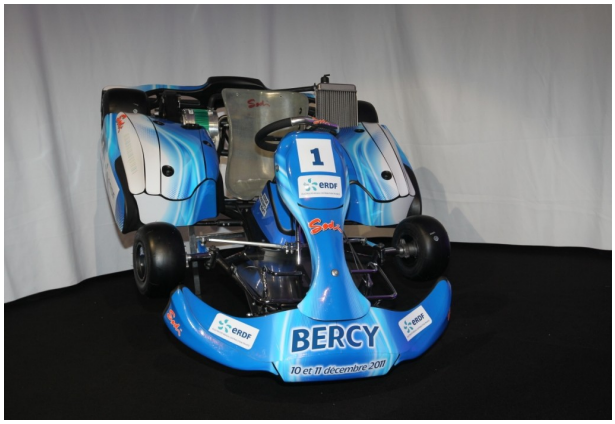
Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle



Étude et Réalisation 2ème année

Balise de mesure de temps pour l'épreuve de 50m départ arrêté

Clément TOURNILLON
Anthony BRAIN
Groupe Q2
2010/2012

Enseignant
Thierry LEQUEU
Philippe AUGER

Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

UNIVERSITE FRANCOIS-RABELAIS
TOURS
Institut Universitaire de Technologie
Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Étude et Réalisation 2ème année

Balise de mesure de temps pour l'épreuve de 50m départ arrêté

Clément TOURNILLON
Anthony BRAIN
Groupe Q2
2010/2012

Enseignant
Thierry LEQUEU
Philippe AUGER

Sommaire

Introduction

Au cours du semestre 3, nous avons choisi de continuer un projet qui n'avait pas encore été terminé. Ce projet est «Balise de mesure de temps pour l'épreuve de 50m départ arrêté». Ce projet est réalisé pour l'association e-kart de M. Thierry LEQUEU afin d'effectuer des chronométrages de karts électriques.

Le but de ce projet sera de créer une liaison entre la borne de départ et d'arrivée afin qu'elles puissent communiquer entre elles.

Après une présentation des balises de mesure de temps, nous présenterons la liaison choisie permettant la communication puis l'étude de la partie programmation.

1. Présentation des balises de mesure de temps

1.1. L'épreuve du 50 mètres départ arrêté

L'épreuve du 50 mètres départ arrêté est une épreuve de vitesse qui peut voir concourir 1 ou 2 karts simultanément. Les karts doivent donc parcourir 50 mères en ligne droite avec un départ arrêté en effectuant le meilleur temps possible.

Pour chronométrer le temps de parcours du kart, on dispose de 2 bornes, une au départ et une à l'arrivée, équipées de capteurs capables de détecter le passage d'un kart. De plus, on dispose d'un afficheur 7 segments pour afficher le temps jusqu'au centième de seconde près, étant donné que les karts parcourent ces 50 mètres en moins de 10 secondes généralement.

1.2. Cahier des charges

Ce projet est réalisé pour le compte du challenge e-Kart présidé par Thierry LEQUEU, afin de fournir un moyen de chronométrage précis et fiable au centième de seconde pour l'épreuve du 50 mètres départ arrêté.

Pour réaliser ce projet, nous devons répondre aux exigences suivantes:

- détection sans contact du passage du kart d'une portée de 3 mètres à l'aide d'un microcontrôleur.
- affichage du temps à l'aide de grands afficheurs à LED.

Pour ce faire, nous disposons d'une borne de départ et d'une borne d'arrivée qui sont capables de communiquer entre elles à l'aide d'une liaison série filaire RS232. Le dispositif devra afficher le temps en secondes et en centièmes de secondes.

Le kart sera détecté par les bornes de départ et d'arrivée à l'aide de capteurs infrarouges. Il y aura 2 capteurs sur chaque borne, le chronomètre se déclenchera lorsque le kart franchira le 2^{e} capteur de la borne de départ, après être resté 10 secondes devant le 1er capteur au préalable, et s'arrêtera lorsqu'il franchira, 50 mètres plus loin, le 1er capteur de la borne d'arrivée.

On pourra éventuellement calculer la vitesse entre les 2 capteurs de la borne d'arrivée et l'afficher sur un second afficheur.

On peut voir les différents cas sur l'illustration suivante:

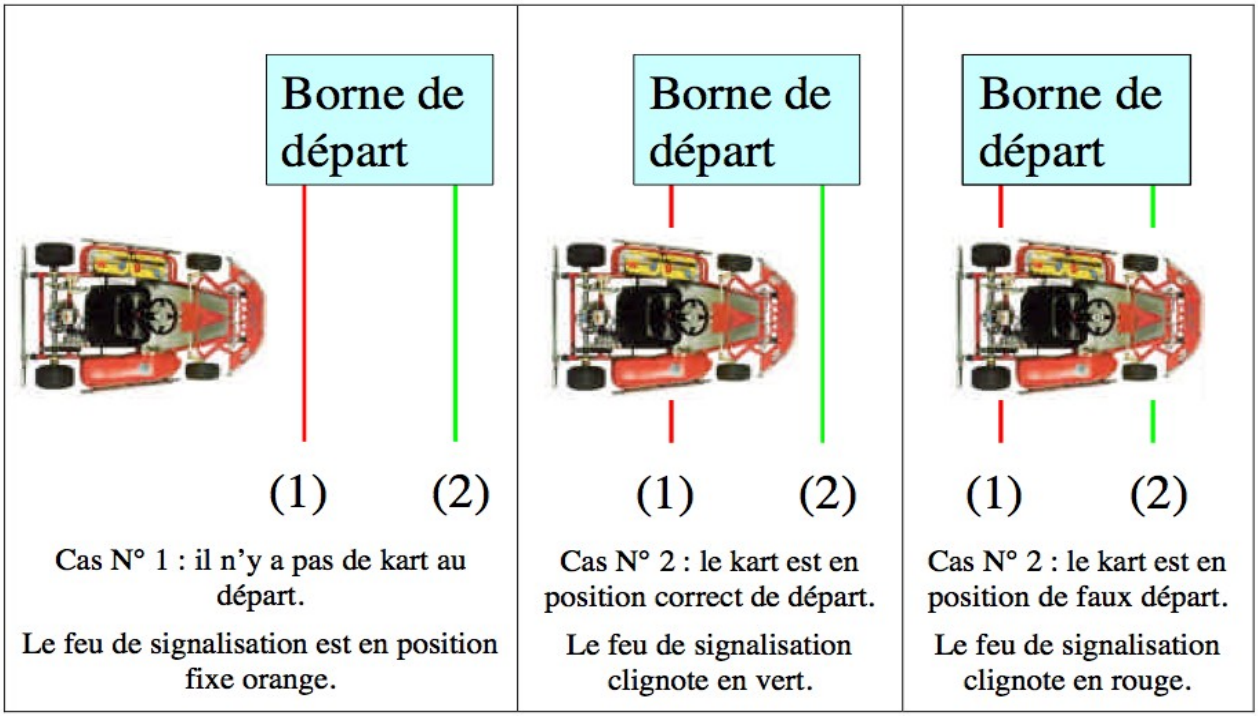


Illustration 1: Schéma de principe de la mesure du temps

On devra réaliser la programmation du microcontrôleur AtMéga8535 à l'aide du logiciel Code Vision AVR. Ce microcontrôleur pourra calculer le temps mis par le kart pour parcourir les 50 mètres, ainsi que sa vitesse, et également l'affichage de ces deux derniers sur des afficheurs à LED. On pourra tester le fonctionnement du microcontrôleur à l'aide de l'écran LCD présent sur la carte.

Nous avons effectué une planification du travail que l'on effectuera à chaque séance sous forme du planning suivant:

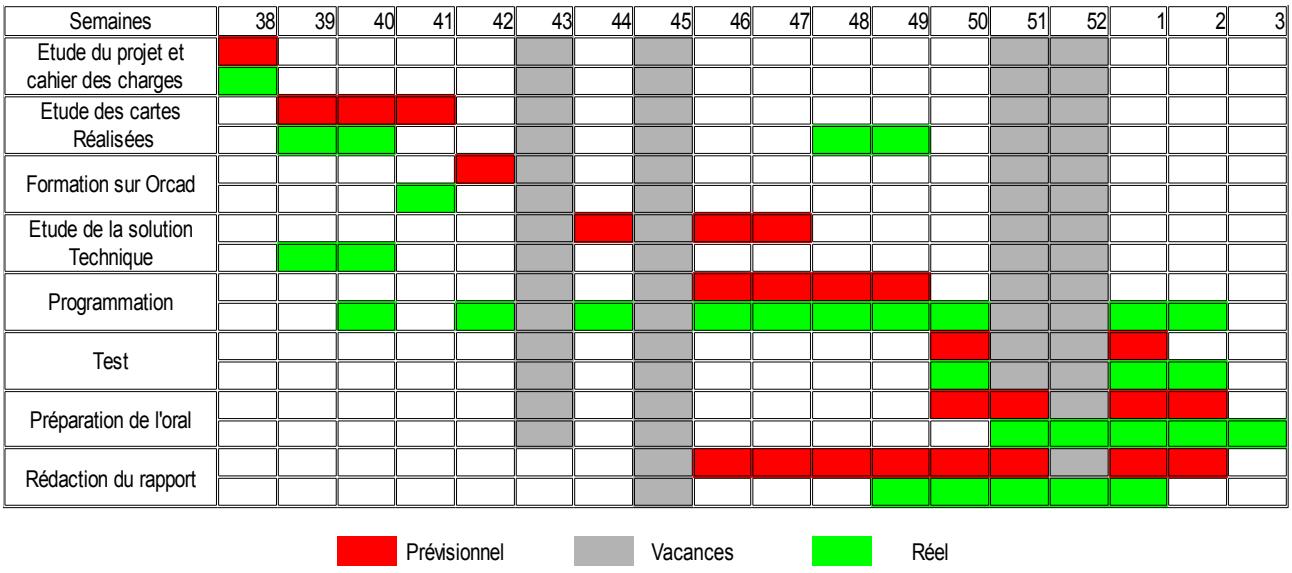


Illustration 2: Planning prévisionnel et réel

Nous pouvons remarquer que notre planning réel a été souvent différent de notre planning prévisionnel. Particulièrement l'étude des cartes et de la solution technique qui ont été effectuées

beaucoup plus rapidement que prévu, car la carte microcontrôleur était déjà réalisée et fonctionnait correctement.

En semaine 48 et 49, nous avons dû effectuer une nouvelle étude des cartes déjà réalisées car nous nous sommes rendu compte que lors de la réalisation du typon des cartes liaison série RS232 les sorties et entrées (TTL et RS232) avaient été inversées ce qui rendait leur fonctionnement impossible.

De plus, la partie programmation a été beaucoup plus longue que prévu, parce qu'on a dû tester un par un les différentes parties du projet (timer, afficheur, Liaison série RS232..) .

2. Communication entre les deux bornes

2.1. Choix du mode de communication

Il est nécessaire d'installer une liaison entre les bornes de départ et d'arrivée pour qu'elles puissent communiquer entres elles, car par exemple c'est le passage du kart devant la borne de départ qui déclenche le chronomètre qui est lui relié à la borne d'arrivée, chaque borne possédant son propre microcontrôleur et son propre programme à traiter.

Des étudiants avant nous ayant travaillé sur le même projet avaient choisi le mode de liaison HF[1] à 433MHz. Leur projet fonctionnait correctement en essai, cependant, lors du challenge e-Kart les étudiants se sont rendu compte que la communication entre les bornes était perturbée par des interférences causées par les karts électriques ainsi que les différents systèmes présents, tel que le wi-fi utilisé par les ordinateurs des juges.

Pour résoudre ce problème, M. Lequeu a donc décidé d'utiliser une liaison filaire entre les 2 bornes. Pour faire le choix de la liaison à utiliser, on sait qu'elle devra pouvoir s'étendre sur plus de 50 mètres, distance minimale entre les deux bornes. C'est pourquoi la solution la plus adaptée est la liaison série RS232, car elle peut s'étendre jusqu'à 100 mètres de distance et de plus nous avons déjà étudié ses caractéristiques et son fonctionnement en 1ère année.

Un étudiant a déjà commencé à travailler sur cette liaison et a réalisé deux cartes identiques permettant d'établir la connexion entre les deux bornes.

1 HF : Haute Fréquence.

2.2. Présentation de la carte liaison série existante

Les cartes liaison série créées par cet étudiant disposent du composant MAX232 qui est un circuit intégré capable de convertir un signal RS 232 en signal de type TTL[2] et inversement. Le constructeur donne la documentation de câblage ci-dessous:

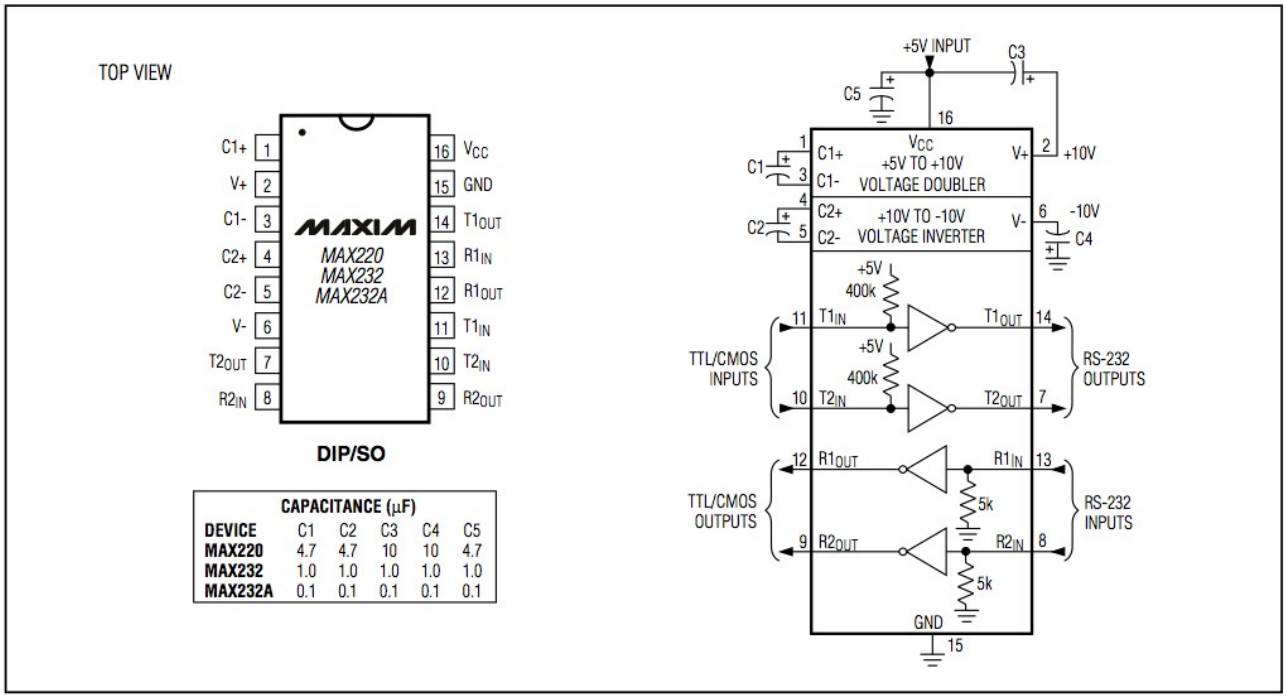


Illustration 3: Documentation constructeur du composant MAX232

On remarque que pour utiliser correctement ce composant, si l'on veut par exemple transformer un signal TTL en signal RS232, il faudra entrer sur la broche 11 et ressortir sur la broche 14 du composant, et pour effectuer l'opération dans le sens inverse, il faudra entrer sur la broche 13 et ressortir sur la broche 12.

Des borniers présents sur les cartes liaison série et sur les cartes microcontrôleur Atméga8535 permettent de les relier entre elles par l'intermédiaire de 4 fils souples, un pour l'alimentation, un pour la masse, un pour la transmission et un pour la réception.

2.3. Schéma de liaison entre les 2 bornes

A partir du schéma constructeur du composant MAX232, nous avons pu réaliser un schéma de câblage permettant de relier les 2 cartes , celle du microcontrôleur et celle de la liaison série

2 TTL : Un signal logique compris entre 0 et 5V.

Le schéma de câblage est donc le suivant:

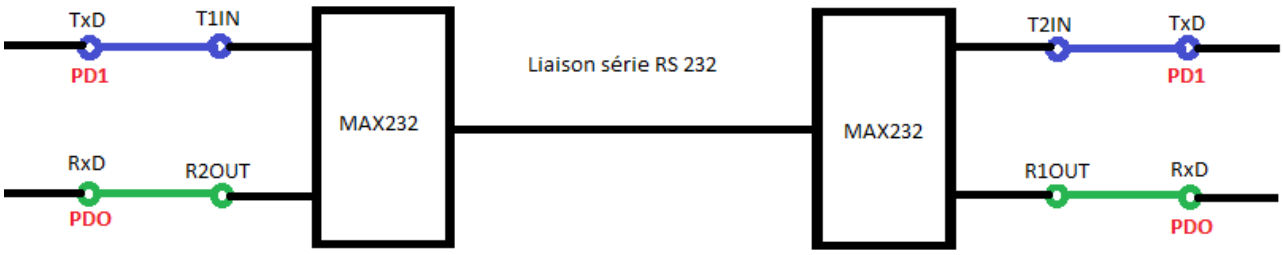


Illustration 4: Schéma de câblage de la liaison série

Sur ce schéma, PDO et PD1 font référence aux broches du microcontrôleur AtMéga8535, PDO étant la broche permettant la réception d'un caractère ou d'une chaîne de caractères et PD1 étant la broche qui permet l'envoi d'un caractère ou d'une chaîne de caractères.

Les lignes en vert et en bleu correspondent à un fil souple qui permet de relier d'un côté la carte microcontrôleur et de l'autre la carte liaison série à l'aide d'un bornier.

Le reste des connexions (celles en noires) sont déjà réalisées sur le circuit imprimé des deux cartes.

3. Programmation

3.1. Présentation du microcontrôleur AtMéga8535

Le microcontrôleur AtMéga8535 est un circuit programmable produit par ATMEL[3]. Ce microcontrôleur de type 8 bits, intègre de nombreux périphériques, ainsi que différents types de mémoire. Nous utiliserons le microcontrôleur de type DIL[4] dont le brochage est le suivant:

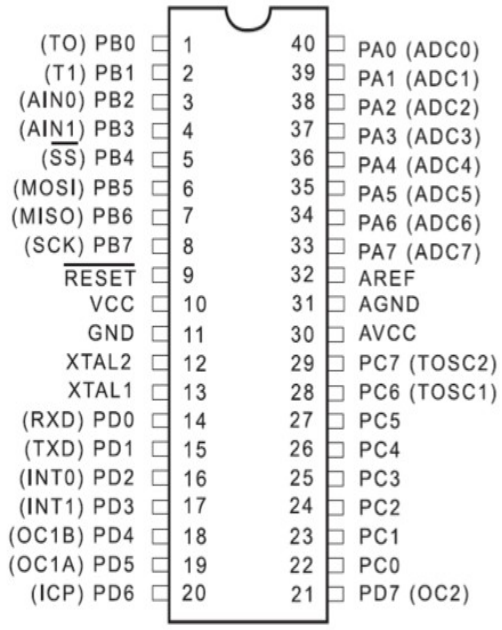


Illustration 5: Brochage de l'AtMéga8535

Il est constitué des fonctions suivantes:

- CPU 8 Bits capable d'exécuter une instruction par cycle d'horloge
- 8 Ko de mémoire programme EEPROM FLASH[5] programmable
- 512 octets d' EEPROM
- 512 octets de RAM statique
- Convertisseur Analogique Numérique 10 bits à 8 entrées multiplexées
- Liaisons séries synchrone (SPI) et asynchrone (SCI)
- 2 TIMERS 8 bits
- 1 TIMER 16 bits
- 1 Comparateur de tensions analogiques
- 2 entrées d'interruptions externes et une entrée de RESET
- 4 Ports d'entrées/sorties 8 bits

Ce microcontrôleur possède 4 ports d'Entrée/Sortie (A, B, C et D) numériques de 8 bits chacun. Chaque bit de ces 4 ports peut être configuré soit en entrée ou soit en sortie, et la

3 ATMEL : Fabricant mondial de composants à semi-conducteur.
4 DIL : Dual In Line.
5 EEPROM FLASH : type de mémoire morte qui permet la programmation de plusieurs espaces mémoires simultanément.

configuration des registres internes qui leurs sont associés permet de mettre en œuvre certaines fonctions secondaires propres à un bit. Dans notre projet par exemple, on utilise les fonctions secondaires RxD et TxD associées respectivement aux bits PD0 et PD1.

Le microcontrôleur AtMéga8535 possède 3 timers (0, 1 et 2). Les timers 0 et 2 sont des compteurs 8 bits pouvant compter de 0 à 255 alors que le timer 1 est un compteur 16 bits comptant de 0 à 65 535. Nous utiliserons donc le timer 1 car nous avons besoin de compter pendant 60 secondes au centième de seconde près, soit de 0 à 6 000 centièmes de seconde.

Il faudra aussi régler et configurer les registres de comparaison du timer ainsi que sa fréquence, définie à partir d'une horloge interne ou externe au timer.

3.2. Les différentes étapes de la programmation

Afin d'aboutir au programme complet, nous avons dû tester une à une les différentes fonctions qui le composent afin d'être sûr de leur fonctionnement et de réduire les risques d'erreurs ou de mauvaise programmation.

3.2.1. Test de la carte microcontrôleur

Pour tester l'écran LCD qui se trouve sur la carte du microcontrôleur nous avons réalisé le programme suivant:

```
#include <mega8535.h>
#include<stdio.h>
#include<delay.h>

#define Vert PIND.6                    // Bouton vert sur la carte microcontrôleur
#define Rouge PIND.7                   // Bouton rouge sur la carte microcontrôleur
#define Bleu PINB.5            // Bouton bleu sur la carte microcontrôleur

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

// Declaration des variables globales
int flag;
char tampon[16];
int C;                                 // Déclaration d'un entier C, représentant un compteur

void TestBouton(void)                  // Cette fonction permet de savoir sur quel bouton on a appuyé
{
        if( flag == 1 )                // Si le flag est à 1
        {
                C++;                   // On incrémente de 1 le compteur C
                lcd_gotoxy(0,2);
                sprintf(tampon, "Temps = %d", C);          // On affiche la valeur de C sur la 3ème ligne de l'écran LCD
                lcd_puts(tampon);
                delay_ms(500);
                return;
        }
}
void main(void)
{
// LCD module initialization
lcd_init(16);                                              // Initialisation de l'écran LCD pour 16 caractères par ligne
        while (1)
        {
                lcd_gotoxy(0,0);
                lcd_putsf("Borne d'arrivee");      // On affiche ce message sur la 1ère ligne
                lcd_gotoxy(0,1);
                lcd_putsf("Temps 50m");                    // On affiche ce message sur la 2ème ligne
                lcd_gotoxy(0,2);
                sprintf(tampon, "Temps = %d", C);          // On affiche ce message sur la 3ème ligne
                lcd_puts(tampon);
                lcd_gotoxy(0,3);
                lcd_putsf("");
                if( Vert == 0)                             // Si on appuie sur le bouton vert
                {
                        flag = 1;                          // On met le flag à 1

                }
                else if (Rouge ==0)                // Si on appuie sur le bouton rouge
                {
                        flag = 0;                          // On met le flag à 0
                }
                 else if( Bleu ==0)                        // Si on appuie sur le bouton bleu
                {
                        flag = 0;                          // On met le flag à 0
                        C = 0;                             // On met le compteur C à 0
                        lcd_gotoxy(0,2);
                        sprintf(tampon, "Temps = %d ", C);                 // On affiche la valeur de C
                        lcd_puts(tampon);
                }
                TestBouton();                                              // On appelle la fonction TestBouton
        }
}
```

Illustration 6: Programme de test de la carte microcontrôleur

Ce programme nous a permis de vérifier le bon fonctionnement des boutons et de l'affichage sur l'écran LCD. Après ces tests, nous avons pu constater que les 2 cartes microcontrôleur déjà réalisées n'avaient aucun défaut de fonctionnement.

3.2.2. Configuration du Timer

Pour configurer le Timer, nous avons repris le travail d'étude déjà réalisé par les étudiants Aurélie BESSE et Jérôme POTELLE. On a repris la configuration suivante:

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 2000,000 kHz
// Mode: CTC top=OCR1A
// OC1A output: Toggle
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x40;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x4E;
OCR1AL=0x20;
OCR1BH=0x00;
OCR1BL=0x00;
```

Illustration 7: Configuration du Timer

Le registre TCCR1B de contrôle permet d'autoriser ou non le timer. Lorsqu'on met TCCR1B = 0x0A, on autorise le timer et lorsqu'on met TCCR1B = 0x08, le timer arrête de compter.

Le registre TCNT1 est le compteur du timer, TCNT1H est le poids fort du compteur et TCNT1L est le poids faible. Comme on commence initialement à 0 secondes, alors on met le compteur à 0 ce qui donne : TCNT1H = TCNT1L = 0x00.

ICR1 est le registre de capture du timer. Pour notre projet, nous n'en avons pas besoin donc on les configure à 0 ce qui donne : ICR1H = ICR1L = 0x00.

La fréquence de l'oscillateur de la carte AtMéga8535 étant de 16MHz, l'horloge aura une période: $T = \frac{1}{f} = \frac{1}{16.10^6} = 62{,}5\,ns$

Comme nous devons compter centième par centième de seconde, le timer devra s'exécuter toutes les 10ms. Il faudra donc compter: $\frac{10.10^{-3}}{62{,}5.10^{-9}} = 160\,000\ périodes\ d'horloge.$

Mais on sait que le timer1 ne peut compter que de 0 à 65 535, pour compter 160 000 périodes d'horloge il faut donc effectuer un prédivision de la fréquence par 8, qui est la plus petite prédivision possible. La fréquence passe donc à 2MHz.

Ce qui donne une période de:

$$T = \frac{1}{2.10^{6}} = 0{,}5\,\mu s$$

Il faut donc compter : $\frac{10.10^{-3}}{0{,}5.10^{-6}} = 20\,000\ périodes\ d'\ horloge.$

Il faut donc avoir une valeur de comparaison de 20 000, ce qui se règle dans le registre de comparaison du timer1 OCR1. Comme on se sert du registre A et que 20 000 correspond à 4E20 en hexadécimal, on a pour le poids fort OCR1AH = 0x4E et pour le poids faible OCR1AL = 0x20.

Étant donné qu'on ne se sert pas du registre B, on a : OCR1BH = OCR1BL = 0x00.

La fonction que le timer exécutera tout les centièmes de seconde est la suivante:

```
unsigned char dizaine, unite, dizieme, centieme;

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
        centieme++;
        if( centieme == 10 )
        {
                dizieme++;
                centieme = 0;
                if( dizieme == 10)
                {
                        unite++;
                        dizieme=0;
                        if(unite>9)
                        {
                                dizaine++;
                                unite=0;
                                if(dizaine == 6)
                                {
                                        dizaine = 0;
                                }
                        }
                }
        }
}
```

Illustration 8: Fonction d'interruption du timer

La fonction principale est donc légèrement modifiée, on affiche désormais les variables dizaine, unité, dizieme et centieme à la place du compteur c qu'on avait créé.

```
while (1)
{
		lcd_gotoxy(0,0);
		lcd_putsf("Borne d'arrivee");
		lcd_gotoxy(0,1);
		lcd_putsf("Temps 50m");
		lcd_gotoxy(0,2);
		sprintf(tampon, "Temps = %d%d,%d%d", dizaine, unite, dizieme,centieme);
		lcd_puts(tampon);
		lcd_gotoxy(0,3);
		lcd_putsf("");
		if( CaptD == 1)
		{
			TCCR1B = 0x0A;
		}
		else if (CaptA ==1)
		{
			TCCR1B = 0x08;
		}
		else if( Bleu ==0)
		{
			lcd_gotoxy(0,2);
			dizaine = 0;
			unite = 0;
			dizieme = 0;
			centieme = 0;
			sprintf(tampon, "Temps = %d%d,%d%d", dizaine, unite, dizieme,centieme);
			lcd_puts(tampon);
		}
}
```

Illustration 9: Fonction principale

3.2.3. Gestion de l'afficheur

Pour afficher le temps de parcours du Kart, nous nous sommes servis d'un afficheur 7 segments qui était déjà conçu. En voici une photo:

Illustration 10: Afficheur 7 segments

La fonction qui permet de gérer l'affichage a été réalisée par Monsieur LEQUEU et est disponible sur son site[6]. Cette fonction est donc la suivante:

```
flash const unsigned char valeur_constant[]=
{
0xEE,0x82,0xDC,0xD6,0xB2,0x76,0x7E,0xC2,0xFE,0xF6,
0xFA,0xFE,0x6C,0xEE,0x7C,0xF8,0x7E,0xBA,0x82,0x86,
0x3C,0x2C,0xEA,0x1A,0xEE,0xF8,0xF2,0xFA,0x76,0x3C,
0x0E,0xAE,0xDC};

flash const unsigned char adresse_constant[16]=
{0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15};

void Afficheur(unsigned char adresse, unsigned char caractere, unsigned char point)
{
        if ( point == 1 )
        {
                PORTA = (valeur_constant[caractere] | 0x01);   //Un point est rajouté après le
                                                               caractère
        }
        else
        {
                PORTA=valeur_constant[caractere];      //Un caractère tout seul
        }
        PORTB=(0b00010000|adresse_constant[adresse & 0x0F]);
        PORTB.4=1;
        PORTB.4=0;
        PORTB.4=1;
        PORTA=0x00;
}
```

Illustration 11: Fonction Afficheur

Cette fonction marchait correctement dès la première utilisation, nous n'avons donc pas apporté de modification.

Dans la fonction principale, pour autoriser et donc allumer l'afficheur, il suffit d'écrire la ligne de code suivante: ENABLE=0. De même, pour ne pas l'autoriser et donc l'éteindre il faut écrire dans notre programme: ENABLE=1.

Pour afficher correctement le temps, le code est le suivant:

```
Afficheur(0,dizaine,0);
Afficheur(1,unite,1);
Afficheur(2,dizieme,0);
Afficheur(3,centieme,0);
```

Illustration 12: Appel de la fonction Afficheur

6 Adresse du site web : http://www.thierry-lequeu.fr/

Les variables passées en paramètre à la fonction Afficheur correspondent à:

- Pour le premier: 0,1,2,3 sont respectivement les premier, deuxième, troisième et quatrième digits[7].
- Pour le deuxième: dizaine,unite, dizieme, centieme correspondent à l'adresse, donc au chiffre, à afficher sur le digit concerné.
- Pour le troisième: on met 1 si l'on souhaite afficher un point après le chiffre et 0 sinon.

3.2.4. Liaison série RS232

Pour faire communiquer les 2 bornes entre elles, nous avons donc réalisé une liaison filaire RS232. Pour notre projet, nous nous contenterons d'émettre et de recevoir des caractères, en sachant que le caractère transmis sera différent suivant les cas de figure ayant lieu devant la borne de départ (faux départ, kart présent ou non, attente de 10 secondes). La configuration de cette liaison dans notre programme se réalise de la manière suivante:

```
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;
```

Illustration 13: Configuration de la liaison série

Les lignes de codes et les valeurs affectées aux variables ci-dessus sont créées automatiquement en passant par le menu du logiciel CodeVisionAVR et en réglant les paramètres de la liaison série. Les paramètres de cette liaison sont:

- 8 bits de données, 1 bit de stop, pas de bit de parité
- Réception et Transmission du microcontrôleur activées
- Horloge Asynchrone[8]
- Débit : 9600 bauds[9]

Les fonctions pour émettre et recevoir un caractère sont également écrites sur le site de Monsieur LEQUEU. La fonction permettant l'envoi d'un caractère est la suivante:

```
void USART_Transmit( unsigned char data )
{
/* Wait for empty transmit buffer */
while ( !( UCSRA & (0x20)) ) // Test de UDRE bit 5
;
/* Put data into buffer, sends the data */
UDR = data;
}
```

Illustration 14: Fonction d'émission d'un caractère

7 Un digit est l'emplacement d'un chiffre sur l'afficheur.
8 Asynchrone: Les parties émission et réception n'utilisent pas la même horloge.
9 Bauds = bit/s

La fonction qui permet la réception d'un caractère est la suivante:

```
unsigned char USART_Receive( void )
{
/* Wait for data to be received */
while ( !(UCSRA & 0x80) ) // Test de RXC bit7
;
/* Get and return received data from buffer */
return UDR;
}
```

Illustration 15: Fonction de réception d'un caractère

Il faudra donc également créer une variable globale[10] de type unsigned char. Dans la fonction principale, l'appel de la fonction d'émission se code de la manière suivante:

```
USART_Transmit(car);    //On passe en paramètre à la fonction la variable car qu'on a créée
```

Quand à la fonction de réception, elle s'appelle de la manière suivante:

```
car=USART_Receive();    //On recopie dans la variable car le caractère que l'on a reçu
```

Nous avons donc ensuite testé les fonctions d'émission et de réception, et nous nous sommes rendu compte qu'elles ne fonctionnaient pas. Pour découvrir le problème, nous avons donc utilisé l'oscilloscope pour suivre point par point suivant les liaisons si le caractère était toujours présent à partir de la broche d'émission PD1. Nous nous sommes ainsi rendu compte que le caractère atteignait bien la carte de la liaison série, mais qu'une fois rentré dans le composant MAX232 il ne ressortait pas.

Nous avons donc réétudié le câblage de la carte liaison série, et en le comparant avec la documentation technique du composant MAX232 nous avons constaté que l'étudiant ayant réalisé ces 2 cartes avait inversé les branchements entre l'émission et la réception. C'est à dire qu'il rentrait sur la broche 11 pour ressortir sur la broche 12 au lieu de la 14, et pour l'opération inverse il rentrait sur la broche 13 pour ressortir sur la broche 14 au lieu de la 12.

Pour que la carte marche correctement, il nous suffisait donc juste d'inverser 2 pistes du circuit imprimé pour retrouver les bonnes connexions. Afin de résoudre ce problème, nous avons coupé au cutter les pistes du circuit imprimé qui étaient mal reliées puis nous avons effectué des vias[11] afin de croiser les 2 pistes et ainsi rétablir une connexion correcte.

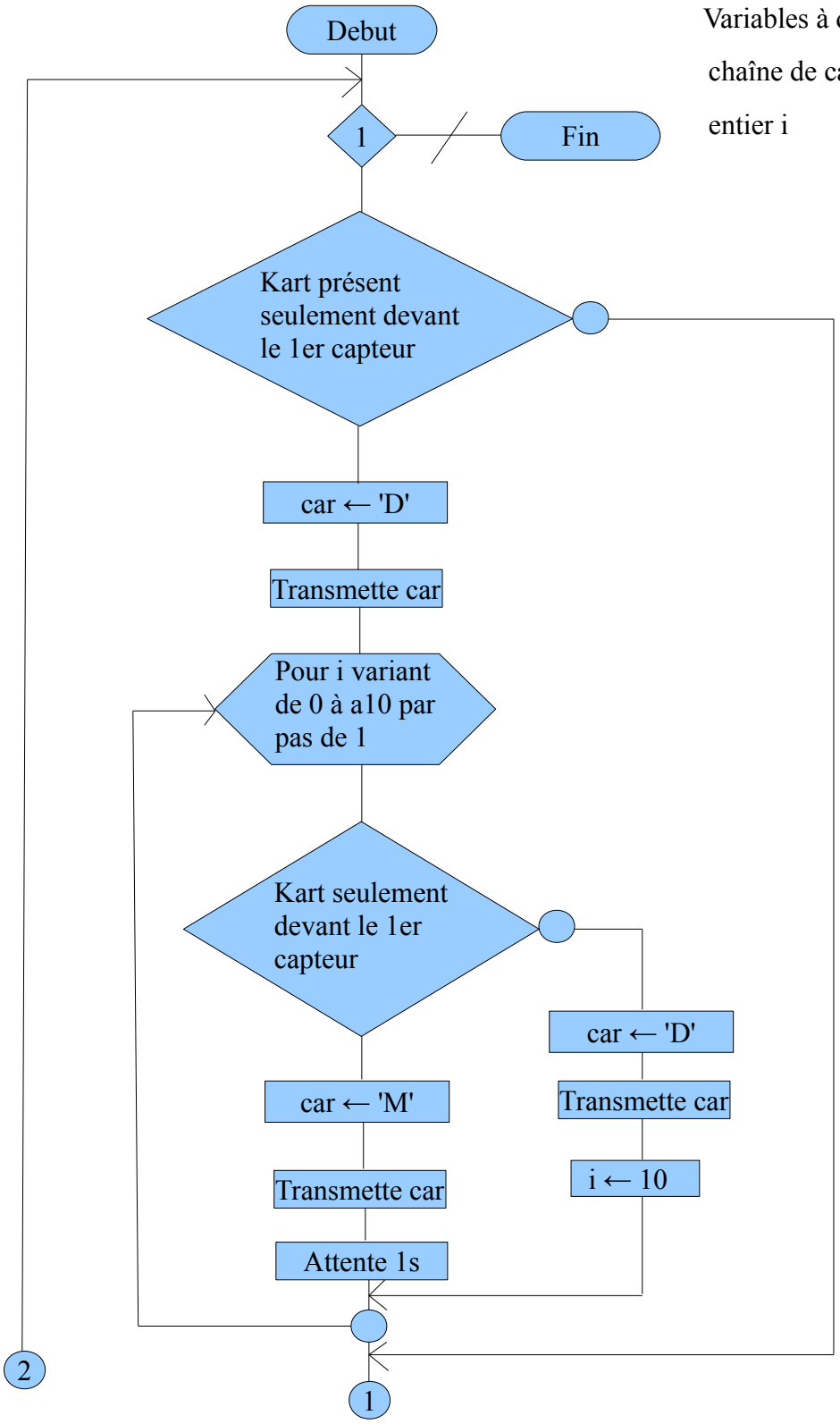
Finalement, nous avons testé les cartes avec les modifications effectuées, et nous nous sommes rendu compte qu'elles fonctionnaient correctement et que nous étions capables d'émettre et de recevoir un caractère entre les 2 microcontrôleurs.

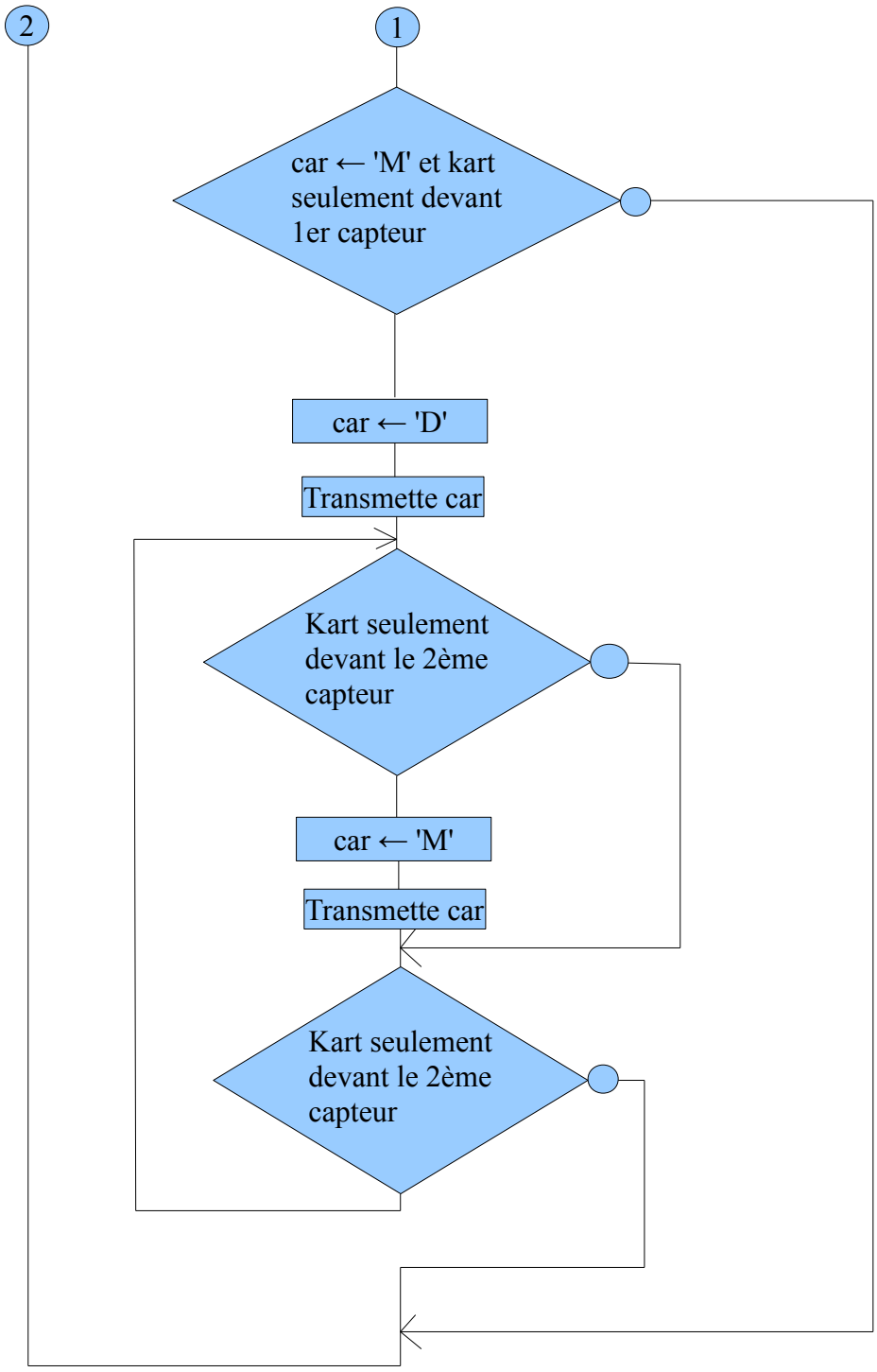
10 Variable globale: C'est une variable qui n'est pas propre à une fonction mais à tout le programme.
11 Vias: fil souple connecté entre 2 points du circuit imprimé.

3.3. *Fonction principale*

Maintenant que nous avons vérifié le bon fonctionnement des différentes fonctions utiles au programme complet, nous pouvons commencer à réaliser la fonction principale afin que le programme réponde aux attentes du cahier des charges.

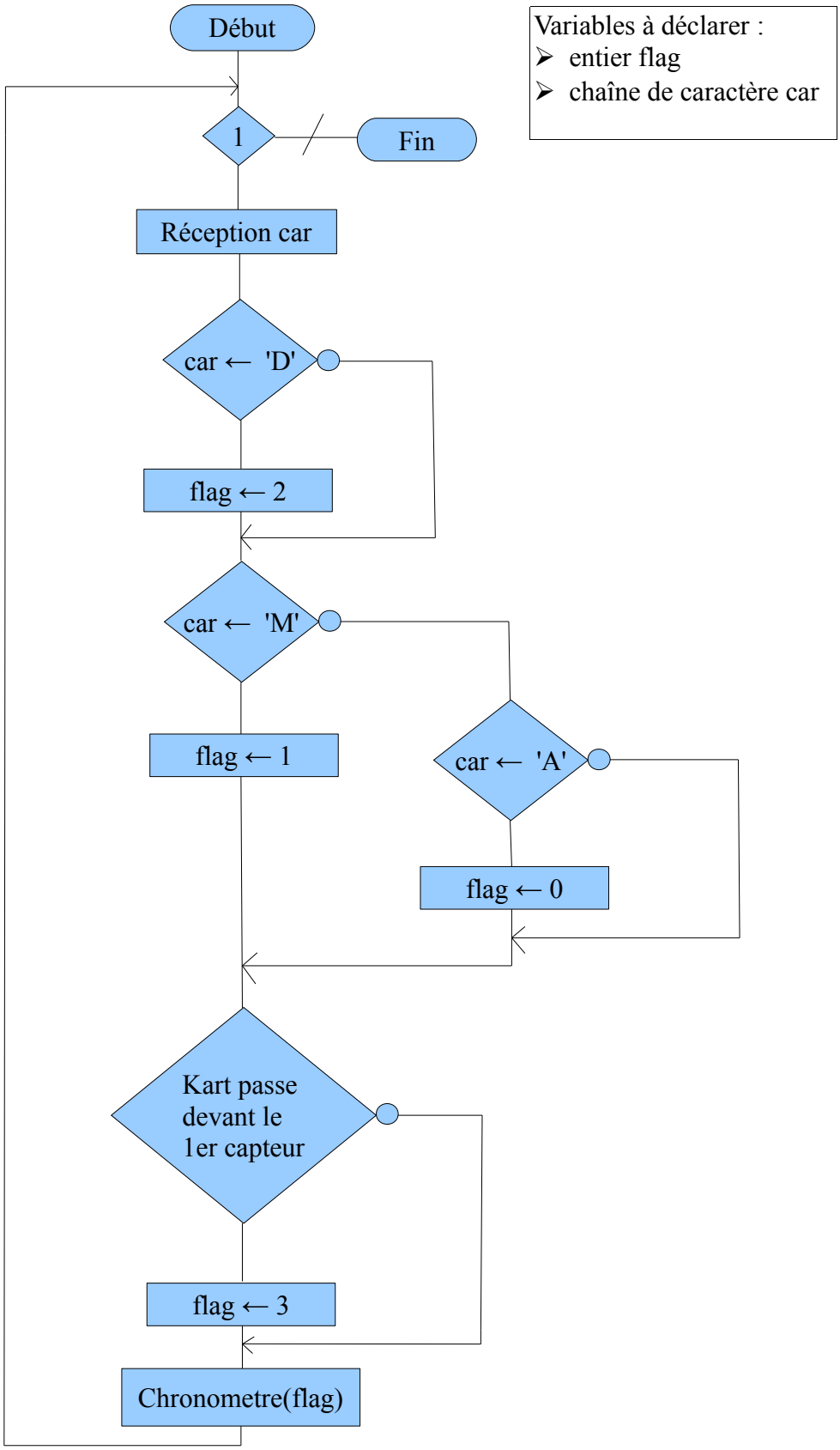
L'ordinogramme de la fonction principale du microcontrôleur relié à la borne de départ et permettant la transmission de caractères est:





Le programme complet du microcontrôleur en langage C relié à la borne de départ se trouve en annexe 1.

La fonction principale du microcontrôleur relié à la borne d'arrivée et permettant la réception des caractères a pour ordinogramme:



Le programme complet du microcontrôleur relié à la borne d'arrivée se trouve en annexe 2.

Conclusion

Ce projet nous a permis de mettre en application toutes nos connaissances en informatique, et en particulier de mettre en œuvre la liaison RS232, que nous avons étudié dans le cadre du semestre 2. Nous avons aussi appris à étudier des cartes électroniques et des circuits imprimés déjà existants à partir de la documentation technique fournie par le constructeur ainsi que du rapport de projet de l'étudiant ayant réalisé la carte.

Sur le binôme, il y en avait seulement un seul qui connaissait le logiciel CodeVisionAVR et donc son utilisation ainsi que la manière de programmer un microcontrôleur. Ce projet a donc permis un échange de connaissance au sein du binôme, chacun apportant ses connaissances personnelles dans le but d'atteindre les objectifs du cahier des charges.

La méthode retenue pour réaliser ce projet, qui était de tester point par point chaque fonction du programme ainsi que les éléments électroniques utilisés dans le projet (carte électronique, afficheur, bornes), s'est avérée longue; mais c'est ce qui nous a permis de réussir notre projet et de faire fonctionner la liaison série. Pour la dernière semaine, il ne nous reste plus qu'à mettre en forme la fonction principale afin qu'elle réponde exactement au cahier des charges imposés pour le challenge e-Kart.

Résumé

Dans le cadre du semestre 3, nous avons fait le choix du projet «Balise de mesure de temps pour l'épreuve de 50m départ arrêté» en Études et Réalisation. Nous avons tous les 2 choisis ce projet car il était centré surtout autour de l'informatique, ce qui est notre préférence comparé à l'électronique, et le fait d'avoir un « réel » client, le challenge e-Kart, rajouté encore plus de réalité et d'envie de mener à bien notre projet afin de le voir fonctionner en situation réelle durant le challenge.

Nous avons passé pas mal de temps durant ce projet a étudier le travail déjà effectué par des étudiants avant nous, afin de savoir nous en servir correctement et pouvoir résoudre les problèmes rencontrés.

Comme certaines fonctions utiles à notre programme avaient déjà été réalisées par Monsieur LEQUEU et qu'elles se trouvaient disponibles sur son site, nous avons gagné du temps dessus, ce qui nous a permis d'étudier plus en détails les composants électroniques utilisés dans ce projet à partir de leur documentation technique.

La plus grosse partie de notre projet a été la réalisation de la liaison série RS232, car nous avons rencontré différents problèmes durant ces tests. Le plus gros problème rencontré étant la carte liaison série qui n'était pas correcte de par son circuit imprimé, nous avons heureusement pu le résoudre rapidement en « bricolant » la carte, comme couper au cutter certaines pistes et mettre des vias pour croiser les pistes.

Finalement, nous avons réussi à faire communiquer les 2 bornes entre elles par cette liaison série RS232. Il ne nous reste plus désormais qu'à tester cette communication avec une liaison filaire sur plus de 50 mètres afin d'être certains qu'elle ne sera pas parasitée par l'environnement l'entourant sur cette distance, et qu'elle sera donc utilisable durant une épreuve 50 mètres départ arrêté.

305 mots

Index des illustrations

Bibliographie

[1] **LEQUEU Thierry,** *La documentation de Thierry sur OVH*, (page consultée en 2011) < http://thierry-lequeu.fr/ >

[2] **BESSE Aurélie, POTELLE Jérôme,** *Balise de chronométrage pour l'épreuve de 50m départ arrêté*, 2007/2009

[3] **MAUGER Vincent,** *Bornes 50 mètres Départ / Arrêté*, 2010/2011

Annexes

Annexe n°1 : Programme complet du microcontrôleur relié à la borne de départ

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.2c Professional
Automatic Program Generator
© Copyright 1998-2004 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project :
Version :
Date    : 03/10/2011
Author  : F4CG
Company : F4CG
Comments:


Chip type           : ATmega8535
Program type        : Application
Clock frequency     : 16,000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 128
*****************************************************/

#include <mega8535.h>
#include <stdio.h>
#include <delay.h>

#define Vert PIND.6
///#define Rouge PIND.7
#define Bleu PINB.5
#define CaptD PIND.4
#define CaptA PIND.5
#define ENABLE PORTD.7

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

// Declare your global variables here
unsigned char car;

void USART_Transmit( unsigned char data )
{
/* Wait for empty transmit buffer */
while ( !( UCSRA & (0x20)) ) // Test de UDRE bit 5
;
/* Put data into buffer, sends the data */
UDR = data;
}
```

```
void main(void)
{
// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTA=0x00;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0x00;
DDRB=0x1F;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=Out Func6=In Func5=In Func4=In Func3=In Func2=In Func1=Out Func0=In
// State7=1 State6=T State5=T State4=T State3=T State2=T State1=0 State0=T
PORTD=0x80;
DDRD=0x82;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 2000,000 kHz
// Mode: CTC top=OCR1A
// OC1A output: Toggle
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x40;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x4E;
OCR1AL=0x20;
OCR1BH=0x00;
OCR1BL=0x00;
```

```
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;


// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x10;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);
lcd_gotoxy(0,2);
lcd_putsf("Q2 - Brain");
lcd_gotoxy(0,3);
lcd_putsf("Tournillon");

while (1)
     {
      int i;
      if( CaptD == 0 && CaptA == 1)
      {
              car = 'D';
         USART_Transmit(car);
         for(i=0;i<10;i++)
         {
            if( CaptD == 0 && CaptA == 1)
            {
               car = 'M';
               USART_Transmit(car);
            }
            else
            {
               car = 'A';
               USART_Transmit(car);
               i=10;
            }
         delay_ms(1000);
         }
```

```
}
      if( car == 'M' && CaptD == 0 && CaptA == 1)
      {
        car = 'D';
        USART_Transmit(car);
        do
        {
              if( CaptA == 0)
                              {
                    car = 'M';
                USART_Transmit(car);
              }
        }
        while( CaptD == 0);

         }
   }
}
```

Annexe n°2 : Programme complet du microcontrôleur relié à la borne d'arrivée

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.2c Professional
Automatic Program Generator
© Copyright 1998-2004 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project :
Version :
Date    : 03/10/2011
Author  : F4CG
Company : F4CG
Comments:


Chip type           : ATmega8535
Program type        : Application
Clock frequency     : 16,000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 128
*****************************************************/

#include <mega8535.h>
#include <stdio.h>
#include <delay.h>

#define Vert PIND.6
///#define Rouge PIND.7
#define Bleu PINB.5
#define CaptD PIND.4
#define CaptA PIND.5
#define ENABLE PORTD.7

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

// Declare your global variables here
unsigned char dizaine, unite, dizieme, centieme,car;

flash const unsigned char valeur_constant[]=
{
0xEE,0x82,0xDC,0xD6,0xB2,0x76,0x7E,0xC2,0xFE,0xF6,
0xFA,0xFE,0x6C,0xEE,0x7C,0xF8,0x7E,0xBA,0x82,0x86,
0x3C,0x2C,0xEA,0x1A,0xEE,0xF8,0xF2,0xFA,0x76,0x3C,
0x0E,0xAE,0xDC};

flash const unsigned char adresse_constant[16]=
{0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15};
```

```
void Afficheur(unsigned char adresse, unsigned char caractere, unsigned char point)
{
        if ( point == 1 )
        {
                PORTA = (valeur_constant[caractere] | 0x01);          // Un point est rajouté après le caractère
        }
        else
        {
                PORTA=valeur_constant[caractere]; //Un caractere tout seul
        }
        PORTB=(0b00010000|adresse_constant[adresse & 0x0F]);
        PORTB.4=1;
        PORTB.4=0;
        PORTB.4=1;
        PORTA=0x00;
}

unsigned char USART_Receive( void )
{
/* Wait for data to be received */
while ( !(UCSRA & 0x80) ) // Test de RXC bit7
;
/* Get and return received data from buffer */
return UDR;
}

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
   centieme++;
   if( centieme == 10 )
   {
      dizieme++;
      centieme = 0;
      if( dizieme == 10)
      {
         unite++;
         dizieme=0;
         if(unite>9)
         {
            dizaine++;
            unite=0;
            if(dizaine == 6)
            {
               dizaine = 0;
            }
         }
      }
   }
   Afficheur(0,dizaine,0);
   Afficheur(1,unite,1);
   Afficheur(2,dizieme,0);
   Afficheur(3,centieme,0);
}
```

```
void Chronometre(int Flag )
{
        if( Flag == 1)
        {
            ENABLE=0;
         TCCR1B = 0x0A;
        }
        else if( Flag == 0)
        {
                TCCR1B = 0x08;
                ENABLE = 1;
                dizaine = 0;
                unite = 0;
        dizieme = 0;
        centieme = 0;
        Afficheur(0,dizaine,0);
          Afficheur(1,unite,1);
          Afficheur(2,dizieme,0);
          Afficheur(3,centieme,0);
        }
        else if (Flag == 2)
        {
                TCCR1B = 0x08;
                ENABLE = 0;
                dizaine = 0;
                unite = 0;
        dizieme = 0;
        centieme = 0;
        Afficheur(0,dizaine,0);
          Afficheur(1,unite,1);
          Afficheur(2,dizieme,0);
          Afficheur(3,centieme,0);
        }
        else if( Flag == 3)
        {
                TCCR1B = 0x08;
                ENABLE=0;
        }
}


void main(void)
{
// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTA=0x00;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0x00;
DDRB=0x1F;
```

```
// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=Out Func6=In Func5=In Func4=In Func3=In Func2=In Func1=Out Func0=In
// State7=1 State6=T State5=T State4=T State3=T State2=T State1=0 State0=T
PORTD=0x80;
DDRD=0x82;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 2000,000 kHz
// Mode: CTC top=OCR1A
// OC1A output: Toggle
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x40;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x4E;
OCR1AL=0x20;
OCR1BH=0x00;
OCR1BL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off˜
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x10;
```

```
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);

#asm("sei")

   while (1)
   {
         int flag;
         car=USART_Receive();
         if (car == 'D')
         {
               flag = 2;
         }
         if (car == 'M')
         {
               flag = 1;
         }
         else if (car == 'A')
         {
               flag = 0;
         }
         if (CaptD == 0)
         {
               flag = 3;
         }
         Chronometre(flag);
   }
}
```

Annexe n°3 : Corpus pouvant aboutir sur une note de synthèse

- http://www.e-kart-troyes.fr/presse/multimedia/index.php?id=30
- http://sitelec.org/cours/abati/aff7seg.htm
- http://meteosat.pessac.free.fr/Cd_elect/temp/a_nettoyer/www.ac-nancy-metz.fr/pres-etab/poinca_r/peda/elec/rs232co.htm

Le 1er lien est une vidéo du challenge e-kart faisant l'objet d'un reportage au JT de 20h sur TF1.

Le 2ème lien est une présentation de ce qu'est un afficheur 7 segments.

Enfin, le 3ème lien est une introduction au fonctionnement de la liaison série RS232.