

Proiectarea cu Microprocesoare

Curs 6

Interfete pentru comunicatie seriala

An 3 CTI

An universitar 2014/2015

Semestrul I

Lector: Radu Dănescu

Dispozitive de Comunicare ATmega64

Serial Peripheral Interface (SPI)

- Comunicare seriala sincrona
- Mod de functionare full duplex
- Configurare Master sau Slave
- Frecventa variabila
- Se poate folosi pentru conexiune intre placi, sau intre placa si diferite module PMOD (ex. DAC extern)

Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART)

- Comunicare seriala sincrona sau asincrona
- Frecventa (baud rate) variabila
- Suporta pachete de date de 5-9 biti, cu sau fara paritate
- Suporta intreruperi pentru controlul transmisiei
- Detectia erorilor de transmisie
- Comunicare intre Cerebot si PC (portul serial)

Dispozitive de Comunicare ATmega64

Two Wire Serial Interface (TWI)

- Protocol de comunicare complex, folosind doar doua fire (clock si data)
- Implementare Atmel a protocolului I2C (Inter Integrated Circuit)
- Controllerul TWI integrat in ATmega64 suporta moduri master si slave
- Adresare pe 7 biti
- Suport pentru arbitrare pentru mai multe dispozitive master
- Adresa slave programabila

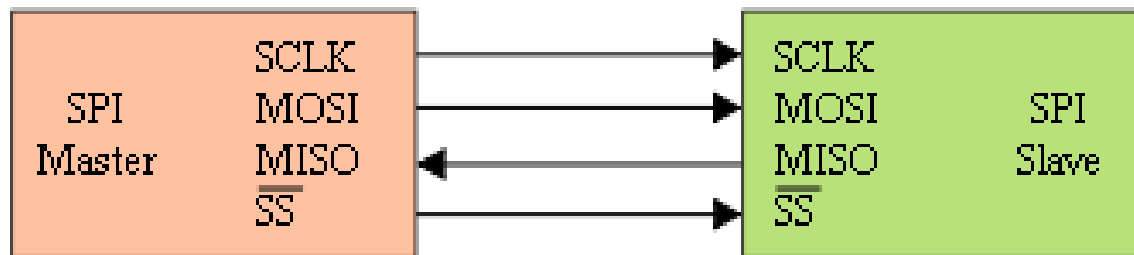
Serial Peripheral Interface (SPI)

Semnale

- SCLK – Serial clock, generat de Master
- MOSI – Master Output, Slave Input, date transmise de Master
- MISO – Master Input, Slave Output, date receptionate de Master
- SS – Slave select – activarea dispozitivului Slave de catre Master, activ pe zero

Functionare

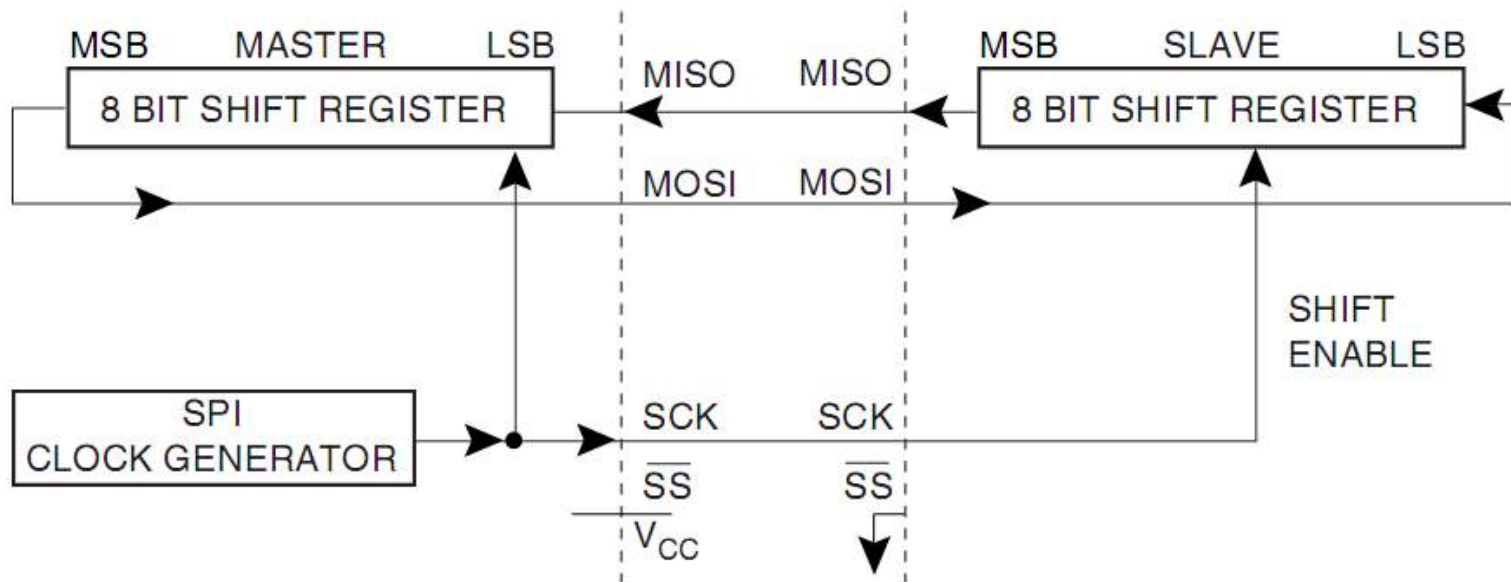
- Master initiaza comunicatia prin activarea SS
- Master genereaza semnalul de ceas SCLK
- Pe fiecare perioada de ceas un bit se transmite de la master la slave, si un bit de la slave la master
- Dupa fiecare pachet de date (8, 16 biti,...) SS este dezactivat, pentru sincronizarea transmisiei



Serial Peripheral Interface (SPI)

Principiul de functionare

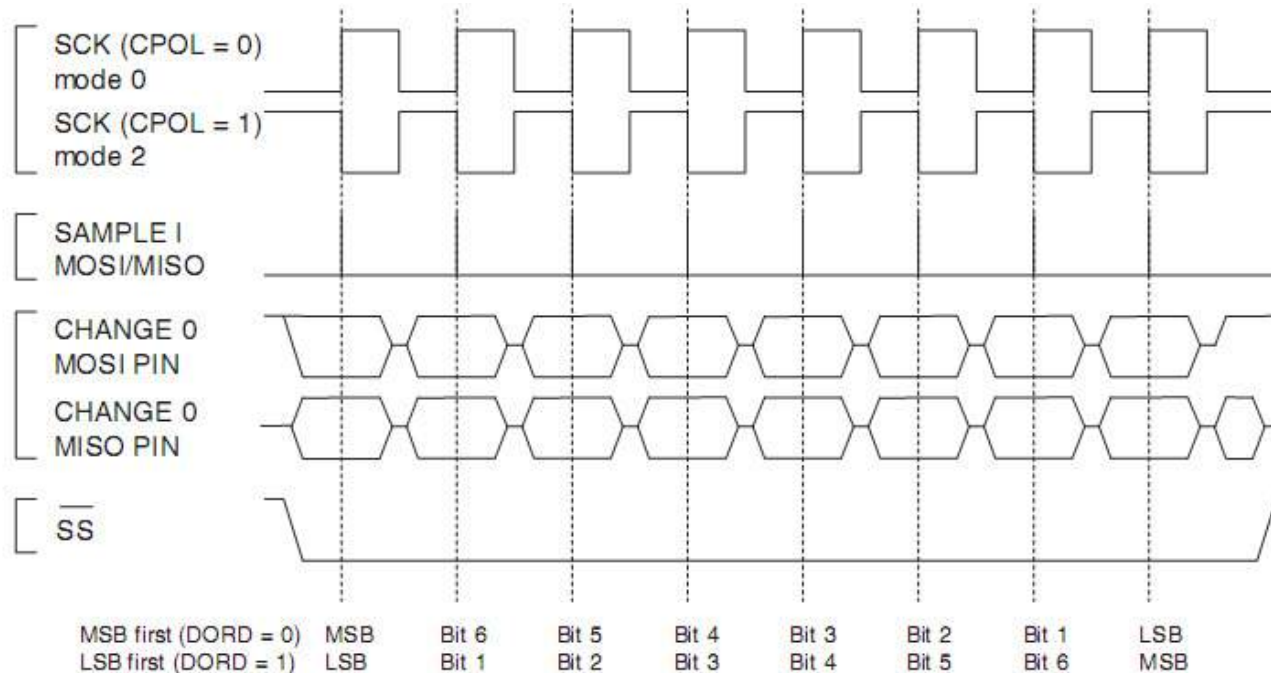
- Ambii parteneri au cate un registru de deplasare intern, iesirile si intrarile fiind conectate prin MISO/MOSI
- Ambii registri au acelasi ceas, SCLK
- Cei doi registri formeaza impreuna un registru de rotatie
- Dupa un numar de perioade de ceas egal cu dimensiunea unui registru, Master si Slave fac schimb de date



Serial Peripheral Interface (SPI)

Sincronizarea datelor cu semnalul de ceas

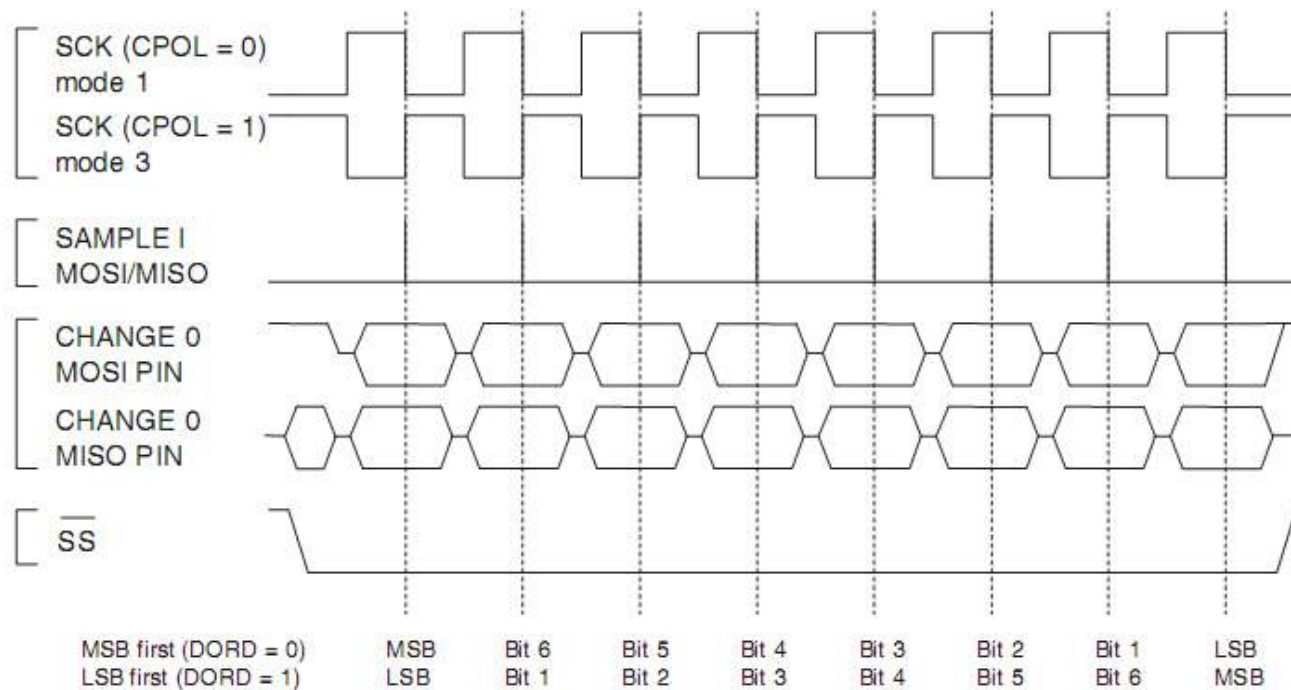
- Deplasarea (shiftare) datelor si preluarea lor se fac pe fronturi opuse
- CPOL – clock polarity – primul front e crescator sau descrescator
- CPHA – clock phase
- Pentru CPHA = 0
 - Pe primul front se face preluarea datelor
 - Pe al doilea front se face stabilizarea (deplasarea)



Serial Peripheral Interface (SPI)

Sincronizarea datelor cu semnalul de ceas

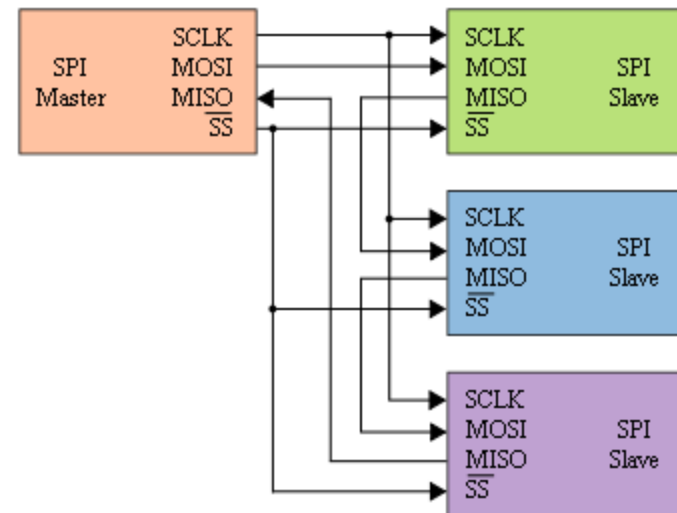
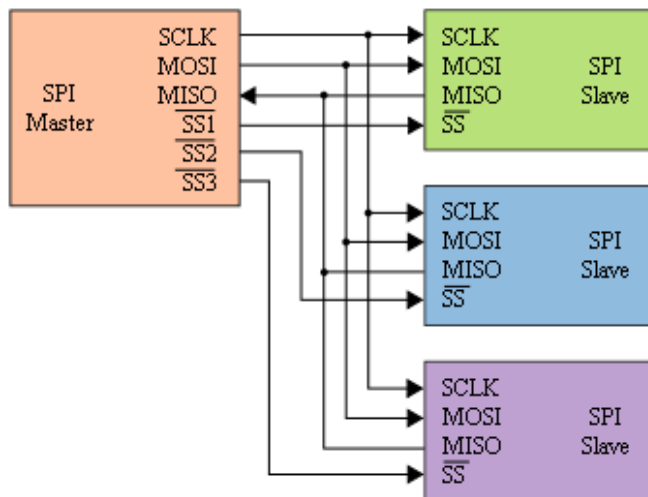
- Pentru $CPHA = 1$
 - Pe primul front se face deplasarea
 - Pe al doilea front se face preluarea datelor



Serial Peripheral Interface (SPI)

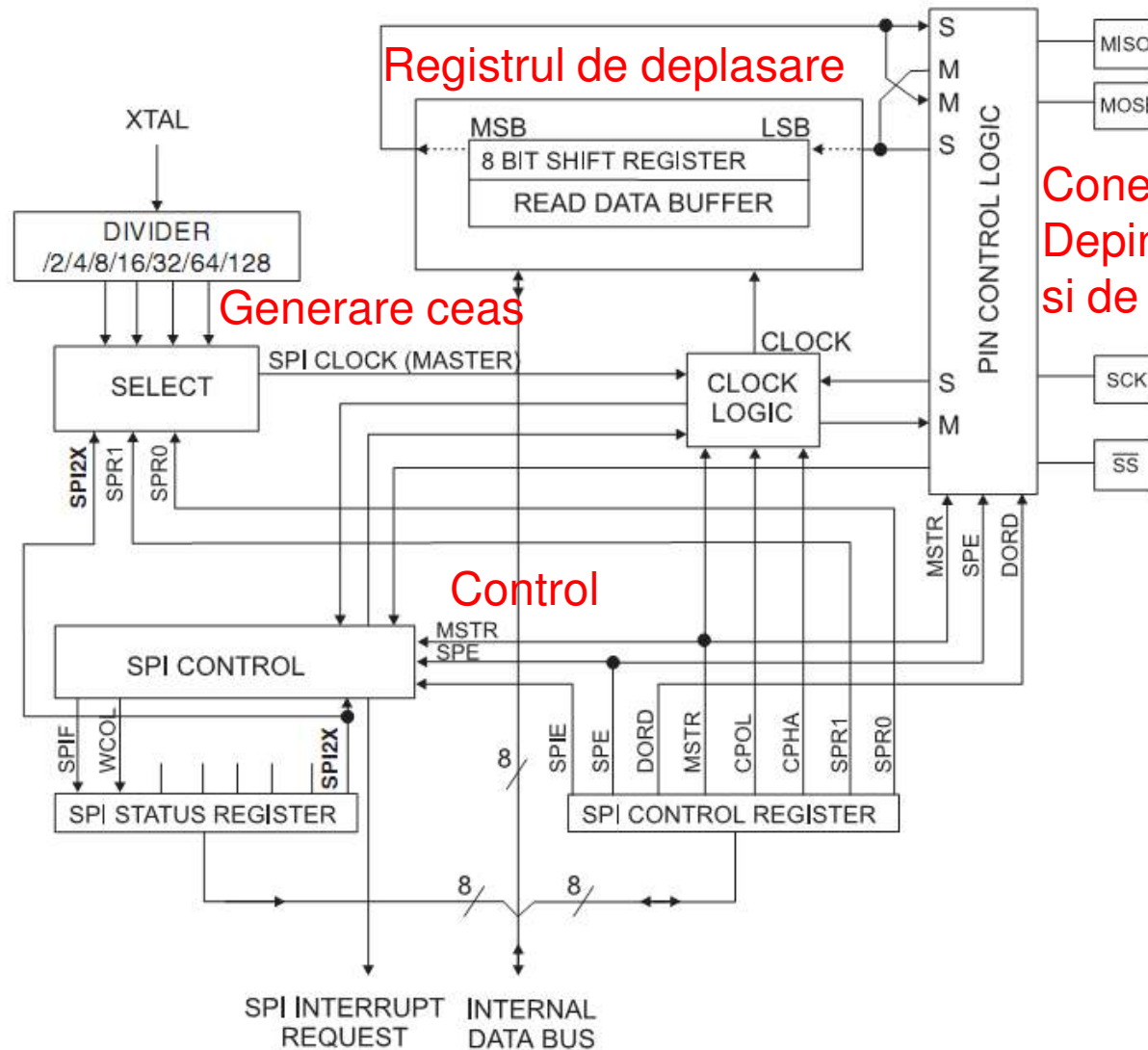
Utilizarea semnalului SS

- Pentru un dispozitiv “slave” **SS** este semnal de intrare
 - SS cu valoare 0 inseamna activarea dispozitivului slave. O tranzitie din 0 in 1 inseamna resetarea ciclului de transfer (marcheaza sfarsitul unui pachet)
 - SS cu valoare 1 – dispozitiv slave inactiv
- Pentru un dispozitiv “master” **SS** poate fi:
 - lesire – prin el activeaza dispozitivul “slave” pentru comunicare
 - Intrare – daca se permit mai multe dispozitive master, o valoare ‘0’ la intrarea SS trece dispozitivul curent in modul “Slave”
- Configuratii cu mai multe dispozitive: - semnale **SS** independente sau “daisy chain”



SPI la ATmega64

Arhitectura sub-sistemului SPI



Registrul de deplasare

Generare ceas

Control

Conectare pini
Depinde de mod (M/S)
si de ordinea datelor

SPI la ATmega64

Configurare SPI

Bit	7	6	5	4	3	2	1	0	
0x0D (0x2D)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Registrul **SPCR**:

SPIE – SPI Interrupt Enable, generare intrerupere la terminarea transmisiei

SPE – SPI Enable. Trebuie setat 1 pentru orice operatie cu SPI

DORD – Data Order. 1=LSB first, 0 = MSB first

MSTR – Master, daca e 1, slave daca e 0

CPOL, CPHA – selecteaza polaritatea si faza semnalului SCLK

	Leading Edge	Trailing Edge
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)

SPR1, SPR0 – regleaza viteza SPI impreuna cu SPI2X din registrul SPSR

SPI la ATmega64

Configurare SPI - Continuare

Bit	7	6	5	4	3	2	1	0	
0x0E (0x2E)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Registrul **SPSR**:
SPI2X – Reglare frecventa ceas, impreuna cu SPR1 si SPR0 din SPCR

WCOL – Write collision – setat daca scriem in **SPDR** inainte SPI sa transfere datele
SPIF – SPI Interrupt flag – setat cand se termina transmisia. Daca SPIE este setat, se genereaza cerere de intrerupere

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

SPI la ATmega64

Utilizare SPI (Master)

1. Configurare directie pini I/O:
Pinii SPI sunt comuni cu pinii portului B

Port Pin	Alternate Functions
PB7	OC2/OC1C ⁽¹⁾ (Output Compare and PWM Output for Timer/Counter2 or Output Compare and PWM Output C for Timer/Counter1)
PB6	OC1B (Output Compare and PWM Output B for Timer/Counter1)
PB5	OC1A (Output Compare and PWM Output A for Timer/Counter1)
PB4	OC0 (Output Compare and PWM Output for Timer/Counter0)
PB3	MISO (SPI Bus Master Input/Slave Output)
PB2	MOSI (SPI Bus Master Output/Slave Input)
PB1	SCK (SPI Bus Serial Clock)
PB0	\overline{SS} (SPI Slave Select input)

2. Configurare – scriere SPCR si SPSR cu valorile corespunzatoare pentru modul de lucru
3. Activare SS (PB(0) <- '0', **explicit!**)
4. Scriere date in SPDR – declanseaza transmisia
5. Asteptare pana SPIF din SPSR este setat – transmisie completa
6. Citire date din SPDR – datele trimise de 'slave'
7. Dezactivare SS (PB(0) <- '1')

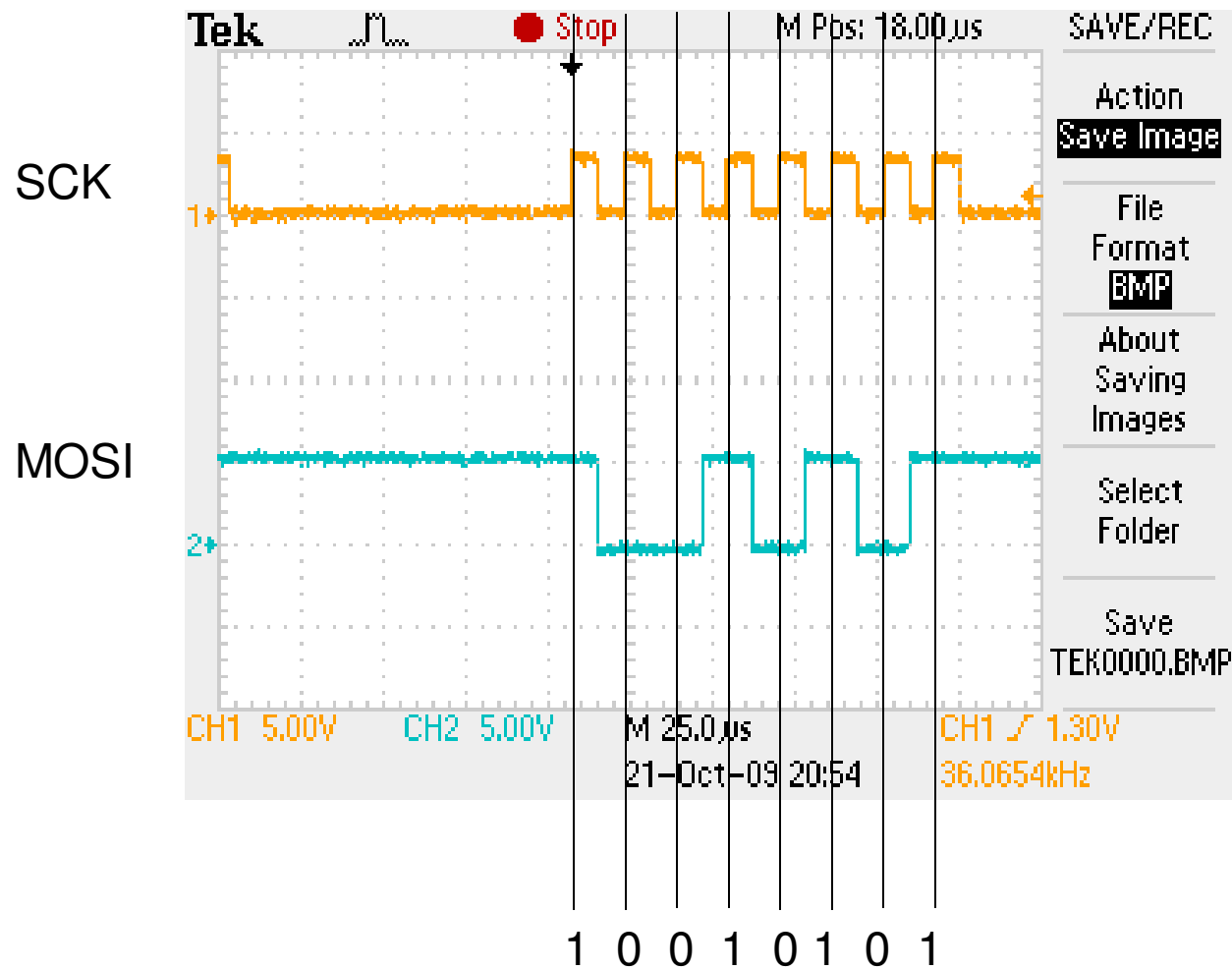
SPI la ATmega64

Utilizare SPI (Master) – Cod sursa

```
.org 0x0000
jmp reset
reset:
ldi r16,0b00000111      ;MISO intrare, MOSI, SCK si SS iesire
out DDRB,r16
ldi r16, 0b00000001      ;Initial, SS<--1, SPI Slave inactiv
out PORTB, r16
cbi SPSR, 0              ; pune bitul zero din SPSR pe zero - pentru frecventa
ldi r16,0b01010011      ;Intreruperi dezactivate, SPI Enabled, LSB first, Master, CPOL=0 – prim front
                           ;crescator, CPHA = 0 – preluare pe primul front, Frecventa cea mai lenta
out SPCR,r16
loop:
  cbi PORTB, 0            ; SS <- 0
  ldi r16, 0b10010101    ; datele de transmis
  out SPDR, r16
wait:
  sbis SPSR, 7            ; bitul 7 din SPSR - transmisie completa
  rjmp wait
  in r16,SPDR
  sbi PORTB, 0            ; SS <- 1
  ldi r18, 0
wait2:                      ; pauza intre transmisii
  dec r18
  brne wait2
rjmp loop
```

SPI la ATmega64

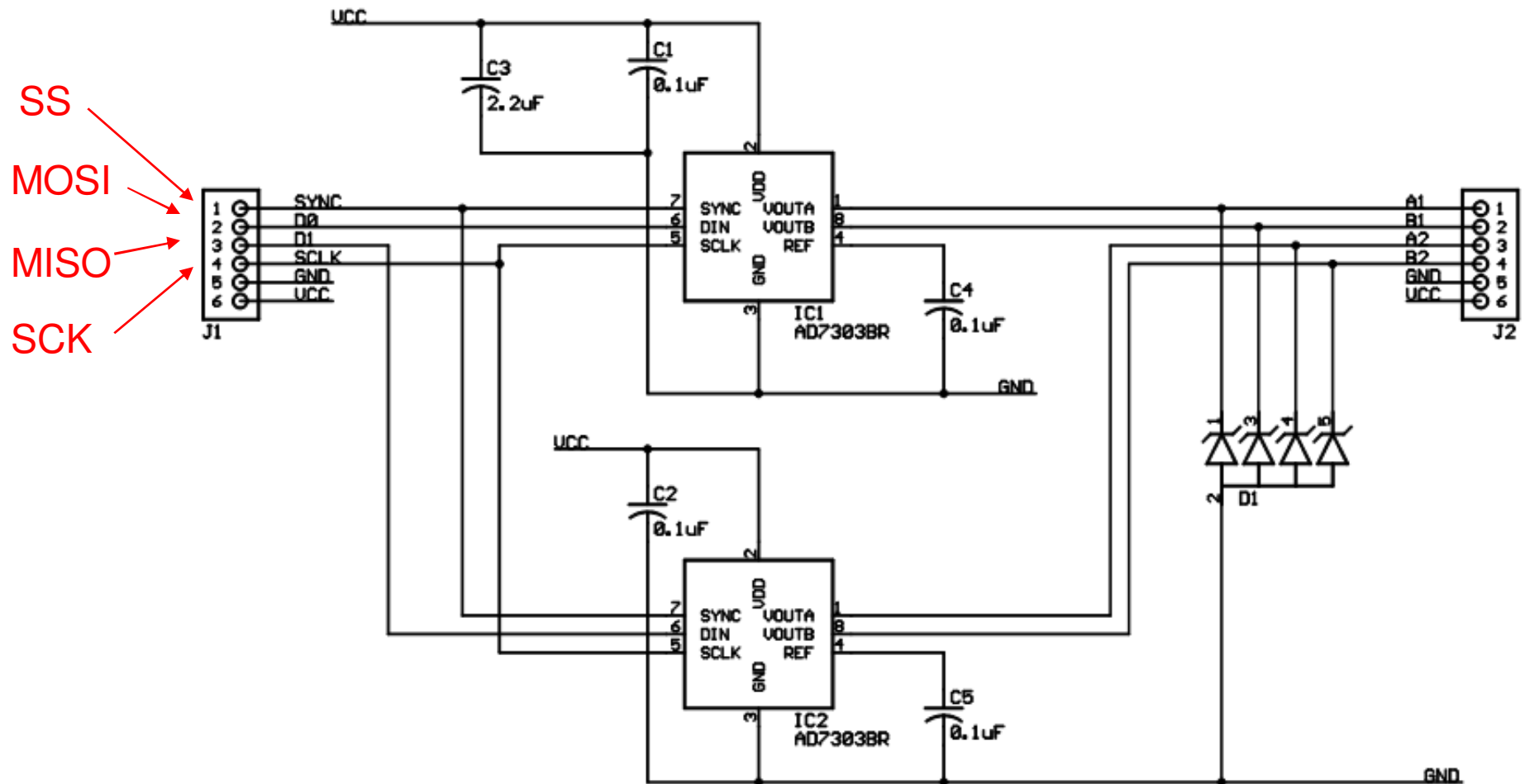
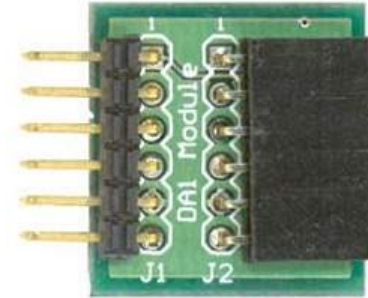
Utilizare SPI (Master) – Rezultat



SPI la ATmega64

Conectare module prin SPI

Digilent PMOD DA1 – Digital to Analog Converter

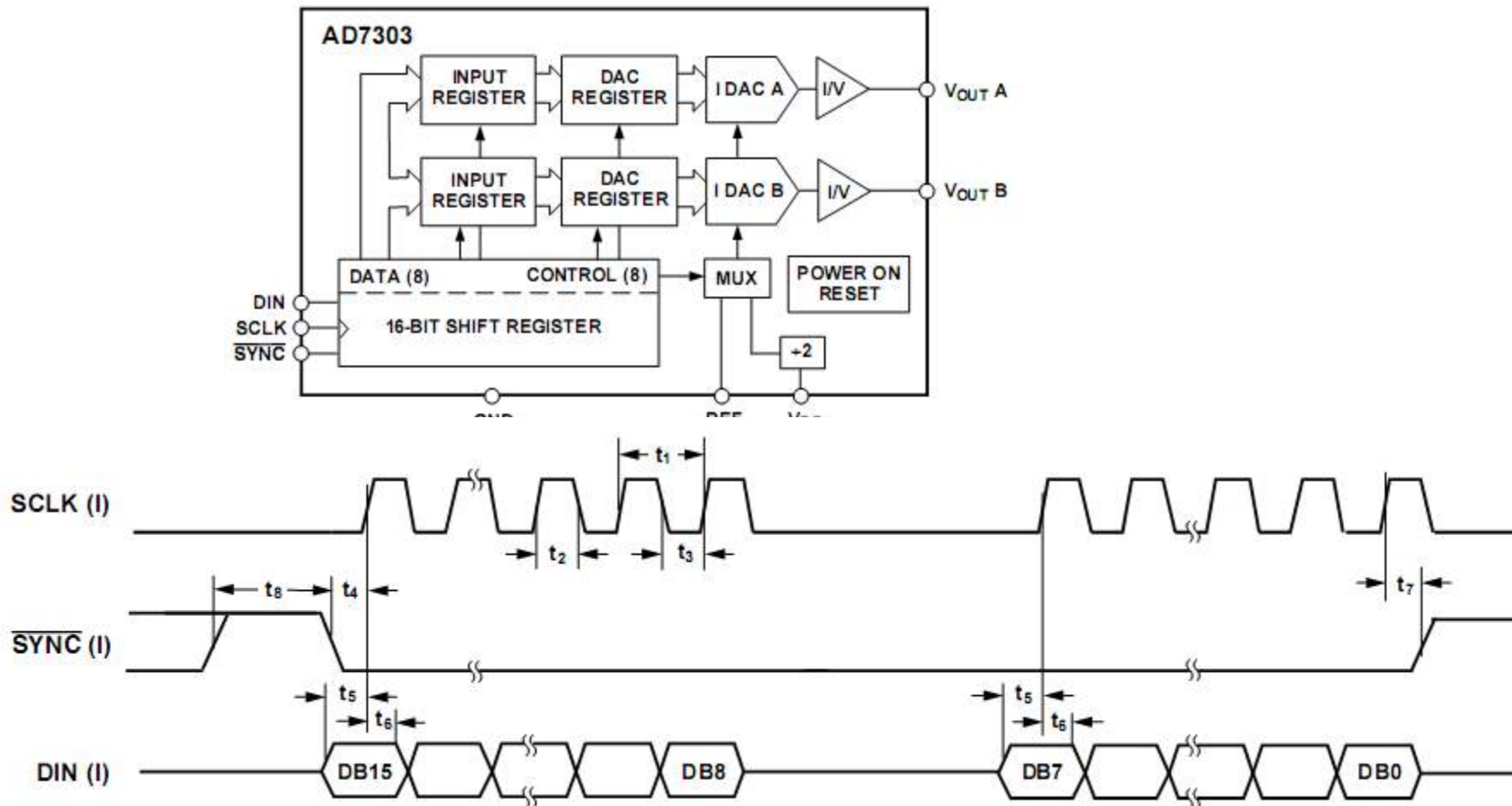


SPI la ATmega64

Conectare module prin SPI

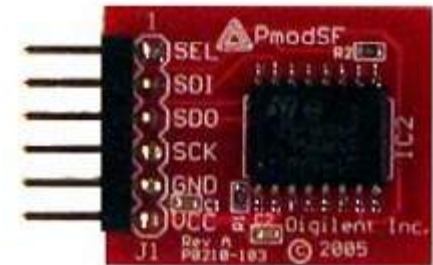
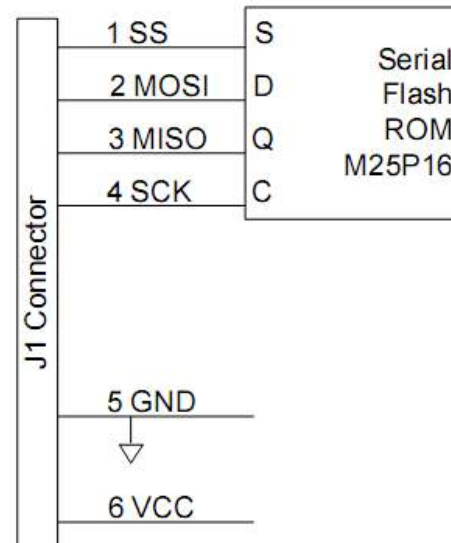
Digilent PMOD DA1 – Digital to Analog Converter

Transmisie 16 biti (2x8 biti) – primii 8 date, urmasorii control

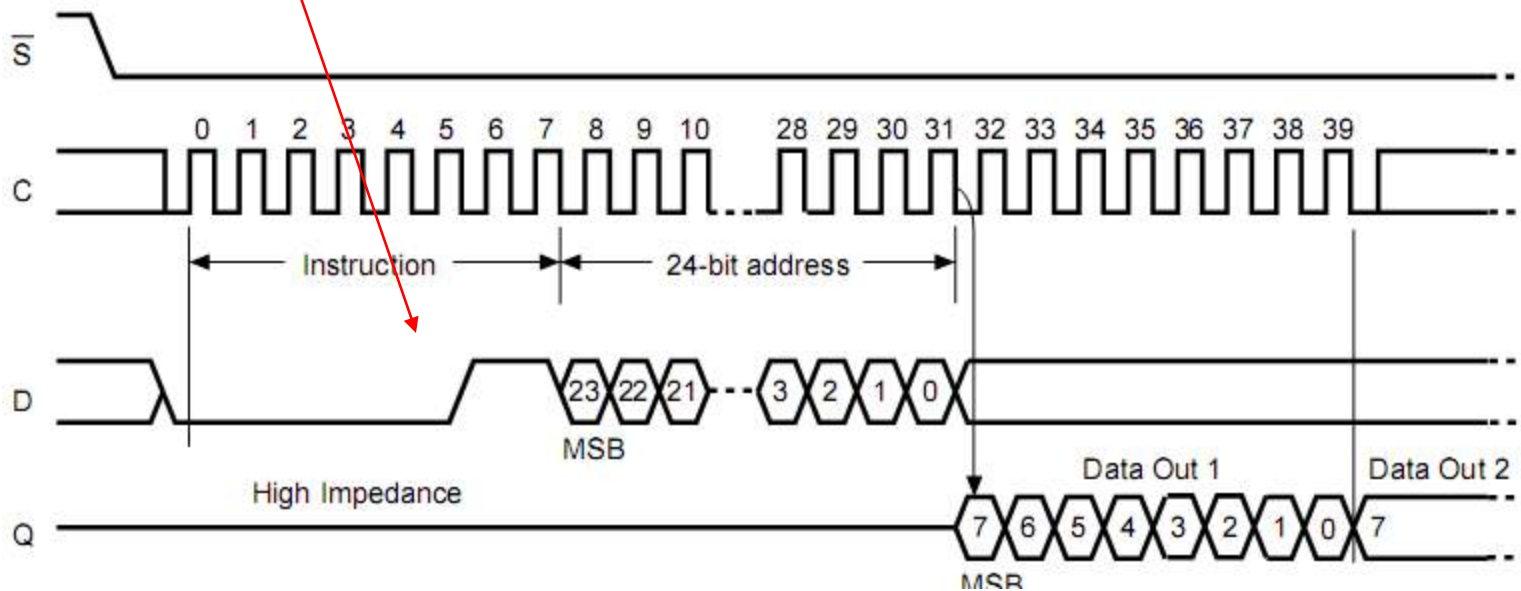


SPI la ATmega64

Conectare module prin SPI
Digilent PMOD SF – Serial Flash



00000011 = 'READ'

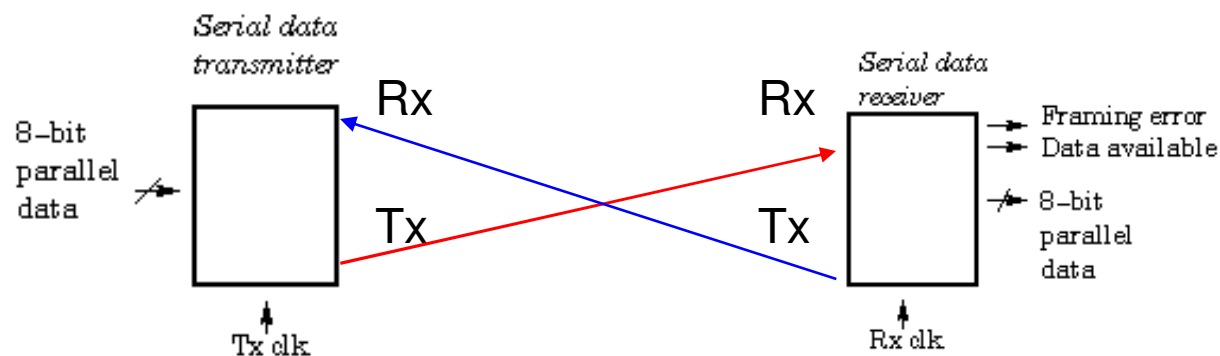


USART

USART – UART cu posibilitate de sincronizare prin semnal de ceas

UART – Interfata pentru comunicare seriala asincrona

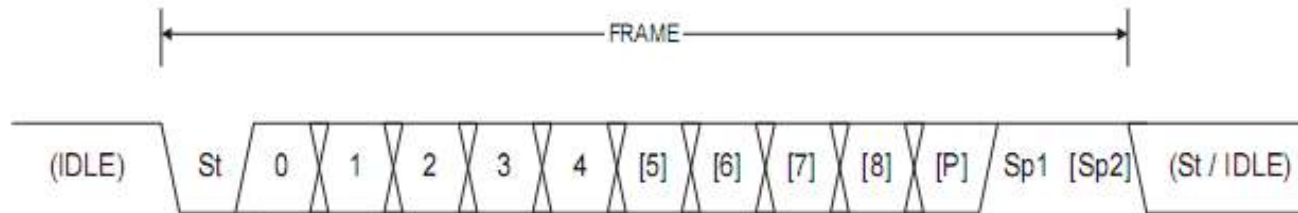
- **Asincron** – intervalul dintre pachete de date poate fi nedefinit. Destinatarul transmisiei detecteaza cand incepe si cand se termina un pachet
- Intervalul de timp dintre biti (frecventa de transmisie a bitilor, **baud rate**) este fixa, si trebuie cunoscuta de ambele parti ale transmisiei
- Transmisia si receptia se pot efectua simultan (full duplex). Fiecare parte a conversatiei poate initializa o transmisie.
- O interfata UART are doua semnale
 - Rx – intrare, receptie
 - Tx – iesire, transmisie
- USART are un semnal in plus, xck (external clock) care poate fi intrare sau iesire, si va sincroniza transmisia si receptia
- Doar functionalitatea comuna cu UART va fi studiata/ implementata



USART

Transmisia datelor: Un pachet (frame) este compus din

- **St:** 1 bit de start, cu valoare '0'
- **D:** Biti de date (5...9, valoare stabilita la ambii participantii la transmisie)
- **P:** 1 bit de paritate. Paritatea poate fi:
 - Absenta: bitul P nu exista
 - Para (*Even*)
 - Impara (*Odd*)
- **Sp:** 1 sau 2 biti de stop, cu valoare '1' – numarul bitilor trebuie stabilit la ambii participantii la transmisie



$$P_{even} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{odd} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

USART

Receptia datelor: - Sistemul care receptioneaza trebuie sa stie care sunt parametrii transmisiei (Baud, Nr biti/frame, Nr stop biti, Paritate)

1. Se detecteaza o tranzitie din '1' in '0' pe linia Rx (receptie)
2. Se verifica mijlocul intervalului pentru bitul de start. Daca este '0', se initiaza secventa de receptie, altfel tranzitia se considera zgomot.
3. Se verifica mijlocul intervalului pentru bitii urmasori (date, paritate, stop), si se reconstruieste pachetul de date
4. Daca in pozitia unde trebuie sa fie bitii de stop se detecteaza valoarea zero, se genereaza eroare de impachetare (**framing error**)
5. Daca paritatea calculata la destinatie nu corespunde cu bitul P, se genereaza eroare de paritate (**parity error**)

Pentru robustete, receptorul esantioneaza semnalul de intrare la o frecventa de 8-16 ori mai mare decat *baud rate*.



Diferente dintre baud rate la transmisie si la receptie pot cauza erori in esantionarea bitilor!

USART

UART si RS232:

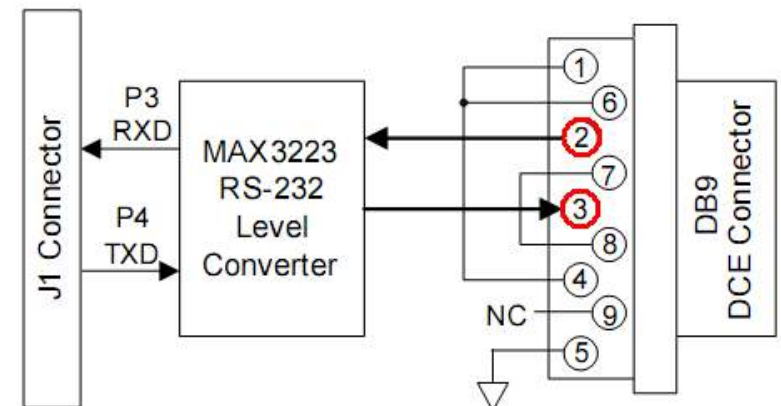
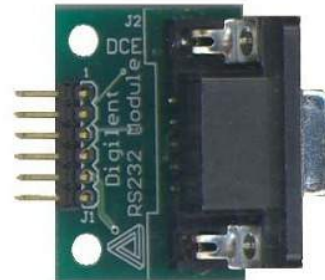
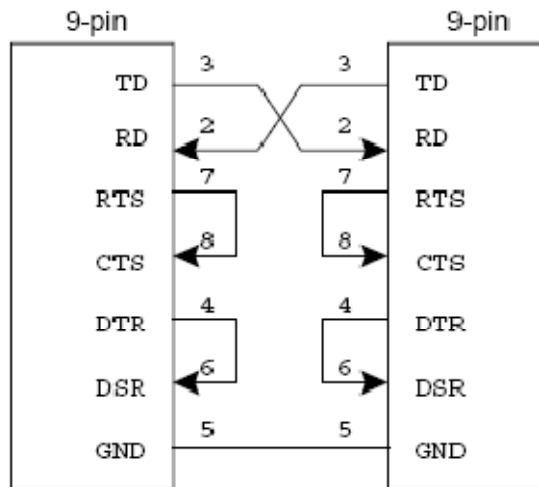
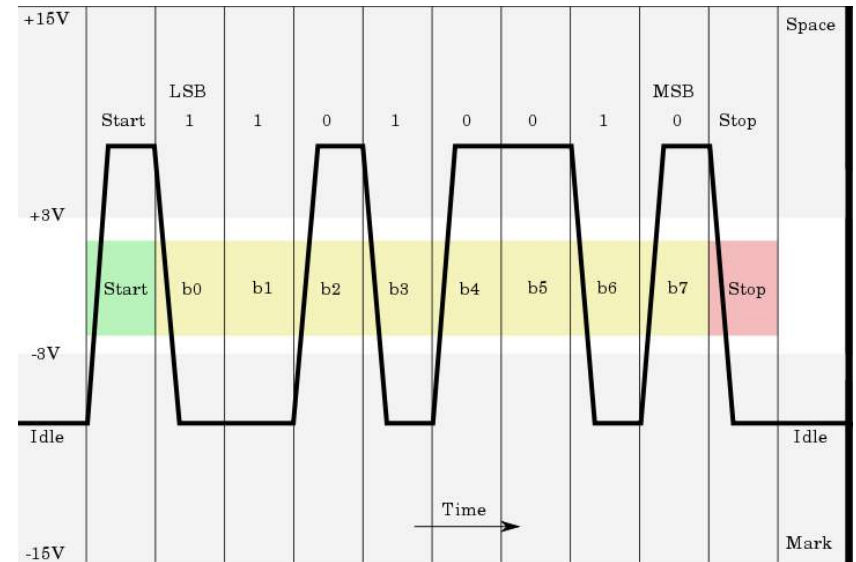
Adaptarea nivelelor de tensiune

RS232 logic '1' -5... -15 V

RS232 logic '0' +5...+15 V

Este nevoie de conversie de la nivelele logice UART la RS232

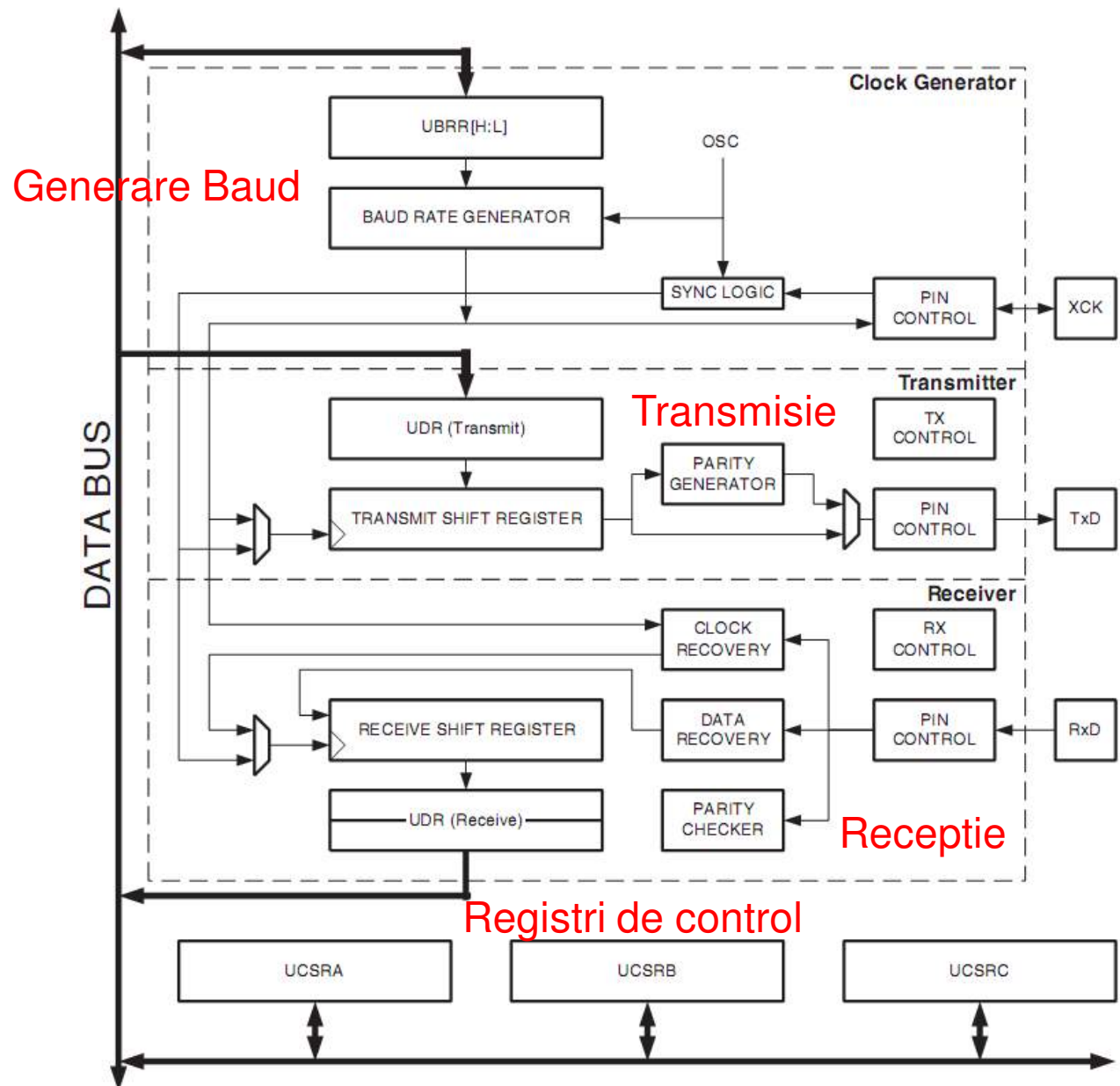
Corespondenta pinilor



USART la ATmega64

Doua unitati USART:
USART0 si USART1

Arhitectura generala:



USART la ATmega64

Configurare sistem:

- Registru de control si stare **UCSRnA**

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **RXCn** – Este ‘1’ cand receptia e completa. Poate genera intrerupere
- **TXCn** – Este ‘1’ cand transmisia e completa. Poate genera intrerupere
- **UDREn** – Data Register Empty, semnaleaza ca registrul poate fi scris
- **FEn** – Semnaleaza eroare de impachetare (Frame Error)
- **DORn** – Data overrun – cand se detecteaza un inceput de receptie inainte ca datele deja receptionate sa fie citite din registrul de date
- **UPEn** – Eroare de paritate (Parity Error)
- **U2Xn** – Valoare ‘1’ = Dublare viteza de transmisie USART
- **MPCMn** – Activare mod de comunicare multiprocesor

USART la ATmega64

Configurare sistem:

- Registru de control si stare **UCSRnB**

Bit	7	6	5	4	3	2	1	0	
	RXCIE_n	TXCIE_n	UDRIE_n	RXEN_n	TXEN_n	UCSZ_{n2}	RXB8_n	TXB8_n	UCSR_{nB}
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **RXCIE_n** – Daca e setat ‘1’, se genereaza intrerupere la terminarea receptiei
- **TXCIE_n** – Daca e setat ‘1’, se genereaza intrerupere la terminarea transmisiei
- **UDRIE_n** - Daca e setat ‘1’, se genereaza intrerupere cand registrul de date e gol
- **RXEN_n** – activare receptie
- **TXEN_n** – activare transmisie
- **UCSZ_{n2}** – combinat cu UCSZ_{n1} si UCSZ_{n0} din **USCR_{nC}** stabileste marimea pachetului
- **RXB8_n** – al 9-lea bit receptionat, cand pachetul are 9 biti
- **TXB8_n** – al 9-lea bit de transmis, cand pachetul are 9 biti

USART la ATmega64

Configurare sistem:

- Registru de control si stare **UCSRnC**

Bit	7	6	5	4	3	2	1	0	
	–	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **UMSELn** – Mod asincron ‘0’ sau sincron ‘1’
- UPMn1 si UPMn0 – Selectia modului de paritate
- **USBSn** – configurare biti de stop: ‘0’ – 1 bit, ‘1’ – 2 biti
- **UCSZn1:UCSZn0** – combinat cu UCSZn2 din UCSRnB, da dimensiunea pachetului

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

USART la ATmega64

Configurare sistem:

- Registrii de control al frecventei: **UBRRnH** si **UBRRnL**
- Formeaza impreuna valoarea **UBRRn**, de 12 biti

Bit	15	14	13	12	11	10	9	8	
	—	—	—	—	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Operating Mode	Equation for Calculating Baud Rate ⁽¹⁾	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{osc}}{16(UBRR + 1n)}$	$UBRRn = \frac{f_{osc}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{osc}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{8BAUD} - 1$

Citire date receptionate / scriere date pentru transmis

- Ambele actiuni se fac prin registrul **UDRn**

USART la ATmega64

Pini comuni cu porturile I/O – Directia este configurata automat prin activarea receptiei si/sau a transmisiei !

Port Pin	Alternate Function
PD7	T2 (Timer/Counter2 Clock Input)
PD6	T1 (Timer/Counter1 Clock Input)
PD5	XCK1 ⁽¹⁾ (USART1 External Clock Input/Output)
PD4	ICP1 (Timer/Counter1 Input Capture Pin)
PD3	INT3/TXD1 ⁽¹⁾ (External Interrupt3 Input or UART1 Transmit Pin)
PD2	INT2/RXD1 ⁽¹⁾ (External Interrupt2 Input or UART1 Receive Pin)
PD1	INT1/SDA ⁽¹⁾ (External Interrupt1 Input or TWI Serial DAta)
PD0	INT0/SCL ⁽¹⁾ (External Interrupt0 Input or TWI Serial CLock)

USART1, portul D

Port Pin	Alternate Function
PE7	INT7/ICP3 ⁽¹⁾ (External Interrupt 7 Input or Timer/Counter3 Input Capture Pin)
PE6	INT6/ T3 ⁽¹⁾ (External Interrupt 6 Input or Timer/Counter3 Clock Input)
PE5	INT5/OC3C ⁽¹⁾ (External Interrupt 5 Input or Output Compare and PWM Output C for Timer/Counter3)
PE4	INT4/OC3B ⁽¹⁾ (External Interrupt 4 Input or Output Compare and PWM Output B for Timer/Counter3)
PE3	AIN1/OC3A ⁽¹⁾ (Analog Comparator Negative Input or Output Compare and PWM Output A for Timer/Counter3)
PE2	AIN0/XCK0 ⁽¹⁾ (Analog Comparator Positive Input or USART0 external clock input/output)
PE1	PDO/TXD0 (Programming Data Output or UART0 Transmit Pin)
PE0	PDI/RXD0 (Programming Data Input or UART0 Receive Pin)

USART0, portul E

USART la ATmega64

Exemplu: comunicare intre ATmega64 si PC – ECHO simplu

Necesar: cablu serial, modul **PMOD RS232**

1. Configurare

Baud: 9600

Marime pachet: 8 biti

Biti de stop: 2

Paritate: fara paritate

$$UBRRn = \frac{f_{osc}}{16BAUD} - 1$$

$$f_{osc} = 8000000$$

$$UBRRn = 51$$

2. Asteptare receptie caracter

- Verificare **RXCn** din **UCSRnA**, asteptare pana devine 1

3. Citire caracter receptionat, din UDRn

4. Scriere caracter de transmis, in UDRn

5. Asteapta transmisie caracter

- Verificare **TXCn** din **UCSRnA**, asteptare pana devine 1

6. Salt la 2

USART la ATmega64

Exemplu: comunicare intre ATmega64 si PC – ECHO simplu

Cod sursa

```
ldi r16, 0b00011000      ; activare Rx si Tx
sts UCSR1B,r16
ldi r16, 0b00001110      ; dimensiune frame 8 biti, fara paritate, 2 biti de stop
sts UCSR1C,r16
ldi r16, 51               ; Baud rate calculat, incapa in primii 8 biti
ldi r17, 0                ; Bitii superiori la UBRR sunt zero
sts UBRR1H, r17
sts UBRR1L, r16
mainloop:

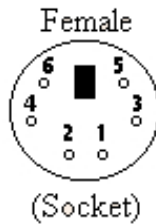
recloop:
    lds r20, UCSR1A
    sbrc r20, 7            ; bitul 7 din UCSR1A – receptie completa
    rjmp recloop

lds r16, UDR1             ; citire date receptionate
sts UDR1,r16              ; scriere date spre transmisie

txloop:
    lds r20, UCSR1A
    sbrc r20, 5            ; asteptare terminare transmisie
    rjmp txloop
rjmp mainloop
```

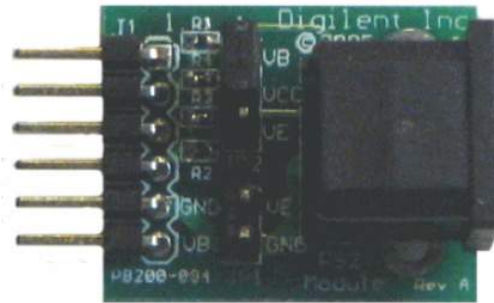
Comunicarea cu tastatura prin PS2

Interfatare fizica

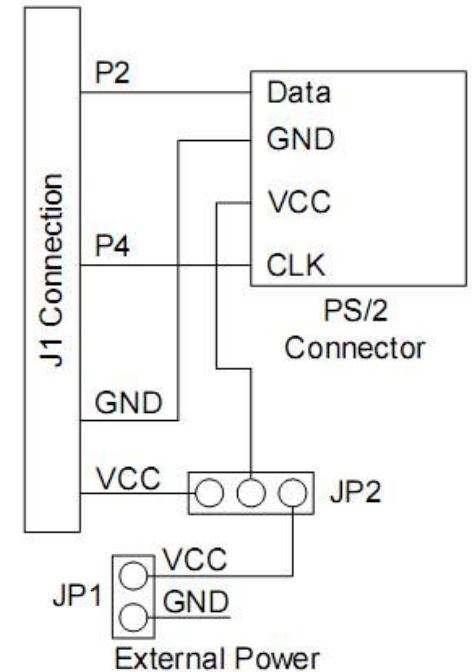


6-pin Mini-DIN (PS/2):

- 1 - Data
- 2 - Not Implemented
- 3 - Ground
- 4 - Vcc (+5V)
- 5 - Clock
- 6 - Not Implemented



PMOD PS2



Semnale logice

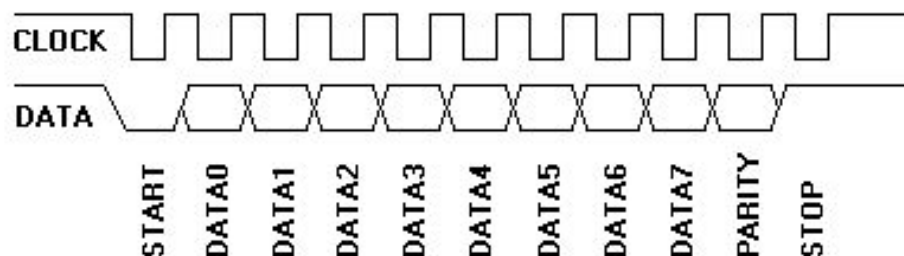
Clock, Data – valoare logica '1' = +5V, valoare logica zero = 0 V
Gnd

In repaus, Clk si Data sunt pe nivelul logic '1' – folosire rezistori PULL UP

Comunicarea cu tastatura prin PS2

Transmisie date de la tastatura la Microcontroller

1. Tastatura verifica daca **Clk** este '1' – linie libera
2. Tastatura semnaleaza inceputul transmisiei prin punerea **Clk** la '0'
3. Tastatura genereaza semnal de clk si trimite datele sincronizate cu acest semnal
Se pune un bit pe linia de date cand clk este '1'
Bitul este citit de microcontroller cand clk este '0'
(Se poate citi bitul la tranzitia dintre '1' si '0')
4. Structura unui pachet: 1 bit start, 8 biti date, 1 bit paritate (odd), 1 bit stop

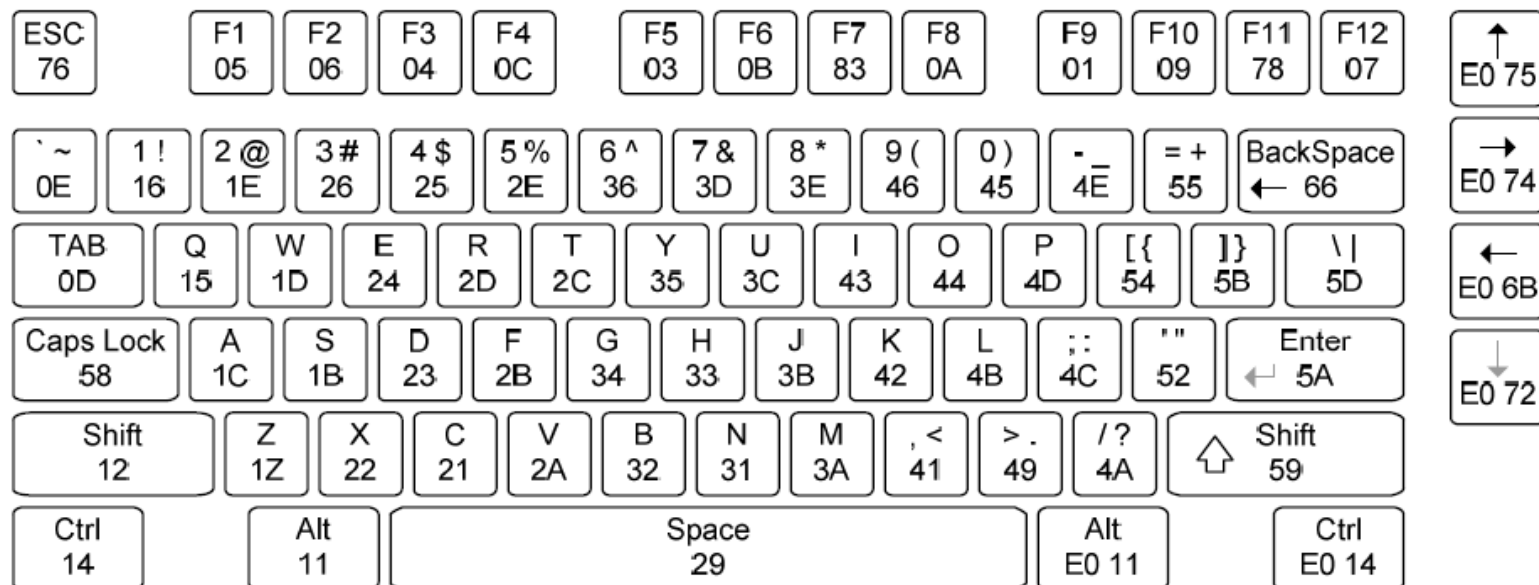


Citirea datelor – se poate utiliza o intrerupere externa pentru tratarea Clk, si datele vor fi citite de pe un pin de intrare in momentul declansarii intreruperii

Comunicarea cu tastatura prin PS2

Transmisie date de la tastatura la Microcontroller

- Tastatura transmite in momentul in care asupra ei se actioneaza (apasare – eliberare tasta)
- La apasarea unei taste (*Make*) se transmite un cod (**scancode**) de 8 biti
- La ridicarea unei taste (*Break*) se transmite codul 0xF0 , urmat de **scancode**
- La taste speciale (sageti, etc) se transmite 0xE0+**Scancode** la apasare, si 0xE0+0xF0+**Scancode** la ridicare
- Toti bitii sunt impachetati cu start, date, paritate si stop
Ex: daca avem ridicarea unei taste speciale, vom avea 3 serii de 11 biti



Comunicarea cu tastatura prin PS2

Coduri ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.LookupTables.com

Comunicarea cu tastatura prin PS2

Coduri ASCII extinse

128	Ç	144	É	161	í	177	⌘	193	⌞	209	⌞	225	ß	241	±
129	ü	145	æ	162	ó	178	⌘	194	⌞	210	⌞	226	Γ	242	≥
130	é	146	Æ	163	ú	179		195	⌞	211	⌞	227	π	243	≤
131	â	147	ô	164	ñ	180	⌞	196	—	212	⌞	228	Σ	244	∫
132	ä	148	ö	165	Ñ	181	⌞	197	⌞	213	⌞	229	σ	245	∫
133	à	149	ò	166	°	182	⌞	198	⌞	214	⌞	230	μ	246	+
134	â	150	û	167	°	183	⌞	199	⌞	215	⌞	231	τ	247	≈
135	ç	151	ù	168	¿	184	⌞	200	⌞	216	⌞	232	Φ	248	°
136	ê	152	—	169	—	185	⌞	201	⌞	217	⌞	233	⊙	249	·
137	ë	153	Ö	170	¬	186	⌞	202	⌞	218	⌞	234	Ω	250	·
138	è	154	Û	171	½	187	⌞	203	⌞	219	■	235	δ	251	√
139	ï	156	£	172	¼	188	⌞	204	⌞	220	■	236	∞	252	—
140	î	157	⌘	173	¡	189	⌞	205	=	221	■	237	φ	253	²
141	ì	158	—	174	«	190	⌞	206	⌞	222	■	238	ε	254	■
142	Ä	159	ƒ	175	»	191	⌞	207	⌞	223	■	239	∧	255	
143	Å	160	á	176	⌘	192	⌞	208	⌞	224	α	240	≡		

Source: www.LookupTables.com

Comunicarea cu tastatura prin PS2

Exemplu

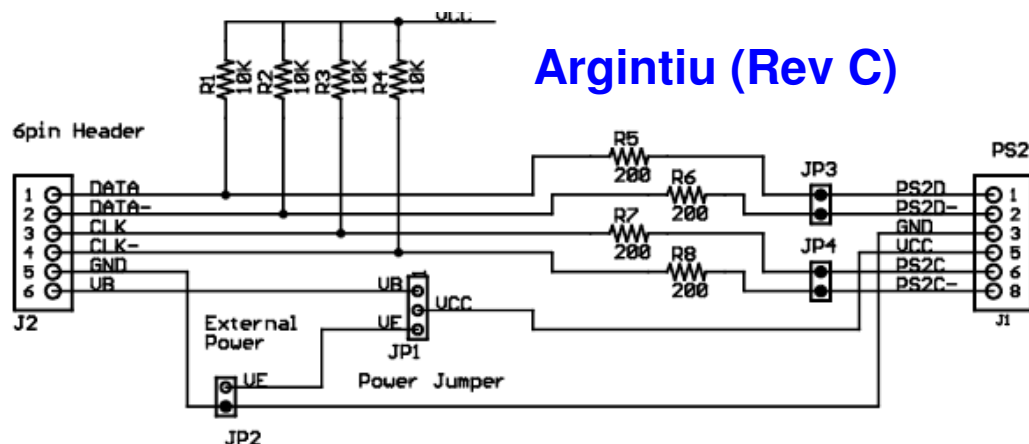
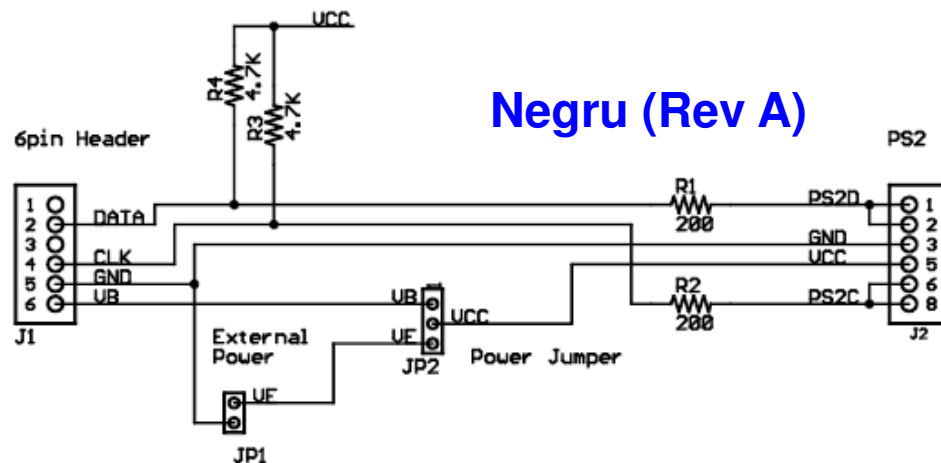
Comunicare PS2 – Cerebot

1. Ce modul PS2 folosim ?

- Fiecare modul are propria schema de conectare

2. Unde conectam ?

- Doar doi conectori au intreruperi externe, JD si JF.
- JD contine interfete pentru UART si SPI, deci alegem JF



Rev A: data, Pin 2, PE4
clk, Pin 4, INT5

Rev C: data, Pin 1, PE6
clk, Pin 3, INT7

JF	1	T3/ INT6	PE 6
	2	OC3B/ INT4	PE 4
	3	ICP3/ INT7	PE 7
	4	OC3C/ INT5	PE 5
	5	GND	
	6	VCC	

Comunicarea cu tastatura prin PS2

Exemplu (valabil pt Pmod PS2 rev C)

```
.include "m64def.inc"
```

```
.org 0x0000
```

```
jmp RESET
```

```
.org 0x0010
```

```
jmp int7isr
```

```
RESET:
```

```
ldi r16, low(RAMEND)
```

```
out SPL, r16
```

```
ldi r16, high(RAMEND)
```

```
out SPH, r16
```

```
ldi r16, 0b10000000 ; INT 7 tratata pe front descrescator
```

```
out EICRB, r16
```

```
ldi r16, 0b10000000 ; activare INT 7
```

```
out EIMSK, r16
```

```
ldi r16, 0 ; folosire PORTE ca intrare
```

```
out DDRE, r16
```

```
ldi r16, 0xff ; activare rezistente PULL UP
```

```
out PORTE, r16
```

```
ldi r16, 0xFF
```

```
out DDRC, r16 ; folosire PORTC ca iesire
```

```
ldi r25, 0 ; numaratorul de biti
```

```
ldi r26, 0 ; registrul de deplasare
```

```
sei ; activare intreruperi
```

```
main:
```

```
rjmp main
```

```
; tratarea semnalelor de la tastatura
```

```
; urmarim detectarea ridicarii unei taste
```

```
; semnale SCANCODE + F0 + SCANCODE
```

```
; format pachet: S + Data 8 + P + Sp
```

```
; total : 33 biti
```

```
; aceasta procedura se executa la fiecare tranzitie a keyboard clock  
(conectat la INT7)
```

```
int7isr:
```

```
inc r25 ; numarator de biti
```

```
cpi r25, 23 ; pana la pozitia 23 nu ne intereseaza
```

```
brlo exit
```

```
cpi r25, 32 ; peste pozitia 31 nu ne intereseaza
```

```
brge skip
```

```
lsr r26 ; de-serializarea datelor, PE(6)
```

```
sbic PINE, 6 ; daca e 0, doar shiftam
```

```
ori r26, 0x80 ; daca e 1 punem 1 in capat la r26
```

```
skip:
```

```
cpi r25, 33 ; am ajuns la capat ?
```

```
brlo exit ; inca nu
```

```
ldi r25, 0 ; daca da, se re-seteaza contorul de biti
```

```
out PORTC, r26 ; pachet complet, afisam datele primite
```

```
exit:
```

```
reti
```