Enseignement Catholique
Lédonien
JEANNE D'ARC • LA SALLE • SAINTE-MARIE

Litzler Samuel

BTS "Systèmes Numériques"

1ère Année

Le 08 / 02 / 2019

Année 2018 - 2019

Timer sur microcontrôleur ATMEGA 32

Objectif **:**

Nous devons tester et réaliser un programme qui consistera à la programmation du Timer 0 sur microcontrôleur ATMEGA32 à l'aide du logiciel de virtualisation ISIS et du logiciel AVR Atmel pour la programmation

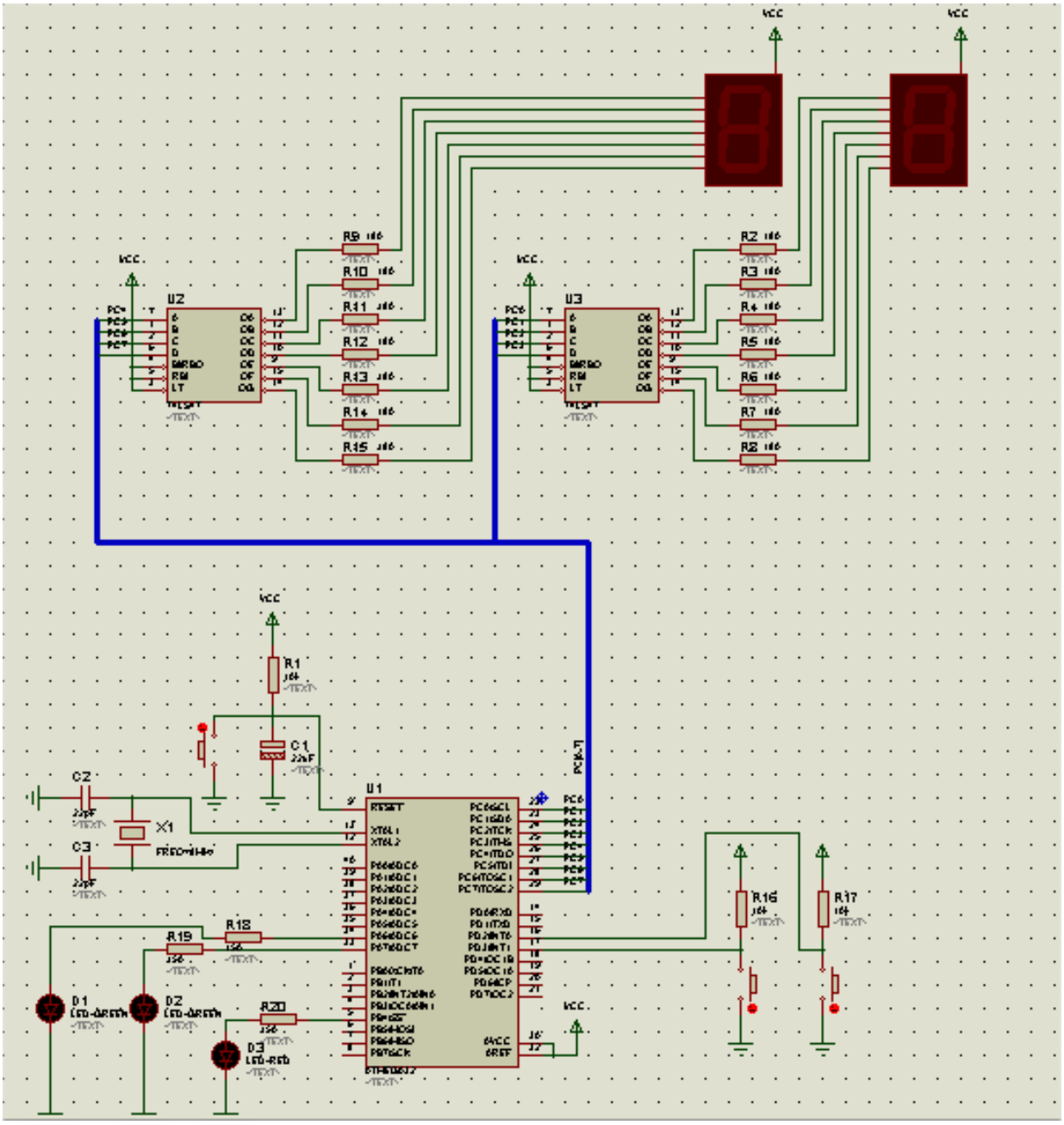
I. Réalisation d'un compteur

II. Modification de la vitesse de comptage

I. Réalisation d'un compteur

1.1 / Réalisation du schéma sur ISIS

Nous avons ci-dessous le shéma créé sur ISIS pour deux afficheurs et un controlleur ATMEGA 32, ainsi que le bouton RESET, INT0 et INT1 puis 2 LED vertes et une rouge.



II. Modification de la vitesse de comptage

On désire utiliser les deux afficheurs 7 segments en réalisant le programme ci-dessous pour compter de **0** à **99** puis qui reboucle à **0** sans utiliser la bibliothèque **« delay.h »** .

Le programme va pouvoir gérer :

- L'incrémentation du compteur qui se fera toutes les secondes d'un **RESET**.

- L'incrémentation du compteur qui se fera toutes les **3** secondes lors d'une interruption **INT0**.

- L'incrémentation du compteur qui se fera toutes les **400** millisecondes lors d'une interruption **INT1**.

On commence par déclarer nos variables, « interruption » est égale à 4000 lors d'un reset.

```
#include <mega32.h>

// Declare your global variables here
int compteur = 0x00;
int quartet;

int tempo = 0;
int interruption = 4000;     // une interruption = 250 µs
```

Ensuite, on voit programme l'interruption INT0 :

```
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
// Place your code here
  interruption = 0;    // on remet la variable à 0 pour éviter les problèmes
  interruption = 12000;      // 12000 correspond à 3 secondes car une seconde = 4000 interruptions
}
```

Pareillement pour INT1 :

```
// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
// Place your code here
  interruption = 0;    // on remet la variable à 0 pour éviter les problèmes
  interruption = 1600; // 1600 correspond à 400ms car une seconde = 4000 interruptions
}
```

Il nous reste donc le programme qui va permettre de faire fonctionner le système :

```
// Timer 0 output compare interrupt service routine
interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
// Place your code here

// une interruption toutes les 250 µs

   if (tempo == interruption)// 4000 interruptions = 1 seconde
   {
   PORTC = compteur;// compteur du PORTC

   quartet = compteur & 0x0F;                //
   if (quartet == 0x09)                      //
      {                                      //
       if (compteur == 0x99)                 //
          {                                  // permet de supprimer l'exadecimal
           compteur = 0x00;                  //
          }                                  //
       else                                  //
            compteur = compteur + 0x07;      //
      }
```

```
   else compteur = compteur + 0x01;
   tempo = 0;
   }
   else
        tempo++;
}
```

Et pour finir on dire au programme de compter :

```
while (1)
      {
      // Place your code here
      PORTC = compteur;
      }
```

Conclusion :

Durant de ce TP, nous avons fait fonctionner un **microcontrôleur ATMEGA 32** grâce aux logiciels **Code Vision AVR** et le simulateur de composants électronique **ISIS.**

Nous avons appris à coder plusieurs programmes avec différentes utilisations :

- L'incrémentation du compteur qui se fera toutes les secondes d'un **RESET**.
- L'incrémentation du compteur qui se fera toutes les **3** secondes lors d'une interruption **INT0**.
- L'incrémentation du compteur qui se fera toutes les **400** millisecondes lors d'une interruption **INT1**.