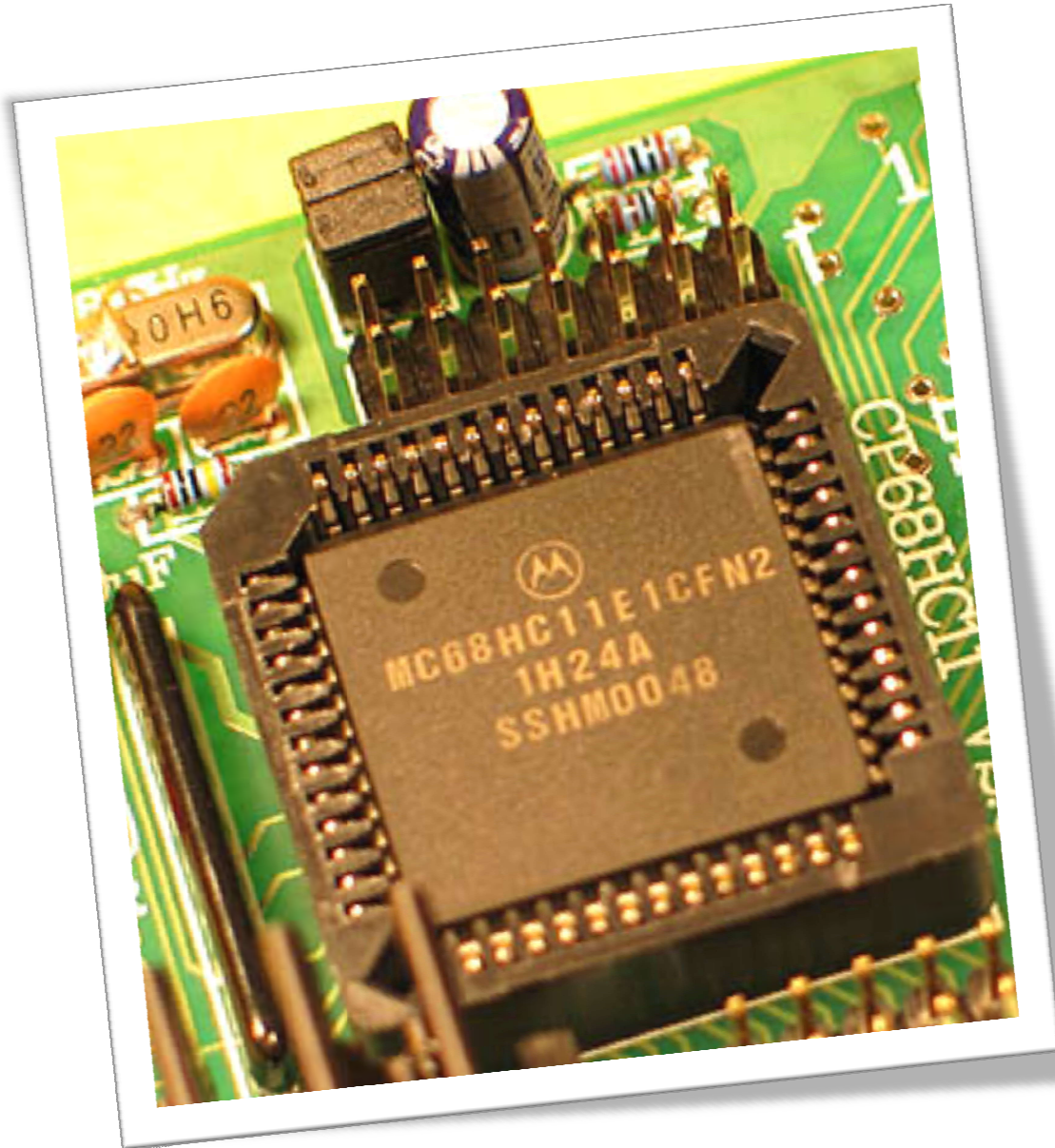


Projet Systèmes Numériques

Implémentation logicielle d'un fréquencesmètre sur un HC11



ABBAS Thomas

TOURRETTE Benjamin

L2 – Groupe A

Mai 2009



Sommaire

Mesure de la période d'un signal	2
Le Polling	2
Structure du Timer :	4
Programme :	6
Application d'interruptions	9
Fonctionnalité du microcontrôleur : LCD	14
Ecran LCD	14
Initialisation	15
Premier affichage	16
Conversion d'un entier en octet	16
Affichage de la période mesurée (le final inaccompli... ☹)	18
Registres de vitesse et de données : BAUD et SCDR	18
Registres d'états : SCSR	18
Registres de contrôle : SCCR1 et SCCR2	18
Saisie d'un caractère sans interruption	19
Saisie d'un caractère avec interruption	19

Mesure de la période d'un signal

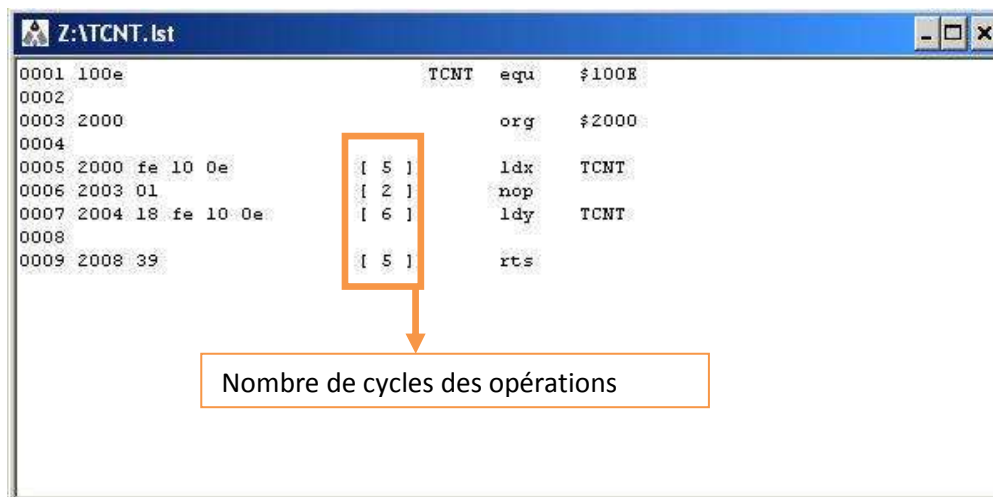
Le but principal de cette partie est de réaliser un mesureur de fréquence (ou de période) : un fréquencemètre. Pour cela, nous exploiterons la minuterie d'un microcontrôleur –et loin d'être méconnu- le 68HC11. Un compteur de 16 bits nous permettra de connaître la période du signal grâce au nombre de cycles machines écoulés dans un intervalle de temps Δt . Bien évidemment, afin de dater les arrivées de fronts et d'en déduire la période, il nous faudra programmer en langage assembleur.

Comment le programme "compte"-t-il?

Grâce à la méthode du Polling! Il scrute : marque un temps t_1 du premier front, puis un temps t_2 du second front, les soustrait, et en déduit la période, donc la fréquence, simple!

Le Polling

Prenons pour exemples les comptages de cycles CPU et Timer avec ce simple programme :



```

0001 100e          TCNT equ    $100E
0002
0003 2000          org     $2000
0004
0005 2000 fe 10 0e [ 5 ]    ldx    TCNT
0006 2003 01      [ 2 ]    nop
0007 2004 18 fe 10 0e [ 6 ]    ldy    TCNT
0008
0009 2008 39      [ 5 ]    rts
  
```

Nombre de cycles des opérations

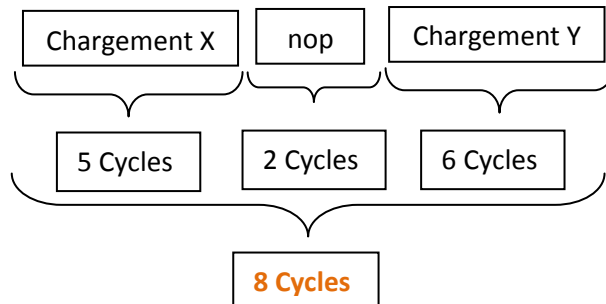




Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencemètre sur un HC1

Nous constatons donc une chose simple : voici le nombre de cycles qui séparent X et Y :



Afin de vérifier les états des registres X et Y, il nous suffit de lancer le programme s'exécutant à partir de l'adresse \$2000, les états nous sont signifiés de cette façon :

```

AxIDE
File Tools Help
[Checkmark] [Upload] CME11E9-EVBU [Build] [Configure] [Program] [Read]

BUFFALO 3.4 (ext) - Bit User Fast Friendly Aid to Logical Operation
>1 t

BUFFALO 3.4 (ext) - Bit User Fast Friendly Aid to Logical Operation
>1 t

done
>call 2000

P-2000 Y-5A23 X-5A1B A-FF B-FF C-D0 S-0041
>
  
```

Si l'on résume donc, dans le registre X se trouve le temps du premier front, dans le registre Y celui du second : la différence est simple à effectuée.

Y	5A23
X	5A1B
Y-X	0008

8 cycles dans l'intervalle, comme trouvé théoriquement !

Etudions un nouveau système. Nous proposons de compter le nombre de cycles d'un processus quelconque. Voici ici le résultat du nombre de cycles minimal :

0001	100e	TCNT	equ	\$100E	TCNT se trouve à l'adresse \$100E
0002					
0003	2000		org	\$2000	On se place à l'adresse \$2000
0004	2000	initial	rmb	2	On réserve 2 octets pour la marque du front de départ (->\$2002)
0005	2002	delta	rmb	2	On réserve 2 octets pour le résultat de la soustraction (->\$2004)
0006					
0007	2050		org	\$2050	
0008					
0009	2050 fc 10 0e	[5]	ldd	TCNT	
0010	2053 fd 20 00	[5]	std	initial	Stockage du front 1
0011		[5]	*processus		**Processus quelconque**
0012	2056 fc 10 0e	[5]	ldd	TCNT	
0013	2059 b3 20 00	[6]	subd	initial	Soustraction du front actuel au premier front
0014	205c fd 20 02	[5]	std	delta	Que l'on stocke dans "delta", l'intervalle Δt
0015					
0016	205f 39	[5]	rts		

On sait qu'à l'adresse \$2002 et \$2004 se trouvent respectivement les valeurs des registres A et B :

```
>call 2050
```

```
P-2050 Y-A5AC X-A5A4 A-00 B-0A C-D0 S-0041
```

On constate que les registres contiennent les valeurs A= $[00]_{16}$ et B= $[0A]_{16}$

L'accumulateur D est sensé être le résultat de la concaténation des registres A et B, nous le confirmons en scannant le contenu de la mémoire à partir de l'adresse \$2002 : "delta" = $[000A]_{16}$.

```
>md 2002
```

```

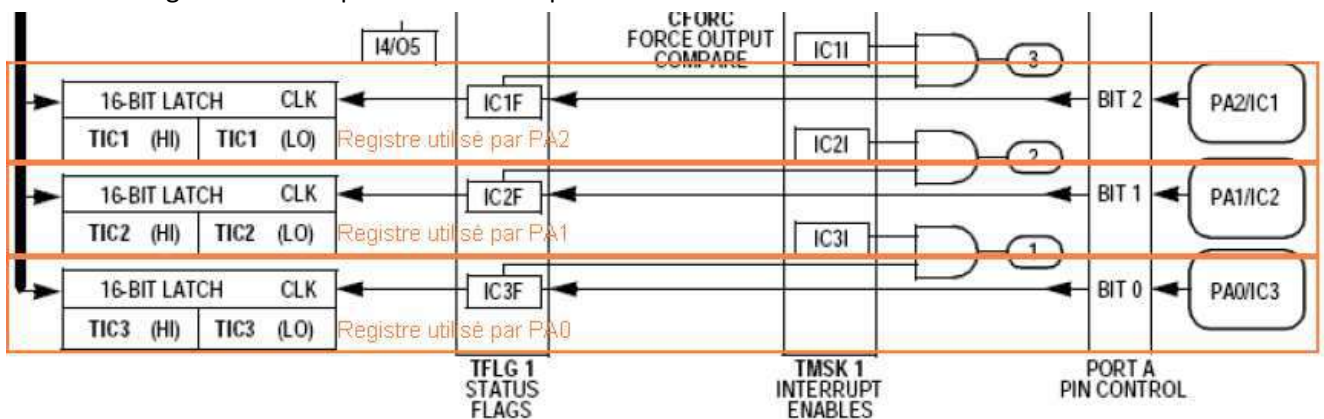
  initial  delta
2000 F5 98 00 0A 18 FE 10 0E 39 F8 53 FB 3E 37 9B E7      9 S >7
2010 1D 37 F9 FF 79 FF FF AD E5 E6 F1 FF 7F FB ED EB 7 y

```

Nous n'oublierons pas de prendre en compte lors du comptage du nombre de cycles de "Processus", le décalage de 10 cycles ($000A_{16}$), pris par le stockage et le chargement de l'intervalle delta.

Structure du Timer :

Voici les registres utilisés par les différents ports du microcontrôleur :



TIC1

Le contenu du TCNT est mémorisé dans TIC1 (registre de capture d'entrée) à l'arrivée d'un front actif (montant ou descendant) sur les entrées respectives PA0, PA1 et PA2 du port A :

TIC1	3 ₁₅	3 ₁₄	3 ₁₃	3 ₁₂	3 ₁₁	3 ₁₀	3 ₉	3 ₈	3 ₇	3 ₆	3 ₅	3 ₄	3 ₃	3 ₂	3 ₁	3 ₀	Adresse :
Registre 16b	MSB à l'adresse 1010								LSB à l'adresse 1011								\$1010-1011

TFLG1

Registre de signalisation dans lequel sont consignés les résultats de comparaisons entre le TCNT et les TOCx ainsi que l'arrivée de fronts actifs sur les entrées PA0, PA1 et PA2 marqués par un 1 ou un 0 :

3C1F	3C2F	3C3F	3C4F	05F 14F	IC1F	IC2F	IC3F	TFLG1 \$1023
Si TOCx = TCNT Alors OcxF=1					Enregistrement des fronts actifs			

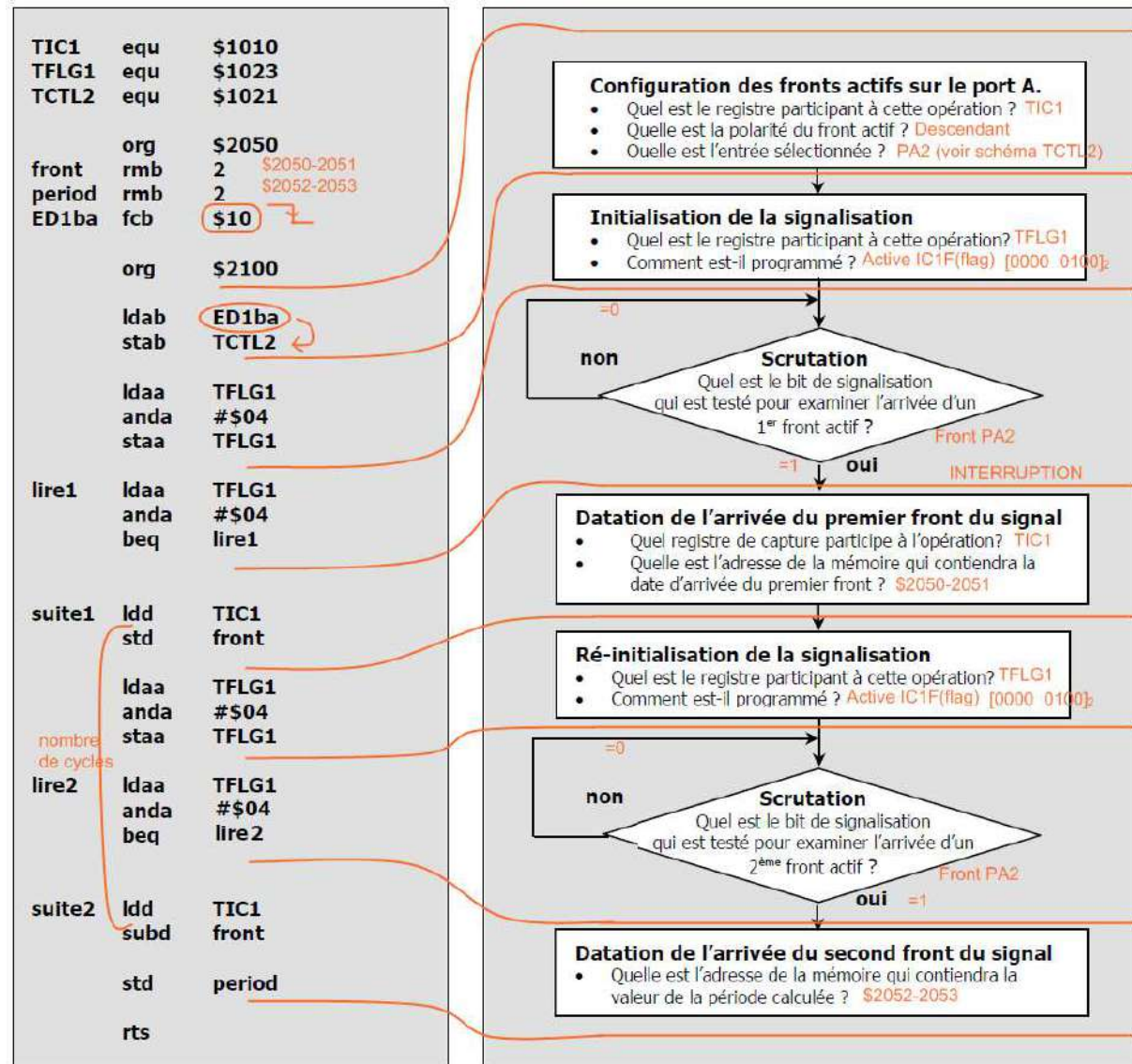
TCTL2

Les fronts (Edge) actifs arrivant sur PA0, 1,2 et PA3 sont indépendamment configurables au moyen du registre de contrôle TCTL2.

TCTL2
\$1021

ED4t	ED4a	ED1t	ED1a	ED2t	ED2a	ED3t	ED3a
		Config PA2		Config PA1		Config PA0	

Voici le programme principal de notre nouvelle étude : la mesure d'une période :





Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencesmètre sur un HC1

Programme :

```

TIC1 equ $1010 * TIC1 se situe a l'@ 1010 (voir la doc)
TFLG1 equ $1023 * TFLG1 se situe a l'@ 1023 (voir la doc)
TCTL2 equ $1021 * TCTL2 se situe a l'@ 1021 (voir la doc)

        org $2050 * on se place à l'adresse $2050
front    rmb 2 * réserver 2 octets pour le front
period  rmb 2 * réserver 2 octets pour la valeur de la période
ED1ba   fcb $10 * front descendant (voir doc)

        org $2100 * début du programme à l'@ $2100

        ldab ED1ba *
        stab TCTL2 *

        ldaa TFLG1 * initialisation de TFLG1
        anda #$04 * avec un ET logique entre la valeur de l'accu A et la valeur 04
        staa TFLG1 * TFLG1 constitue un temp

lire1    ldaa TFLG1 * initialisation de TFLG1
        anda #$04 * avec un ET logique entre la valeur de l'accu A et la valeur 04
        beq lire1 * aller à lire1 si égal

suite1   ldd TIC1 * Stockage du front
        std front * de TIC1

        ldaa TFLG1 * initialisation de TFLG1
        anda #$04 * avec un et logique entre la valeur de l'accu A et la valeur 04
        staa TFLG1 * réinitialisation du temp

lire2    ldaa TFLG1 *
        anda #$04 *
        beq lire2 *

suite2   ldd TIC1 * Front actuel...
        subd front * moins front stocké...

        std period * Donne la période

        rts *

```

Il convient préalablement de noter que le temps de notre processus est ici de 29 cycles :

0001	1010		TIC1 equ	\$1010
0002	1023		TFLG1 equ	\$1023
0003	1021		TCTL2 equ	\$1021
0004				
0005	2050		org	\$2050
0006	2050		front rmb	2
0007	2052		period rmb	2
0008	2054 10		ED1ba fcb	\$10
0009				
0010	2100		org	\$2100
0011				
0012	2100 f6 20 54	[4]	ldab	ED1ba
0013	2103 f7 10 21	[4]	stab	TCTL2
0014				
0015	2106 b6 10 23	[4]	ldaa	TFLG1
0016	2109 84 04	[2]	anda	#\$04
0017	210b b7 10 23	[4]	staa	TFLG1
0018				
0019	210e b6 10 23	[4]	lire1 ldaa	TFLG1
0020	2111 84 04	[2]	anda	#\$04
0021	2113 27 f9	[3]	beq	lire1
0022				
0023	2115 fc 10 10	[5]	suite1 ldd	TIC1
0024	2118 fd 20 50	[5]	std	front
0025				
0026	211b b6 10 23	[4]	ldaa	TFLG1
0027	211e 84 04	[2]	anda	#\$04
0028	2120 b7 10 23	[4]	staa	TFLG1
0029				
0030	2123 b6 10 23	[4]	lire2 ldaa	TFLG1
0031	2126 84 04	[2]	anda	#\$04
0032	2128 27 f9	[3]	beq	lire2
0033				
0034	212a fc 10 10	[5]	suite2 ldd	TIC1
0035	212d b3 20 50	[6]	subd	front
0036				
0037	2130 fd 20 52	[5]	std	period
0038				
0039	2133 39	[5]	rts	

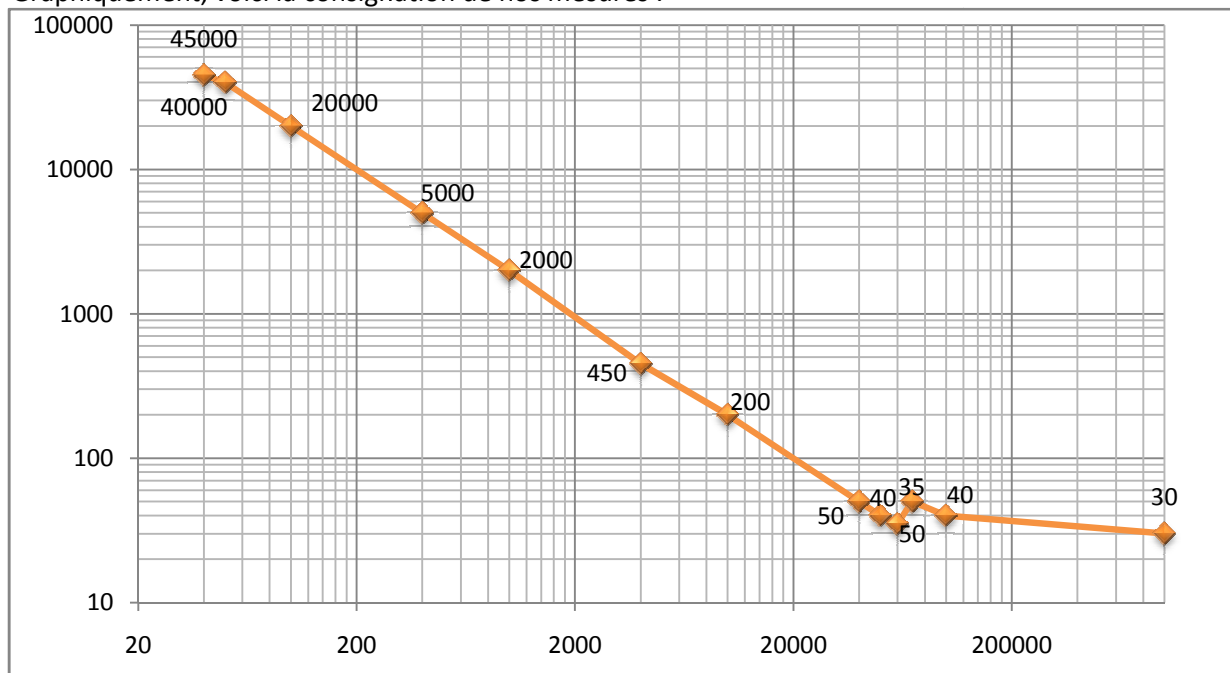
29 cycles

Nous avons, grâce aux scrutations mémoire (valeurs stockées), nous obtenons donc le nombre de cycles pour une fréquence de base donnée. Ils sont exprimés en hexadécimal dans l'accumulateur D. Il nous a donc simplement suffi de les convertir sous forme binaire.

Nous proposons de changer la fréquence d'exécution du programme : voici un tableau récapitulant le nombre de cycles observés pour des fréquences de mesures allant de 40 Hz à 1MHz.

Fréquence de mesure	Registre D, Delta, Polling, ou encore Nombre de cycles :	
40 Hz	[AFC8] ₁₆ = [45000] ₁₀	44,4 Hz
50 Hz	[9C40] ₁₆ = [40000] ₁₀	50 Hz
100 Hz	[4E20] ₁₆ = [20000] ₁₀	100 Hz
400 Hz	[1388] ₁₆ = [5000] ₁₀	400 Hz
1 kHz	[07D0] ₁₆ = [2000] ₁₀	1 kHz
4 kHz	[01C2] ₁₆ = [450] ₁₀	4,4 kHz
10 kHz	[00C8] ₁₆ = [200] ₁₀	10 kHz
40 kHz	[0032] ₁₆ = [50] ₁₀	40 kHz
50 kHz	[0028] ₁₆ = [40] ₁₀	50 kHz
60 kHz	[0023] ₁₆ = [35] ₁₀	57,1 kHz
70 kHz	[0032] ₁₆ = [50] ₁₀	40 kHz
100 kHz	[0028] ₁₆ = [40] ₁₀	50 kHz
1 MHz	[001E] ₁₆ = [30] ₁₀	66,6 kHz

Graphiquement, voici la consignation de nos mesures :



On sait que dès le départ, les bits PR1 et PR2 du registre TMSK2 (adresse \$1024) sont initialisés à "0". Ainsi, on obtient grâce à la documentation constructeur, la période d'un cycle TCNT correspondante : ici 500ns. On observe d'ailleurs, comme indiqué dans le tableau de mesures, que la fréquence théorique correspond à celle de la pratique pour une période de TCNT à 500ns (sauf valeurs en rouge, erronées)!

La courbe est donc linéaire (affine) et diminue (rapport) quelle que soit la fréquence comprise entre

les fréquences de limites haute et basse : $f = \frac{1}{\text{Polling} * \text{cycle}(\text{constante})}$. Il ne vous aura donc pas échappé la confusion des mesures aux fréquences extrêmes (hautes et basses).



Projet de Systèmes Numériques Implémentation logiciel d'un fréquencemètre sur un HC1

Nous allons donc chercher à déterminer les fréquences minimale et maximale de fonctionnement correct du programme.

Nous savons qu'au minimum, le programme exécute 29 cycles, puisque un cycle prend 500ns (voir doc constructeur), on obtient le temps d'exécution minimal, on peut en déduire la fréquence maximale qu'on peut lui fournir en entrée :

Au minimum, on a 29 cycles :

$$f_{\max} = \frac{1}{T} = \frac{1}{29 \cdot 1 \text{ cycle}} = \frac{1}{29 \cdot 5 \cdot 10^{-7}} = \mathbf{69 \text{ kHz}}$$

Au-delà de cette fréquence, le Timer calcule beaucoup trop lentement par rapport à la fréquence de mesure. Les valeurs sont erronées.

Au maximum, nous ne pourrions pas dépasser $(\text{FFFF})_{16} \cdot 1 \text{ cycle}$ puisque notre compteur est un registre de 16bits : soit **65 535 cycles**.

Au maximum, on a $(\text{FFFF})_{16}$ cycles :

$$f_{\min} = \frac{1}{T} = \frac{1}{\text{FFFF} \cdot 1 \text{ cycle}} = \frac{1}{65\,535 \cdot 5 \cdot 10^{-7}} \approx \mathbf{30 \text{ Hz}}$$

Au-dessous de cette valeur, la valeur codée sur 16bits est erronée.

En nous référant à nos mesures et notre graphique, nous constatons en effet qu'en dessous de 30Hz et au dessus de 70kHz, le programme ne fonctionne plus.



Projet de Systèmes Numériques Implémentation logiciel d'un fréquencesmètre sur un HC1

Application d'interruptions

TIC1	equ	\$1010	
TFLG1	equ	\$1023	
TCTL2	equ	\$1021	
TMSK1	equ	\$1022	
	org	\$2050	
front	rmb	2	
period	rmb	2	
ED1ba	fcf	\$10	
		org	\$00E8 ;
		jmp	issprog ;
		org	\$2100
		sei	
Les instructions suivantes ne peuvent être interrompues			
		lds	#\$0200
	ldab	ED1ba	IC1F
	stab	TCTL2	
	ldaa	TFLG1	auto interrupt sur ...
	anda	#\$04	
	staa	TFLG1	
	ldaa	#\$04	IC1i
	staa	TMSK1	
	cli		
boucle	ldd	#\$0000	Place 0000 à l'adresse \$2000, en boucle
	std	\$2000	
	bra	boucle	
issprog	ldd	#\$ACDC	Place ACDC à l'adresse \$2000 sur interruption
	std	\$2000	
	rti		

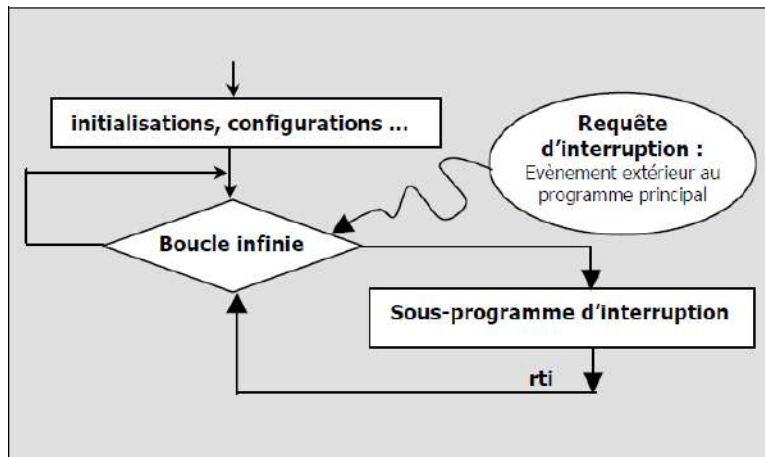
Cette instruction est importante de par ses conséquences. En effet, dès lors que l'on va recevoir une capture d'entrée 1 (vecteur d'interruption à l'adresse \$00E8), la sous-routine de mise à "\$ACDC" va être effectuée. On pourra ainsi savoir si l'interruption a été effective.

INTERRUPT VECTOR	FIELD
Serial communications Interface (SCI)	\$00C4-\$00C6
Serial Peripheral Interface (SPI)	\$00C7-\$00C9
Pulse Accumulator Input Edge	\$00CA-\$00CC
Pulse Accumulator Overflow	\$00CD-\$00CF
Timer Overflow	\$00D0-\$00D2
Timer Output Compare 5	\$00D3-\$00D5
Timer Output Compare 4	\$00D6-\$00D8
Timer Output Compare 3	\$00D9-\$00DB
Timer Output Compare 2	\$00DC-\$00DE
Timer Output Compare 1	\$00DF-\$00E1
Timer Input Capture 3	\$00E2-\$00E4
Timer Input Capture 2	\$00E5-\$00E7
Timer Input Capture 1	\$00E8-\$00EA
Real Time Interrupt	\$00EB-\$00ED



Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencemètre sur un HC1



```

TIC1 equ $1010      *TIC1 se situe a l'@ 1010 (voir la doc)
TFLG1 equ $1023     *TFLG1 se situe a l'@ 1023 (voir la doc)
TCTL2 equ $1021     *TCTL2 se situe a l'@ 1021 (voir la doc)
TMSK1 equ $1022     *TMSK1 se situe a l'@ 1022

      org $2050      *curseur a l'@ 2050
front rmb 2          *réserve de 2octets pour front
period rmb 2         *réserve de 2octets pour period
ED1ba fcb $10        * active fronts descendants

      org $00E8
      jmp issprog

      org $2100      *curseur a l'@ 2100(début du programme)
      sei            *début d'interruptions interdites

      lds #$0200

      ldab ED1ba     *active fronts descendants
      stab TCTL2     *de PA2

      ldaa TFLG1     *enregistrement des
      anda #$04      * fronts actifs de IC1F
      staa TFLG1     *([04]10 = [100]2)

      ldaa #$04      *autorise interruptions
      staa TMSK1     * de IC1i
      cli           *fin d'interruptions interdites

boucle ldd #$0000    *Place "0"
      std $2000      *à l'adresse 2000
      bra boucle     *en boucle

issprog ldd #$ACDC   *place "ACDC"
      std $2000      *à l'adresse 2000

      rti            *retourne au programme
  
```



Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencemètre sur un HC1

Après avoir chargé ce programme en mémoire, nous observons les changements effectués suite aux interruptions volontaires :

Avant la compilation, voici les états mémoire :

```
>md 2000

2000 14 F7 9C 51 EB FE FF 7E BF F8 53 FB 3E 37 9B A7
2010 1D 37 F9 EF 79 FF FF AD E5 E6 F1 FF 7F FB ED AB
2020 9A 9E 57 ED D6 9F 53 41 FB FE FF B7 FF D7 67 FF
2030 7F FB FD 6F F7 9F 7F F9 55 3F F5 9D 7A F5 F3 54
2040 6F F9 96 ED DF EF EF BF 29 AF 67 FB DE F9 EF FF
2050 5B DF 28 9F 10 FF FF 79 DE 9E B8 2D DC C7 8D F5
2060 D2 EB FD D7 FB 7F 9A 76 2C D7 96 EF B2 98 CB 9C
2070 F3 FD EC 67 54 DD 7F F4 8E BF 47 DF BB 47 DB FF
2080 FF 16 3F 3E FD EC FD C1 C6 FA 96 65 AF DA FE DF
>md 00E8

00E0 E3 71 7E E3 71 7E E3 71 7E 21 20 7E E3 71 7E E3
00F0 71 7E E3 71 7E FA 7A 7E E3 71 7E E3 71 7E E3 71
```

Après la compilation, voici les états mémoire :

- Sans interruption

```
BUFFALO 3.4 (ext) - Bit User Fast Friendly Aid to Log
>md 2000   Chargement de #0000 : Boucle
           effectuée sans interruption
2000 00 00 9C 51 EB FE FF 7E BF F8 53 FB 3E 37 9B A7
2010 1D 37 F9 EF 79 FF FF AD E5 E6 F1 FF 7F FB ED AB
```

- Avec interruption

```
>md 2000   Chargement de #ACDC : Boucle
           effectuée avec interruption
2000 AC DC 9C 51 EB FE FF 7E BF F8 53 FB 3E 37 9B A7
2010 1D 37 F9 EF 79 FF FF AD E5 E6 F1 FF 7F FB ED AB
```

Les valeurs ont bel et bien changé, le vecteur d'interruption a donc bien fait effet puisque nous sommes rentrés dans la boucle!



Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencemètre sur un HC1

Puisque nous connaissons maintenant les manipulations d'interruptions, il est temps de mesurer la période de notre signal!

On distingue ici deux types de tâches : celles réalisées par le programme principal et celles de la routine d'interruption. Voici le programme que nous proposons :

```

TIC1      equ    $1010      *TIC1 se situe a l'@ 1010 (voir la doc)
TFLG1     equ    $1023      *TFLG1 se situe a l'@ 1023 (voir la doc)
TCTL2     equ    $1021      *TCTL2 se situe a l'@ 1021 (voir la doc)
TMSK1     equ    $1022      *TMSK1 se situe a l'@ 1022

front1     org    $2050      *curseur a l'@ 2050
front1     rmb    2          *réserve de 2octets pour front1
front2     rmb    2          *réserve de 2octets pour front2
period     rmb    2          *réserve de 2octets pour period
ED1ba     fcb    $10        * active fronts descendants
N          rmb    1          *

          org    $00E8
          jmp    issprog

          org    $2100      * curseur a l'@ 2100(début du programme)

          sei              *début d'interruptions interdites

          lds    #$0200

          ldab   ED1ba      *active fronts descendants
          stab   TCTL2      *de PA2

          ldaa   TFLG1      *enregistrement des
          anda   #$04        * fronts actifs de IC1F
          staa   TFLG1      * ([04]10=[100]2)

          ldaa   #$04        *autorise interruptions
          staa   TMSK1      * de IC1i
          cli      *fin d'interruptions interdites

          ldab   #2          *
          stab   N           *N=[2]10

boucle     tst    N          *boucle de test
          bne    boucle      *de N

          clr    TMSK1      *toute interruption possible

          ldd    TIC1        * TIC1
          std    front2      * stocke le front 2

          subd   front1      *on soustrait le front 1
          std    period      *on obtient la période

          rts              *retour au programme

issprog    ldaa   TFLG1      *enregistrement des
          anda   #$04        * fronts actifs de IC1F
          staa   TFLG1      * ([04]10 = [100]2 )

          dec    N           *décrémenter N
          beq    sortie      *sortie du programme si N=0

          ldx    TIC1        * TIC1 stocke
          stx    front1      * le front 1

sortie     rti

```



Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencesmètre sur un HC1

Nous réitérons donc l'étude en variance de fréquence précédente sur notre nouveau système :

Voici un tableau récapitulant le nombre de cycles observés pour des fréquences de mesures allant de 40 Hz à 1MHz. Cette fois-ci, la période sera observée de cette manière : (exemple à environ 1kHz)

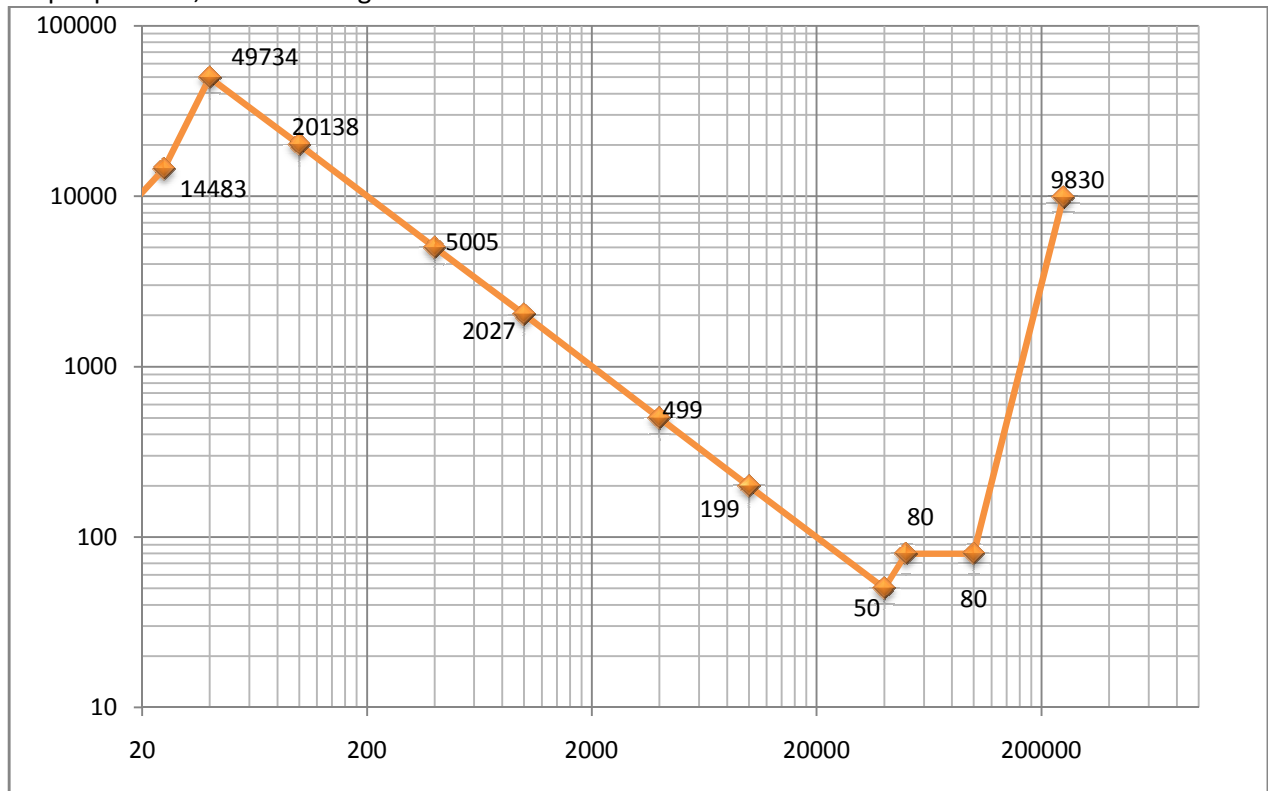
```

front2 front1 periode 07CF=AB0A - A33B
2050 A3 3B AB 0A D7 CF 10 00 DE 9E 18 29 88 87 8D A1
2060 D0 A9 AD 57 DB 7F 9A 6E 2C 82 12 EF 32 98 8B 9C
2070 50 F5 E8 07 54 95 7B F4 88 AC 06 9C BB 43 DB BE
2080 F5 12 3B 1E FD EE D9 C1 C6 FA 06 04 AC DA FE 9B
2090 B3 9E 3F C4 FC 66 1F 3C 3B 37 45 03 E3 1F 1C 3D
20A0 87 67 36 CE F8 24 77 64 3C 5F C6 84 D4 01 2F BA
20B0 D2 6D 06 AA 99 5D EF FC 93 EC D1 BD 8F 4E 6B 37
20C0 76 B9 3F FF 03 4C 79 FA 49 FE BE AA 56 D4 C4 4E
20D0 F1 BE 4E 74 87 EC DE CF 85 EC E7 BB AC EE 53 59
>
  
```

Voici notre tableau de mesures :

Fréquence de mesure	Interruption
10 Hz	$[0F57]_{16} = [3927]_{10}$
25 Hz	$[3893]_{16} = [14483]_{10}$
40 Hz	$[C246]_{16} = [49734]_{10}$
100 Hz	$[4EAA]_{16} = [20138]_{10}$
400 Hz	$[138D]_{16} = [5005]_{10}$
1 kHz	$[07EB]_{16} = [2027]_{10}$
4 kHz	$[01F3]_{16} = [499]_{10}$
10 kHz	$[00C7]_{16} = [199]_{10}$
40 kHz	$[0032]_{16} = [50]_{10}$
50 kHz	$[0050]_{16} = [80]_{10}$
100 kHz	$[0050]_{16} = [80]_{10}$
250 kHz	$[2666]_{16} = [9830]_{10}$

Graphiquement, voici la consignment de nos mesures :





Projet de Systèmes Numériques Implémentation logiciel d'un fréquencesmètre sur un HC1

On recherche à obtenir de nouveau les fréquences minimale et maximale de fonctionnement du programme.

Nous savons qu'au minimum, le programme exécute 29 cycles, puisque un cycle prend 500ns, on obtient le temps d'exécution minimal, on peut en déduire la fréquence maximale qu'on peut lui fournir en entrée.

Au minimum, on a 29 cycles :

$$f_{max} = \frac{1}{T} = \frac{1}{29 \cdot 1 \text{ cycle}} = \frac{1}{29 \cdot 5 \cdot 10^{-7}} = \mathbf{69 \text{ kHz}}$$

Nous savons qu'au maximum, nous ne pourrions pas dépasser (FFFF*1 cycle) puisque notre compteur est un registre de 16bits : soit 65 535 cycles.

Au maximum, on a FFFF cycles :

$$f_{min} = \frac{1}{T} = \frac{1}{FFFF \cdot 1 \text{ cycle}} = \frac{1}{65\,535 \cdot 5 \cdot 10^{-7}} \approx \mathbf{30 \text{ Hz}}$$

En nous référant à nos mesures et notre graphique, nous constatons en effet qu'en dessous de 30Hz et au dessus de 70kHz, le programme ne fonctionne plus.

Fonctionnalité du microcontrôleur : LCD

Nous voici donc capables de compter les cycles, gérer, quantifier le nombre de cycles et interrompre un programme, en un mot (trois précisément) : mesurer la période! Mais nous n'apprenons jamais assez du 68HC11! Un de ses fonctionnalités est l'affichage par le biais d'un écran LCD. Vous nous voyez venir, nous allons afficher la mesure sur l'écran!

Ecran LCD

Le système LCD a la connectique suivante :

Il est accessible suivant deux registres (bien connus) :

- de commande situé à l'adresse \$B5F0
- de données situé à l'adresse \$B5F1

B5F0 Command Register	+5V	2	1	GND
B5F1 Data Register	A0	4	3	Vee
	/LCDCS	6	5	/RW
	D1	8	7	D0
	D3	10	9	D2
	D5	12	11	D4
	D7	14	13	D6

TABLE 1. LCD Command Codes

Command	Code	Delay
Clear Display, Cursor to Home	\$01	1.65ms
Cursor to Home	\$02	1.65ms
Entry Mode:		
Cursor Decrement, Shift off	\$04	40us
Cursor Decrement, Shift on	\$05	40us
Cursor Increment, Shift off	\$06	40us
Cursor Increment, Shift on	\$07	40us
Display Control:		
Display, Cursor, and Cursor Blink off	\$08	40us
Display on, Cursor and Cursor Blink off	\$0C	40us
Display and Cursor on, Cursor Blink off	\$0E	40us
Display, Cursor, and Cursor Blink on	\$0F	40us
Cursor / Display Shift: (nondestructive move)		
Cursor shift left	\$10	40us
Cursor shift right	\$14	40us
Display shift left	\$18	40us
Display shift right	\$1C	40us
Display Function (default 2x40 size)	\$3C	40us
Character Generator Ram Address set	\$40-\$7F	40us
Display Ram Address set	\$80-\$FF	40us

temporisation
longue

temporisation
courte

Voici les quelques commandes à retenir de l'écran d'affichage.

Nous verrons plus tard les temporisations dues à un certain temps de latence lié au système. Elles correspondent à certains types de commandes signifiés ici.



Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencesmètre sur un HC1

Initialisation

Bien évidemment, ce dispositif ne fonctionne pas tout seul, il faut l'initialiser! Voici les quelques procédure à suivre sont indispensables au bon fonctionnement :

- LCDCMD equ \$b5f0 variable correspondant à l'adresse mémoire où se placera la commande que nous utiliserons.
- LCDDAT equ \$b5f1 variable correspondant à l'adresse mémoire où se placera la donnée que nous utiliserons (caractères).
- L4xC20, CLRCH, DCBON, CINSHOF, SCSHR, CSRHOM et leurs adresses respectives : on attribue des variables à certaines commandes-type bien plus simples à utiliser que des adresses mémoire (il est peu parlant de se servir de "\$14").
- Le reste est décrit dans les commentaires du programme suivant :

```

LCDCMD      equ    $b5f0      *
LCDDAT      equ    $b5f1      *
L4xC20      equ    $3c * 4x20  *

CLRCH       equ    $01        *
DCBON       equ    $0f        *
CINSHOF     equ    $06        *
SCSHR       equ    $14        *
CSRHOM      equ    $02        *

                ORG    $2100    *

init         ldaa    #L4xC20    * LCD size
                staa    LCDCMD   * =
                jsr    courte     * 4x20

                ldaa    #CLRCH    *
                staa    LCDCMD   * Clear
                jsr    longue     *

                ldaa    #DCBON    *
                staa    LCDCMD   * Blink off
                jsr    courte     *

                ldaa    #CINSHOF  *
                staa    LCDCMD   * Shift off
                jsr    courte     *

                ldaa    #SCSHR    *
                staa    LCDCMD   * Shift right
                jsr    courte     *

                ldaa    #CSRHOM   *
                staa    LCDCMD   * Home
                jsr    longue     *

*
longue       ldx     #50000      *
loop_l       dex     *
                bne    loop_l     *
                rts              *

*
courte       ldx     #1000       *
loop_c       dex     *
                bne    loop_c     *
                rts              *
*

```



Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencemètre sur un HC1

Premier affichage

Prêt? Allons-y! Tentons donc de rendre notre tout premier caractère, à la suite de l'initialisation, on insert le code suivant :

```
ldaa #'A'      * on garde le caractère 'A'
staa LCDDAT    * il s'agit de l'application d'un caractère
jsr  courte    * puisque c'est un caractère
```

Victoire !



Nous allons maintenant tenter de produire un affichage correcte avec une phrase et, bien évidemment une place pour notre période que nous (ne perdons pas de vue) devons récupérer!

```
org $2000      * A partir de l'adresse $2000
texto fcc 'La periode =' * Réserver la place nécessaire en
                                * octets pour cette chaîne de caractères
                                * Réserve d'octets multiples : ici pour 0
fin fcb #0
[...]
ldy #texto     * stockage de la chaîne de caractères
chaîne ldaa 0,y * observation du registre Y
staa LCDDAT    * commande de données
jsr  courte    * temporisation courte (caractères)
iny           * incrémente Y (caractère suivant)
bra  chaîne    * retourner à Y (tant que la fin n'est pas
[...]          * atteinte.
```



Conversion d'un entier en octet

Le plus simple est passé, fini de faire joujou avec l'écran, il va falloir l'afficher cette période! Et ce n'est pas si simple qu'on ne pourrait le penser : la période est stockée sous forme d'un octet (hexadécimal)! On sait que la période n'atteindra jamais plus de 2octets...

Voici le –très long- programme nous permettant de convertir (cela vaut toutes les explications !) :

```
*** [Initialisation]
*** [Réservation de "texto"]

org $2020

table fcc '0'      *
      fcc '1'      * Réserve d'un table
      fcc '2'      * correspondant à tous
      fcc '3'      * les caractères affichables
      fcc '4'      * (en hexadécimal).
      fcc '5'      *
      fcc '6'      * Chaque caractère est
      fcc '7'      * implanté de façon à ce
      fcc '8'      * que sa valeur corresponde
      fcc '9'      * à (l'adresse de départ) $2020
      fcc 'A'      * PLUS sa valeur.
      fcc 'B'      *
      fcc 'C'      * Exemple :
      fcc 'D'      * Pour afficher "9", on se placera
```



Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencemètre sur un HC1

```

        fcc      'E'          * à l'adresse $2020 + 9
        fcc      'F'          * = $2029 !

octet    fcb      #$36        * à partir de l'adresse $2036, on
q_haut   rmb      1          * réserve par exemple la valeur "36",
q_bas    rmb      1          * 1 octet pour le quartet bas et haut

        ORG      $2100        *

conv     lda      #table      * stockage valeur de la table : $2020
        ldaa     octet        * stockage valeur de l'octet [36]16
        anda     #$0F         * ET (0000 1111) : [0F]16
        adda     0,x          * donnera toujours q.bas : [06]16
        staa     q_bas        * on additionne à $2020 : $2026 : '6'
                                * cela nous donne le caractère q.bas

        ldaa     octet        * stockage valeur de l'octet [36]16
        asra     4            * décalage 4 fois d'un bit :
        asra     4            * 36 = 0011 0110
        asra     4            *      0011 0110
        asra     4            *      0011 0110
        anda     #$0f         * on obtient [03]16, ET [0F]16 = [03]16
        adda     0,x          * on additionne à $2020 : $2023 : '3'
        staa     q_haut       * cela nous donne le caractère q.haut

*** [Initialisation des commandes]

        ldy      #texto      * on retient le texte
        jsr      chaine      * on passe à la sous-routine "chaine"

*** [Sous-routine "longue"]
*** [Sous-routine "courte"]
*** [Sous-routine "chaine"]

suite    ldaa     q_haut      *
        staa     LCDDAT      * affichage du quartet haut
        jsr      courte      *

        ldaa     q_bas        *
        staa     LCDDAT      * affichage du quartet bas
        jsr      courte      *

```

-Again- Victory!

La periode = 36_



Affichage de la période mesurée (le final inaccompli...☹)

Maintenant que nous connaissons parfaitement l'utilisation du Timer et du LCD, il faut apprendre comment déclencher de l'affichage à partir de la saisie d'un caractère...

Pour cela, on utilise –comme précédemment- la liaison série SCI. Cette interface utilise le port D (en binaire) pour communiquer avec le processeur. Cinq registres sont sollicités :

Registres de vitesse et de données : BAUD et SCDR

Le registre BAUD gère la vitesse de liaison grâce aux bits SCP0-1-2 et SCR0-1-2, à l'adresse \$102B.

BAUD				SCP1	SCP0		SCR2	SCR1	SCR0
\$102B									

Ici, initialisé de cette manière : $[0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]_2 = [30]_{16}$

Le registre SCDR consigne la donnée à transmettre ou recevoir, à l'adresse \$102F.

R-17	R-16	R-15	R-14	R-13	R-12	R-11	R-10	SCDR
								\$102F

Ici, laissé aux valeurs par défaut.

Registres d'états : SCSR

Il s'agit du *Serial Status Com. Register* : l'état de liaison, géré par RDRF(réception) et TDRE(émission). Lorsqu'ils sont à 1, ils sont actifs. Ici, ils sont initialisés à $[00]_{16}$, puis mis à $[20]_{16}$:

$[0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]_2 = [20]_{16}$								
TDRE		RDRF						SCSR
								\$102E
Etat du SCSR								

Registres de contrôle : SCCR1 et SCCR2

Nous utilisons les registres SCCR1 et SCCR2. SCCR1 nous indique seulement que nous fonctionnons en mode 8bits de données + 1bit de start + 1 stop. SCCR2, lui, comprend 5 valeurs notables :

- RE : Réception activée à 1
- TE : Emission activée à 1
- TIE : Interruption autorisée par le bit TDRE à 1
- RIE : Interruption autorisée par le bit RDRF à 1

Ici, nous l'initialisons à $[0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0]_2 = [0C]_{16}$

TIE		RIE		TE	RE		SBK	SCCR2
								\$102D
Interruptions				Activation		0		

Puis à $[0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0]_2 = [2C]_{16}$

Autrement dit, les interruptions sont autorisées par le bit RDRF à 1, en activant la réception et l'émission.



Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencemètre sur un HC1

Saisie d'un caractère sans interruption

On cherche, grâce au programme suivant, à sortir d'une boucle infinie lors de la saisie d'un caractère, de le stocker et de le restituer sur l'écran LCD.

```

BAUD equ $102b      *
SCDR equ $102f      *variables pratiques
SCSR equ $102e      *correspondant aux adresses
SCCR1 equ $102c     *des différents registres
SCCR2 equ $102d     *

        org $2200    * début du programme
        clr SCCR1    * Initialisé [00]16 : mode 8bits

        ldaa #$30    * BAUD = [30]16
        staa BAUD    *

        ldab #$0C    * SCCR2 = [0C]16
        stab SCCR2   *

        clr $2000    * valeur [00]16 à l'adresse $2000

loop    ldaa SCSR     *
        anda #$20    * boucle infinie mettant RDRF à 1 :
        beq loop     * la donnée peut être lue

        ldaa SCDR     * stockage de la donnée
        staa $2000    * à l'adresse $2000

        rts          * fin du programme

```

On l'exécute...rien ne se passe (boucle infinie de réception)...on appuie sur la touche 'a' (envoi de la donnée)...on redémarre... Allons vérifier en mémoire l'adresse de stockage (\$2000)...suspense...

```

>call 2200
Saisie du caractere A
BUFFALO 3.4 (ext) - Bit User Fast Friendly Aid to Logic
>md 2000
= 'A'
2000 61 61 20 70 65 72 69 6F 64 65 3D 00 DB 97 69 FE aa
2010 EF FD F7 E5 5F 6A 4B B6 D3 DE FB BB B2 FF 57 F7

```

Encore une victoire !

Saisie d'un caractère avec interruption

Quelle est la suite? Et bien fonctionner en interruption, vous avouerez que redémarrer le microcontrôleur est assez barbare!

Le vecteur d'interruption se situe à l'adresse [00C4]₁₆, la preuve sur la documentation :

Interrupt Vector Jump Table	
INTERRUPT VECTOR	FIELD
Serial communications Interface (SCI)	\$00C4-\$00C6
Serial Peripheral Interface (SPI)	\$00C7-\$00C9
Pulse Accumulator Input Edge	\$00CA-\$00CC
Timer Overflow	\$00DD-\$00DE
Timer Output Compare 5	\$00D3-\$00D5

En implémentant la méthode d'interruption, on obtient le programme final suivant :

```

*** [Initialisation des commandes]
        org $00C4    * vecteur d'interruption SCI
        jmp issprog  * aller à la sous-routine issprog

        org $2200    * début du programme
        sei          * interruptions OK

        clr SCCR1    * Initialisé [00]16 : mode 8bits
        clr SCSR     * Initialisé [00]16

```



Projet de Systèmes Numériques

Implémentation logiciel d'un fréquencemètre sur un HC1

```

        ldaa    #$30          * BAUD = [30]16
        staa    BAUD          *
                                   *
        ldaa    #$2C          * SCCR2 = [2C]16 :réception activée
        staa    SCCR2        * interruptions activées
                                   *
        clr     $2000          *
        cli                                   *
loop    bra     loop          * Boucle infinie (attente d'interruption)

issprog clr    SCSR          *
        ldab    SCDR          * stockage de la valeur
        stab    $2000        *

*** [Sous-routine d'initialisation]
*** [Sous-routine "longue"]
*** [Sous-routine "courte"]
*** [Sous-routine "chaîne"]

        rti

```

On l'exécute...rien ne se passe (boucle infinie de réception)...on appuie sur la touche 'A' (envoi de la donnée)...**on ne redémarre pas !** Mais...que voyons nous apparaître sur l'écran...?



C'est bien ce que nous venons de taper! Et si l'on continue, on peut même envoyer plusieurs lettre d'affilé :

