**Université du Québec en Outaouais INF4103**

Rapport : mini-projet

Minuterie du four à micro-ondes sur Dragon 12

**Arnaud Niyonkuru NIYA26109309  
Vestine Mukeshimana MUKV01568701**

**21 avril 2016**

**Responsable de laboratoire A. Shaneen-Professeur A.-M. Cretu**

# Table des matières

Contents

[Rapport : mini-projet 1](#_Toc448928865)

[Minuterie du four à micro-ondes sur Dragon 12 1](#_Toc448928866)

[Table des matières 2](#_Toc448928867)

[Le cahier de charges: 3](#_Toc448928868)

[Description du logiciel en termes de fonctionnalités 3](#_Toc448928869)

[Description du matériel requis 3](#_Toc448928870)

[Le choix de conception 3](#_Toc448928871)

[La conformité de la conception par rapport au cahier de charges 4](#_Toc448928872)

[Les prototypes intermédiaires et finaux 5](#_Toc448928873)

[Prototype pour l’affichage 7-segments 6](#_Toc448928874)

[Prototype pour le son 6](#_Toc448928875)

[Prototype pour les boutons poussoirs et interrupteurs 7](#_Toc448928876)

[Pseudocode pour le programme final 7](#_Toc448928877)

[Le code 10](#_Toc448928878)

[Les tests effectués ainsi que les résultats du projet accompagnés par des captures d’écran et/ou des images. 13](#_Toc448928879)

[Difficultés 13](#_Toc448928880)

[Résultats 13](#_Toc448928881)

[Conclusion 14](#_Toc448928882)

[Références 15](#_Toc448928883)

[Annexe A - Programme principale 16](#_Toc448928884)

[Annexe B – Utilitaires 19](#_Toc448928885)

# Le cahier de charges:

## Description du logiciel en termes de fonctionnalités

Voici les fonctionnalités de base d’une minuterie qu’on aimerait développer pour le four à micro-ondes.

1. La minuterie devra afficher les minutes et les secondes.
2. La minuterie permettra l’entrée des minutes et des secondes.
3. La minuterie permettra d’amorcer le décompte.
4. La minuterie permettra d’arrêter le décompte.
5. La minuterie permettra d’initialiser à zéro les minutes et les secondes.

## Description du matériel requis

* Un microcontrôleur HSC12 sur la carte d'entraînement Dragon-12 pour le traitement.
* 3 Boutons sur la carte d'entraînement Dragon-12 pour la lecture des données
  1. Le premier bouton pour la commande *Start* (Démarrer) de la minuterie.
  2. Le deuxième bouton pour la commande *Stop/Clear* (Arrêt/Initialisation) de la minuterie.
  3. Le troisième bouton pour la commande *Number* (Nombre) pour entrer les chiffres dans la minuterie.
* 4 Interrupteurs sur la carte d'entraînement Dragon-12 pour la lecture des données pour composer les chiffres 0 à 9 à entrer dans la minuterie sous forme binaire.
* 4 afficheurs 7-segments pour affiche des résultats.
* Un haut-parleur sur la carte d'entraînement Dragon-12 pour les sons.

# Le choix de conception

Pour gérer la logique de notre minuterie, nous avons utilisé une machine à état fini. La machine à état fini appelé aussi automate fini est une construction abstraite, susceptible d'être dans un nombre fini d'états, un seul état à la fois; l'état où il se trouve est appelé l'« état courant ». Le passage d'un état à un autre est dirigé par un événement ou une condition; ce passage est appelé une « transition ». Un automate particulier est défini par la liste de ses états et par les conditions des transitions. [[1]](#_Références)



Figure 1 Diagramme d'états

La minuterie du four à micro-ondes présente un nombre fini d’états et un seul état est valide à la fois. Voici les 5 états de la minuterie :

* État d’initialisation
* État de décalage
* État de décompte
* État de pause
* État final.

Les transitions d’un état à l’autre sont assurées par les commandes données au moyen des boutons ***Start***, ***Stop/Clear***et ***Number*** ainsi que les délais données comme les 2 secondes entre l’état Final et Initialisation.

# La conformité de la conception par rapport au cahier de charges

La minuterie du four à micro-ondes est un circuit composé par 4 afficheurs 7-segments, 4 boutons poussoirs et 4 boutons interrupteurs pour différentes commandes. Une fois programmée, elle sert à gérer le temps de cuisson. Au début sur les afficheurs s’affiche 0000. En appuyant sur les boutons ***Number***, on programme la minuterie avec les chiffres binaires que représentent les interrupteurs. Quand on appuie sur le bouton ***Start*** le décompte automatique. Pour interrompre la minuterie on appuie sur le bouton ***Stop***. Une fois la minuterie arrêté on a l’option d’appuyer encore sur le bouton Stop pour réinitialiser la minuterie à 0000 ou bien appuyer sur Start pour continuer le décompte. À la fin du décompte le mot **END** s’affiche sur les 7-segments.

Voici comment la minuterie conçue est conforme au cahier de charges.

Le logiciel de minuterie du four à micro-ondes présente 5 états :

1. Si la minuterie est dans l’état Initialisation où les afficheurs 7-segments affichent 0000
   1. Si le bouton ***Start*** est appuyé, la minuterie reste dans l’état Initialisation
   2. Si le bouton ***Stop*** est appuyé, la minuterie reste dans l’état Initialisation
   3. Si le bouton ***Number*** est appuyé, la minuterie va dans l’état Décalage et le petit son est déclenché.
2. Si la minuterie est dans l’état décalage où les afficheurs 7-segments sont programme de droite à gauche
   1. Si le bouton ***Start*** est appuyé, la minuterie va dans l’état décompte et le petit son est déclenché.
   2. Si le bouton ***Stop*** est appuyé, la minuterie va dans l’état Initialisation
   3. Si le bouton ***Number*** est appuyé, la minuterie reste dans l’état décalage et le petit son est déclenché.
3. Si la minuterie est dans l’état décompte où le compte à rebours s’effectue
   1. Si le bouton ***Start*** est appuyé, la minuterie reste dans l’état décompte
   2. Si le bouton ***Stop*** est appuyé, la minuterie va dans l’état pause
   3. Si le bouton ***Number*** est appuyé, la minuterie reste dans l’état décompte
   4. Quand le décompte arrive à 0 minutes et 0 secondes la minuterie se met dans l’état final et les afficheurs 7-segments affichent END (Fin) et sonne 2 fois.
4. Si la minuterie est dans l’état pause où les afficheurs 7 segments ne bougent plus
   1. Si le bouton ***Start*** est appuyé, la minuterie va dans l’état décompte
   2. Si le bouton ***Stop*** est appuyé, la minuterie va dans l’état Initialisation
   3. Si le bouton ***Number*** est appuyé, la minuterie reste dans l’état pause
5. Si la minuterie est dans l’état final les afficheurs 7-segments affichent END (Fin) et sonne 2 fois et elle va directement dans l’état initialisation.

# Les prototypes intermédiaires et finaux

Pour développer notre minuterie, on n’est pas directement allé au programme final. On a testé différentes composantes de Dragon 12 dans le cadre de notre projet. On a utilisé les examples du cours pour tester l’afficher 7-segments et le son, ensuite pour l’usage des boutons poussoirs n’a pas été enseigné dans le cadre de notre cours, on a utilisé le cours de l’université New Mexico Institute of Mining and Technology[[2]](#_Références). Avant de s’attaquer à la programmation en assembleur le pseudocode a été développé pour faciliter le raisonnement.

## Prototype pour l’affichage 7-segments

#include "hcs12.inc"

org $1500

start lds #$1500

movb #$FF,DDRB ; configuration du port B

movb #$3F,DDRP ; configuration du port P

forever ldx #DispTab ; X pointe vers le tableau à afficher

loopi movb 1,X+ PTB ; sortie du schéma de segments

movb 1,X+ PTP ; sélection de l’afficheur

ldy #1

jsr delayby1ms ; attend 1 ms

cpx #DispTab+8 ; fin du tableau?

bne loopi

bra forever

#include "delay.asm"

DispTab dc.b $7D, $37 ; écrire 6=$7D sur afficheur 4

dc.b $06, $3B ; écrire 1=$06 sur afficheur 3

dc.b $3F, $3D ; écrire 0=$3F sur afficheur 2

dc.b $5B, $3E ; écrire 2=$5B sur afficheur 1

end

## Prototype pour le son

; génération d’un forme d’onde périodique avec une fréquence de 1KHz

#include "hcs12.inc"

org $1500

start lds #$1500

bset DDRT,BIT5 ; configuration de PT5 comme sortie

forever bset PTT,BIT5 ; remettre PT5 à haut

ldy #10 ; attendre 0.5 ms

jsr delayby50us

bclr PTT,BIT5 ; mettre PT5 à bas

ldy #10 ; attendre 0.5 ms

jsr delayby50us

bra forever

#include "delay.asm"

end

## Prototype pour les boutons poussoirs et interrupteurs

;web\_sw\_to\_led.asm, 11/2006, ref. Dragon12 schematics

;Ignoring bounce issues, echo switch value on LEDs

;PJ1 = 1 enables 7-segment while PJ0 = 0 enables LEDs

;PH0 thru PH7 connected to DIP switches

; but also PH0 thru PH3 are pushbuttons (in parallel)

portb equ $0001 ;LEDs and 7-seg LED port (Refer back to Port Use Slide)

portj equ $0268 ;7-segment and LEDs Enable/Disable

porth equ $0261 ;push button and rocker/DIP switches

portp equ $0258 ;connected to cathodes of 7-segments (DIG0 – DIG3)

ddrb equ $0003

ddrh equ $0262

ddrj equ $026A

ddrp equ $025A

output equ $FF

input equ $00

org $1500

movb #output,ddrj

movb #output,ddrb

movb #output,ddrp

movb #input,ddrh

movb #00,portj ;enables LEDs

movb #$FF,portp ;turns off 7-seg LED display using DIG0-DIG33

poll ldaa porth

staa portb

bra poll ;never ends

end

## Pseudocode pour le programme final

Configuration des ports(entrées/sorties)

Initialisation des variables

WHILE le programme pas forcement interrompus

DO

Lire la commande venant de boutons poussoirs

IF le bouton START appuyé

THEN

IF état courant est INITIALISATION

THEN

GO TO etat INITIALISATION

ENDIF état courant est INITIALISATION

IF état courant est DECALAGE

THEN

GO TO etat DECOMPTE

ENDIF état courant est DECALAGE

IF état courant est PAUSE

THEN

GO TO etat DECOMPTE

ENDIF état courant est PAUSE

ENDIF le bouton START appuyé

IF le bouton STOP appuyé

THEN

IF état courant est INITIALISATION

THEN

GO TO etat INITIALISATION

ENDIF état courant est INITIALISATION

IF état courant est DECALAGE

THEN

GO TO etat INITIALISATION

ENDIF état courant est DECALAGE

IF état courant est DECOMPTE

THEN

GO TO etat PAUSE

ENDIF état courant est DECOMPTE

IF état courant est PAUSE

THEN

GO TO etat INITIALISATION

ENDIF état courant est PAUSE

ENDIF le bouton STOP appuyé

IF le bouton NUMBER appuyé

THEN

IF état courant est INITIALISATION

THEN

GO TO etat DECALAGE

ENDIF état courant est INITIALISATION

IF état courant est DECALAGE

THEN

GO TO etat DECALAGE

ENDIF état courant est DECALAGE

IF état courant est DECOMPTE

THEN

GO TO etat DECOMPTE

ENDIF état courant est DECOMPTE

IF état courant est PAUSE

THEN

GO TO etat PAUSE

ENDIF état courant est PAUSE

ENDIF le bouton NUMBER appuyé

IF aucun bouton appuyé

THEN

IF état courant est DECOMPTE

THEN

GO TO etat DECOMPTE

ENDIF état courant est DECOMPTE

ENDIF aucun bouton appuyé

BEGIN etat INITIALISATION

changer l'état à INITIALISATION

Initialiser les variables d'affichage à 0

END etat INITIALISATION

BEGIN etat DECALAGE

changer l'état à DECALAGE

Lire le nombre sur les interrupteurs

Inserer le nouveau nombre à droite tout en décallant les autres à droite

Faire un son tout en affichant sur l'afficheur 7-segment

Attendre une seconde tout en affichant sur l'afficheur 7-segment

END etat DECALAGE

BEGIN etat DECOMPTE

changer l'état à DECOMPTE

IF les variables d'affichage sont à 0

THEN

GO TO etat FINAL

ELSE

Décrimenter les minutes et les secondes au besoin

Faire un son

Attendre une seconde

ENDIF les variables d'affichage sont à 0

END etat DECOMPTE

BEGIN etat PAUSE

changer l'état à PAUSE

Faire un son tout en affichant sur l'afficheur 7-segment

Attendre une seconde tout en affichant sur l'afficheur 7-segment

END etat PAUSE

BEGIN etat FINAL

changer l'état à FINAL

Faire un son long tout en affichant END sur l'afficheur 7-segment

Attendre 500ms sans rien afficher

Faire un son long tout en affichant END sur l'afficheur 7-segment

END etat FINAL

ENDO

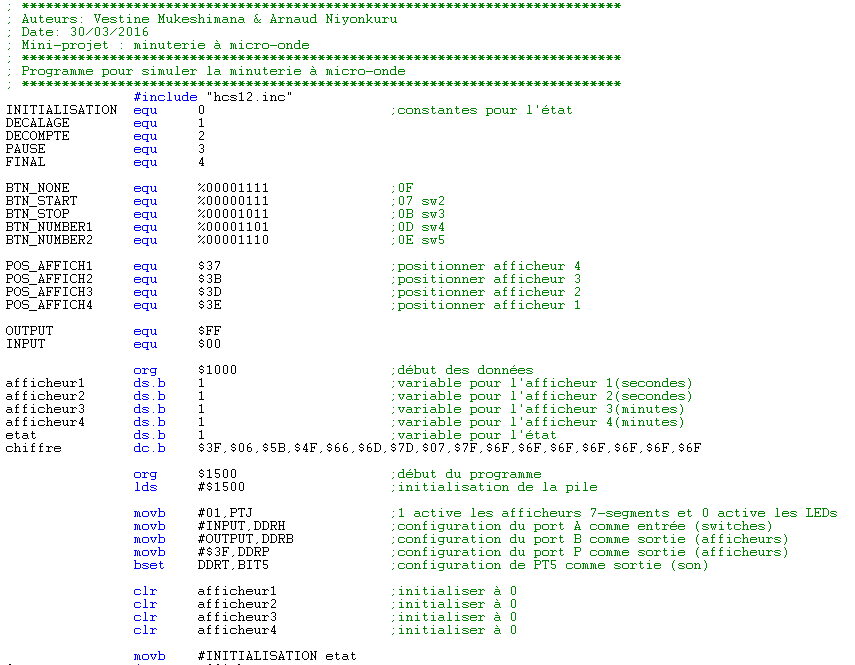
END WHILE le programme pas forcement interrompus DO

# Le code

Le code final détaillé dans Annexe A et B, est composé par les parties suivantes :

1. La définition des constantes
2. La déclaration des variables
3. La configuration des entrées et sorties
4. Initialisation des variables
5. Boucle sans fin qui gère la machine d’états fini

Ce programme utilise les sous-routines d’affichage et de production de son qui sont assemblé dans leur propre fichier.



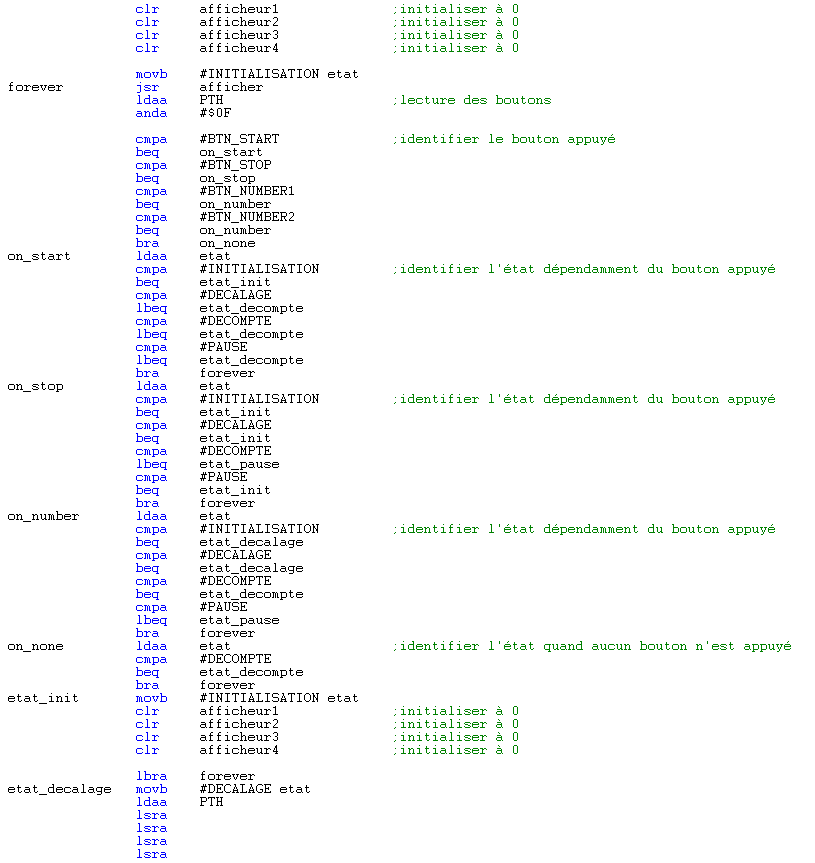
**Définition des constantes**

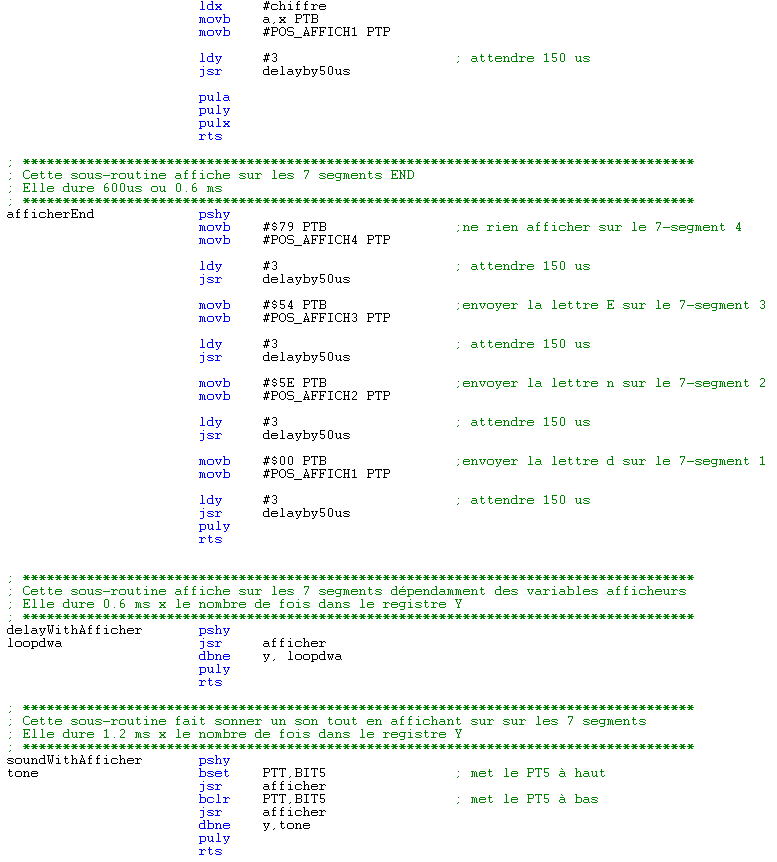
**Déclaration des variables**

**Configurations entrées/sorties**

**Boucle sans fin qui gère la machine d’état**

**Initialisation des variables**





**Sous routines pour la gestion de l’affichage   
7-segments, sons et délais**

# Les tests effectués ainsi que les résultats du projet accompagnés par des captures d’écran et/ou des images.

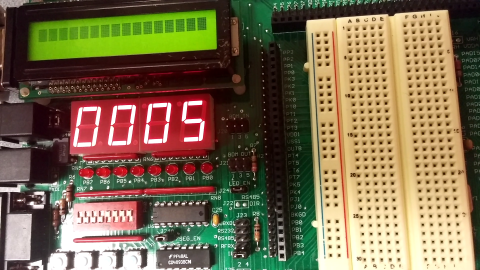
Tous les tests ont été effectués sur le microcontrôleur HSC12 sur la carte d'entraînement Dragon-12. En testant les prototypes qui se concentrent sur les modules utilisés dans le cadre de notre projet, on a pu confirmer que notre carte Dragon-12 était en bon état.

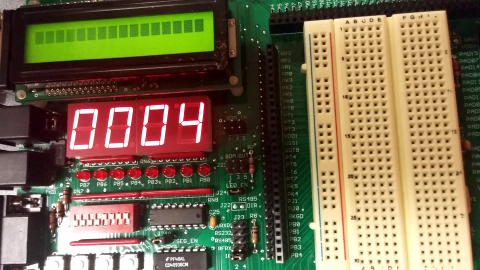
## Difficultés

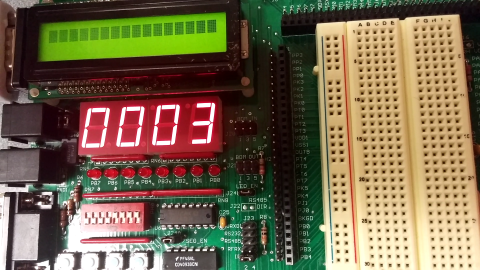
La grande difficulté rencontré c’est afficher sur 7-segments on a réalisé que les 4 7-segments ne s’affichent pas à la fois et à chaque fois qu’une sous-routine de délaie était utilisée, seulement le dernier segment montrait un chiffre. Pour remédier à ce problème, on a intégré l’affichage dans les sous-routines de délaies et dans celles de génération du son.

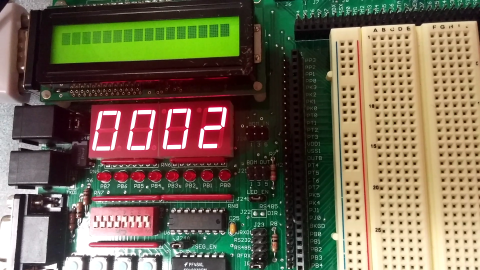
## Résultats

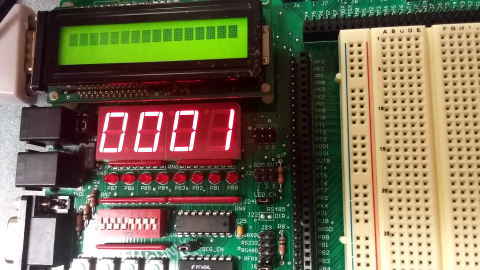
Les images suivantes ainsi que la vidéo soumise prouvent que le projet a été un succès.

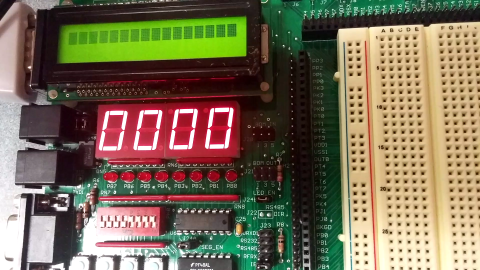


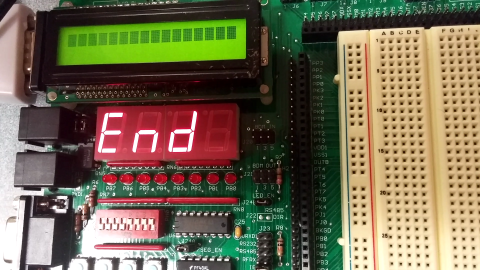












# Conclusion

Dans le cadre de ce mini-projet on a pu concevoir, développer, tester et évaluer de prototypes sur le microcontrôleur HCS12. On a pu :

* mettre en œuvre des investigations documentaires, des expériences et/ou des prototypes
* créer des modèles, simulations, prototypes, et faire des tests
* vérifier la conformité de la conception par rapport au cahier des charges

# Références

[1] Wikipedia. Automate fini, [En ligne]. <https://fr.wikipedia.org/wiki/Automate_fini> (Page consultée le 9 avril 2016).

[2] New Mexico Institute of Mining and Technology. Introduction à la programmation de MC9S12, <http://www.ee.nmt.edu/~erives/308_13/Lecture10_S13.pdf> (Document consulté le 29 mars 2016).

# Annexe A - Programme principale

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Auteurs: Vestine Mukeshimana & Arnaud Niyonkuru

; Date: 30/03/2016

; Mini-projet : minuterie à micro-onde

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Programme pour simuler la minuterie à micro-onde

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "hcs12.inc"

INITIALISATION equ 0 ;constantes pour l'état

DECALAGE equ 1

DECOMPTE equ 2

PAUSE equ 3

FINAL equ 4

BTN\_NONE equ %00001111 ;0F

BTN\_START equ %00000111 ;07 sw2

BTN\_STOP equ %00001011 ;0B sw3

BTN\_NUMBER1 equ %00001101 ;0D sw4

BTN\_NUMBER2 equ %00001110 ;0E sw5

POS\_AFFICH1 equ $37 ;positionner afficheur 4

POS\_AFFICH2 equ $3B ;positionner afficheur 3

POS\_AFFICH3 equ $3D ;positionner afficheur 2

POS\_AFFICH4 equ $3E ;positionner afficheur 1

OUTPUT equ $FF

INPUT equ $00

org $1000 ;début des données

afficheur1 ds.b 1 ;variable pour l'afficheur 1(secondes)

afficheur2 ds.b 1 ;variable pour l'afficheur 2(secondes)

afficheur3 ds.b 1 ;variable pour l'afficheur 3(minutes)

afficheur4 ds.b 1 ;variable pour l'afficheur 4(minutes)

etat ds.b 1 ;variable pour l'état

chiffre dc.b $3F,$06,$5B,$4F,$66,$6D,$7D,$07,$7F,$6F,$6F,$6F,$6F,$6F,$6F,$6F

org $1500 ;début du programme

lds #$1500 ;initialisation de la pile

movb #01,PTJ ;1 active les afficheurs 7-segments et 0 active les LEDs

movb #INPUT,DDRH ;configuration du port A comme entrée (switches)

movb #OUTPUT,DDRB ;configuration du port B comme sortie (afficheurs)

movb #$3F,DDRP ;configuration du port P comme sortie (afficheurs)

bset DDRT,BIT5 ;configuration de PT5 comme sortie (son)

clr afficheur1 ;initialiser à 0

clr afficheur2 ;initialiser à 0

clr afficheur3 ;initialiser à 0

clr afficheur4 ;initialiser à 0

movb #INITIALISATION etat

forever jsr afficher

ldaa PTH ;lecture des boutons

anda #$0F

cmpa #BTN\_START ;identifier le bouton appuyé

beq on\_start

cmpa #BTN\_STOP

beq on\_stop

cmpa #BTN\_NUMBER1

beq on\_number

cmpa #BTN\_NUMBER2

beq on\_number

bra on\_none

on\_start ldaa etat

cmpa #INITIALISATION ;identifier l'état dépendamment du bouton appuyé

beq etat\_init

cmpa #DECALAGE

lbeq etat\_decompte

cmpa #DECOMPTE

lbeq etat\_decompte

cmpa #PAUSE

lbeq etat\_decompte

bra forever

on\_stop ldaa etat

cmpa #INITIALISATION ;identifier l'état dépendamment du bouton appuyé

beq etat\_init

cmpa #DECALAGE

beq etat\_init

cmpa #DECOMPTE

lbeq etat\_pause

cmpa #PAUSE

beq etat\_init

bra forever

on\_number ldaa etat

cmpa #INITIALISATION ;identifier l'état dépendamment du bouton appuyé

beq etat\_decalage

cmpa #DECALAGE

beq etat\_decalage

cmpa #DECOMPTE

beq etat\_decompte

cmpa #PAUSE

lbeq etat\_pause

bra forever

on\_none ldaa etat ;identifier l'état quand aucun bouton n'est appuyé

cmpa #DECOMPTE

beq etat\_decompte

bra forever

etat\_init movb #INITIALISATION etat

clr afficheur1 ;initialiser à 0

clr afficheur2 ;initialiser à 0

clr afficheur3 ;initialiser à 0

clr afficheur4 ;initialiser à 0

lbra forever

etat\_decalage movb #DECALAGE etat

ldaa PTH

lsra

lsra

lsra

lsra

movb afficheur3 afficheur4 ;décaller les digits

movb afficheur2 afficheur3

movb afficheur1 afficheur2

staa afficheur1

ldy #50 ;son court (60ms)

jsr soundWithAfficher

ldy #1567 ;940ms = 1567 x 0.6ms

jsr delayWithAfficher

lbra forever

etat\_decompte movb #DECOMPTE etat

ldab #$0 ;vérifier si la décompte est à 0

addb afficheur1

addb afficheur2

addb afficheur3

addb afficheur4

cmpb #0

beq etat\_final

dec afficheur1 ;afficheur1 -= 1

ldab afficheur1

cmpb #-1

bne wait1sec

ldab #9

stab afficheur1

dec afficheur2 ;afficheur2 -= 1

ldab afficheur2

cmpb #-1

bne wait1sec

ldab #5

stab afficheur2

dec afficheur3 ;afficheur3 -= 1

ldab afficheur3

cmpb #-1

bne wait1sec

ldab #9

stab afficheur3

dec afficheur4 ;afficheur4 -= 1

ldab afficheur4

cmpb #-1

bne wait1sec

ldab #5

stab afficheur4

wait1sec ldy #50 ;son court (60ms)

jsr soundWithAfficher

ldy #1567 ;940ms = 1567 x 0.6ms

jsr delayWithAfficher

lbra forever

etat\_pause movb #PAUSE etat

ldy #50 ;son court (60ms)

jsr soundWithAfficher

ldy #1567 ;940ms = 1567 x 0.6ms

jsr delayWithAfficher

lbra forever

etat\_final movb #FINAL etat

ldy #833 ;sonner la 1ere fois (1000ms)

jsr soundWithAfficherEnd

ldy #5 ;silence de 500ms

jsr delayby100ms

ldy #833 ;sonner la 2e fois (1000ms)

jsr soundWithAfficherEnd

lbra etat\_init

swi

#include "delay.asm"

#include "utilitaires.asm"

end

# Annexe B – Utilitaires

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Auteurs: Vestine Mukeshimana & Arnaud Niyonkuru

; Date: 30/03/2016

; Mini-projet : minuterie à micro-onde

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Utilitaires :7-segments, son et delai combinés

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Cette sous-routine affiche sur les 7 segments dépendamment des variables afficheurs

; Elle dure 600us ou 0.6 ms

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

afficher pshx

pshy

psha

ldaa afficheur4 ;envoyer le contenu d'afficheur4 sur le 7-segment

ldx #chiffre

movb a,x PTB

movb #POS\_AFFICH4 PTP

ldy #3 ; attendre 150 us

jsr delayby50us

ldaa afficheur3 ;envoyer le contenu d'afficheur3 sur le 7-segment

ldx #chiffre

movb a,x PTB

movb #POS\_AFFICH3 PTP

ldy #3 ; attendre 150 us

jsr delayby50us

ldaa afficheur2 ;envoyer le contenu d'afficheur2 sur le 7-segment

ldx #chiffre

movb a,x PTB

movb #POS\_AFFICH2 PTP

ldy #3 ; attendre 150 us

jsr delayby50us

ldaa afficheur1 ;envoyer le contenu d'afficheur1 sur le 7-segment

ldx #chiffre

movb a,x PTB

movb #POS\_AFFICH1 PTP

ldy #3 ; attendre 150 us

jsr delayby50us

pula

puly

pulx

rts

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Cette sous-routine affiche sur les 7 segments END

; Elle dure 600us ou 0.6 ms

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

afficherEnd pshy

movb #$79 PTB ;ne rien afficher sur le 7-segment 4

movb #POS\_AFFICH4 PTP

ldy #3 ; attendre 150 us

jsr delayby50us

movb #$54 PTB ;envoyer la lettre E sur le 7-segment 3

movb #POS\_AFFICH3 PTP

ldy #3 ; attendre 150 us

jsr delayby50us

movb #$5E PTB ;envoyer la lettre n sur le 7-segment 2

movb #POS\_AFFICH2 PTP

ldy #3 ; attendre 150 us

jsr delayby50us

movb #$00 PTB ;envoyer la lettre d sur le 7-segment 1

movb #POS\_AFFICH1 PTP

ldy #3 ; attendre 150 us

jsr delayby50us

puly

rts

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Cette sous-routine affiche sur les 7 segments dépendamment des variables afficheurs

; Elle dure 0.6 ms x le nombre de fois dans le registre Y

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delayWithAfficher pshy

loopdwa jsr afficher

dbne y, loopdwa

puly

rts

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Cette sous-routine fait sonner un son tout en affichant sur sur les 7 segments

; Elle dure 1.2 ms x le nombre de fois dans le registre Y

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

soundWithAfficher pshy

tone bset PTT,BIT5 ; met le PT5 à haut

jsr afficher

bclr PTT,BIT5 ; met le PT5 à bas

jsr afficher

dbne y,tone

puly

rts

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Cette sous-routine fait sonner un son tout en affichant END sur sur les 7 segments

; Elle dure 1.2 ms x le nombre de fois dans le registre Y

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

soundWithAfficherEnd pshy

toneE bset PTT,BIT5 ; met le PT5 à haut

jsr afficherEnd

bclr PTT,BIT5 ; met le PT5 à bas

jsr afficherEnd

dbne y,toneE

puly

rts