UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Enginyeria Minera, Industrial i TIC

Comunicació sèrie

Dispositius Programables — Enginyeria de Sistemes TIC

Eric Blanco

Grau en Enginyeria de Sistemes TIC
Quadrimestre 3
Dispositius programables
Curs 2023-24, G11

Tasca prèvia 1 Llegiu detalladament l'apartat 20 de [ATmega328p].

Llegit i entes, tal i com es pot veure en la explicació del codi de exemple del següent previ…

Tasca prèvia 2 Descriviu detalladament què fa p5-exemple.s.

```
DDRB_o = 0x4
PORTB_o = 0x5
DDRD_o = 0x0a
PORTD_o = 0x0b
UDR0 = 0xc6
UBRR0H = 0xc5   ;inicialitza USART I/O Data Register
UBRR0L = 0xc4   ;inicialitza USART Baud Rate Register Low (Registre que diu bits/S de la transmisió)
UCSR0C = 0xc2   ;inicialitza registre
UCSR0B = 0xC1   ;inicialitza registre
UCSR0A = 0xC0   ;inicialitza registre

.global main


/* rutina de recepció de bytes, el valor es recull al registre r16 */
rx:     lds r16,UCSR0A
        sbrs r16,7              ; Si el bit 7 esta a 1 (rebem dada i salta el flag) el programa saltar
a una posicició i carregara la dada a r16
        rjmp rx
        lds r16,UDR0
        ret

/* rutina de transmissió de byte, el valor a transmetre està al registre r16 */
tx:     lds r17,UCSR0A
        sbrs r17,5              ;indica si el bit 5 esta activat, es a dir, si esta llest per rebre no
va data i per tant llest per transmetre el que ja te
        rjmp tx
        sts UDR0,r16
        ret

main:
        /* set baud rate a 9600*/
        ldi r16, 0
        sts UBRR0H,r16          ;son bits reservats per a definir els bps de transferencia
        ldi r16, 103            ;el mateix que l'anterior pero aqui el definirem a 9600 qu es el valor
 103 a UBRR0n
        sts UBRR0L,r16
        /* set frame format */
        /* tot i que el valor dels registres després d'un reset ja és correcte  (asíncron, 8 bits de d
ades, 1 bit de parada, sense paritat,velocitat normal, comunicació no multiprocessor)assegurem aquesta
 configuració escrivint el valor als registres*/

        ldi r16, 0b00100000     ;carregarem això a (USART Control and Status Register n A)-  interesa
activar el Bit 5 - que es el que fa que el flag UDREn s'activi quan esta llest per rebre noves dades,
el bit 1 tambe ens interesa tenirlo a 0 perque la nostre transmisió serà sincrona i tots els demes a 0
 tambe per tant r16 = 0b00100000 (nomes activem el bit 5)
```

```
        sts UCSR0A,r16          ;carreguem bits explicats en la instrucció anterior en UCSR0A
        ldi r16, 0b00000110     ; carreguem això a UCR0C, volem activar a transmisio i recepsió de 8 b
its segguint la taula del manual ens indica que és activant els bits 1 i 2
        sts UCSR0C,r16          ; carreguem els bits a UCSR0C

        /* enable rx, tx, sense interrupcions */
        ldi r16, 0b00011000     ; carreguem això a UCR0B, nomes interesa activar els bits 3 i 4, per a
ctivar la transmisió (TX) amb el bit 3, i activar la recepció (RX) amb el bit 4
        sts UCSR0B,r16          ;carreguem lo anterior a UCR0B utilitzant sts per poder escriure en un
 registre de memoria reservat

        /* configuració dels pins */
        ldi r16,0b00000010      ;carreguem la pota TX com a sortida i RX com a sortida
        out DDRD_o,r16

loop:                           ;el que fara el loop es que estarà esperant una dada (RX) i despres aq
uesta maateixa sera enviada (TX) i tornarà a eserar la dada
        call rx
        call tx
        rjmp loop
```

Tasca prèvia 3 Modifiqueu p5-exemple.s de manera que la configuració de la comunicació passi de velocitat 9600 bps i 8 bits de dades sense paritat, a velocitat 115 Kbps i 7 bits de dades amb paritat. Anomeneu el fitxer modificat p5-exemple-mod.s

Per començar el que demana el previ es posar la velocitat transmisió a 115 Kbps, per fer-ho haurem de consultar la següent taula que trobem al manual del AVR:

Table 20-7. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max. (1)	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

1. UBRRn = 0, Error = 0.0%

A nosaltres només ens interessa la primera columna que es la de la freqüència del nostre AVR 16MHz:

Table 20-7. Examples of U

Baud Rate (bps)	f_{osc} = 16.0 U2Xn = 0	
	UBRRn	Error
2400	416	-0.1%
4800	207	0.2%
9600	103	0.2%
14.4k	68	0.6%
19.2k	51	0.2%
28.8k	34	-0.8%
38.4k	25	0.2%
57.6k	16	2.1%
76.8k	12	0.2%
115.2k	8	-3.5%
230.4k	3	8.5%
250k	3	0.0%
0.5M	1	0.0%
1M	0	0.0%
Max. (1)	1Mbps	

1. UBRRn = 0, Error = 0.

En aquesta taula hem de localitzar els 115 Kbps que ens demanen. Un cop localitzat el que ens indica la taula es que haurem de posar el valor de UBRRn a 8. En el cas d'exemple teniem 9600 → es a dir el valor 103. El canviarem per 8. 8 en binari es 1000 **per tant omplirem el registre UBRR0H amb 0 i el UBRR0L el posarem a 00001000 per dir que volem el valor 8 i per tant 115 Kbps.**

Seguidament ens demana que en comptes de enviar 8 per transmissió, només enviem 7. A mes a mes, en comptes de sense paritat ara ho farem amb paritat. Això ho aconseguirem modificant el registre UCSRTn. Seguint la taula del registre que configura els modes de paritat veiem que per activar-lo hem de activar el bit 5 a '1', i el bit 4 a '0', tal i com observem en la seguent taula del manual:

1 0

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	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	

Table 20-9. UPMn Bits Settings

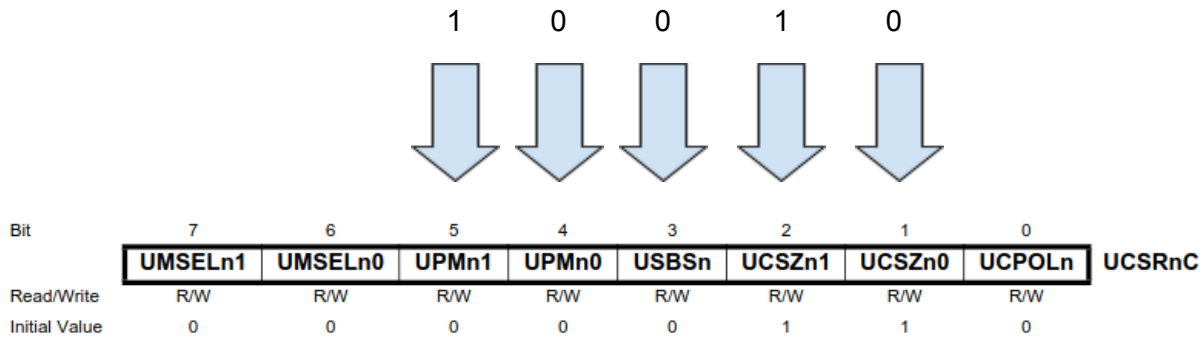
UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Ara que tenim el mode de paritat activat anem a activar la transmisió de 7 bits, mirem la taula següent i ens mostra que hem de activar el bit 1, i el bit 0 i 2 deixar-los a 0.

Table 20-11. UCSZn Bits Settings

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

Per tant el registre UCSRnC quedaria de la següent manera:



Y el tros de codi que he modificat ha quedat així:

```
ldi r16, 0b00100100     ; carreguem això a UCR0C, volem activar a transmisio i recepsió de 8 bits
                        i el mode de paritat
sts UCSR0C,r16          ; carreguem els bits a UCSR0C
```

La resta queda igual que en l'exemple…

Tasca prèvia 4 Feu un programa que quan detecti que s'ha polsat la lletra l o L encengui el LED de l'Arduino i quan rebi qualsevol altre dada l'apagui. Useu la mateixa configuració del programa p5-exemple.s. Anomeneu aquest fitxer p5-codi1.s.

Començarem per agafar el codi d'exemple de la pràctica per escriure el nou codi allà, aprofitant així les configuracions de 8 bits de transmissió i de 9600 bps de velocitat.

Primer de tot començarem definint les subrutines led_on / led_off:

- quan les cridem encendran o apagaran el led

```
Unset
led_on:
  sbi PORTB_o, 0x05
  ret

led_off:
  cbi PORTB_o, 0x05
  ret
```

seguidament definirem la subrutina que comprovarà si la dada que li esta entrant pel port serie es l o L en aquest ordre.

Utilitzarem la instrucció CPI (compare immediat) per comparar si el valor de r16 (dada de input) es igual a la lletra l o L codificada en codi ASCII.

- si r16 es igual a l, farà branch a led_on per encendre el led. si no pasarà a comprovar L.
- si r16 es igual a L, farà branch a led_on per encendre el led. si no és igual cridara a la subrutina led_off per apagar el led ja que no ha trobat coincidencia.

```
Unset
check_L:
  cpi r16, 0b01101100 /* l */
  breq led_on
  cpi r16, 0b01001100 /* L */
  breq led_on
  call led_off
  ret
```

Com que la subrutina RX i TX ja han estat definides prèviament, i les inicialitzacions en el main també, només queda definir el loop:

```
Unset
loop:
       call rx       ; esperem a rebre dades i quan rebem guardem a r16
       call check_L  ; cpi r16 amb l i L, si si --> led on / si no--> led_off
       call tx
       rjmp loop     ; tornem a mirar si hi ha dades
```

En aquest cas no caldria cridar a TX, ja que aquesta subrutina només envia dades, i nosaltres el que volem és encendre el led, pero el posarem igualment per veure-hi el echo..

Tasca prèvia 5 Feu un programa que quan detecti que s'ha polsat la lletra n o N respongui amb els caràcters corresponents als nombres 0,1,2,3,4,5,6,7,8,9. Quan es rebi qualsevol altre valor, respongui amb el caràcter N. Anomeneu aquest fitxer p5-codi2.s.

Per fer aquesta tasca previa aprofitarem agunes subrutines del codi anterior com RX, TX, check_L. A la subrutina check_L la modificarem per a que revisi N o n.

N = 110 en la taula asci → 01101110 en binari
n = 78 en la taula ascii → 01001110 en binari

revisa_N: revisara si es N o n, si ho és enviara 0123456789 cridant la subrutina envia_num, si no ho és enviara N cridant a la subrutina envia_N

```
Unset
revisa_N:
  cpi r16, 0b01101110 /* N */
  breq envia_num
  cpi r16, 0b01001110 /* n */
  breq envia_num
  call envia_N
  ret
```

Ara farem la subrutina que retornarà N quan no rebi la lletra ‘n’ o ‘N’.

```
Unset
envia_N:
	ldi r16, 0b01101110 /* N */
	call tx
	ret
```

Ara, hem de fer la subrutina que enviï la secuencia de números desde el 0 fins el 9 cuan es rebi el valor N o n.

```
Unset
envia_num:
	ldi r16, 0b00101111		; posem el valor asci 47, el de abans
del 0
int_num:
	inc r16			; incrementa 1
	call tx			; transmet r16
	cpi r16, '9'		; compara si es 9 (ultim valor)
	brne int_num		; si no es fa branch a int_num, si es pc+2
	ret
```

Només queda fer el loop, el bucle principal del programa.

La estrategia a seguir feta en pseudocodi sera:

- mirar la dada amb rx
- revisar si es N o n
- if N → tx envia: (0123456789)
- else → tx envia: N

} Això ja ho fa revisa_N

El loop quedaria simplement així:

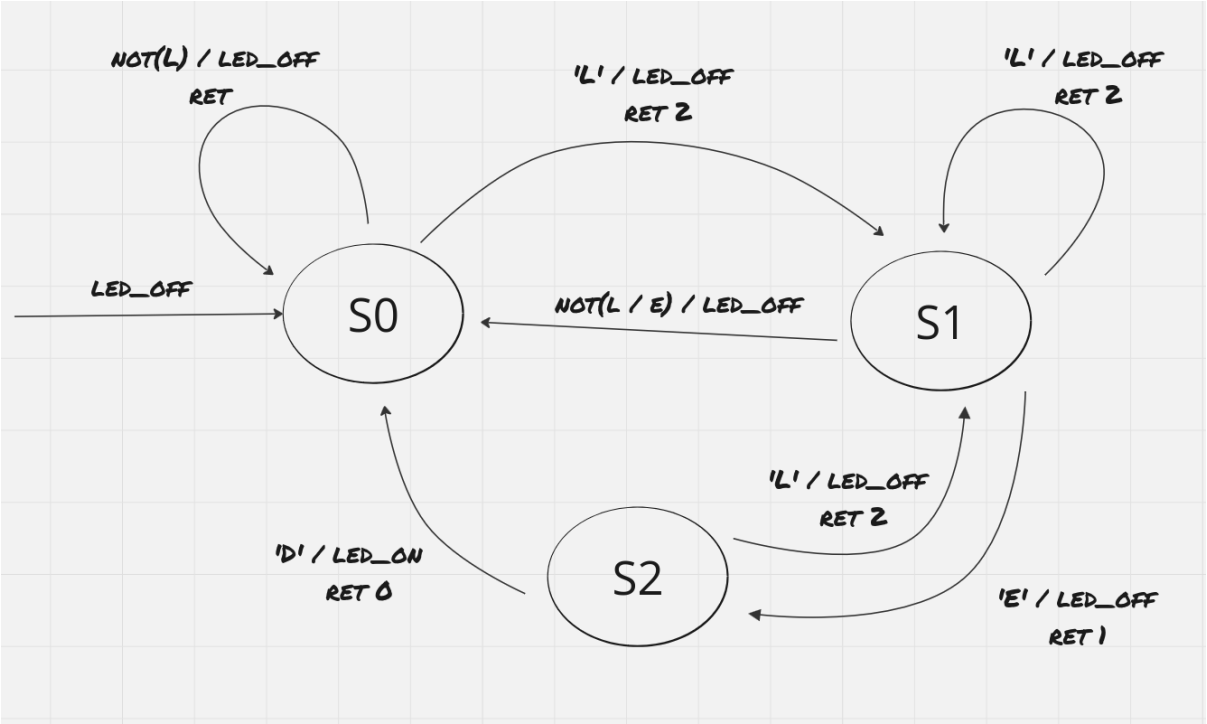
```
Unset
loop:
	call rx
	call check_N
	rjmp loop
```

Tasca prèvia 6 Feu un programa que quan detecti exactament la seqüència de lletres led encengui el LED i quan rebi qualsevol altre seqüència l'apagui. La resposta de l'Arduino cap al ordinador a cada pulsació hauria de ser el nombre de lletres que queden per teclejar fins aconseguir la seqüència led. Implementeu aquest programa com una màquina d'estats: definiu els estats, dibuixeu el diagrama d'estats i declareu qui mantindrà el valor de l'estat del sistema. Anomeneu aquest fitxer p5-codi3.s

primer de tot haurem de fer el diagrama de estats, per saber que ha de fer el programa en cada moment.

Utilitzarem per a fer això un diagrama de tipus mealy perquè ens serà més fàcil i entenedor d'aplicar en ensamblador. Ja que aquest model genera una sortida basant se en l'estat actual i en l'entrada.

El diagrama es el següent:



El codi de la màquina d'estats estara tot dins del loop, així aconseguirem un bucle simple que vagi variant d'estat segons l'entrada.

El codi en qüestió serà el següent:

```
Unset
loop:
    call led_off


s0:    ldi r16, '3'
    call tx
    call rx
    call led_off
    call check_L
    breq s1
    rjmp s0

s1:    ldi r16, '2'
    call tx
    call rx
    call check_E
    breq s2
    call check_L
    breq s1
    rjmp s0

s2:    ldi r16, '1'
    call tx
    call rx
    call check_L
    breq s1
    call check_D
    brne s0
    call led_on
    ldi r16, '0'
    call tx
    call rx
    rjmp s0
```

Tasca 7 Assembleu p5-exemple.s i comproveu el seu funcionament.

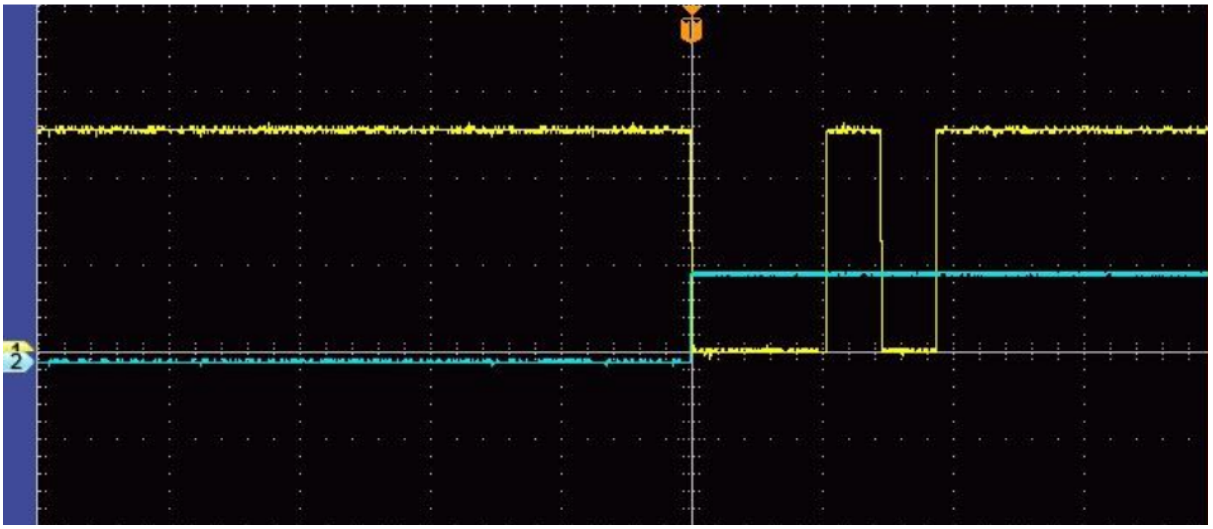
El codi funciona perfectament tal i com veiem. retorna cada lletra que li enviem. podem comprovar en l'osciloscopi, veiem que que esta en mode repòs (5V) i just després veiem el bit start. A continuació veiem tots els bits de la lletra que li enviem des del bit de menor pes fins el de major pes, en aquest cas la 'a'.

Tasca 8 Comproveu el correcte funcionament de p5-exemple-mod.s.

Funciona igual que el codi d'exemple, al executar picocom li haurem de posar que ara transmitir en 115 kbps i en 7 bits.

Tasca 9 Comproveu el correcte funcionament de p5-codi1.s.

funcione també a la perfecció, cuan prenem la lletra l o L s'encén el LED:



Tasca 10 Comproveu el correcte funcionament de p5-codi2.s.

També funciona igual que el previ descrit. Quan prenem qualsevol lletra que no sigui n 0 N ens retorna N, i si prenes les lletres n o N ens retorna els numeros 0123456789:

```
baudrate is    : 9600
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
hangup is      : no
nolock is      : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-a] [C-h] to see available commands
Terminal ready
0123456789NNNNNNNNN0123456789012345678901234567890123456789NNNNNNNNNNNNNNNNNNNNN0123456789
NNNNNNN0123456789NNNNNNN0123456789NNNNNNNNNNNN0123456789NNNNNN0123456789
```

Tasca 11 Comproveu el correcte funcionament de p5-codi3.s.

la màquina d'estats ha funcionat bé, amb tres estats i les accions es fan en els canvis de estat. premem les lletres L - E - D i si la secuencia del input es la correcte s'encendrà el led. si no es la correcte torna a l'estat 0 sense importar en quin estat estiguem. si prenem L de la mateixa manera com que es la primera lletra ens canvia automàticament a l'estat 1 per poder continuar la secuencia sense haver de passar per l'estat 0.