

Rapport de TP1 IN 201

Utilisation du PTM

GIVERNAUD Omar - ROSSETTO Anthony

23 octobre 2007

Table des matières

1	Introduction	3
2	Matériel utilisé	3
3	Le programme	4
3.1	Le PTM	4
3.1.1	Paramétrage du PTM	5
3.1.2	Calcul de période	6
3.2	Les interruptions	7
4	Le compteur	8
5	Divers	10
5.1	Problèmes rencontrés	10
6	Code source du programme	10

1 Introduction

Le but premier du TP est de nous apprendre à utiliser un périphérique sur un bus du 68k et plus précisément le PTM (Programmable Timer Module).

Pour mettre en application la théorie vue en cours, nous avons pour objectif d'afficher dans la console (shell) le résultat d'un compteur en boucle allant de 0 à 99 et ce toutes les secondes. Etant donné qu'il est impossible d'avoir une période d'une seconde avec la méthode que nous avons vue pour le moment, nous ferons l'affichage toutes les 0.33 secondes dans un premier temps. Le dernier exercice proposé nous demandait de trouver une méthode permettant un affichage précis toutes les secondes.

2 Matériel utilisé

- Plaquette 68000
- PC sous Linux
- Ide "kit" et commande assemblage pour 68k

3 Le programme

3.1 Le PTM

Le PTM 6840 (Programmable Timer Module) est un timer programmable. Il permet de générer des signaux périodiques ou monostables à partir de l'horloge qui lui est fournie. Le PTM est donc un diviseur d'horloge (un compteur).

Ce composant comporte trois compteurs binaires de 16 bits, trois registres de contrôles (un pour chaque compteur, bien qu'ils aient des spécificités pour chacun d'entre eux) ainsi qu'un registre d'état.

Chaque compteur dispose de son propre registre déterminant sa période. Ce registre est divisé en deux sous registre : MSB (Most Significant Byte) qui est l'octet de poids fort, ainsi qu'un LSB (Least Significant Byte) qui est l'octet de poids faible. La raison de ceci est que le bus de données du PTM est un bus de données de 8 bits (D0-D7).



FIG. 1 – Initialisation du PTM

Afin de générer les interruptions, nous avons choisi le compteur 3 du PTM qui a l'avantage d'avoir un prédiviseur d'horloge par 8. Cette fonctionnalité est activable en mettant au niveau logique "1" le premier bit du registre CR3 (registre de contrôle du compteur 3), soit $CR30 = 1$.

3.1.1 Paramétrage du PTM

Premièrement, nous devons arrêter tous les compteurs. Soit positionner le premier bit de CR1 à 0. Cependant, il n'est pas possible d'accéder directement aux registre CR1 et CR3. Il faut d'abord mettre le bit CR20 à 1 pour écrire dans CR1 et à 0 pour CR3.

Nous plaçons le mot 43h (01000011b) dans ce registre.

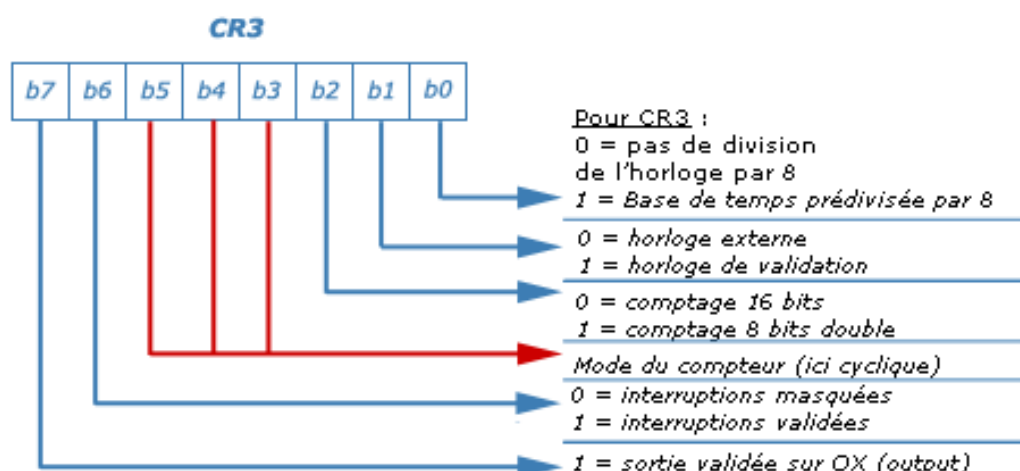


FIG. 2 – Registre de contrôle CR3

Nous avons choisi pour notre montage le mode cyclique puisque nous souhaitons générer des signaux périodiques. Il nous reste maintenant à définir ce temps de 0.33 secondes qui définit la durée de la période.

3.1.2 Calcul de période

$$T_{int} = T_{clock} * N$$

- Le paramètre T_{int} est celui qui nous intéresse puisqu'il correspond à la durée de la période d'interruption, celle qui dit « toutes les T_{int} secondes, le sous programme d'interruption sera appelé ».
- Le T_{clock} lui dépend uniquement de la fréquence du processeur et éventuellement de la division de la fréquence d'horloge par huit. Nous sommes dans ce cas, le T_{clock} sera donc égal (sachant que notre processeur 68000 est cadencé à 1 MHz) à $\frac{8}{10^6}$ soit $8 * 10^6$.
- Il nous reste ensuite pour trouver N à effectuer le calcul de T_{int} / T_{clock} . Dans notre cas, nous trouvons en hexadécimal la valeur **A2C1**.

$$\frac{1}{3} = \frac{8 * x}{10^6} x = \frac{10^6}{24} = (41666,666)_{10} = (A2C1)_{16}$$

L'octet de poids fort de cette valeur (A2) sera placé dans le registre correspondant à l'octet de poids fort du compteur 3, tandis que l'octet de poids faible (C1) sera placé dans le registre de poids faible du compteur 3.

3.2 Les interruptions

Il existe deux manières de recevoir les signaux du PTM : la scrutation ou l'utilisation des interruptions. Nous avons choisi la seconde méthode pour des raisons pédagogiques (apprendre à utiliser les interruptions vectorisées). Et que bien que notre programme soit mono-processus, cela le rend plus claire.

A chaque interruption, le 68000 sauvegarde le contenu du registre PC (Program Counter) ainsi que le registre SR (Status Register) afin de pouvoir revenir au programme principal une fois la routine d'interruption traitée.

Cette phase est appelée « sauvegarde du contexte » et le contenu de ces registres est mémorisé à l'adresse pointée par A7 (Stack Pointer, ou pointeur de pile) du 68000, plus exactement dans USP (User Stack Pointer) si l'interruption avait eu lieu pendant que le microprocesseur fonctionnait en mode utilisateur ou SSP (Supervisor Stack Pointer) dans le cas du mode superviseur.

Dans tous les cas, juste après la sauvegarde du contexte, le 68000 passe en mode superviseur, **sans pour autant revenir automatiquement au mode utilisé avant l'interruption.**

Dans notre cas, toutes les 0.33 secondes, une interruption est générée par le PTM. Nous devons donc traiter l'interruption via une routine appelée "sous programme d'interruption". Ce sous programme sera appelé à chaque interruption.

Nous avons utilisé des interruptions auto-vectorisées, c'est à dire que nous avons fait appel à un registre du 68000 appelé VBR (Vector Base Register), ce registre contenant à son adresse de base plus un offset de 70H en hexadecimal, l'adresse mémoire du sous programme d'interruption correspondant au PTM. La taille des vecteurs est 4 octets, soit $70h/4 = 1C$, soit l'interruption 28.

L'interruption que nous avons à gérer était de niveau 4 (niveau indiqué par les intervenants de TP).

La documentation technique du 68000 stipulant que tout sous-programme d'interruption de niveau 4 doit se trouver à l'adresse de base du registre VBR + un offset de 70H, nous avons placé l'adresse de notre sous-programme d'interruption dans la table des vecteurs d'interruptions en conséquence.

Le schéma visible ci-après aidera à la compréhension du principe de traitement d'une interruption auto-vectorisée :

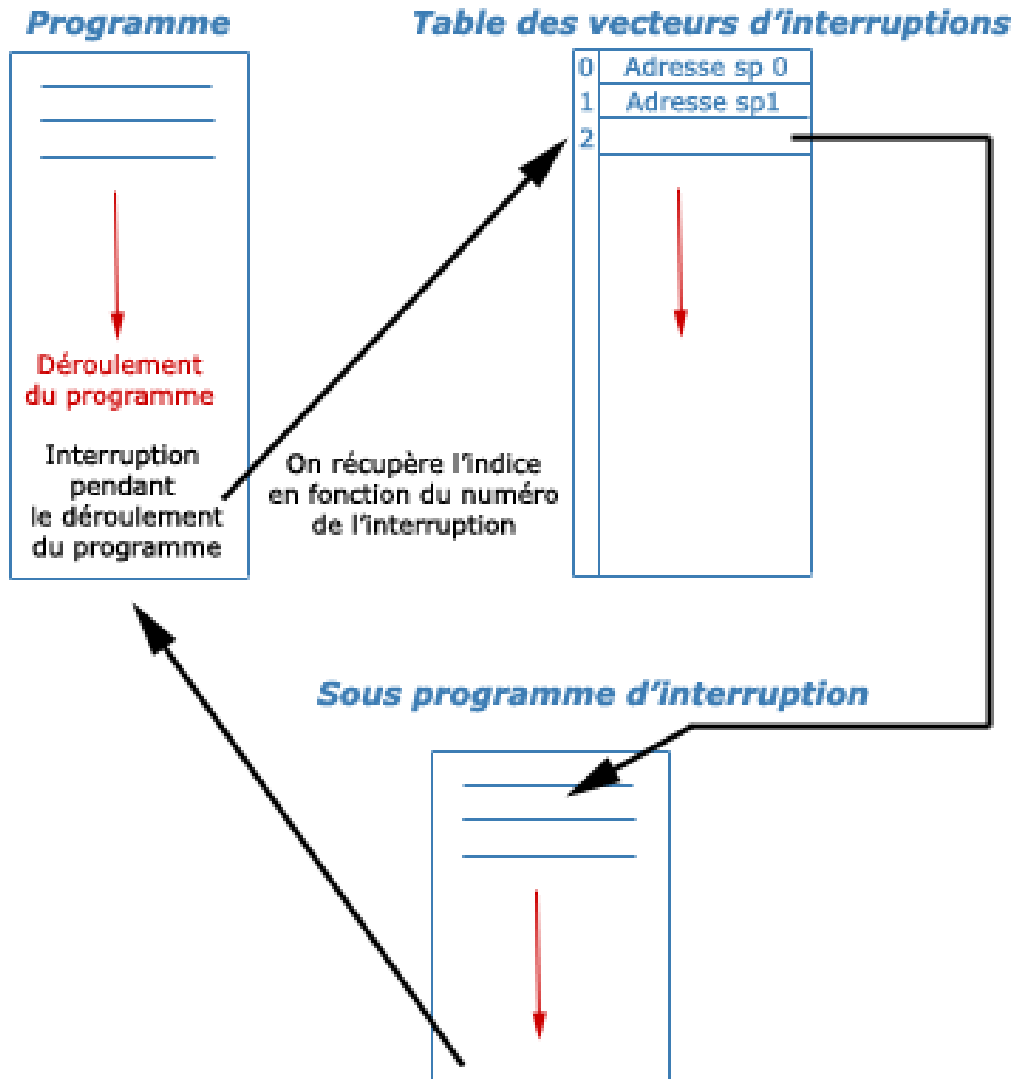


FIG. 3 – Principe de traitement d’une interruption auto-vectorisée

4 Le compteur

Nous commençons par un pré-diviseur par trois pour obtenir une période d’une seconde ($\frac{1}{3} * 3 = 1$) Puis par un compteur en boucle de 0 à 99.

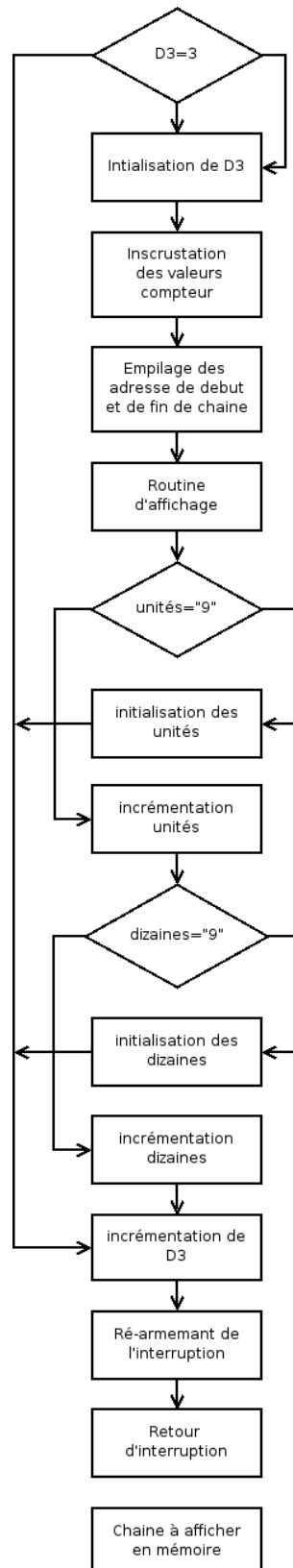


FIG. 4 – Schéma du compteur

5 Divers

5.1 Problèmes rencontrés

- Plaquette bogguée : reboot brutale
- Présence d'un TRAP #15 dans le fichier à inclure : faire deux fois got dans le kit

6 Code source du programme

```
1  include ptmldadc.s
;-----;
;INITIALISATION DES ADRESSES MEMOIRES DES ;
;REGISTRES          ;
;-----;
6
;adresse de base : 140000H
; tous les registres sont situes a des adresses
;impaires (A0 = 1)

11 CR3 EQU 140001H
   CR2 EQU 140003H
   CR1 EQU CR3
   SRPTM EQU CR2
   N EQU A2C1H ;A2 octet de poids fort , C1 octet de poids faible ,
               proc cadence a 1 MHz
16 OUTSTR EQU $0021 ; code de l'appel noyau OUTSTR

;-----;
;gestion du compteur 1 ;
;-----;

21 ;WMSB1 EQU 140005H ; ecriture dans le msb1

;RCPT1 EQU WMSB1 ; lecture dans le compteur 1
;WLCPT1 EQU 140007H ; ecriture dans les latches du compteur 1 (
   LSB)

26 ;RLSB1 EQU WLCPT1 ; lecture du buffer LSB1

;-----;
;gestion du compteur 2 ;
;-----;
```

```
31 ;-----;
;WMSB2 EQU 140009H ; ecriture dans le msb2
;RCPT2 EQU WMSB2 ; lecture dans le compteur 2
36 ;WLCPT2 EQU 14000BH ; ecriture dans les latches du compteur 2 (
    LSB)
;RLSB2 EQU WLCPT2 ; lecture du buffer LSB2
;-----;
41 ;gestion du compteur 3 ;
;-----;
WMSB3 EQU 14000DH ; ecriture dans le msb3
46 RCPT3 EQU WMSB3 ; lecture dans le compteur 3
WLCPT3 EQU 14000FH ; ecriture dans les latches du compteur 3 (
    LSB)
RLSB3 EQU WLCPT3 ; lecture du buffer LSB3
51 ;-----;
;On ecrit dans CR1 donc ;
;il faut que CR20 = 1 ;
;-----;
56 MOVE.B #01H,CR2
;-----;
;On met CR10 a 1 ;
;On arrete le timer ;
61 ;-----;
MOVE.B #01H,CR1
;-----;
66 ;On ecrit dans CR3 donc ;
;il faut que CR20 = 0 ;
;-----;
MOVE.B #00H,CR2
71
```

6 CODE SOURCE DU PROGRAMME

```

;-----;
;Initialisation de CR3 ;
;-----;

76 MOVE.B #43H,CR3 ; 0100 0011

;-----;
;Initialisation de SR ;
;du 68000 ;
81 ;-----;

MOVE.W #2300H,SR

;-----;
86 ;Ecriture de MSB3 et LSB3 ;
;-----;

MOVE.B #0A2H,WMSB3
MOVE.B #0C1H,WLCPT3
91

;-----;
;On ecrit dans CR1 donc ;
;il faut que CR20 = 1 ;
;-----;

96 MOVE.B #1H,CR2

;-----;
;On initialise le timer ;
101 ;par une remise a zero ;
;-----;

MOVE.B #0H,CR1

106 ;-----;
;definition de l'adresse ;
;du programme d'interruptions dans VBR ;
;-----;

111 MOVEC.L VBR,A0
ADD.L #070H,A0
MOVE.L #SPINT,(A0)
```

```

LOOP  BRA LOOP
116
;-----;
;Gestion du sous programme;
;d'interruption      ;
121 ;(appelle toutes les 0.33s);
;-----;

SPINT MOVE.B  SRPTM,D0
      MOVE.B  RLSB3,D0
126  MOVE.B  WMSB3,D0
      ADD.B  #1,compteur+1
      CMP.B  #'9',compteur+1

      BLE SUITE
131  MOVE.B  #'0',compteur+1
      ADD.B  #1,compteur
      CMP.B  #'9',compteur
      BLE SUITE
      MOVE.B  #'0',compteur
136
SUITE PEA fin    ; empile l'adresse du dernier caract re + 1
      PEA compteur ; empile l'adresse du premier caract re
      TRAP #15    ; appel noyau
      DC.W  OUTSTR
141  RTE
      compteur DC.B  '00',13,10 ; message, retour chariot, saut de
      ligne
      fin DC.B  0
END

```