République Algérienne démocratique et Populaire

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université des Sciences et de la Technologies d'Oran « Mohamed Boudiaf »

FACULTE DE GENIE ELECTRIQUE

Département d'Automatique

TRAVEAUX PRATIQUE SYSTEME EMBARQUES.

Réalisé par :

- RAHMOUN LOKMANE NOUR EL ISLEM.
  .

Année universitaire 2019/2020

1-Introduction aux systèmes embarqués :

1-Definition d’un Système embarqué:

Embarqué signifie quelque chose lié à autre chose, et un système embarqué est un système spécialement conçu pour effectuer une tâche spécifique ou un certain nombre de tâches spécifiques, c'est-à-dire qu'il est conçu à des fins spécifiques, et c'est le contrôleur ou le processeur basé sur le système qui est conçu pour effectuer une tâche spécifique ou plusieurs tâches spécifiques.

On peut considérer que le système embarqué est un ordinateur matériel avec le logiciel intégré. Un système embarqué peut être un système autonome ou une partie d'un autre grand système, c'est-à-dire être intégré dans un produit plus grand de sorte qu'il puisse être invisible pour l'utilisateur.

Par exemple: une alarme incendie est un système compact, des machines industrielles, des appareils agricoles, des équipements médicaux, des caméras et des appareils électroménagers, ainsi que des appareils mobiles, ainsi que des routeurs et des exemples de systèmes intégrés dans un produit plus grand, un ordinateur, un lecteur DVD et une carte. Le LAN.

Les systèmes embarqués, certains d'entre eux contiennent une interface utilisateur (UI) et d'autres pas. Par exemple, les appareils qui sont conçus pour effectuer une tâche ne contiennent pas souvent d'interface utilisateur, et d'autres qui sont plus complexes et sont conçus pour effectuer un plus grand nombre de tâches telles que les appareils mobiles sont conçus Avec une interface utilisateur graphique (GUI).

(L'interface utilisateur (UI) est définie comme un ensemble de moyens par lesquels une personne interagit avec une machine).

2-Le système embarqués contient trois composants:

- Partie physique (équipement) Matériel.
- Applications logicielles d'application.
- RTOS.

(RTOS) est une abréviation de Real Time Operating System, qui est responsable de la gestion des ressources matérielles d'un ordinateur et de l'hébergement des applications s'exécutant sur cet appareil. RTOS effectue ces tâches, mais est spécialement conçu pour exécuter des applications avec un timing très précis et avec un haut degré de fiabilité. Cela peut être particulièrement important dans les systèmes de mesure et d'automatisation où les temps d'arrêt sont coûteux ou un retard de programme peut entraîner un danger pour la sécurité.

3-Caractéristiques du système embarqué:

1. Fonction unique: un système compact qui exécute généralement un processus spécialisé et fait la même chose encore et encore, c'est-à-dire qu'il répète le processus de la même manière.
2. Contraintes: tous les systèmes informatiques ont des limites sur les normes de conception, mais ceux du système embarqué ont des limites en particulier. Par exemple, la taille, la puissance et les performances, telles que les performances du système, sont suffisamment rapides pour traiter les données à RT (limite de temps) avec une consommation d'énergie minimale pour prolonger la durée de vie de la batterie.
3. Réactivité et temps (approprié) (réel) (RT): l'interactivité est que de nombreux systèmes embarqués interagissent en permanence avec les changements dans l'environnement du système et que les résultats sont calculés à temps (RT) sans aucun délai. Par exemple, les systèmes intégrés dans les voitures, notamment chargés de surveiller les vitesses et les capteurs de freinage, le calcul différé des vitesses, par exemple, peuvent entraîner des dysfonctionnements et un manque de contrôle de la voiture.
4. Mémoire: cela signifie fournir de la mémoire système, et la ROM est généralement une mémoire en lecture seule.
5. Connecté: il doit être connecté aux périphériques des périphériques d'entrée et de sortie.
6. Matériel - Systèmes logiciels: Le système est caractérisé par la présence d'un système matériel et logiciel à l'intérieur, chacun avec des tâches spécifiques.

4-Caractéristiques des systèmes embarqués:

- Facilité de personnalisation (personnalisation du système pour effectuer des tâches spécifiques).
- Faible consommation d'énergie.
- À bas prix.
- bonne performance.

L'un des principaux inconvénients des systèmes embarqués est que leurs efforts de développement et de production sont importants.

5-Architecture interne d’un AVR :

Le microcontrôleur est un ordinateur intégré sur une seule puce utilisé pour contrôler un groupe d'autres appareils. Comme tous les ordinateurs, le microcontrôleur contient les mêmes composants internes de l'ordinateur, mais avec des capacités différentes en termes de quantité et de puissance. L'image suivante représente la structure interne de l'AVR.

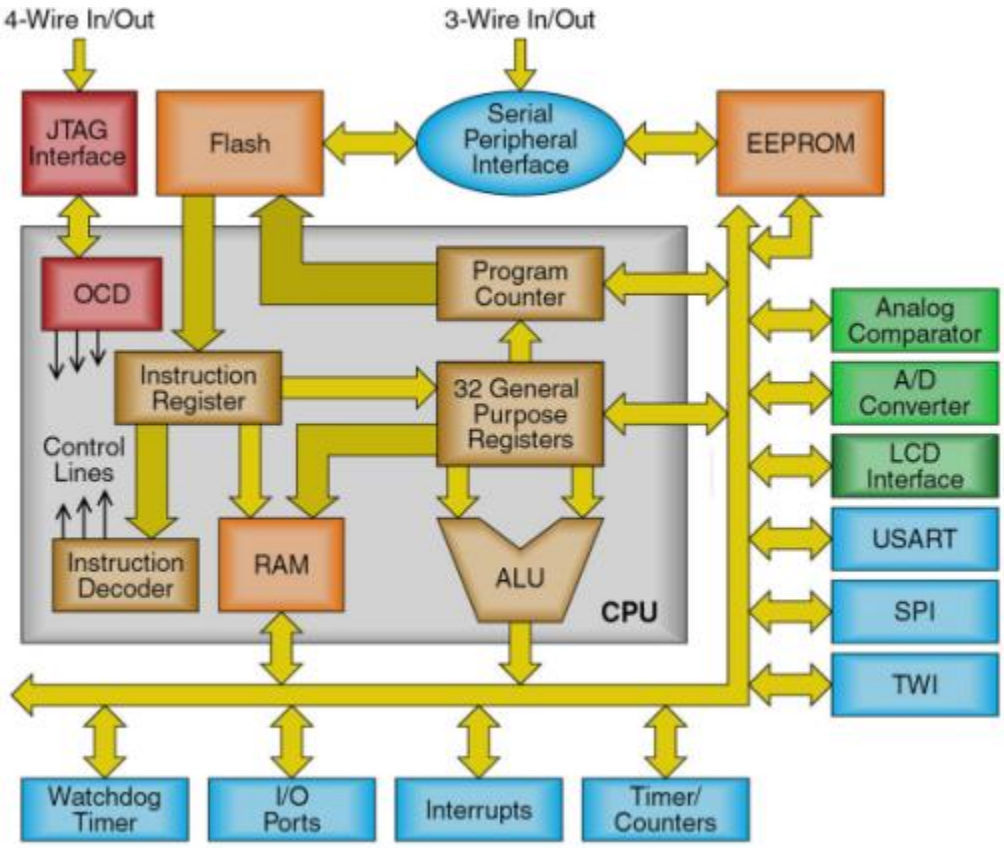


Figure 1.1 : Structure interne d'un AVR.

6-Les pines d'un ATMEGA16 :

Il contient 40 pins distribué sur 4 ports : PORTA, PORTB, PORTC et PORTD.

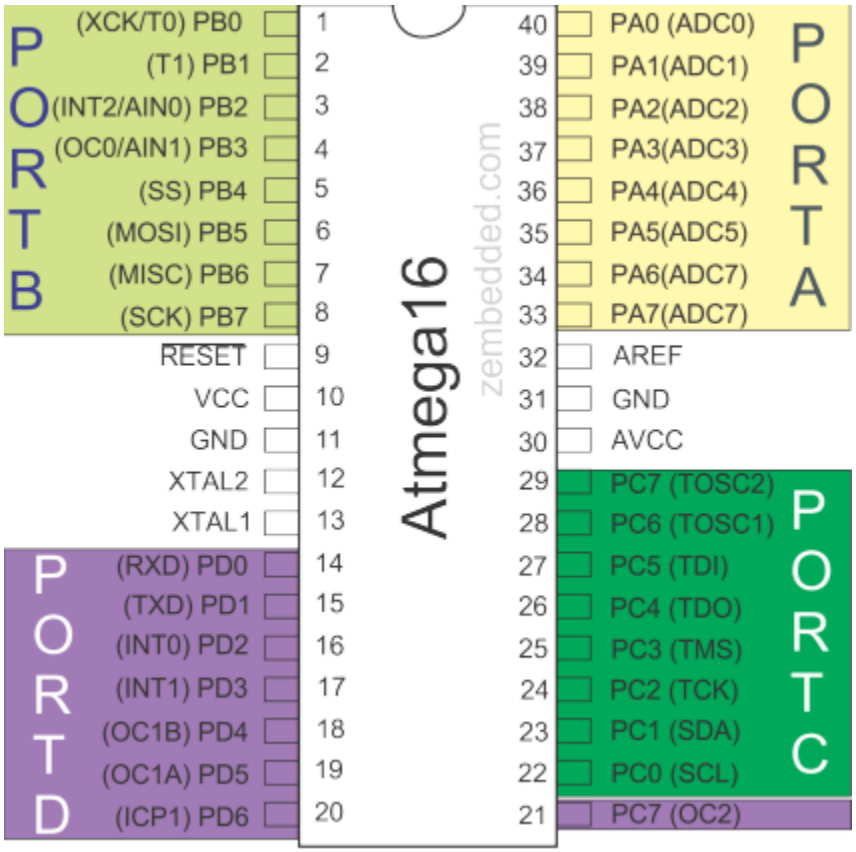
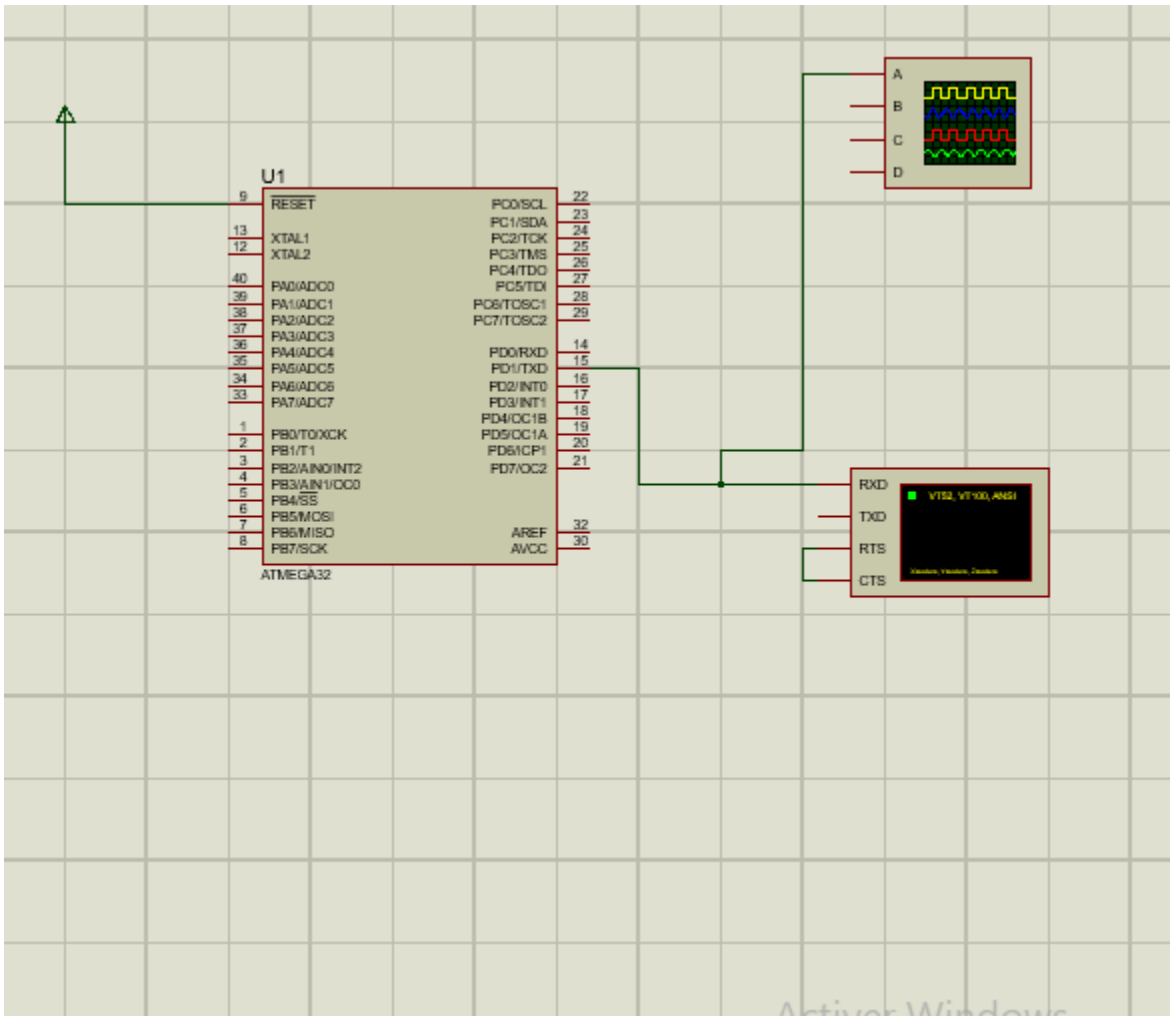


Figure 1.2 : Les pines d'ATMEGA 16.

2-Résolution de l'exercice :

Dans notre exercice il est demandé d'envoyer une lettre 'A' à partir d'un microcontrôleur ATMEGA32 avec une vitesse de transmission de 9600 Baud et une vitesse de réception du terminal en 9600 comme premier essai puis on va varier la vitesse de transmission et de réception du ATMEGA32 et de terminal pour visualiser l'influence de la vitesse de transmission et de réception sur la trame de donnés envoyé.

La première des choses on va faire un montage sur proteus pour effectuer cette simulation :



Ensuite on va charger un programme dans le microcontrôleur pour commencer cette simulation,

Dans un premier essai on va choisir une vitesse de transmission dans le programme de

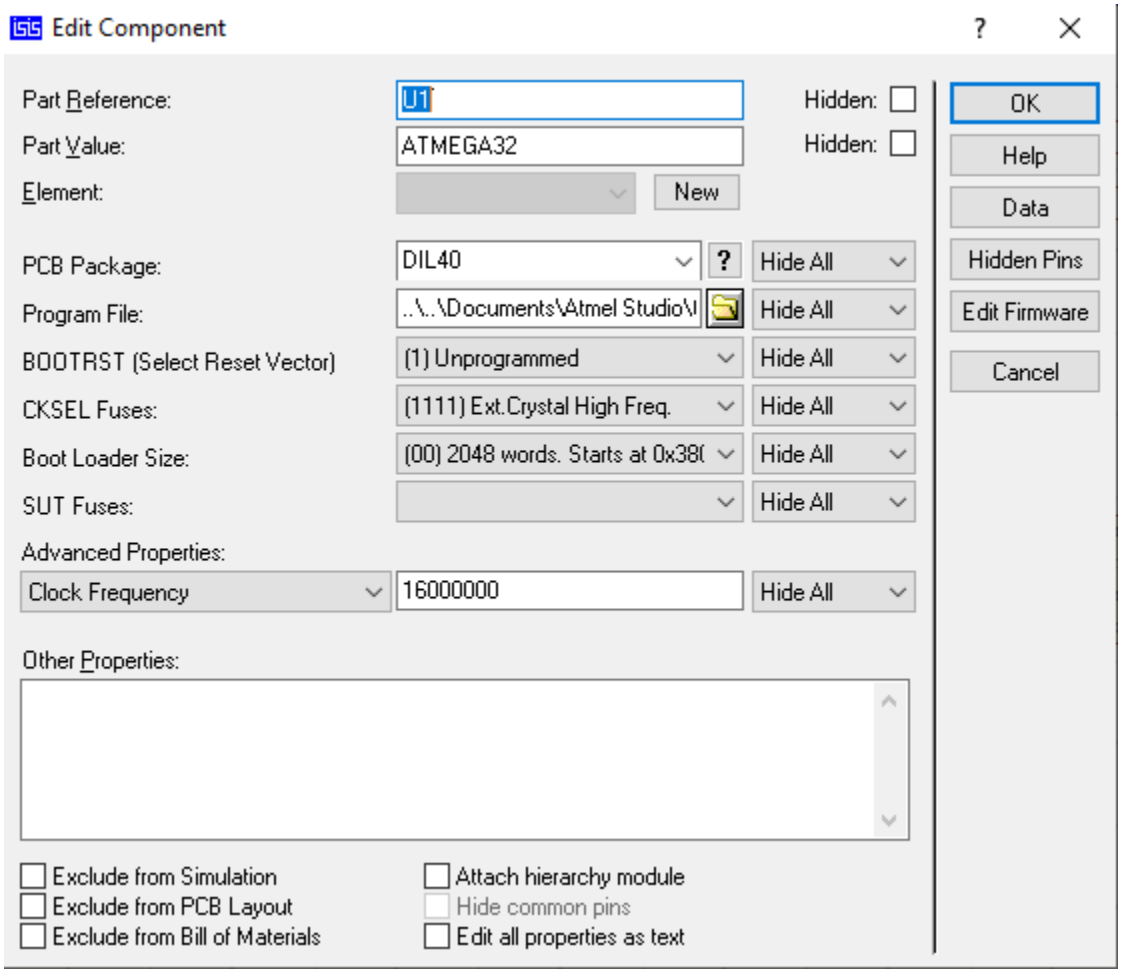
9600 BAUD et on va configurer le terminal pour 9600 Baud pour la réception.

Le programme utilisé est le suivant :

```
#define F_CPU 16000000UL // Clock Speed

#include <avr/io.h>
#include <util/delay.h>

#define BAUD 9600
#define MYUBRR F_CPU/16/BAUD-1

void USART_Init(unsigned int);
```

```c
void USART_Transmit(unsigned char);
unsigned char USART_Receive();
void USART_Transmit_String(char*);


char Welcome[]="  WelCome to USART Practice !!! \n\r\n";

int main(void)
{
	USART_Init (MYUBRR);
	// Display a Welcome Message
	USART_Transmit_String(Welcome);
	_delay_ms(10000);
	while(1)
	{
		USART_Transmit('A');
		_delay_ms(10);
	}
	return 0;
}

void USART_Init(unsigned int ubrr)
{
	/* Set baud rate */
	UBRRH = (unsigned char)(ubrr>>8);
	UBRRL = (unsigned char)ubrr;
	/* Enable receiver and transmitter */
	UCSRB = (1<<RXEN)|(1<<TXEN);
	/* Set frame format: 8data, 1stop bit */
	UCSRC = (1<<URSEL)|(3<<UCSZ0);
}

void USART_Transmit(unsigned char data)
{
	/* Wait for empty transmit buffer */
	while (!(UCSRA & (1<<UDRE)));
	/* Put data into buffer, sends the data */
	UDR = data;
}

unsigned char USART_Receive( void )
{
	/* Wait for data to be received */
	while (!(UCSRA & (1<<RXC)));
	/* Get and return received data from buffer */
	return UDR;
}

void USART_Transmit_String(char* sendString)
{
	/* Send string */
	while(*sendString != 0x00)
	{
		USART_Transmit(*sendString);
		sendString++;
	}
}
```

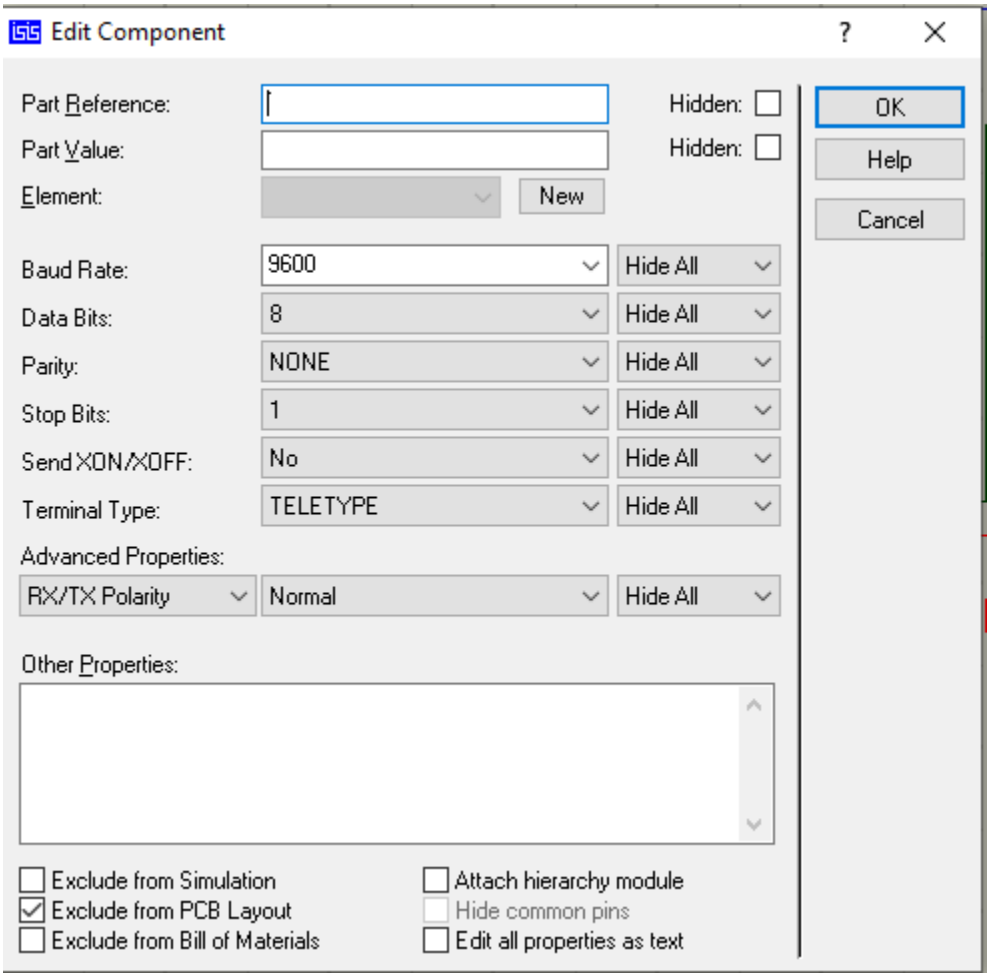
Les paramètres de configuration de l'ATMEGA sont les suivants :



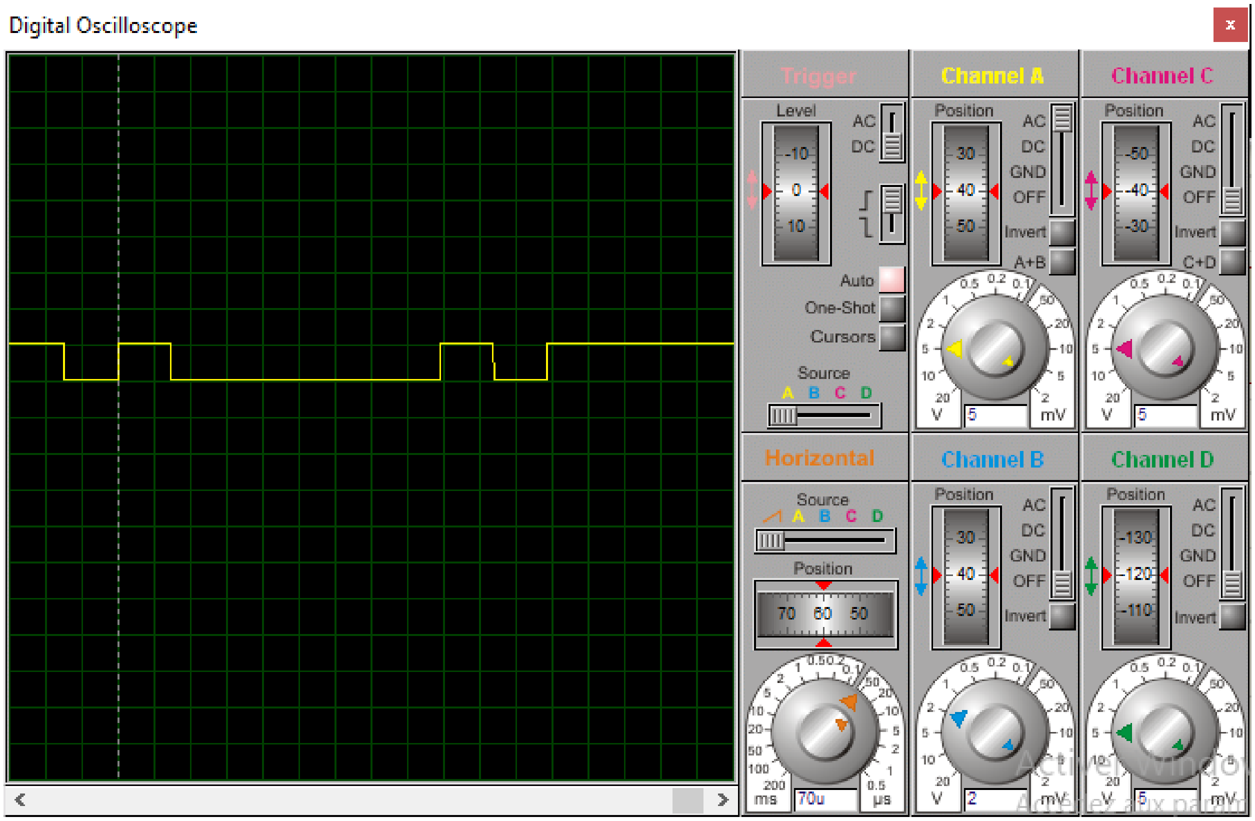
La trame transmettre par l'ATMEGA est caractérisé par :

- Un bit de START.
- 8 bits de données.
- un seul bit de stop.

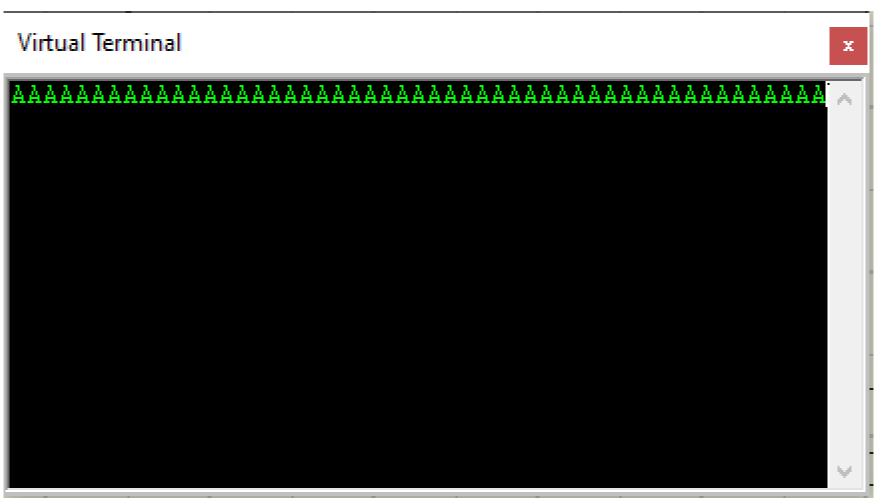
Le Virtual terminal doit être configuré de la manière suivante :



Après le lancement de la simulation le résultat obtenu est le suivant :



Cette figure présente la trame de donné transmettre par le microcontrôleur.



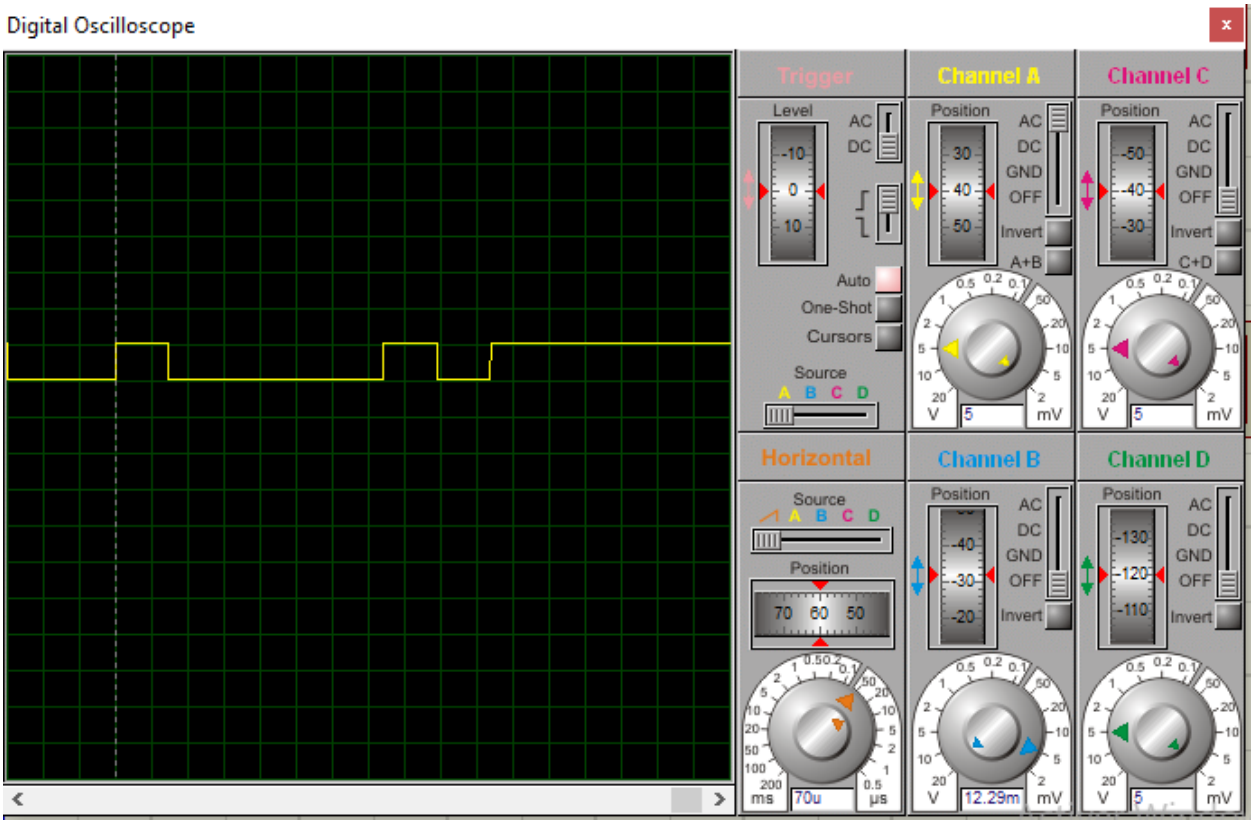
Donc on peut bien voir qu’à la réception on a obtenu le résultat désiré.

Maintenant on va changer la lettre de ‘A’ vers ‘B’ en modifiant le programme dans l’instruction suivante :

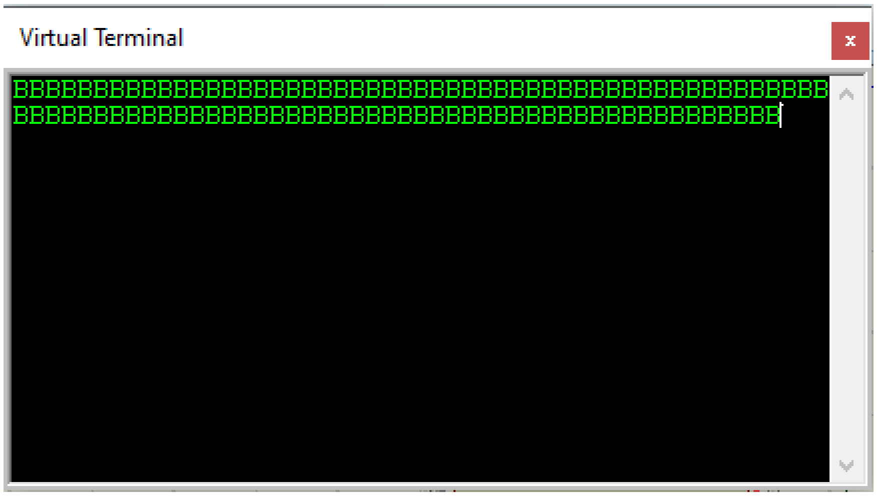
```
USART_Transmit('B');
```

Le résultat obtenu est le suivant :

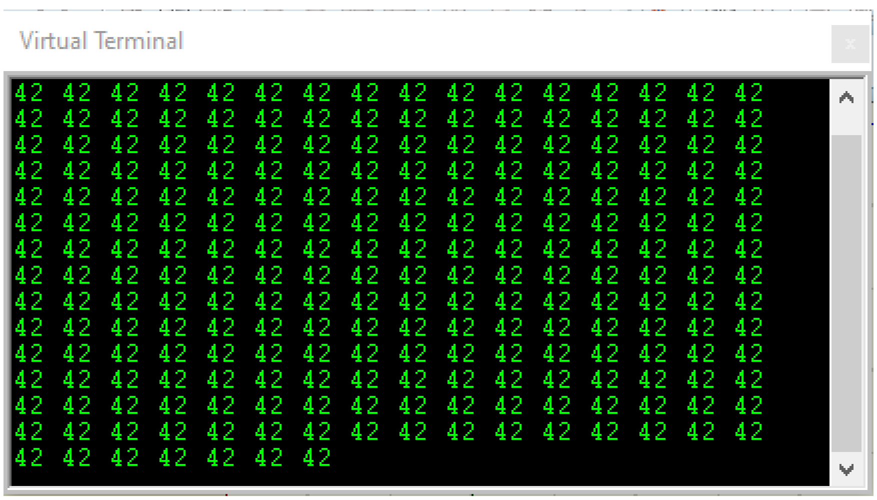
La première figure présente la trame de donné transmettre par le microcontrôleur.



La deuxième figure présente le résultat affiché dans le terminal.



ET on peut changer l'affichage en héxadécimal le résultat obtenu est le suivant :

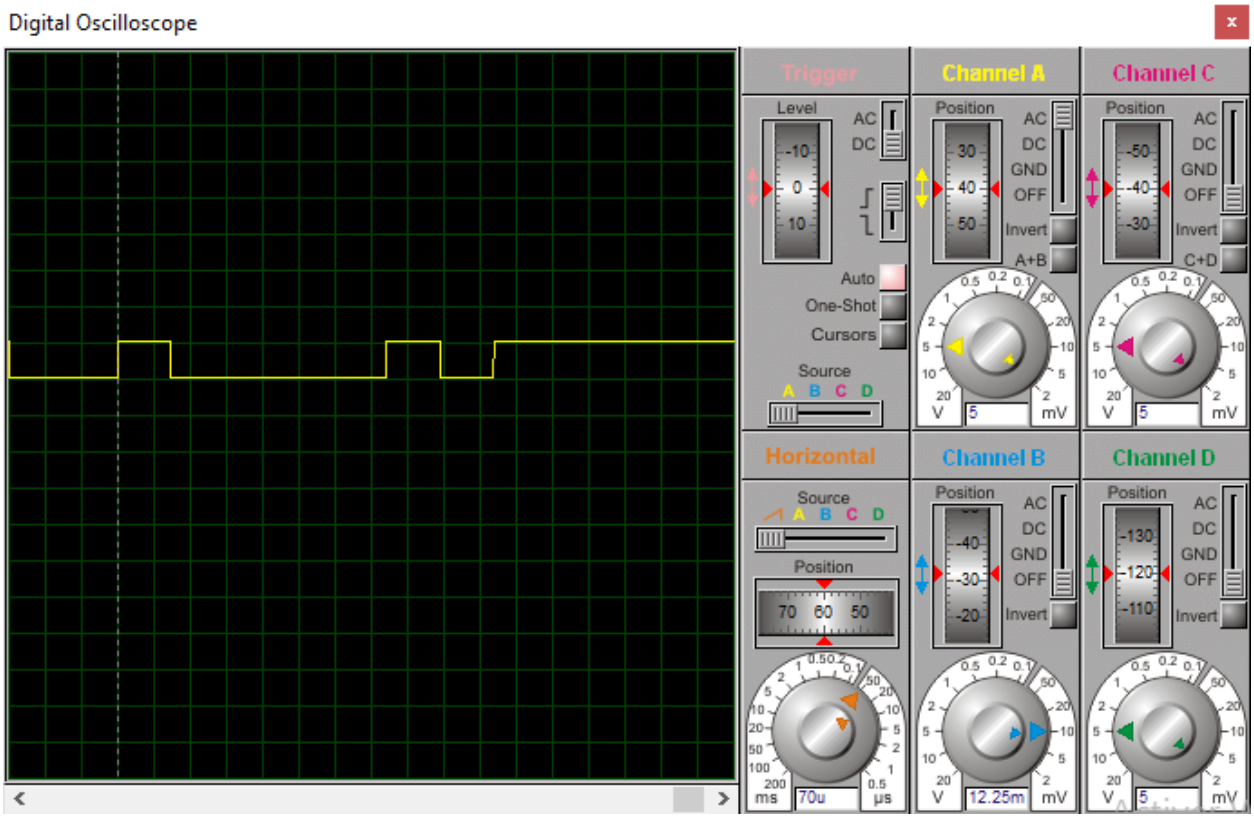


On a recu dans le terminal 0x42 ce qui est équivalent à 0b 01000010.

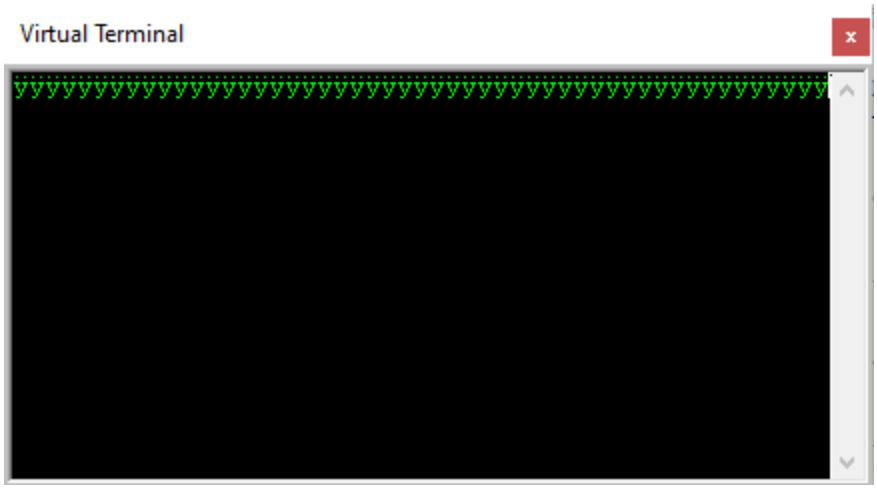
Maintenant on va changer la vitesse de réception du terminal je vais la mettre en 2400 BAUD,

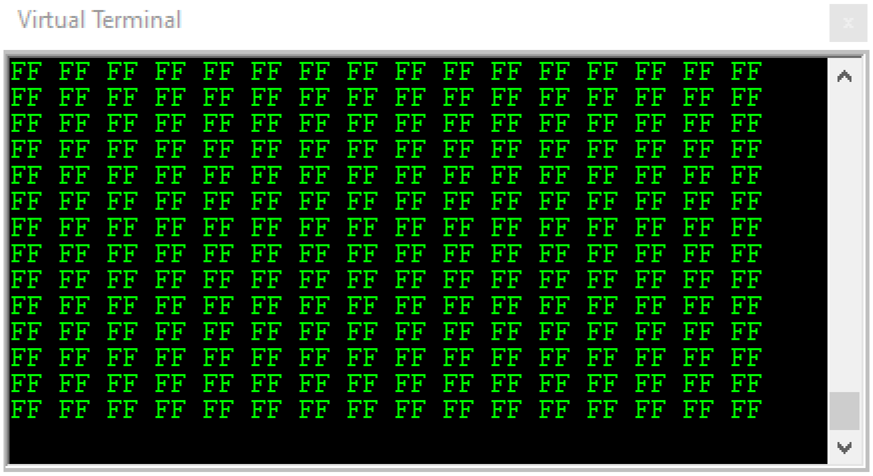
Le résultat obtenu est le suivant :

La première figure présente la trame de donné transmettre par le microcontrôleur.



La deuxième figure présente le résultat affiché dans le terminal.





On remarque qu’on reçoit plus la lettre ‘B’.

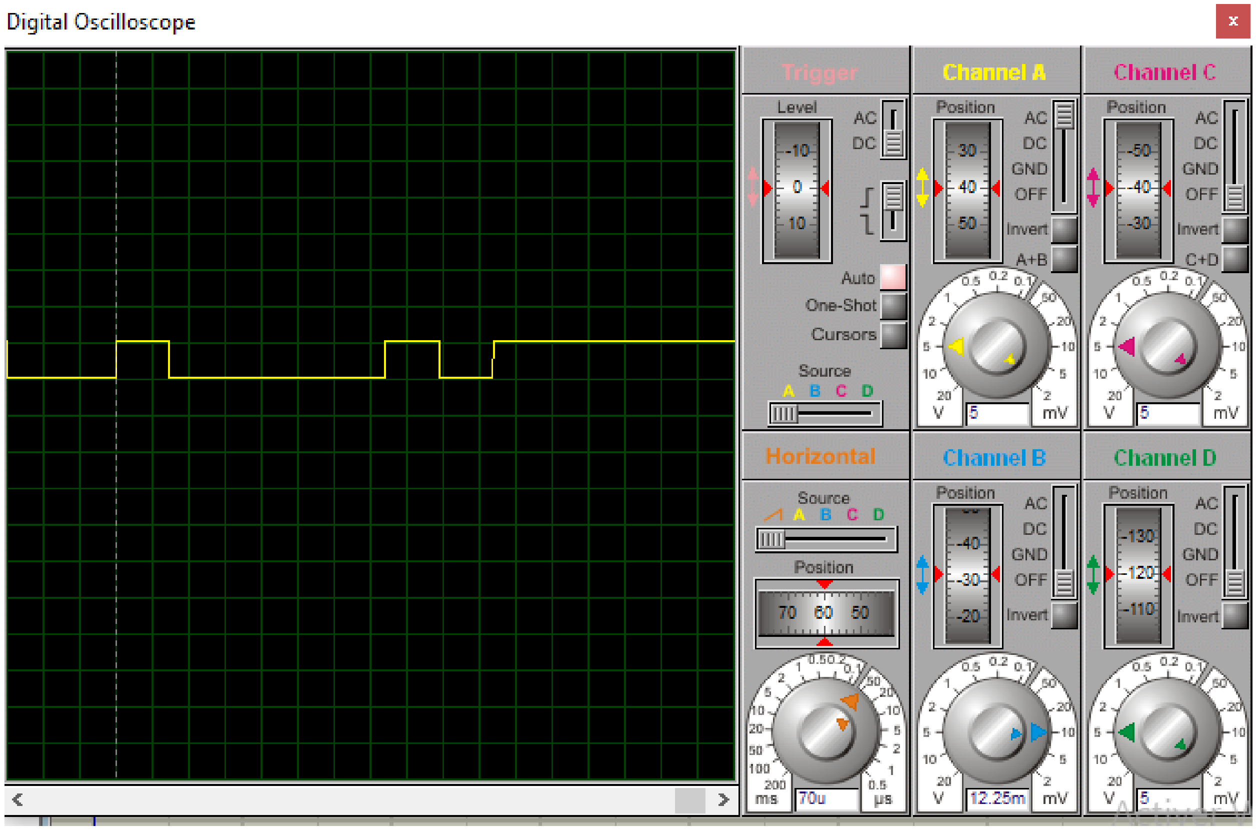
On a reçu dans le terminal 0xFF ce qui est équivalent en binaire 0b 11111111.Donc à cause du changement de la vitesse de transmission on a un changement des valeurs des bits (0, 2, 3, 4, 5 et 8) de 0 vers 1.

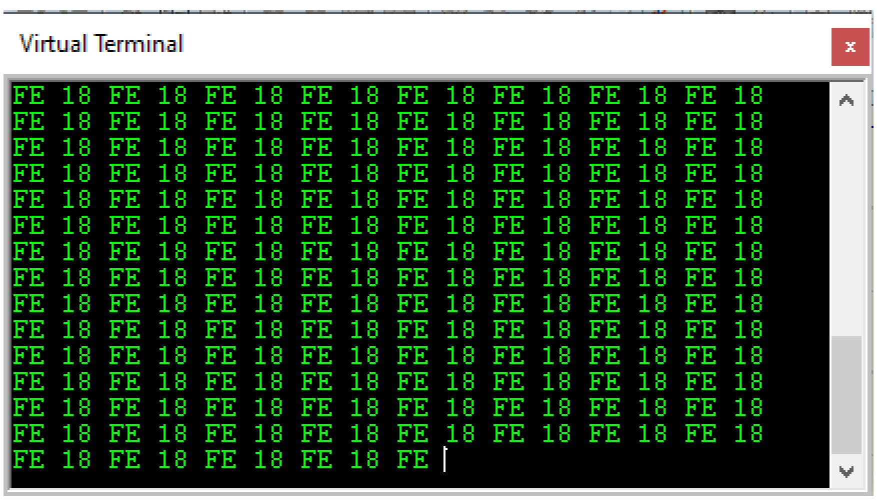
On a la vitesse d’émetteur est supérieur à la vitesse du récepteur donc on a un écrasement de donnés ce qui est appelé data over run.

Data over run : elle se produit lorsqu'un autre octet de données arrive avant même que l'octet précédent n'ait été lu à partir du tampon de réception de l'UART. Cela est principalement dû au temps mis par la CPU pour traiter l'interruption UART afin de supprimer les caractères du tampon de réception. Si la CPU ne traite pas l'interruption UART assez rapidement et ne lit pas l'octet reçu avant l'arrivée du suivant, le tampon se remplit et une erreur de dépassement se produit.

On va changer la vitesse de réception du terminal je vais la mettre en 19200 BAUD,

Le résultat obtenu est le suivant :





On a obtenu FE 18.

La vitesse de la transmission est inférieur à la vitesse de réception donc on a une duplication de donné ce qui est appelé frame error.

Frame error :

Il signifie que l'USART a vu un bas alors qu'il s'attendait à un haut (au milieu de l'impulsion de bit STOP).

Conclusion :

Dans ce TP on a pu effectuer une transmission de donnés entre un microcontrôleur et un récepteur terminal pour faire une étude simple sur la transmission des donnés.

Pour effectuer une bonne transmission de donnés il faut que la vitesse d'émetteur et la vitesse du récepteur soit la même par ce que si la vitesse de la transmission est inférieur à la vitesse de réception donc on a une duplication de donné ce qui est appelé frame error.

Dans le cas contraire si la vitesse d'émetteur est supérieur à la vitesse du récepteur donc on a un écrasement de donnés ce qui est appelé data over run.

Pour éviter l'écrasement de donné on peut appliquer quelque conseils :

1. Exécutez le CPU à la vitesse maximale possible. Cela accélérera l'exécution de l'interruption UART (et de toute autre interruption également).

2. Gardez l'UART ISR efficace et aussi court que possible. Par exemple, l'ISR pourrait simplement lire à partir du tampon RX de l'UART et le transférer dans un tampon RAM et définir un indicateur. Le traitement des données reçues pourrait se faire au premier plan. Le temps nécessaire pour exécuter l'ISR doit être inférieur au temps nécessaire pour recevoir l'octet de données suivant.

3. L'écriture d'un C ISR pousse et fait apparaître tous les registres virtuels, les registres de pagination et les registres A et X. Cela augmente le temps nécessaire pour exécuter un ISR. Écrivez l'ISR UART dans l'assemblage pour le rendre plus efficace.

4. D'autres interruptions dans le programme peuvent également créer des latences d'interruption qui pourraient entraîner des erreurs de dépassement. Dans de telles circonstances, la réduction du débit en bauds de l'UART empêchera également l'erreur de dépassement.