KOMUNIKAZIO SERIALA AVR-n (USART)

Unai Fernandez

Contents

List of Figures

USART, ingelesezko Universal Syncronous Asynchronous Receiver-Transmitter sigletatik dator. Mikrokontrolagailua, ordenagailuarekin komunikatzeko balioko du.

Komunikazioaren azalpena

USART bidezko komunikazioa simplea da azaltzen, komunikazioa hasteko start bita erabiliko da, hau da beheranzko aldaketa bat detektatzen denean seinalean, UART-ek interpretatu egingo du, hurrengo bitak datuak direla. Azkenik stop bitaren bidez, mezua bukatu dela adieraziko da. Stop bita, 1-eko konstantea izango da. Ondorengo irudian ikusi daiteke aurretik azaldutakoa.

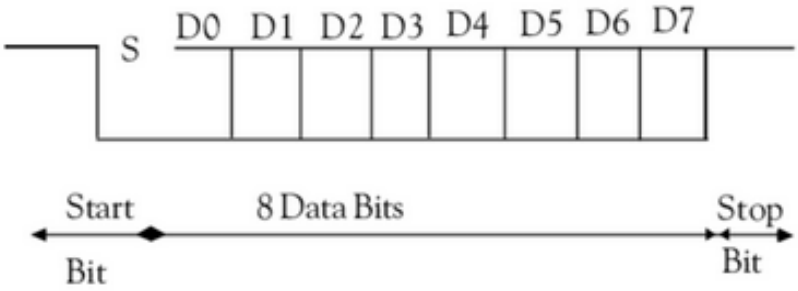

Figure 1: USART

AVR mikrokontrolagailu batekin komunikazio seriala gauzatzeko erabiliko diren kodearen eta programen azalpen hau bi ataletan banatuko da.

- AVR-n konfigurazioa
- Screen serie terminalaren instalazioa eta oinarrizko erabilera

AVR-n konfigurazioa

Konfigurazioarekin hasteko, erregistroak behar diren bezala hasieratu beharko ditugu, mikrokontrolagailuak izatea nahi dugun portaera lortzeko.

DATU TRANSMISIOA ETA DATU JASOKETA GAITU

Bit hauek UCSR0B erregistroaren barruan daude, eta datu transmisioa(TX) edo jasoketa(RX) gaitzeko ahalmena daukate, bitak 1 balioa daukatenean. Biten balioa 1-era jartzeko honako hau izango da kodea:

```
UCSR0B |= (1 << TXEN0);
UCSR0B |= (1 << RXEN0);
```

USART OPERAZIO MODUA AUKERATU

Atmega328p mikrokontrolagailuak hainbat operazio modu desberdinetan funtzionatzeko konfigura daiteke, ondoko taulan ikus daitekeen bezala

Irudian antzematen den moduan, UCSR0C erregistroko UMSEL0[1:0] bitei balio desberdinak emanda aukeratuko dugu operazio modua. Gure kasuan asinkronoa erabiliko dugu, beraz UMSEL01 = 0 eta UMSEL00 = 0 izan beharko dute.

UMSEL0[1:0]	Mode
00	Asynchronous USART
01	Synchronous USART
10	Reserved
11	Master SPI (MSPIM)[1]

Figure 2: USART poerazio moduak

```
UCSR0C &=~ (1 << UMSEL00);
UCSR0C &=~ (1 << UMSEL01);
```

PARITATEA

Datuak elkartrukatzerakoan erroreak detektatzeko, paritate bitak erabiltzen dira.

UPM0[1:0]	ParityMode
00	Disabled
01	Reserved
10	Enabled, Even Parity
11	Enabled, Odd Parity

Figure 3: USART paritatea

Ikus daitekeen bezala, bi paritate mota daude, bakoitia eta bikoitia. Gure kasuan ez dugu erabiliko, beraz UPM01 = 0 eta UPM00 = 0 izan beharko dute.

```
UCSR0C &=~ (1 << UPM00);
UCSR0C &=~ (1 << UPM01);
```

STOP BITA KONFIGURATU

Hasieran aipatu den bezala, transmisio bakoitza noiz hasten den eta bukatzen den jakiteko start eta stop bitak erabiltzen dira. Stop bitaren kasuan bita bat edo bi izan daitezke. Luzeera hori USBS0 bitean balioak aldatuz zehaztuko da. Gure kasuan bateko luzeera jarriko diogu, beraz USBS0 = 0 izan beharko da.

```
UCSR0C &=~ (1 << USBS0);
```

USBS0	Stop Bit(s)
0	1-bit
1	2-bit

Figure 4: USART stop bita

DATUAK IZANGO DUEN BIT KOPURUA ZEHAZTU

Datuak izango duen bit kopurua zehazteko UCSR0C eta UCSR0B erregistroetako UCSZ0[2:0] bitei balio desberdinak emango zaizkie, nahi den luzeeraz lortzeko. Ondorengo taulan bit hauen balioak eta bakoitzari dagokion bit kopurua agertzen dira.

UCSZ0[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

Figure 5: Datu size

Gure kasuan datuak 8 bit izango ditu, horretarako UCSZ0[2:0] biten balioak 011 izan behar dute hurrenez hurren

```
UCSR0C |= (1 << UCSZ00);
UCSR0C |= (1 << UCSZ01);
UCSR0B &=~ (1 << UCSZ02);   //kasu honetan behar den azkeneko
            //bita UCSR0B erregistroan dago kokatuta.
```

BAUDRATE

Baud rate, komunikazio kanal batetik transmititzen diren datuen maiztasuna zehazten du. Atmega328p mikrokontrolagailuan UBRR0 erregistroan balio bat zehaztu behar da nahi den baud rate-aren arabera. Balio hori kalkulatzeko ondorengo formula erabiliko da.

Gure kasuan 9600-ko baud rate-a nahi dugu, beraz hau izango litzateke UBRR0-n jarri beharko genukeen balioa:

$$UBRR0 = \frac{16000000}{8 * 9600} - 1 = 207,3$$

Operating Mode	Equation for Calculating Baud Rate(1)	Equation for Calculating UBRRn Value
Synchronous Master mode	$\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{2(\mathbf{UBRR}n + 1)}$	$\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{2\mathrm{BAUD}} - 1$

Figure 6: BAUD formula

Taulan ikusten den bezala lortutako emaitza, taulan 3 lerroan eta 4 zutabean dagoen balioa da. Gure formulan ikusten den bezala baud rate-a gordetzeko erabiltzen diren 16 bitak erabili beharrean zati 8 egiten da. Hori U2X0 bita 1 balioa daukalako da. Bit hori modu asinkronoan erabiltzen da eta baud ratearen zatitzailean 16 tik 8 ra pasatzen du, transmisio maiztasuna bikoiztuz.

Honako hau izango litzateke beharrezko kodea, 207 balioa zuzenean jarriz:

```
UCSR0A |= (1 << U2X0);
UBRR0 = 207;
```

Zenbakia zuzenean jarri nahi bada, formula jarri dezakegu:

==KONTUZ: formulako biderketa jartzen badugu overflow gertetuko da, biderketako emaitza adierazteko ez dago bit nahikoa. Hori ekiditeko, zatiketak erabiliz egingo da kalkulua.==

```
#define F_CPU 16000000
#define BAUD 9600

UCSR0A |= (1 << U2X0);
UBRR0 = (F_CPU/16/BAUD)-1;
```

ASCII balioak C-n

C lengoaian karaktere baten ascii kodea lortzea nahiko erraza da, honako hau izango litzateke zenbaki baten ascii kodea lortzeko kodea:

```
zenb = '0' + 3;
printf("%d\n", zenb);   //honek ascii kodearen zenbakia
            //pantailaratuko du
printf("%c\n", zenb);   //honek ascii kodeari dagokion
            //karakterea pantailaratuko du
```

Screen serie terminalaren instalazioa eta oinarrizko erabilera

Screen, terminal multiplexore bat da. Window manager baten antzekoa, baina sesio desberdinak terminalean irekitzen utziko digu. Gure kasuan Komunikazio serialean trukatzen diren mezuak pantailaratzeko erabiliko dugu.

Instalazioa banaketa desberdinetan

Programa honen instalazioa oso simplean da, programa linuxeko banaketa gehienetako erreposito-rioetan baitago.

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	

Figure 7: BAUD table

Arch-en oinarritutako banaketetan:

```
pacman -S screen
```

Debian-en oinarritutako banaketak (Debian, Ubuntu):

```
sudo apt update
sudo apt install screen
```

Erabilera

Gure kasuan, komando bakarra erebiliko dugu egindako programako komunikazio serialean trukatzen diren mezuak bistaratzeko.

```
screen /dev/ttyACM0 115200
```

Komandoak honako parametroak dauzka:

1. portua
2. baud-rate

UART moduluko Funtzioak

UART moduluan 4 funtzio nagusi egongo dira definituta, UART-a hasieratzeko, datuak bidaltzeko eta hasieratzeko.

Hasieratu UART

UART modulua hasieratzeko Init_USART funtzioa sortu da. Parametro moduan freskatze maiztasuna pasako zaio. Defektuzko konfigurazioan Start bit bat eta Stop bit bat egongo dira mezuaren hasiera eta bukaeran. Mezuak 8 bitekoak izango dira.

Adibidea:

```
Init_USART(115200); //USART initialization with 115200 baudrate
```

Transmititu karakterea

Funtzio honen bidez UART moduluak karaktere bat transmitituko du Tx pinetik. Argumentu moduan bidali nahi den karakterea jasoko du.

Adibidea:

```
USART_tx('u'); // Send 'u' character
```

String bat transmititu

Funtzio honen bidez string bat transmitituko da. Argumentu moduan string bat jasoko du eta aurretik sortutako USART_tx funtzioa erabilita karaktereak banan-banan transmititzen ditu.

Adibidea:

```
USART_string("hello world!"); // Sends the entire sting to the receiver.
```

Karakterea jaso

Funtzio honen bidez mikrokontrolagailuak karaktere bat jasoko du funtzioa exekutatzen den momentuan. Momentuz funtzioa ez da erabiliko, datu jasoketa etenen bidez egingo delako.

Adibidez:

```
    char data = USART_rx(); //receive information in data variable
```