



01/12/2017

Compte rendu

TP SEM1



Léo

[NOM DE LA SOCIÉTÉ]

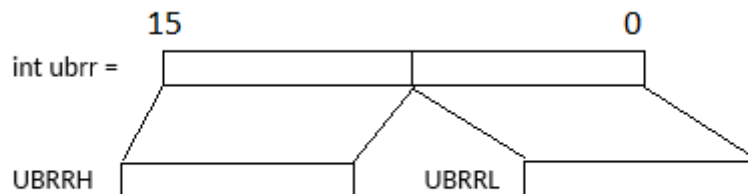
Chapitre 2 : Communication série RS-232

1. Configuration de la communication série

int ubrr =

UBRRH

 UBRL



```
#include <avr/interrupt.h>
volatile unsigned char x;
ISR(INT0_vect){
    x=1;
}

int main(void){
    DDRB=0xFF;
    DDRD=0xFB;
    MCUCR |= (1<<ISC01);
    MCUCR &= ~(1<<ISC00);
    GICR |= (1<<INT0);
    sei();
    x=1;
    while(1){
        PORTB=~x;
        x=x<<1;
        _delay_ms(300);
        if (x==0){
            x=1;
        }
    }
    return 1;
}
```

2. Envoi d'une chaîne de caractères et d'une chaîne de caractères

```
void usart_init(unsigned int debit);
void usart_putc(char c);
void usart_puts(char*s);
char usart_getc(void);

int main(void){// on initialise le serial
    usart_init(57600);//freq

    char chaine = "leo"; //nom de la chaîne
    while(1){
        usart_puts(chaine);//on appelle la fonction d'envoi de chaîne
    }
    return 1;
}

void usart_init(unsigned int debit){
    UCSRB |= (1<<RXEN) | (1<<TXEN); //def USCSRB (doc)
    UCSRC |= (1<<URSEL) | (1<<UCSZ1) | (1<<UCSZ0); //def USCSRC (doc)
    unsigned int ubrr = (F_CPU/debit/16)-1;

    UBRR1 = (unsigned char)ubrr; //on met l'octet de poids faible dans UBRR1, on prend les 8 premiers (on part de la droite)
    UBRRH = (unsigned char)(ubrr>>8);
}

//fonction 1 char
void usart_putc(char c){
    while ((UCSRA&(1<<UDRE))==0){
    }
    UDR = c;
}

//fonction chaîne de caractère
void usart_puts(char*s){
    int i=0;
    while(s[i]!='\0'){
        usart_putc(s[i]);
        i++;
    }
}
```

4. Réception d'un caractère

```
void usart_init(unsigned int debit);
void usart_putc(char c);
void usart_puts(char*s);
char usart_getc(void);

int main(void){// on initialise le serial
    usart_init(57600);//freq

    usart_puts("tapez");
    while(1){
        char c = usart_getc();//on appelle la fonction d'envoi de chaîne
        DDREB = 0xFF; // init
        PORTB = c; //on stock "c" qui va allumer les diodes selon la valeur de c (
    }
    return 1;
}

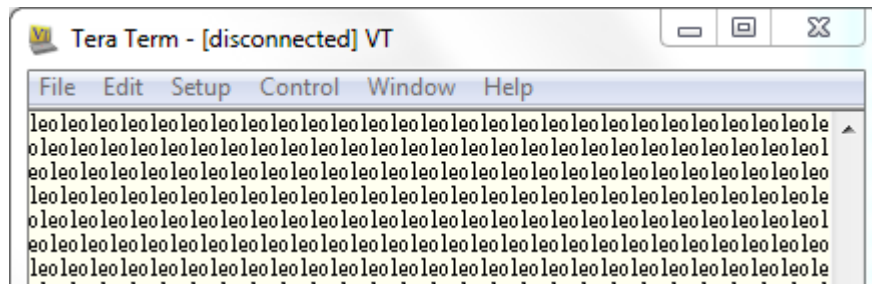
void usart_init(unsigned int debit){
    UCSRB |= (1<<RXEN) | (1<<TXEN); //def USCSRB (doc)
    UCSRC |= (1<<URSEL) | (1<<UCSZ1) | (1<<UCSZ0); //def USCSRC (doc)
    unsigned int ubrr = (F_CPU/debit/16)-1;

    UBRR1 = (unsigned char)ubrr; //on met l'octet de poids faible dans UBRR1, on prend les 8 premiers (on part de la droite)
    UBRRH = (unsigned char)(ubrr>>8);
}

//fonction 1 char
void usart_putc(char c){
    while ((UCSRA&(1<<UDRE))==0){ // tant que il n'y a pas de reception, on ne fait rien
    }
    UDR = c; //sinon on stock c
}

//fonction chaîne de caractère
void usart_puts(char*s){
    int i=0;
    while(s[i]!='\0'){ //tant que la chaîne de caractères n'est pas finie
        usart_putc(s[i]);
        i++;
    }
}

char usart_getc(void){ //on recoit sur le micro le code correspondant à ce que l'on a tapé
    while((UCSRA&(1<<RXC))==0){ //si c'est égale à 0 on fait rien
    }
    return UDR; // si ==1 on recoit la donnée
}
```



Chapitre n°3 : Télémètre à ultrasons

```
#include<avr/io.h>;
#define F_CPU 3686400
#include <util/delay.h>
void pulse(void);

int main(void){
    while(1){
        _delay_ms(50); //on attend 50ms avant de générer un nouveau pic
        pulse();
        _delay_us(12); //délai de 12ms avant de redescendre
        PORTD &=~(1<<PD2); //remise à zéro
    }
    return 1;
}
```

```
void pulse(void){
    DDRD |= (1<<PD2); //on utilise PD2 = 0 | 4
    PORTD |= (1<<PD2); //on a just PD2 en sortie
}
```

```
#include<avr/io.h>
#define F_CPU 3686400
#include <util/delay.h>
#include<avr/interrupt.h>
void pulse(void);
ISR(TIMER1_CAPT_vect);

ISR(TIMER1_CAPT_vect){
    DDRB = 0xFF;
    PORTB |= (1<<PB3);
}

void pulse(void){
    DDRD |= (1<<PD2) ; //on utilise PD2 = 0 | 4
    PORTD |= (1<<PD2); //on a juste PD2 en sortie
}

int main(void){
    TIMSK |= (1<<TICIE1); //1 par source d'interruption
    TCCR1B |= (1<<CS10)|(1<<ICES1);
    sei(); //interrupt enable global, ici il est activé

    while(1){
        _delay_ms(50); //on attend 50ms avant de générer un nouveau pic
        pulse();
        _delay_us(12); //délai de 12ms avant de redescendre
        PORTD &=~(1<<PD2); //remise à zéro
    }
    return 1;
}
```

```

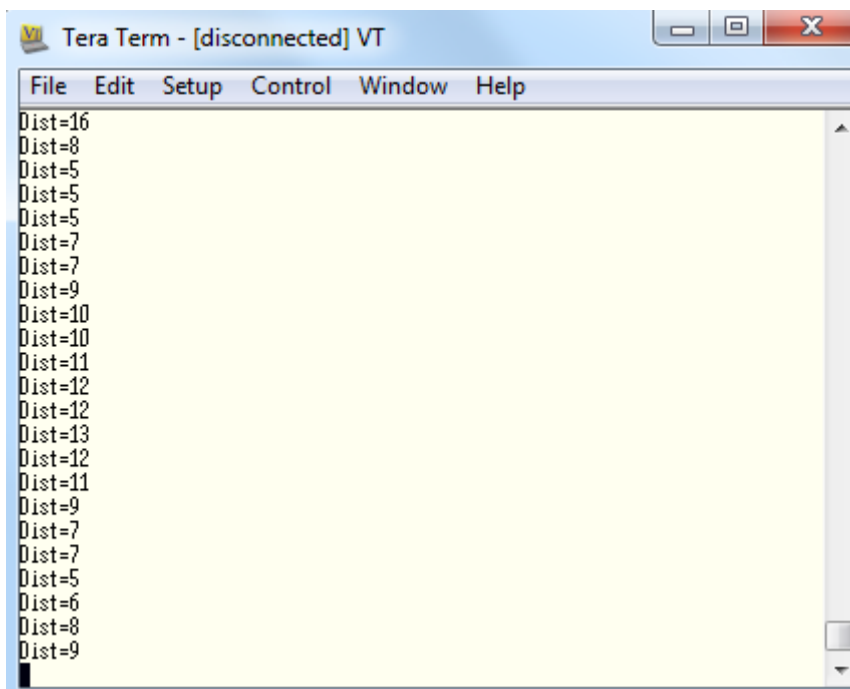
#include<avr/io.h>
#define F_CPU 3686400
#include <util/delay.h>
#include<avr/interrupt.h>
#include <stdio.h>
void pulse(void);
volatile unsigned int t1;
volatile unsigned int t2;
volatile unsigned int t;
int dist;
ISR(TIMER1_CAPT_vect){
void usart_putc(char c);
void usart_puts(char*s);
void usart_init(unsigned int debit);
ISR(TIMER1_CAPT_vect){

    if ((TCCR1B & (1<<ICES1)) == 0){

        t2=ICR1;
        t=t2-t1;
        dist = t/(3.6*58); //l'horloge est à 3.6 GHz donc il faut divisé par 3.6 pour avoir le temps
        PORTB = t/(3.6*58);
        char s[20];
        sprintf(s, "Dist=%d \n\r", dist);
        usart_puts(s);
        TCCR1B |= (1<<ICES1);
    }
    else{
        t1=ICR1;
        TCCR1B &=~ (1<<ICES1);
    }
}

void pulse(void){
    DDRD |= (1<<PD2); //on utilise PD2 = 0 | 4
    PORTD |= (1<<PD2); //on a juste PD2 en sortie
}

```



The screenshot shows a Tera Term window titled "Tera Term - [disconnected] VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The main text area displays a series of distance measurements, each on a new line, starting with "Dist=" followed by a number. The values are: 16, 8, 5, 5, 5, 7, 7, 9, 10, 10, 11, 12, 12, 13, 12, 11, 9, 7, 7, 5, 6, 8, and 9. The text is black on a light yellow background.

```

Dist=16
Dist=8
Dist=5
Dist=5
Dist=5
Dist=7
Dist=7
Dist=9
Dist=10
Dist=10
Dist=11
Dist=12
Dist=12
Dist=13
Dist=12
Dist=11
Dist=9
Dist=7
Dist=7
Dist=5
Dist=6
Dist=8
Dist=9

```