Step 1

Une directive d’assembleur est une instruction qui permet de :

Definir une variable

Definir une constante ou un symbole

Les section (.txt) et .data

2.2 Delay.s

1.VarTime dcd 0 initialisation de la variable varTime (definition du mot var time)

TimeValue equ 900000 donne le nom timeValue à la constante 900000

2.Les fonctions :

Delay\_100ms : copie time value à l’adresse 0 (var time)

Delay\_100ms proc

ldr r0,=VarTime Ro contient VarTIme

ldr r1,=TimeValue r1 contient TimeValue

str r1,[r0] timeValue est copié à l’adresse VarTime

Bouble tempo enleve 1 à son parametre jusqu’à ce qu’on arrive à 0

BoucleTempo

ldr r1,[r0] copie le premier parametre à r1

subs r1,#1 r1=R1 - 1

str r1,[r0] R1 est copié dans r0

bne BoucleTempo Boucle while ( tant qu’on a pas 0 on recommence

bx lr retour de foncion (dans r0)

endp

END

3. role de bx lr

(lr ligne du code à laquelle on revient

(si y avait un retour de fonction on l’aurait copié dans r0 avant de faire bx lr ( c’est le x qui dit qu’on doit sauvegarder le retour)

BX (Branch and Exchange) :Op ́eration de retour de sous programme`a partir de la valeur d’unregistre

LR c’est l’adresse de retour de fonction

Cette ligne d’instruction sert

3. Travail sur cible simulation

1.

2. on a fait build

3. en placant les points d’arrêt au GPIO\_set et au clear la période est dee 100ms ( avce timeValue =900000) en doublant time value on double le temps.

5. varTime est egal à 900000 et est décrémenté de 1 jusq’a ce qu’on arrive à 0 (delay fontion)

6. Adresse de VarTime 0x20000000 Data 4 delay.o(maram)

7. on observe la valuer de VarTime qui fait comme en 6.

8. observer le signal electronique

On voit un singal carré de période 200ms

4.

C’est la sonde avec Keil qu’on utilise

On observe la même chose qu’avec Simu

11. observation du signal

Mettre un fil dans le port pb.1 puis relier à l’oscillo et lancer la simulation

Step2

STEP Son

Periode du timer = 6552 = 91 \* 10^-6 \* 72 \* 10^6

Equivalent des echantillons dans l’intervalle de la PMW

[-32768; + 32767] 🡪 [0; 719]

Donc p= (32767 + 32768)/720

Offsett= 359.5

X🡪 x1/p + offset

Dans notre tableau les valeurs ont une taille de 2octets alors que les registres font 4octets (on va donc charger 2 valeurs consecutives au lieu d’une seule

Etant donné que les données sont signées, le processeur regarde la valeur du bit de signe qui est devant (dans les msb)

[0, (2^15-1)/2 ] le reste sont des valeurs negatives

4.2. Observation de la PMW

Le signal de la PWM est un signal carré entre 0 et 3.3V

Celui de CompValue est une sinusoide entre 0.6V et environ 2.7V

Celui du Compteur est en dents de scie entre 0 et 3.3V

8.Condition pour que le signal de la PWM soit à l’état haut : la tension CompValue soit supérieure à la tension Compteur

Condition pour que le signal de la PWM soit à l’état bas : CompValue soit inférieur à Compteur

9. si le signal qu’on peut transmettre est le signal CompValue alors le signal obtenu est de même nature mais il est en retard et l’amplitude crête à crête est un peu atténuée

Comment la PWM fonctionne :

Le rapport cyclique de la PWM varie de façon sinusoidale parce que le signal qu’on veut avoir c’est une sinusoide. Il compare le compteur Compteur avec CompValue (en numerique c’est un nombre)

4.3.1 Configuration de la PWM

10. la période de la PWM est T = 720/72MHz = 10 µs

La fréquence est de 100kHz

12. si probleme au include utiliser la baguette magique > ASM > Paths

4.4 Test du programme sur cible réelle

15.

Fc= 1 / 2piRC

H(p)= 1/ 1+ pRC

En prenant R=3.9kOhm => C = 10.2nF

16. A l’oscilloscope, après le son, on observe un signal triangulaire qui est normal c’est à cause du signal de la PMW qui sera filtré par notre passe bas.

(ce qu’on observe c’est la PMW)

4.5 Finalisation du module GeestionSon.s

17. Pour lancer le son, il suffit de remettre la variable Index à 0.

CallbackSon s’exécute de façon infinie en continuant d’incrémenter Index.

StartSon s’exécute en même temps que CallbackSon c’est pour ça qu’on avait juste un son

En mettant une condition (celle sur le projet) on voit qu’on a bien un autre son (il faut le mettre dans le while)

Step DFT

* 1. Petit rappel

1.On sait que frequence = k /(M/Te)

K1=17

K2 =18

K3 =19

K4=20

K5=23

K6=24

1.2.2 2) principe du codage au format 1.15

De 0x0 à 0x7FFF on a les nombres de 0 à 32767 en les divisant par 2^F on obtient le décimal

Et de 0x8000 à 0xFFFF on a les nombres de 32768 à 65535 auxquels on doit retrancher 2^(E+F)puis les diviser par 2^F pour avoir le nombre décimal.

Pour parcourir les différentes valeurs

Angle = 2pip/M des qu’on est supérieur à 2PI on fait angle = 2pi p / M mod 2pi

Pour passer au p au divise par 2pi/M

P = angle mod 64. Et l’angle sera égal au multiple de k

Un modulo 10 de 19756454 c’est le dernier chiffre, par 100 les deux derniers et donc en décimal par 64 = 2^6 ce sont les 6 derniers.

Pour trouver le k normalisé il faut faire k= ( F / Fe) \* 64

F = kFe/64