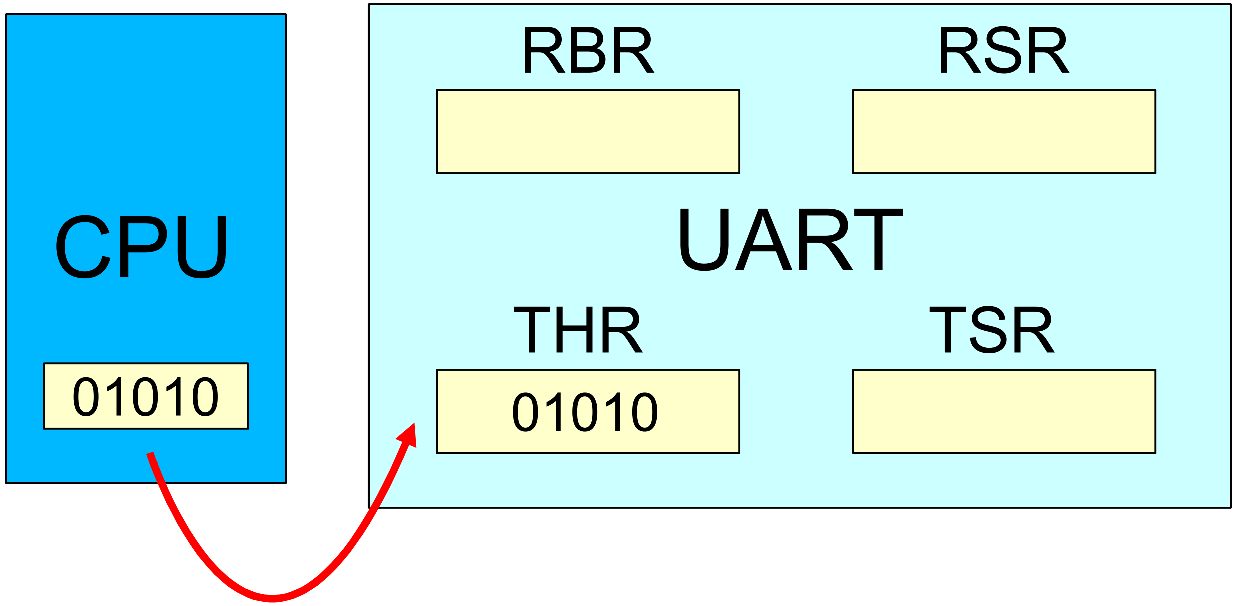
THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)

(6)

6.6. Asynchronous serial port (IX)

- The CPU writes the data byte into the THR.



RBR (Receiver Buffer Register)

RSR (Receiver Shift Register)

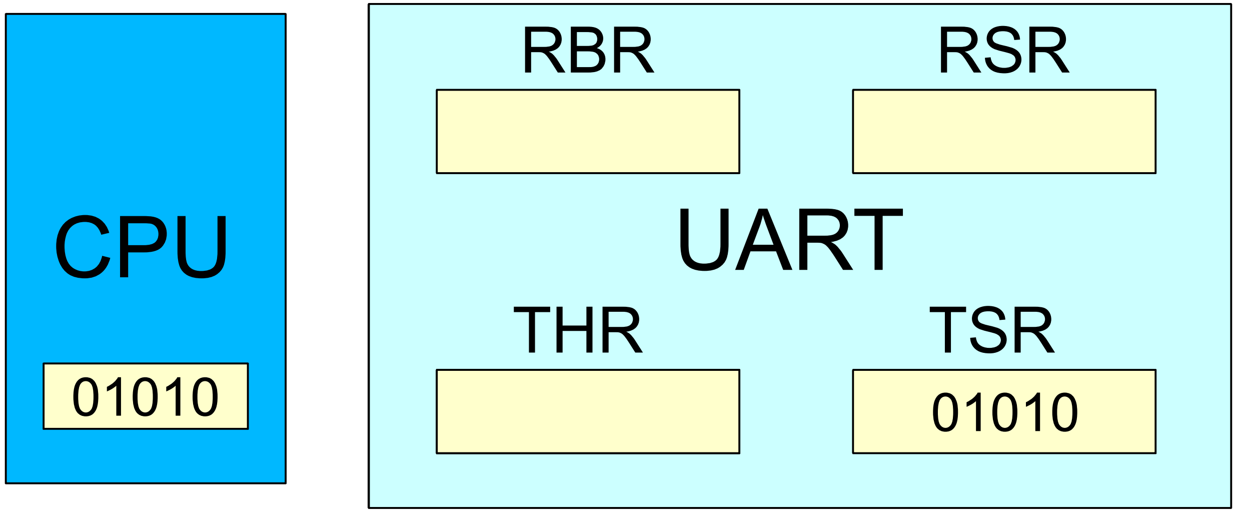
THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)

(6)

6.6. Asynchronous serial port (X)

- The UART automatically moves the data byte to the TSR.



RBR (Receiver Buffer Register)

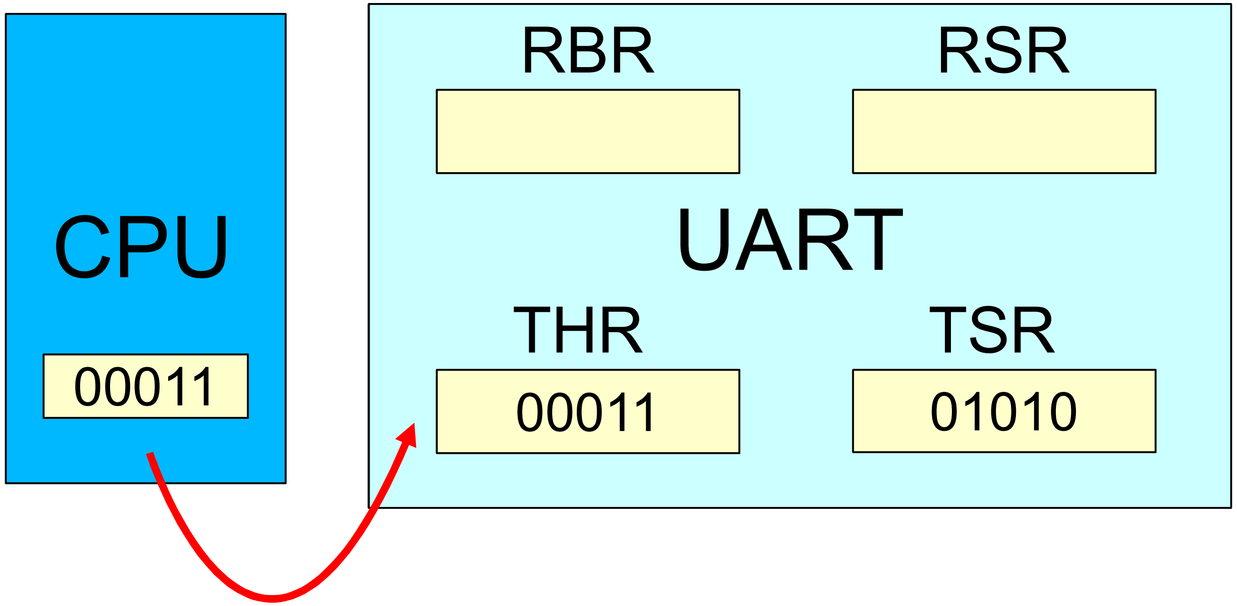
RSR (Receiver Shift Register)

THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)

6.6. Asynchronous serial port (XI)

- The CPU can write another data byte to the THR.

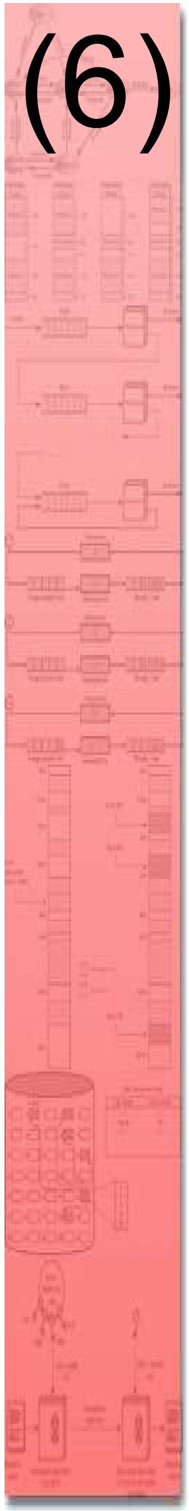


RBR (Receiver Buffer Register)

RSR (Receiver Shift Register)

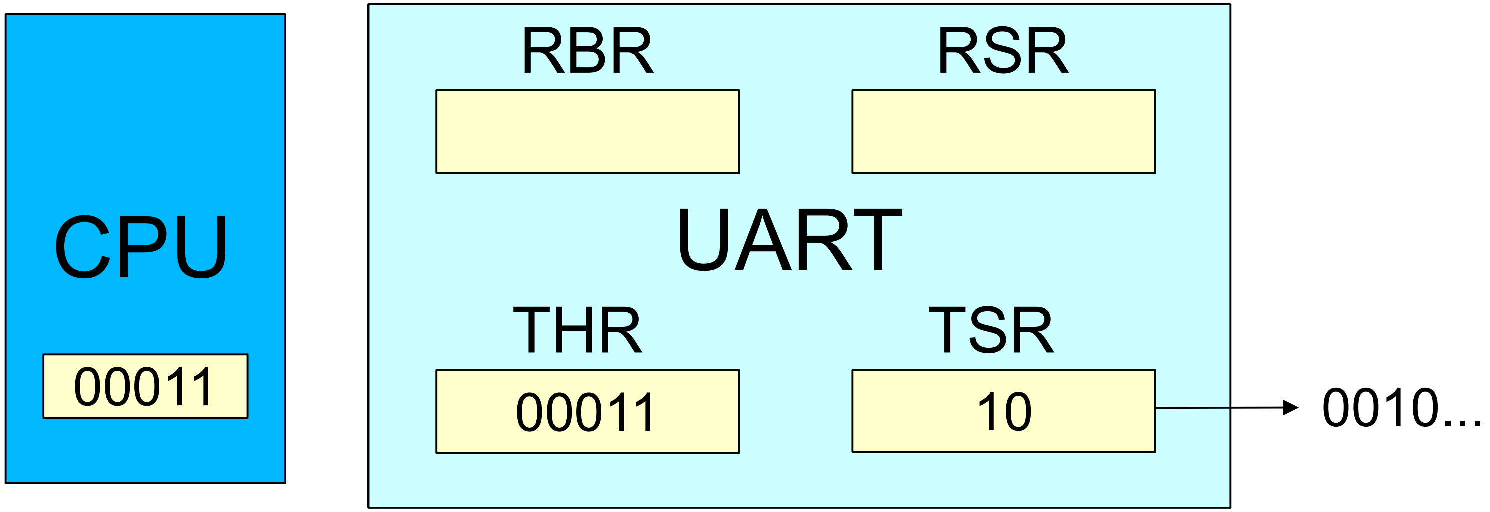
THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)



6.6. Asynchronous serial port (XII)

- The UART sends the start bit.
- It then sends all bits starting with the least significant one.



RBR (Receiver Buffer Register)

RSR (Receiver Shift Register)

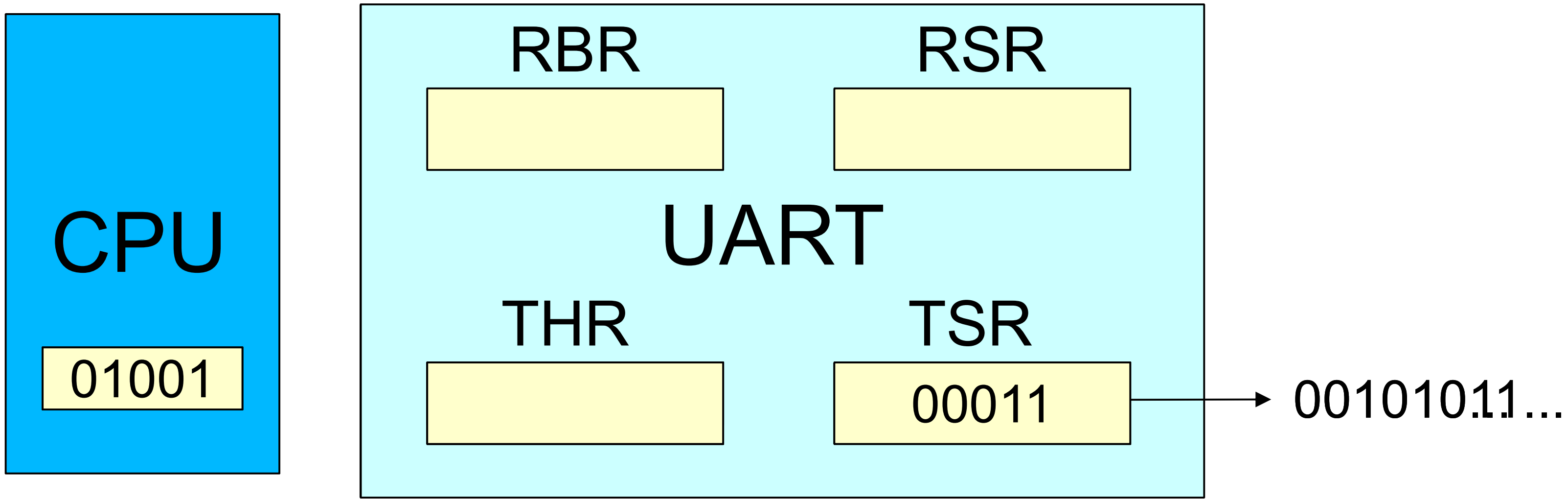
THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)

(6)

6.6. Asynchronous serial port (XIII)

- After sending the data byte, the UART adds the parity bit.
- When the TSR is emptied, the UART writes a new data byte into the TSR.
- The CPU can write into the THR again.



RBR (Receiver Buffer Register)

RSR (Receiver Shift Register)

THR (Transmitter Holding Register)

TSR (Transmitter Shift Register)

6.6. Asynchronous serial port (XIV)

- UART's external communication lines:

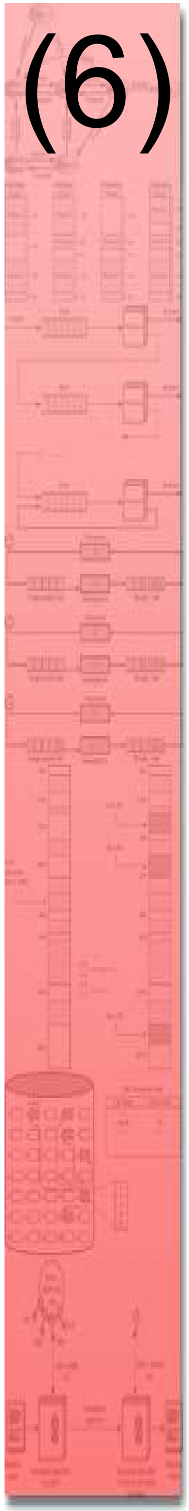
Lines	Meaning
SOUT	Serial transmission line (*)
SIN	Serial reception line (*)
OUT1	Digital output
OUT2	Digital output

- Lines with * have TTL logical levels and it is necessary a chip (buffer) to convert them to EIA levels before leaving through the serial transmission line.
- According to EIA RS232-C norm, a logical 1 is converted to –12V and a logical 0 to +12V.

6.6. Asynchronous serial port (XV)

- UART's external protocol lines:

Lines	Direction	Meaning
DTR	Output	(Data Terminal Ready) The UART tells the modem that it is ready for communication
DSR	Input	(Data Set Ready) The modem tells the UART that it is ready for communication
RTS	Output	(Request To Send) The UART tells the modem that it is ready to send it a data byte
CTS	Input	(Clear To Send) The modem tells the UART that it is ready to send a data byte to the line
RI	Input	(Ring Indicator) Modem's phone is ringing
DCD	Input	(Data Carrier Detect) Remote phone is picked up


(6)

6.6. Asynchronous serial port (XVI)

Registers	A2	A1	A0	DLAB	Action
THR	0	0	0	0	Data output (THR)
RBR	0	0	0	0	Data input (RBR)
DLL	0	0	0	1	Frequency divider (low byte)
IER	0	0	1	0	Enable interrupts
DLH	0	0	1	1	Frequency divider (high byte)
IIR	0	1	0		Interrupt identification
LCR	0	1	1		Line control (parity, data length...)
MCR	1	0	0		Model control (DTR, RTS)
LSR	1	0	1		Line state
MSR	1	1	0		Modem state (DSR, CTS, ...)
SCR	1	1	1		Free use (“Scratchpad”)

- Base addresses of serial port defined by BIOS:

  COM1: 0000h:0400h and 0000h:0401h

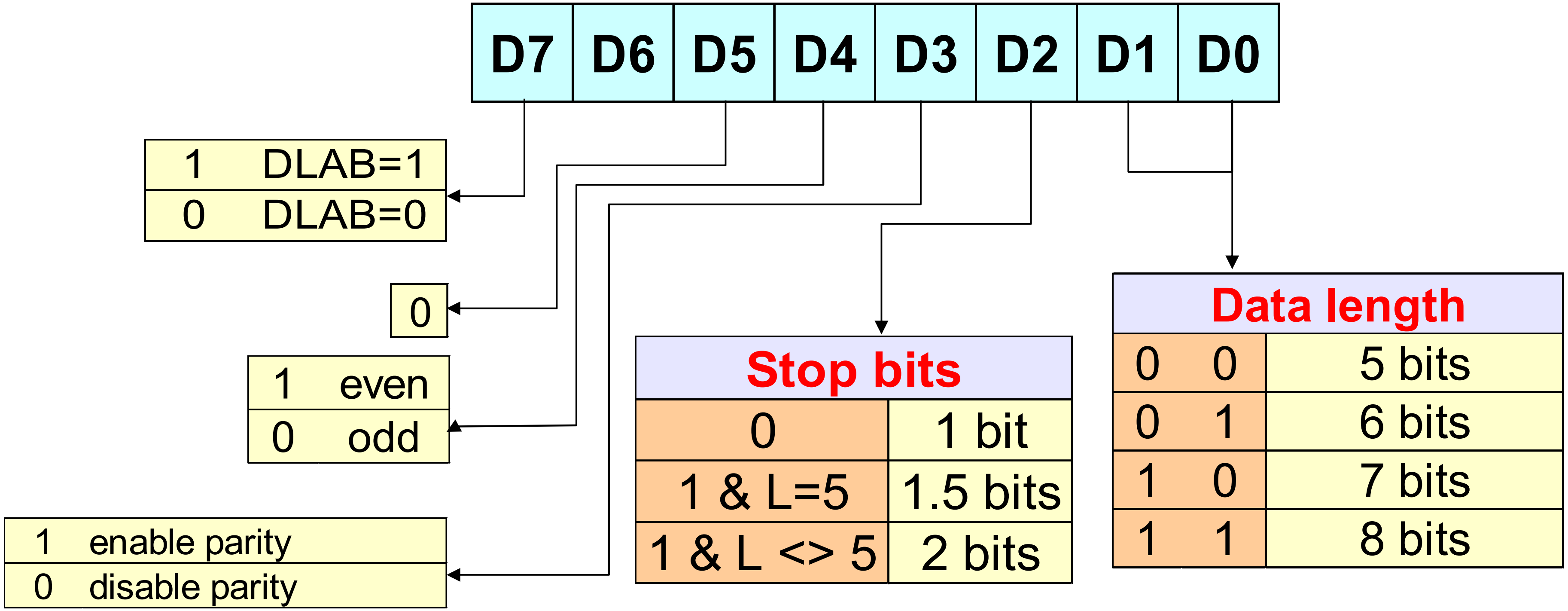
  COM2: 0000h:0402h and 0000h:0403h

  COM3: 0000h:0404h and 0000h:0405h

  COM4: 0000h:0406h and 0000h:0407h

6.6. Asynchronous serial port (XVII)

- Line Control Register (LCR): Base address + 3
  - Write only.



D7	D6	D5	D4	D3	D2	D1	D0

D7:

1	DLAB=1
0	DLAB=0

D6: 0

D4:

1	even
0	odd

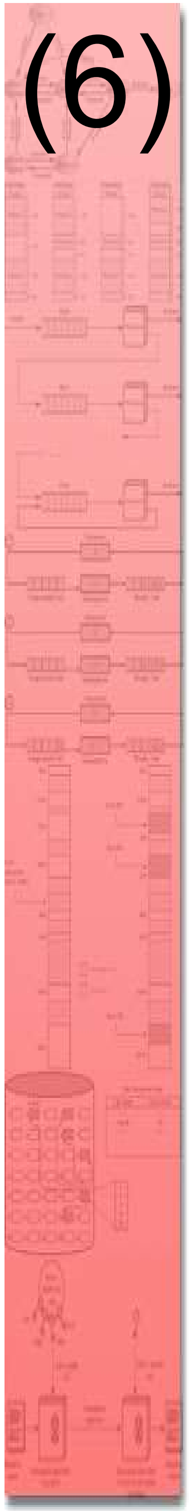
D3:

1	enable parity
0	disable parity

Stop bits	
0	1 bit
1 & L=5	1.5 bits
1 & L <> 5	2 bits

Data length		
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

  - When D6 is set to 1, a BREAK condition is generated in the output line SOUT, which consists in setting SOUT to 0 independently of the current transmission state. This change can be detected by the receiver.



6.6. Asynchronous serial port (XVIII)

- Frequency divider registers (DLL, DLH): Base address and Base address + 1 (DLAB=1)
  - Write only.
  - They divide the transmission frequency.
  - The transmission speed in bps is:

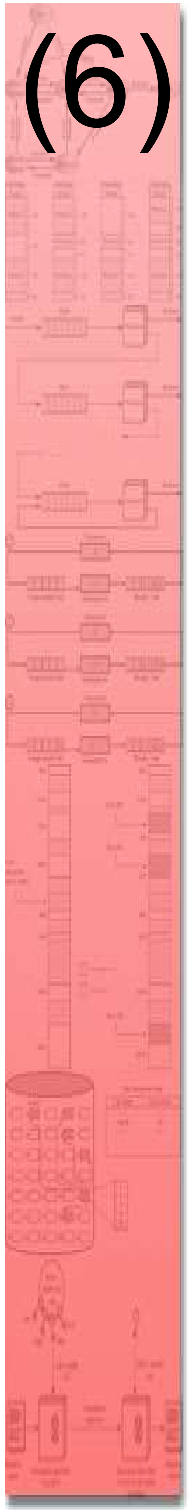
$$v = \frac{1843200}{16\left(256\ DLH + DLL\right)}$$

$$DLH = \left(1843200 / 16v\right)\ \text{div}\ 256$$

$$DLL = \left(1843200 / 16v\right)\ \text{mod}\ 256$$

  - Example: DLH = 6 and DLL = 0

$$v = \frac{1843200}{16\ \left(256 \times 6 + 0\right)} = 75\ \text{bps}$$

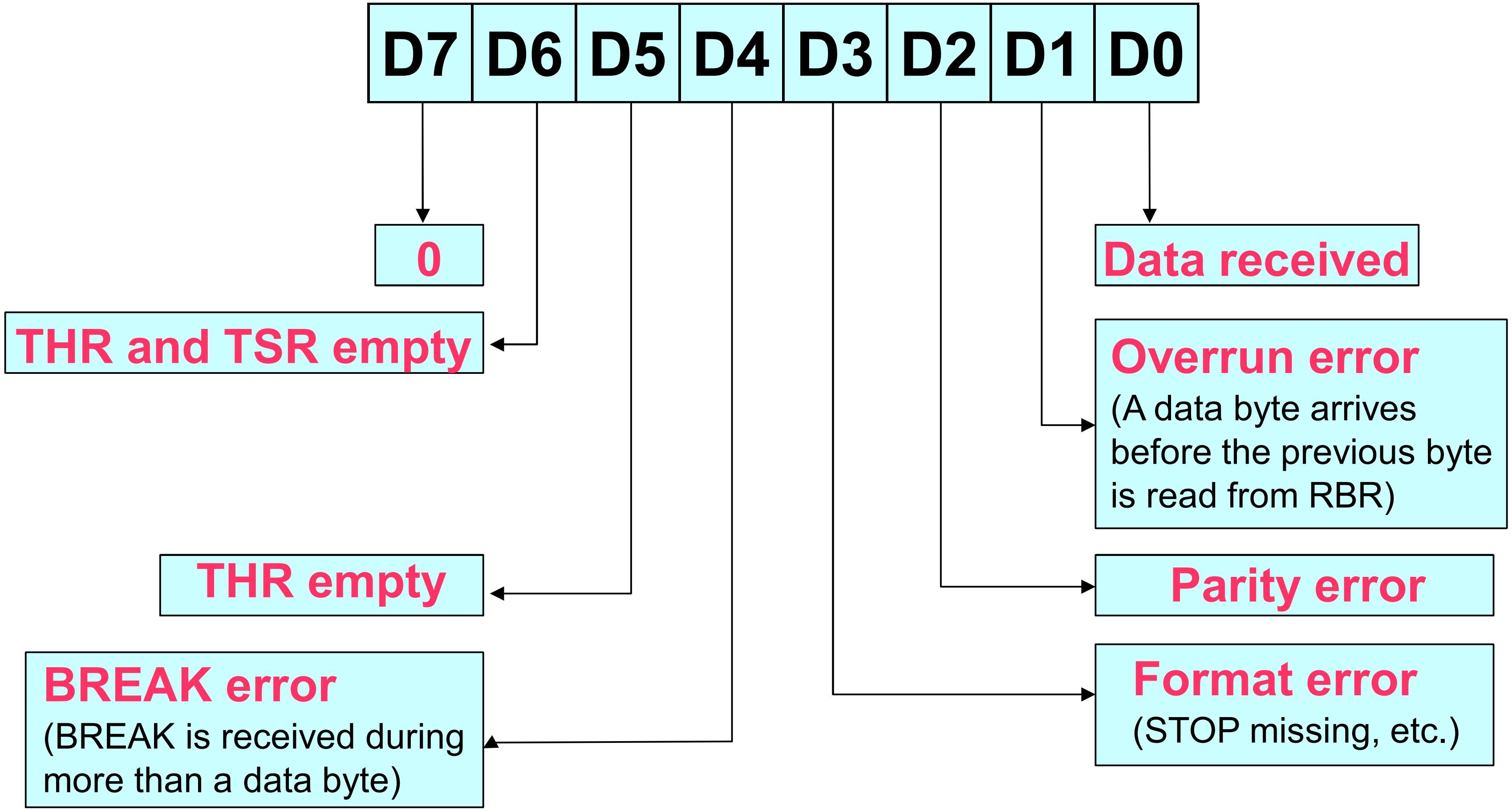


6.6. Asynchronous serial port (XIX)

- Data read register (RBR) and write register (THR): Base address (DLAB=0).
  - RBR when read.
  - THR when written.

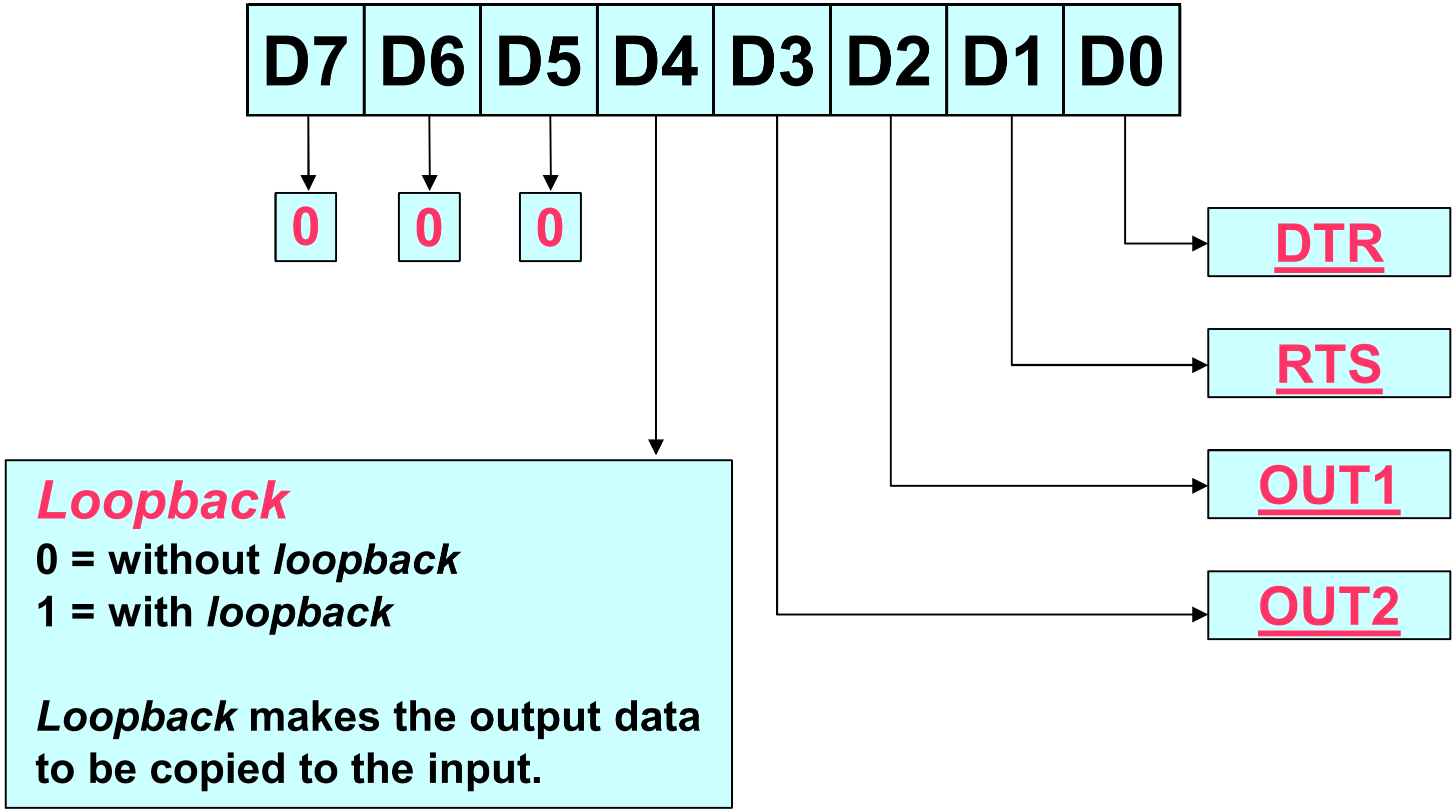
6.6. Asynchronous serial port (XX)

- Line State Register (LSR): Base address + 5
  - Read only.



6.6. Asynchronous serial port (XXI)

- Modem Control Register (MCR): Base address + 4.
  - Write only.

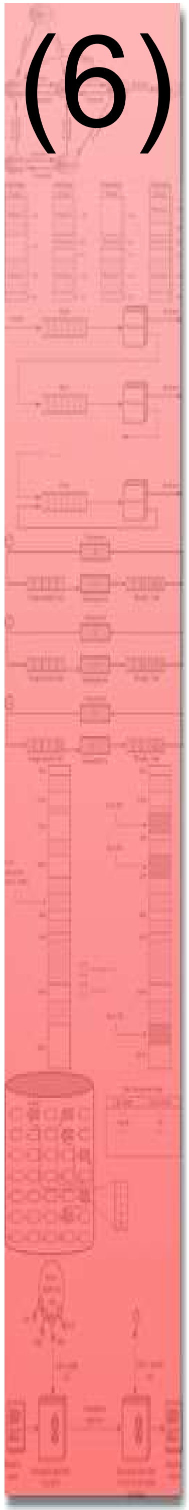


Loopback

0 = without *loopback*

1 = with *loopback*

***Loopback* makes the output data to be copied to the input.**



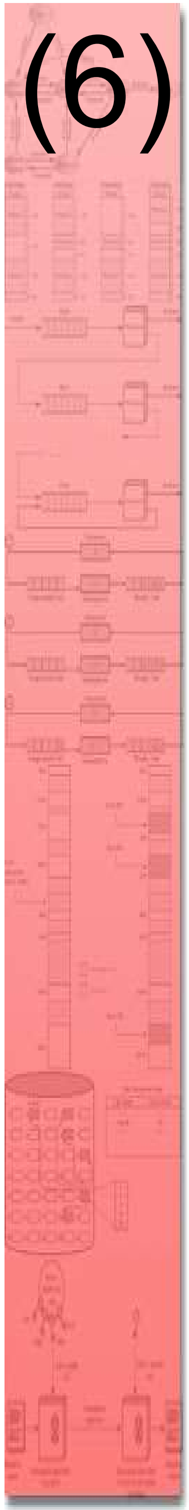
6.6. Asynchronous serial port (XXII)

- Modem State Register (MSR): Base address + 6.
  - Read only.

D7	D6	D5	D4	D3	D2	D1	D0
DCD	RI	DSR	CTS	DDCD	DRI	DDSR	DCTS

- Inverted logical state of terminals.

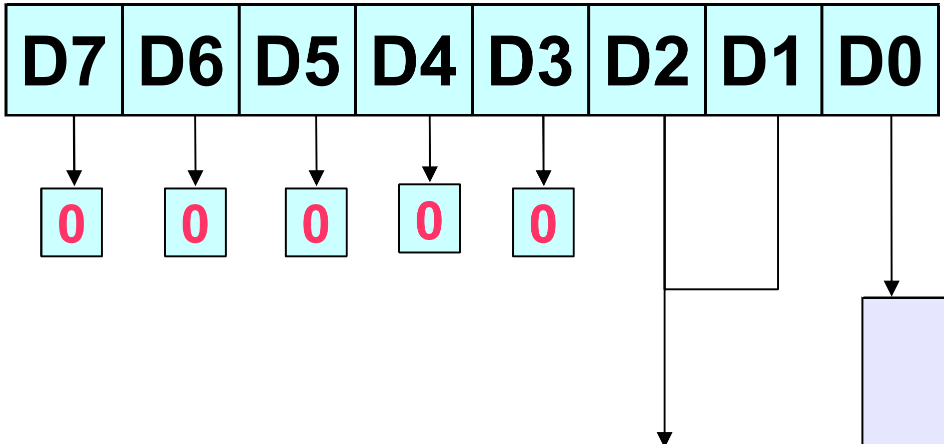
- They inform about a change of state since the last time MSR was read.
  - When MSR is read, the bits are set to 0 and they only indicate transitions from 0 to 1.



(6)

6.6. Asynchronous serial port (XXIII)

- Interrupt Identification Register (IIR): Base address + 2
  - Read only.

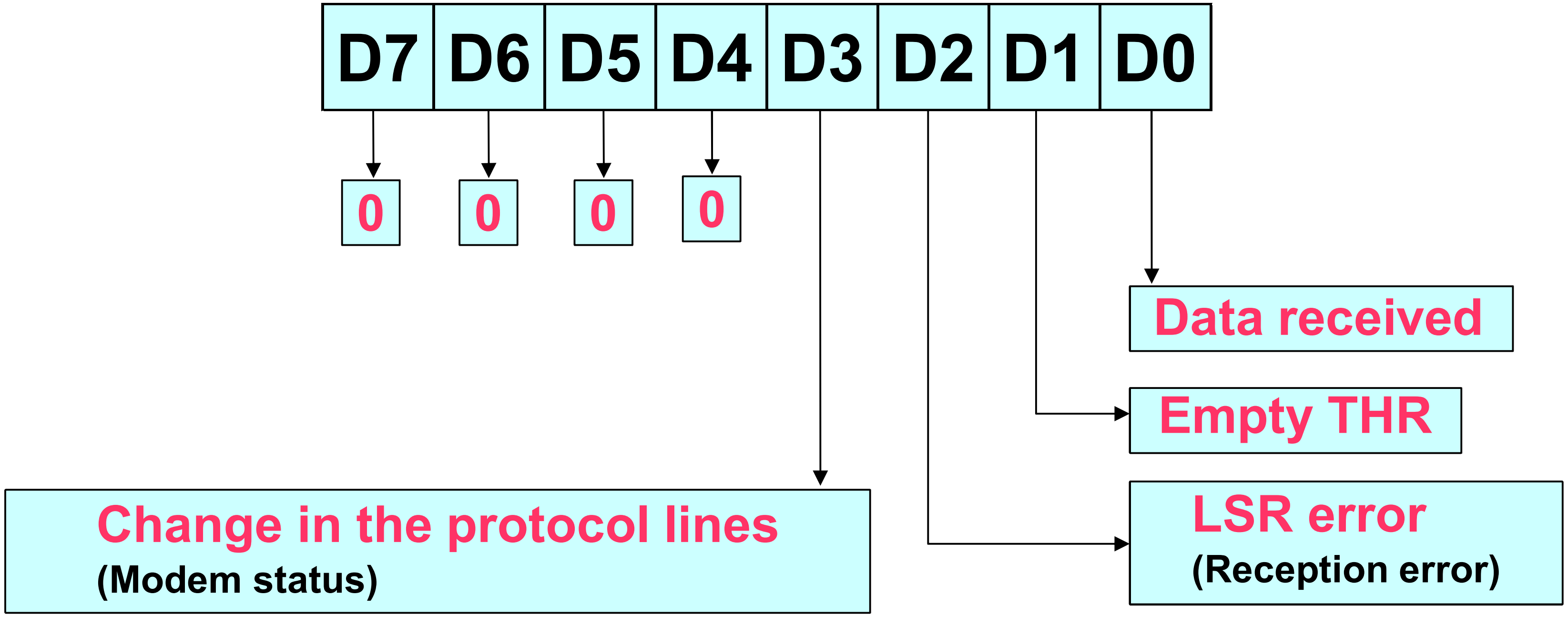


Interrupt cause	
1 1	Error in LSR
1 0	Data received
0 1	Empty THR
0 0	Detected CRS, DSR, RI or DCD

Pending interrupt	
1	Yes
0	No

6.6. Asynchronous serial port (XXIV)

- Interrupt Enable Register (IER): Base address + 1 (DLAB=0).
  - Write only.
  - When set to 1, they enable the corresponding interrupt. Otherwise, the interrupt is disabled.


D7 D6 D5 D4 D3 D2 D1 D0

0 0 0 0

Data received

Empty THR

LSR error
(Reception error)

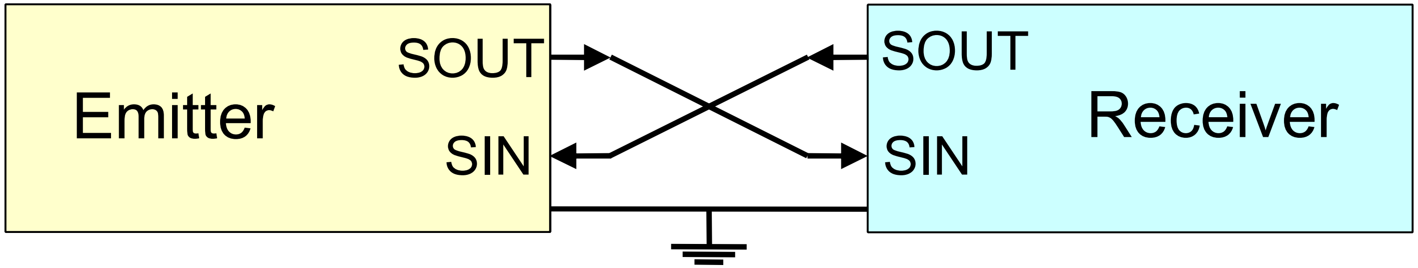
Change in the protocol lines
(Modem status)

6.6. Asynchronous serial port (XXV)

- Scratch Register (SCR): Base address + 7.
  - Read and write register.
  - It does not affect the UART.
  - Useful for storing some data byte of interest to the programmer.

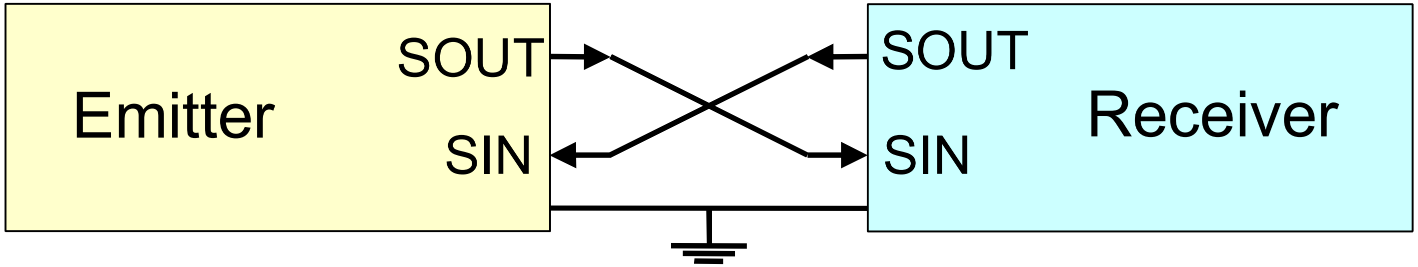
6.6. Asynchronous serial port (XXVI)

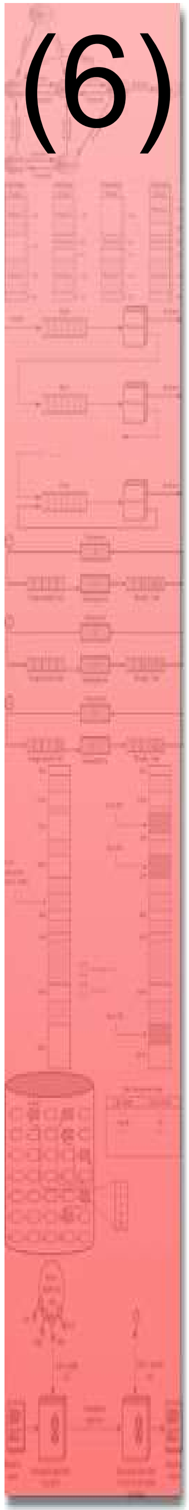
- XON/XOFF protocol
  - Flux control used when the transmission is unidirectional (e.g., between a computer and a serial printer).
  - Control lines (DTR, RTS, …) are not used.
  - Simplest serial connection (3 lines):



6.6. Asynchronous serial port (XXVII)

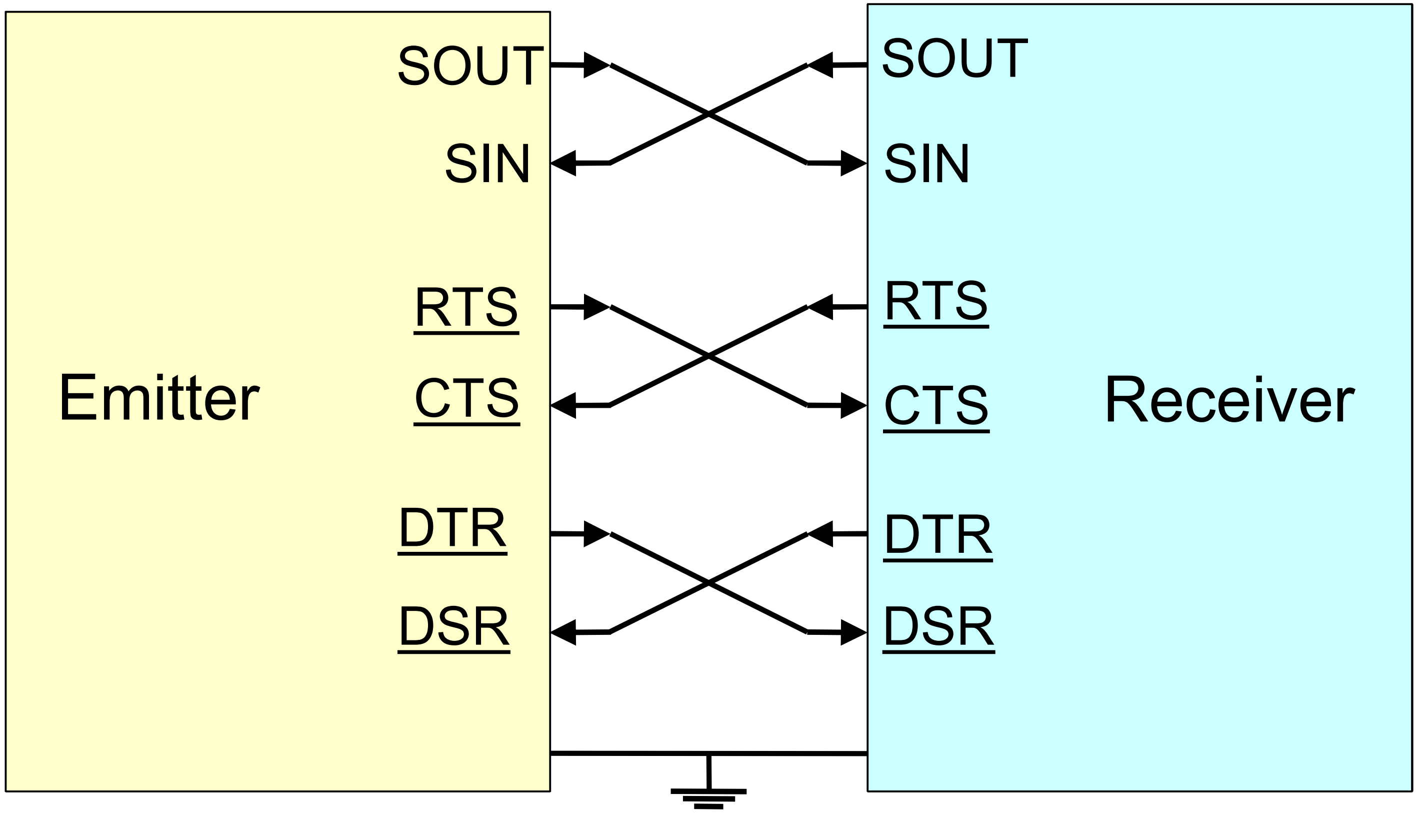
- XON/XOFF protocol
  - Emitter sends data.
  - If receiver cannot accept data at the cadence imposed by the emitter, it sends Control-S (XOFF character) to the emitter.
  - Emitter stops sending data.
  - When receiver is ready to receive data, it sends Control-Q (XON character).
  - Emitter starts sending again.
- Before writing a data byte into the THR, the emitter must:
  - Verify that it has not received a Control-S.
  - Verify that THR is empty.





6.6. Asynchronous serial port (XXVIII)

- Protocol using control lines.
  - Transmission can be Simplex (unidirectional) or Duplex (bidirectional)

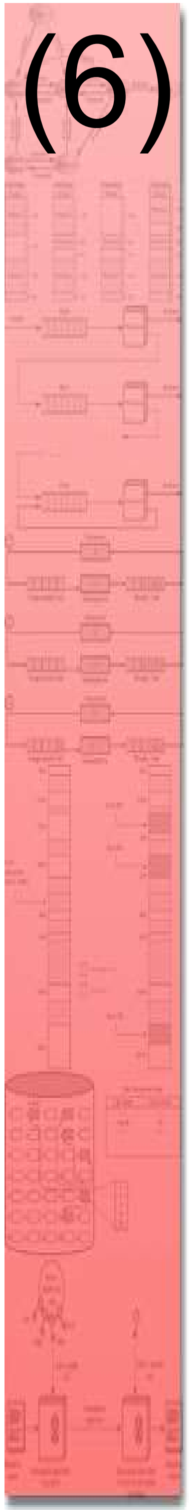


NULL modem connection

6.6. Asynchronous serial port (XXIX)

Simplex protocol

- Emitter
  - DTR is set to 0 indicating it is ready to communicate.
  - RTS is set to 0 indicating it is ready to send a byte.
  - It verifies that both DSR and CTS are 0 (they must be set by the receiver).
  - It sends the byte.
- Receiver
  - DTR is set to 0 indicating it is ready to communicate.
  - RTS is set to 0 indicating it is ready to receive a byte.
  - If the receiver wants to hold off the emission, it sets RTS to 1. For resuming the emission, RTS is set to 0.



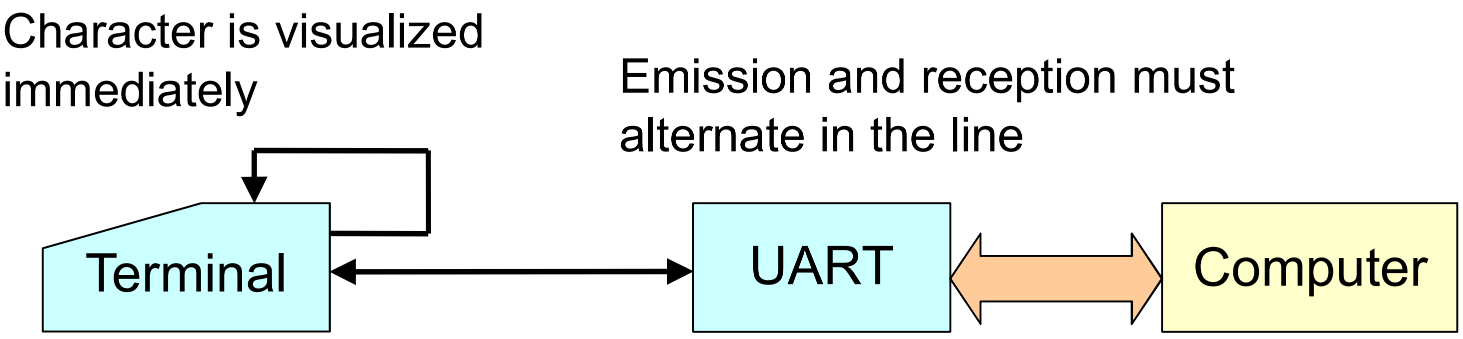
6.6. Asynchronous serial port (XXX)

Duplex protocol

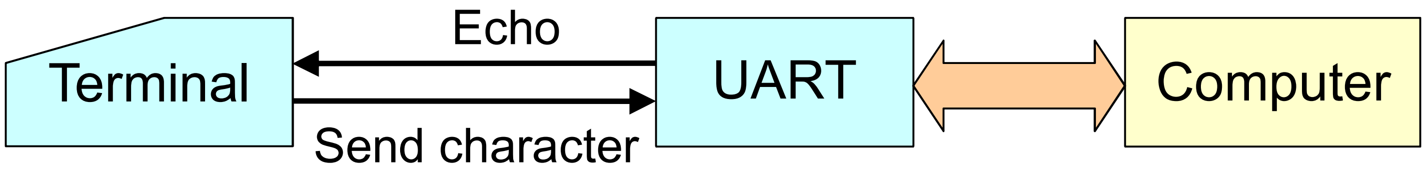
- Both devices act as emitters / receivers
  - Both set DTR to 0 indicating they are ready to communicate.
  - Both set RTS to 0 indicating they are ready to send data.
  - Both must verify that both DTS and CTS are 0.
  - If any of them wants to hold off the emission, it will set RTS to 1. To resume the emission, it will set RTS to 0.

6.6. Asynchronous serial port (XXXI)

- Half Duplex (alternated emission and reception)


Character is visualized immediately
Emission and reception must alternate in the line
Terminal
UART
Computer

- Full Duplex (simultaneous emission and reception)


Echo
Terminal
UART
Computer
Send character