- ATAHELP-NEWS -

COURS N°1 : Possibilités de POKEY en utilisation du type liaison série ASYNCHRONE. Soit : **liaison ATARI XL/XE vers Minitel**

	$D20F SKCTL			$D208 AUDCTL					(Tableau A)		
	D6	D5	D4	D6	D5	D4	D3	D0			
SEROUT	1	1	1	X		0		0	AUDF2	F64	(1)
	1	1	1	X		0		1	AUDF2	F15	(2)
(Sortie											
série)	**1**	**1**	**1**	**1**		**1**		**X**	**AUDF1+2**	**F1.78**	**(3)**
	1	1	1	0		1		0	AUDF1+2	F64	(4)
	1	1	1	0		1		1	AUDF1+2	F15	(5)
SERIN	1	1	1		X		0	0	AUDF4	F64	(6)
	1	1	1		X		0	0	AUDF4	F15	(7)
(Entrée											
série)	**1**	**1**	**1**		**1**		**1**	**X**	**AUDF3+4**	**F1.78**	**(8)**
	1	1	1		0		1	0	AUDF3+4	F64	(9)
	1	1	1		0		1	1	AUDF3+4	F15	(10)

CONVENTIONS : F1.78, F64 et F15 correspondent à des fréquences internes à votre ATARI. Hélas il y a plusieurs sortes de modèles et/ou de quartz : PAL, SECAM, NTSC. Pour les modèles PAL (cas de mon 130XE (un des premiers) j'ai considéré que F1.78 était égal à : 1.773447 Mhz, alors que le quartz interne est à 8 fois cette fréquence : 14.187576 Mhz. Pour les modèles NTSC (seulement pour les USA, mais je n'ai que les plans de cette version) F1.78 = 1.7897725 Mhz soit un quartz interne à 14.31818 Mhz. Pour les modèles SECAM, il vous faudra ouvrir votre machine pour chercher la fréquence du style 14.xxxxxx Mhz et la diviser par 8 pour en déduire F1.78 SECAM. ATTENTION : n'ayant pas de modèle SECAM sous la main TOUS mes CALCULS sont basés sur le modèle PAL donc F1.78 = 1.773447 Mhz. On obtient F64 par la formule F64=F1.78 divisé par 28, et F15 par F15 = F1.78 divisé par 114.

PARTIE COMMUNE : Pour réaliser une telle liaison série il faut s'occuper des bits 6, 5 et 4 du registre SKCTL ainsi que des bits 6, 5, 4, 3 et 0 du registre AUDCTL de POKEY (Voir les articles sur les liaisons série déjà publiés dans CENACLE-NEWS 1 à 5). Au total il y a 8 bits ce qui fait 256 possibilités à tester ! Après de nombreuses heures de travail j'ai pu réduire à 10 les cas les plus probables, et ce sont eux qui sont résumés dans le tableau ci-dessus.

En fait en ce qui nous concerne, seuls les cas (3) et (8) conviennent. La précision est suffisante de par l'utilisation d'un pseudo-registre de 16 bits, elle couvre les vitesses suivantes : 1200, 2400, 4800, 9600 et 19200 bauds. 300 bauds est couvert par les cas (1) et (6), 75 bauds est couvert par les cas (2) et (7). Les cas 4, 5, 9 et 10 couvrent des fréquences trop basses.

- ATAHELP-NEWS -

CAS de la SORTIE SERIE : ici AUDF1 et AUDF2 sont couplés pour obtenir un compteur 16 bits (nommé AUDF) cadencé à la même fréquence que SALLY, le microprocesseur 6502 de votre Atari.

J'ai utilisé les formules : (cas 3 4 5 8 9 et 10 du tableau A)

AUDF = (Fin/(2*Fout))-7.
Fout = Fin/2(AUDF+7)

Pour les cas 1 2 6 et 7, remplacez (formules ci-dessus) le chiffre "7" par un "3"

Exemple de calcul d'AUDF pour une sortie série à 1200 bauds.

AUDF = (1.773447 Mhz/(2*1200))-7 = 738.93-7 = 731.93
732 sera la valeur choisie pour AUDF

Vérifions dans l'autre sens :

Fout2 = 1.773447 Mhz/(2*(732+7)) = 1.773447 Mhz/1478
Fout2 = 1199.896

L'erreur relative est donnée par : ER% = 100*(Fout2-Fout)/Fout

pour AUDF = 732 on obtient ER% = - 0.0086 %
pour AUDF = 733 on obtient ER% = - 0.14 %

Application pratique : mettre 732 dans AUDF (ou $02DC en héxadécimal) revient à mettre $02 dans AUDF1 (LDA #$02 STA AUDF1 en assembleur) puis $DC dans AUDF2 (LDA #$DC STA AUDF2)

Résumé des différents cas pour F1.78 = 1.773447 Mhz (PAL)

vitesse de SEROUT	valeur AUDF	valeur Hexa AUDF1	valeur Hexa AUDF2	erreur ER%	référence tableau A
1200	732	$02	$DC	-0.0086%	(3)
2400	363	$01	$6B	-0.14%	(3)
4800	178	$00	$B2	-0.14%	(3)
9600	86	$00	$56	-0.68%	(3)
19200	40	$00	$28	-1.74%	(3)
19200	39	$00	$27	+0.40%	(3)
300	103	non utilisé	$67	-0.41%	(1) F64
75	101	non utilisé	$65	-0.30%	(2) F15

(TABLEAU B)

Nota : Il n'est pas possible de piloter la liaison série en utilisant F1.78 relié à AUDF2 (AUDF2 étant alors considéré comme un registre simple de 8 bits). D'un autre coté s'il est possible de relier AUDF1 à F1.78, celui-ci n'a hélas pas de lien avec la sortie série ! C'est donc un cas d'exclusion mutuelle.

Pour 19200 bauds, F64 est inutilisable sur un compteur 8 bits car la fréquence obtenue plafonne à 10556 bauds, avec une très mauvaise précision sur toute la gamme (±31.5 µs).

- ATAHELP-NEWS -

CAS de l'ENTREE SERIE : ici c'est AUDF3 et AUDF4 qui sont couplés pour obtenir un compteur 16 bits. Il s'agit de transformer un circuit synchrone (POKEY) en asynchrone ! L'astuce consiste à obtenir ER% = -5%. Ce décalage de 5% sur le prélèvement de la 1ère valeur SERIN est de 10% au 2ème coup, 15% au 3ème, etc... On se retrouve donc avec un décalage de 45% au 9ème coup, celui-ci correspondant à peu près au milieu du BIT de STOP sur SERIN. Seul l'événement "START" provoquera une remise à zéro du système de prélèvements de POKEY, en rechargeant automatiquement les bonnes valeurs dans AUDF3 et AUDF4.

ER% = -5% signifie que Fout2 = Fout*0.95

ce qui donne pour audf :

AUDF=(FIN/(2*0.95*Fout))-7 ou AUDF=(FIN/(1.9*Fout))-7

Résumé des différents cas pour F1.78 = 1.773447 Mhz (PAL)

vitesse de SERIN	valeur AUDF	valeur Hexa AUDF3	valeur Hexa AUDF4	erreur ER%	référence tableau A
1200	770	$03	$02	-4.90%	(8)
2400	381	$01	$7D	-4.78%	(8)
4800	187	$00	$BB	-4.78%	(8)
9600	90	$00	$5A	-4.78%	(8)
19200	41	$00	$29	-3.78%	(8)
19200	42	$00	$2A	-5.26%	(8)
300	108	non utilisé	$6C	-4.90%	(6) F64
75	106	non utilisé	$6A	-4.85%	(7) F15

(TABLEAU C)

Nota : Il y a toujours un problème de précision aux alentours de 19200 bauds comme expliqué page 2 (sortie série).

MEMENTO des ADRESSES UTILES:

AUDF1 $D200 \
AUDF2 $D202 \ accès en écriture seulement, pour y inscrire les valeurs
AUDF3 $D204 / indiquées dans les tableaux B et C
AUDF4 $D206 /

AUDCTL $D208 \ contrôle du port série (valeurs du tableau A)
SKCTL $D20F / Attention à ne pas modifier les autres bits !

STIMER $D209 réinitialise chaque compteur avec sa valeur AUDFx
SKRES $D20A réinitialise à 1 les bits 5, 6 et 7 de IRQST.

SERIN $D20D en lecture seulement
SEROUT $D20D en écriture seulement
Il faut écrire dans SEROUT juste après l'évènement "IRQ SEROUT VIDE" indiqué par le bit 4 de IRQST, pour garder un bon rendement.

- ATAHELP-NEWS -

CONSEILS D'UTILISATION :

Il faut lire SERIN juste après l'événement "IRQ SERIN PLEIN" indiqué par le bit 5 de IRQST, afin d'éviter que l'octet suivant ne vienne écraser celui qui est en attente dans SERIN. Le temps maximum à ne pas dépasser est donné par : T= 10 divisé par la vitesse en bauds. Il serait même préférable de remplacer le 10 par un 9 dans la formule.

Problèmes liés à la lecture correcte des données arrivant sur la liaison série :

1-NMI (Interruption Non Masquable), par nature on ne peut pas empêcher son déroulement, mais il est possible de détourner l'un de ces vecteurs, celui qui consomme beaucoup de temps machine (à peu près 2000 cycles). Il s'agit du vecteur VVBLKI (Vecteur de Blanking TV Immédiat) qui s'exécute obligatoirement toutes les 20 milisecondes (50 Hz en Europe). Ce programme met à jour l'horloge du système ($12, $13 et $14), gère le mode protection de l'écran par rotation des couleurs, décrémente le "TIMER" logiciel CDTMV1 ($218 $219), et passe éventuellement par un sous-programme CDTMA1 "JSR ($0226)" lors du passage à zéro de CDTMV1, d'où une autre perte éventuelle de temps ! Ensuite l'indicateur "CRITIC" en $42, est testé pour exécuter ou non la deuxième partie de VBLANK, passant par le vecteur VVBLKD (Vecteur de Blanking TV Différé). Il est évidemment dans votre intérêt d'empêcher l'exécution de la partie VVBLKD, d'autant que la gestion du "bip" clavier utilise l'adresse $D40A à tout va, ce qui bloque le 6502 pendant n fois 64µs ! Inconvénients : 15 registres, 3 timers et 24 variables "système" ne seront plus mis à jour ! **Ne désespérez pas**, je compte prochainement vous expliquer en détail les 403 octets de VVBLKI/VVBLKD...

2-IRQ, la priorité de ces routines (en passant par VIMIRQ $0216 $0217) est la suivante : VSEROR, VSERIN, VSEROC, seules les 2 premières nous intéressent.

L'IRQ VSEROR nous indique que la sortie série est vide, et passe le contrôle au sous programme via le vecteur situé en $020C et $020D, donc situé en RAM et modifiable par vous même, pour exécuter votre propre programme.

L'IRQ VSERIN indique que l'entrée série est pleine, et passe via le vecteur $020A et $020B (idem ci dessus).

IRQST $D20F en lecture seulement donne l'origine d'une IRQ
IRQEN $D20E en écriture seulement, 0 masque une IRQ, 1 la valide.
Bit 5 = IRQ entrée série pleine. Bit 4 = IRQ sortie série vide.

A vous de "jouer" maintenant, il vous reste à réaliser le cable entre le connecteur 13 broches "périphéral" de votre 800XL ou 130XE, et le connecteur DIN 5 broches de votre minitel. Vous devrez vous pencher sur le livre "DE RE" d'ATARI, pour apprendre à installer correctement vos programmes sous interruptions, modifier et/ou connaitre VVBLKI, VIMIRQ, le vecteur NMI du 6502.

A SUIVRE : dans mes prochains numéros d'ATAHELP-NEWS, je compte détailler la cartographie "HARDWARE" complète des XL/XE, équations comprises ; la façon de faire une cartouche en EPROM (8Ko, 16Ko bank select) ; expliquer en détail certaines routines du système ; les plans et le fonctionnement d'un clavier de 130XE pour y ajouter des touches de fonctions, etc...

Pour des raisons de temps, je ne peux répondre à vos questions que de manière succincte et par l'intermédiaire du 3615 code TILT choix HELP puis ATARI.
"Réponses sous 48 heures" & "Accès rapide à VOTRE réponse par pseudo minitel"