1 But

Le but de l'analyse qui suit est de définir un circuit capable de décoder le signal employé par le service de présentation du numéro ou du nom de l'appelant. Ces services utilisent une modulation par sauts de fréquences à phase continue (MDF/CPFSK), standard V23 à 1200 bauds.

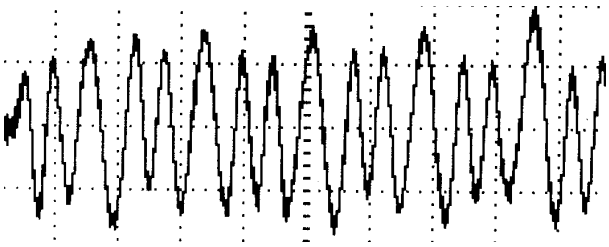
L'analyse considère donc une entrée analogique, et une sortie binaire, soit un train à 1200 bauds.

2 PRINCIPES DE DEMODULATION MDF/CSFSK

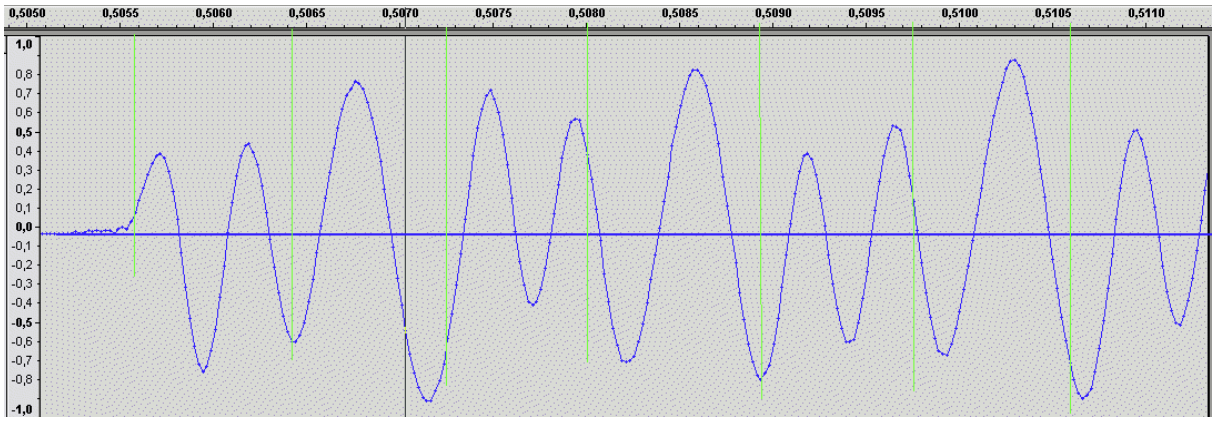
2.1 Examen du protocole V23/1200 bauds

Rappel : Le débit est de 1200 bauds, 833.3 µs

La modulation V23, est de type à Déplacement de Fréquence MDF (FSK) ; c'est à dire que l'état logique 1 est codé par une fréquence de 1300 Hz+/-10Hz, et l'état logique 0 est codé par une fréquence de 2100 Hz+/-10Hz. Cette modulation est bien connue : au début de la micro informatique le code Kansas-City a été utilisé pour sauver des programmes sur bande magnétique , un zéro était codé comme 2 périodes de 1200Hz , un 1 comme 4 périodes de 2400Hz.



Nota : on observe une continuité de phase, la modulation serait donc plutôt du CPFSK que du FSK simple. Cette continuité de phase entre bits, n'est pas spécifiée par la norme V23.



Principes de décodage
La modulation MDF se décode traditionnellement par auto-corrélation et détection de seuil. Un principe plus simple est possible par filtrage sélectifs et détection, si chaque symbole contient suffisamment de périodes. Il est également possible si on a régénéré convenablement l'horloge bit d'effectuer une détection par comptage, on compte les périodes pendant une durée bit .
Cf cours de Mr AUVRAY – Jussieu IST SETI (CSE023_sur_porteuse.pdf).

2.2 Décodage par auto-corrélation

Dans le cas d'un décodeur par auto-corrélation, le signal est multiplié par le même signal k échantillons avant.

$$\sin(\omega t).\sin(\omega t-\omega k/Fe)=[\cos(\omega k/Fe)-\cos(2\omega t-\omega k/Fe)]/2$$

On voit apparaître un terme constant en $\cos(\omega k/Fe)/2$ qui va faire l'essentiel du signal en sortie, et un terme résiduel en $\cos(2\omega t-\omega k/F_e)/2$ qu'il faut minimiser.
Puis on effectue la somme des M derniers produits.

Evidemment, il faut que la fréquence d'échantillonnage soit telle que le nombre d'échantillons par symbole soit supérieur à la profondeur K ou M de l'auto-corrélation. Sinon on passe une frontière de bit. En considérant une fréquence d'échantillonnage de l'ordre de 8000 Hz, on a toujours plus de 6 échantillons par baud.

2.2.1 Détermination de la profondeur de corrélation (paramètre K)

Un simple tableur suffit pour montrer qu'une profondeur de 6 (k=6) est celle qui donne le meilleur contraste entre les signaux 1300 et 2100 Hz, avec un magnifique changement de signe.

freq	1300	1300	1300	1300	2100	2100	2100	2100	distance			
k	4	5	6	7	4	5	6	7	4	5	6	7
7800	-0,50	0,50	1,00	0,50	0,89	-0,57	-0,75	0,75	1,39	1,07	1,75	0,25
7900	-0,55	0,44	1,00	0,58	0,92	-0,48	-0,83	0,64	1,47	0,92	1,82	0,06
8000	-0,59	0,38	0,99	0,65	0,95	-0,38	-0,89	0,52	1,54	0,77	1,88	0,13
8100	-0,63	0,32	0,97	0,71	0,97	-0,29	-0,94	0,40	1,60	0,61	1,91	0,32
8200	-0,67	0,26	0,95	0,77	0,99	-0,19	-0,97	0,26	1,65	0,46	1,93	0,51
8300	-0,70	0,21	0,93	0,82	1,00	-0,09	-0,99	0,13	1,70	0,30	1,92	0,69
8400	-0,73	0,15	0,90	0,87	1,00	0,00	-1,00	0,00	1,73	0,15	1,90	0,87

Cos(ωk/Fe) selon k et Fe, pour des signaux d'amplitudes unité.

2.2.2 Détermination de l'horizon d'intégration (paramètre M) et de la fréquence d'échantillonnage

La meilleure annulation du terme résiduel est réalisée pour une somme 3 ou 6 avec F=1300 et une somme 2,4 ou 6 pour F=2100. Une somme sur un horizon de 6 semble donc être un bon compromis. On peut le démontrer plus finement en analysant l'amplitude des résiduels 1300/2100 en fonction de la profondeur de la somme mais aussi de la fréquence d'échantillonnage :

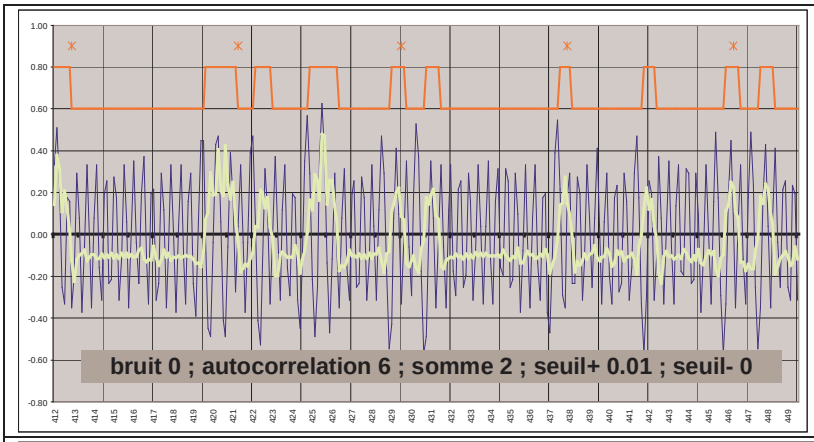
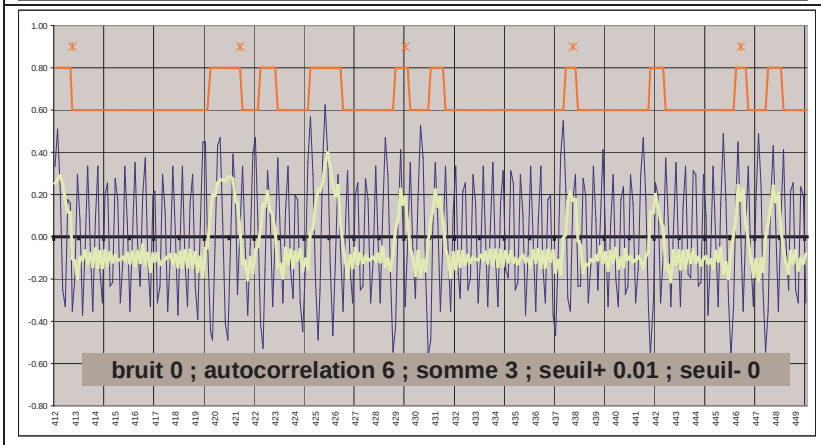
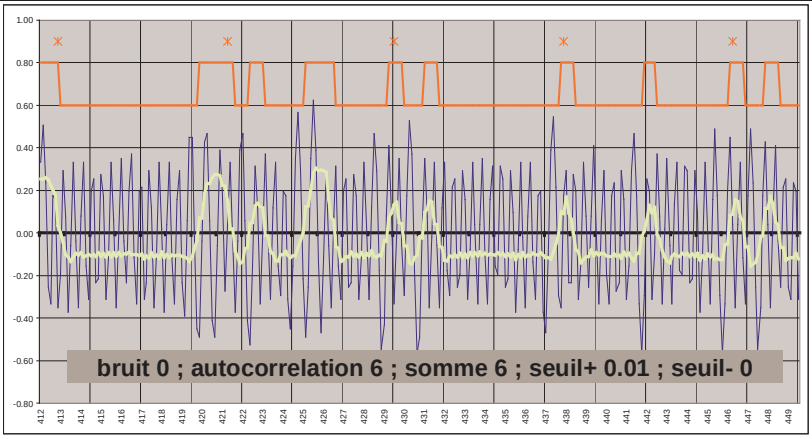
Fe	1300					2100					Somme des carrés				
	S2	S3	S4	S5	S6	S2	S3	S4	S5	S6	S2	S3	S4	S5	S6
7800	1,50	0,00	1,50	1,50	0,00	0,48	1,86	0,92	1,63	1,32	2,48	3,46	3,10	4,91	1,74
7900	1,99	0,09	1,90	2,02	0,18	0,40	1,92	0,78	1,77	1,13	4,12	3,69	4,22	7,21	1,31
8000	2,08	0,18	1,89	2,17	0,37	0,31	1,95	0,62	1,85	0,91	4,42	3,83	3,96	8,13	0,97
8100	2,13	0,27	1,84	2,24	0,54	0,23	1,97	0,46	1,91	0,68	4,58	3,96	3,58	8,65	0,76
8200	2,17	0,36	1,78	2,29	0,72	0,15	1,99	0,31	1,96	0,46	4,73	4,09	3,26	9,09	0,73
8300	2,20	0,45	1,71	2,35	0,88	0,07	2,00	0,13	1,99	0,19	4,84	4,20	2,94	9,48	0,81
8400	2,25	0,54	1,64	2,38	1,05	0,00	2,00	0,00	2,00	0,00	5,06	4,29	2,69	9,66	1,10

Amplitudes crête à crête des résiduels en 2omega K=6, M de 2 à 6, selon Fe

On constate qu'une somme 6 est effectivement la seule qui baisse les résidus franchement, de plus, une fréquence d'échantillonnage de 8100Hz est plus favorable.

2.2.3 Simulation : sensibilité à la profondeur d'intégration (horizon, paramètre M)

Une simulation a été codée pour visualiser le signal de corrélation selon les paramètres K,M. Ici on cherche à visualiser l'effet de l'horizon d'intégration.

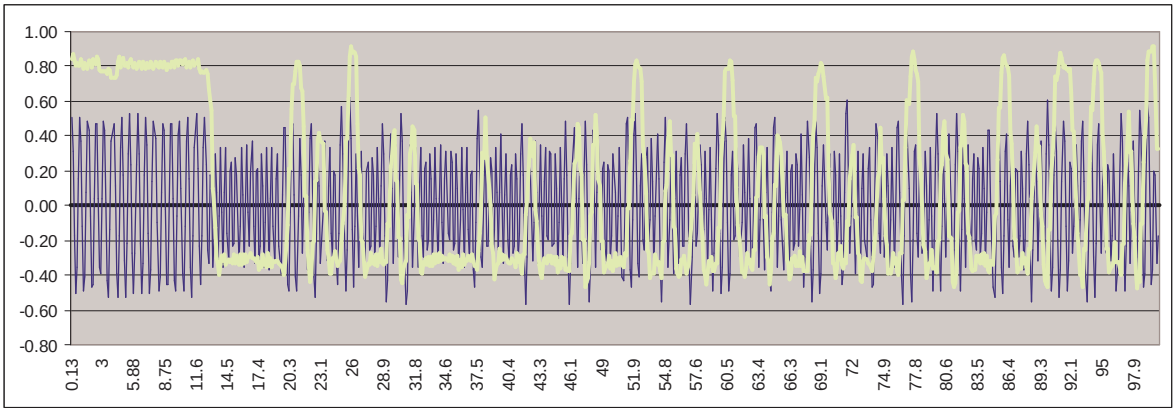
	Autocorrélation avec un délai de 6/Fe et sommation sur 2 produits consécutifs. Le bruit sur le 2100 Hz est atténué, mais pas celui sur le 1300 Hz.
	Autocorrélation avec un délai de 6/Fe et sommation sur 3 produits consécutifs. Le bruit sur le 1300 Hz est atténué, mais pas celui sur le 2100 Hz.
	Autocorrélation avec un délai de 6/Fe et sommation sur 6 produits consécutifs. Les bruits sur le 1300 Hz et sur le 2100 Hz sont atténués au maximum.

Dans ces 3 cas, et même si la théorie dit qu'une profondeur de 6 est meilleure, on notera que l'allure du signal numérique reconstitué ne varie guerre.

2.2.4 Simulation : auto-corrélation K=6 et M=6

Donc en considérant K=6 et M=6, on n'est pas loin de l'optimal.
On voit que le signal utile est de l'ordre de +/-3.V², V étant l'amplitude des sinus élémentaires, le résiduel est lui proche de +/-0.35.V². On constate un rapport « signal à bruit » supérieur à 8.

Exemple avec un échantillon 8 bits 8kHz. Le signal numérique apparaît parfaitement bien. On peut même lire le premier octet à 80h.



La détection se fait ensuite avec un seuillage à zéro. On constate que le signal est propre, très véloce au moment du passage à zéro, c'est le gage d'une détection sans ambiguïté, et peu sensible aux bruits.

2.2.5 Algorithme de niveau bit

L'autocorrélation est un processus continu, qui tourne à la même vitesse que l'échantillonnage, par exemple 8100 Hz, soit un échantillon tous les 123.4567µs (pas voulu, mais remarquable suite…).
A chaque échantillon :

- Décale la pile des 6 dernières acquisitions, et on met la plus récente
- Multiplie Ack(t) avec Ack(t-k)
- Décale la pile des 6 derniers produits, et on met le plus récent
- Somme les M derniers produits ensemble
- Seuillage à zéro, et on décide si le bit courant est à UN ou ZERO ; Attention au signe qui dépend de K

A ce niveau, on peut déjà commencer à traiter des particularités d'un train V23/CLID en détectant le préambule de bits à UN :

- Si le bit vaut UN
  - Augmente le compteur de UN
  - Si le compteur de UN dépasse un seuil
    - Séquence MARK détectée
- Si le bit vaut ZERO
  - RAZ le compteur de UN, RAZ séquence MARK

Comme le train binaire à décoder est à 1200 bauds, on a 6,75 échantillons par bit à reconstituer. C'est bien suffisant pour reconstruire le train série, mais ce n'est pas rond.
Pour éviter les erreurs, on va créer un compteur incrémenté à chaque échantillon, et pour des valeurs prédéfinies de ce compteur, on échantillonnera un bit logique. Evidemment, avant de faire cela, il faut se synchroniser et détecter un bit start.

Exemple avec Fech=8100 Hz et 1200 bauds

- Premier front 1→0 détecté, on suppose qu'on est au bord du bit start
- 3 coups plus loin on vérifie que le niveau courant est ZERO, on est au milieu du start

- 10 coups plus loin on échantillonne le bit 0 (LSB)
- 16 coups plus loin on échantillonne le bit 1
- …
- 64 coups plus loin on vérifie que le niveau courant est UN, on est au milieu du stop, et l'octet reçu et valide, et mis à disposition du traitement faible vitesse (10/1200 sec).
- …et puis on attend le front 1→0 suivant.

Cet algorithme est très robuste, il peut encore être amélioré en faisant des échantillonnages multiples voire des votes majoritaires autour de la position milieu de chaque bit. Par exemple, vérifier qu'à 2, 3 et 4 coups du début, on a bien un niveau ZERO.

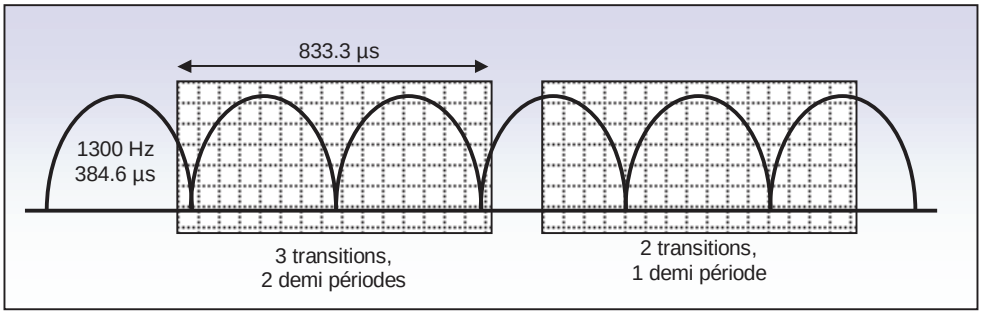
2.2.6 Implantation

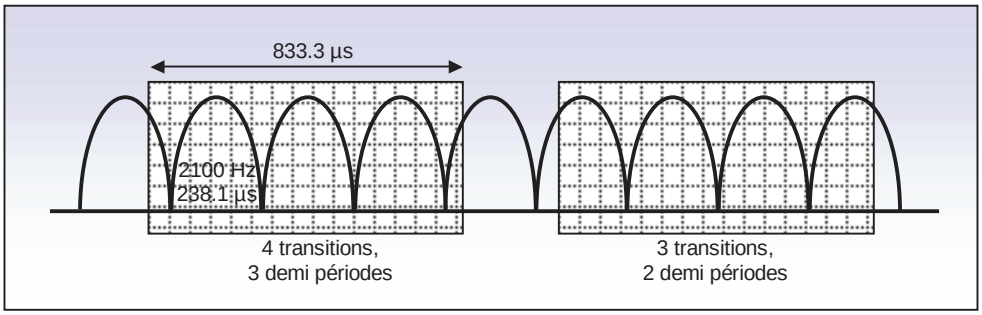
L'autocorrélation n'est pas compliquée à coder.
La partie rapide est insérer directement dans la routine interruption ADC.

2.3 Décodage par mesure de demi-périodes

Si l'autocorrélation est le procédé le plus performant, il n'est pas le plus simple à mettre en œuvre. La première grandeur accessible facilement est la mesure temps réel des demi périodes, entre 2 passages à zéro. Il faut partir de cela et imaginer un principe qui consiste à fortement amplifier le signal, puis à mesurer les demi périodes. Selon la fréquence (1300/2100) on peut observer 1, 2 ou 3 demi périodes :

- au moins 1 demi période de 346.6 µs avec la fréquence de 1300 Hz (état 1)
- au moins 2 demi périodes de 238.1 µs avec la fréquence de 2100 Hz (état 0)

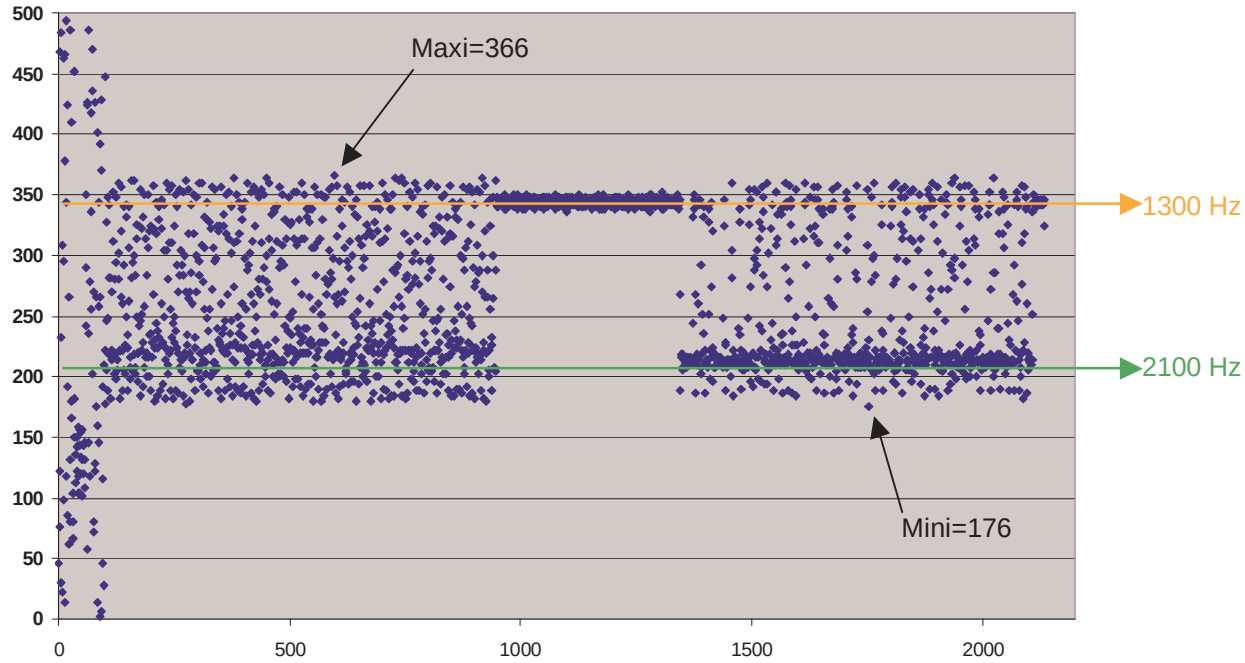




2.3.1 Mesure d'une séquence V23

Le tracé suivant est le résultat d'une acquisition réelle, chaque point représente la mesure d'une demi période entre 2 passages à zéro. L'échelle est en tic d'horloge, soit ici 4/37595000 = 1.1147 µs.
Une fréquence de 1300 Hz a une demi période de 384µs et donnera une mesure de 345 tics.
Une fréquence de 2100 Hz a une demi période de 238µs et donnera une mesure de 218 tics.

On distingue parfaitement :
- Du bruit en amont
- La séquence de synchronisation, ~850 demi périodes alternées (suite de U)
- La séquence repos (mark, niveau UN constant, 1300 Hz)
- Le message proprement dit, avec une forte densité de ZERO (2100 Hz)
- Et la terminaison de quelques bits à UN



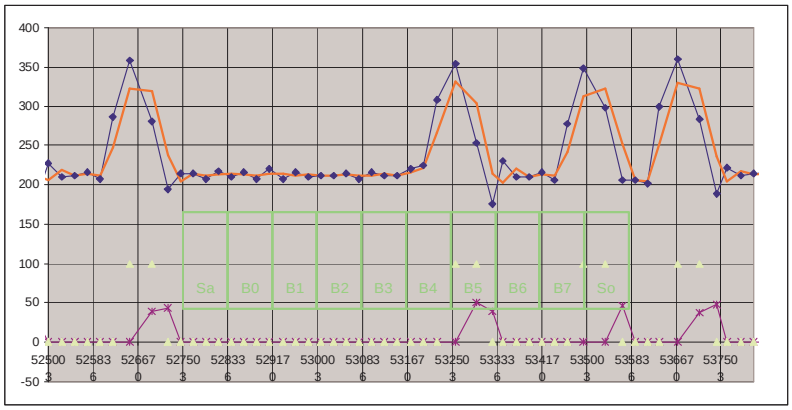
On constate que :
- On peut avoir des mesures au delà des fréquences pures ; cela vient d'un décalage en tension des demi périodes mesurées : 366 maximum soit +6.8%, et 176 minimum soit –19.3%
- Que cet écart est plus fort sur les fréquences hautes, sans doute à cause d'une amplitude plus faible et donc une sensibilité plus grande aux offsets

2.3.2 Algorithmes

A partir de cette mesure de demi-période, il faut recréer un train binaire qui a une chance d'être vu comme une train série correct. Tout va être conditionné par :
- la détection du bit start
- la façon de décider qu'on a un bit UN ou un bit ZERO

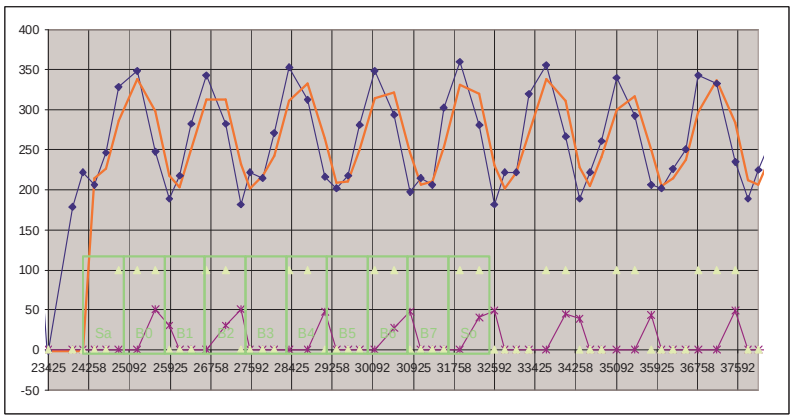
Dans tous les cas, on va avoir 2 processus au niveau du microprocesseur :
- Un processus qui mesure les demi-périodes, et détecte le bit start le cas échéant
- Un processus qui échantillonne le train série à 1200 bauds pour retrouver les données, et détecte la fin d'un octet pour le mettre à disposition des traitements ultérieurs.

Zoom aux alentours du minimum (176 @1752)



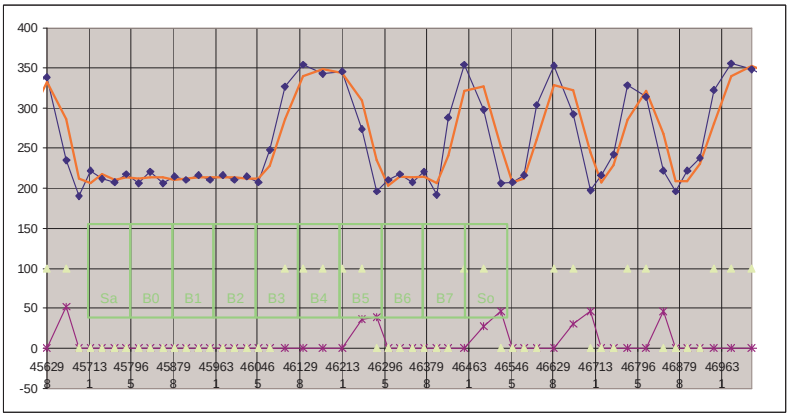
Moyenne de 2
Delta = Moyenne – courant
Début bit start si Delta >25 et Moyenne <279
Ou
Début bit start sur deuxième ZERO détecté

Zoom aux début de la zone synchro (@101)



Moyenne de 2
Delta = Moyenne – courant
Début bit start si 2 mesures consécutives <250
Puis
Début bit start si Delta >25 et Actuel <250

Zoom aux 5ème octet du message (30h @1474) *octet toujours en erreur avant*



Moyenne de 2
Delta = Moyenne – courant

Start si Delta >25 et Moyenne <279 ?????????
Début bit start sur premier ZERO détecté

2.3.2.1 Première solution simple

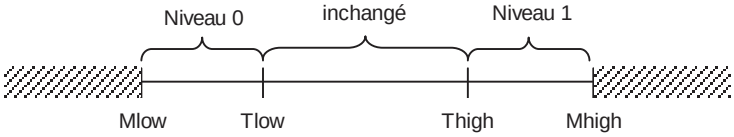
L'algorithme de détection des niveaux logiques est simple :

- si la demi période mesurée est supérieure à Th, le niveau devient 1,
- si 2 demi périodes consécutives mesurées sont inférieures à Tl, le niveau devient 0,
- sinon, on ne change pas le niveau, c'est à dire que le niveau précédent perdure.

A ces deux seuils, on rajoutera des seuils d'acceptabilité de la mesure.

- si la demi période est inférieure à Ml, la mesure est invalide
- si la demi période est supérieure à Mh, la mesure est invalide

Pb : un état UN est détecté plus vite qu'un état ZERO

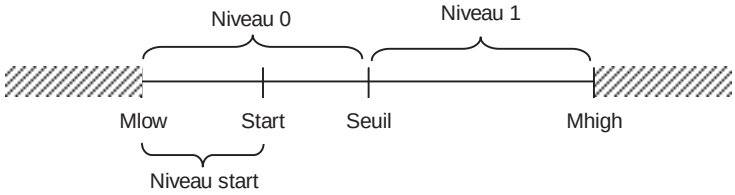


Le bit start est détecté dès qu'on a un niveau ZERO valide, c'est à dire dès la première demi période en dessous de Tlow.

Cette solution est ok si on décode le train série à l'extérieur, par un PC par exemple. Elle ne donne pas des résultats satisfaisants même si on compense le délai de détection, lorsqu'on cherche à décoder le train série dans le même micro.

2.3.2.2 Deuxième solution

L'algorithme de détection des niveaux logiques utilise 2 mesures consécutives, dont il fait la moyenne. Ceci permet de s'affranchir un peu des offsets et décalages de détection. Un seuillage simple est utilisé, il correspond à la moyenne des 2 périodes.



Le bit start est détecté si la moyenne de deux demi périodes successives est inférieure à un seuil assez bas, typiquement quelques pourcent au dessus de la demi période 2100Hz.

Le train série est correctement décodé, sans aucune erreur, la somme de contrôle est correcte.

2.3.3 Valeurs des seuils

Les erreurs sur la détection des seuils sont récapitulées ci après :

	1300 Hz 384µs	2100 Hz 238µs	Influe sur moyenne 2 demi périodes
Fréquences 1300/2100, avec leurs tolérances à +/- 10Hz	6µs	2.2µs	Oui
Résolution de la mesure, LSB du timer	0-2µs	0-2µs	Oui
Précision de le mesure numérique, durée de la boucle de recherche des fronts	0-3µs	0-3µs	Oui
Précision de l'horloge locale	0	0	
Déphasages dus aux filtres RC LC	11.5µs	0.1µs	Oui
Décalages, offsets	+/-19µs	+/-12µs	Non
Retards	+/-10µs	+/-10µs	Non
Variation de la valeur moyenne	+/-19µs	+/-12µs	Non
Total sur 1/2 période	**70µs (19%)**	**43µs (18%)**	
Total sur période	**22µs (3%)**	**9µs (2%)**	

Ces tolérances sont en accord avec la mesure réelle où on a capté du 2100Hz à –19% et du 1300Hz à +7%

On prendra :

- Mlow (2100 Hz+erreurs) = 238µs – 25%
- Mhigh (1300 Hz-erreurs) = 384µs + 10%

2.3.4 Implantation logicielle

2.3.4.1 Version simple sans interruption

Le code consiste à attendre les transitions 1→0 et 0→1 avec des boucles logicielles.
Chaque boucle dure 3 cycles au minimum, soit 3µs avec un quartz de 4 MHz. Noter que ces boucles sont éternelles, elles supposent qu'il existera toujours un bruit suffisant sur la ligne pour ne pas rester coincé, on pourra aussi utiliser un watchdog pour sortir de la boucle.

La gamme de mesure vaut 256 fois la période du timer, soit 512 µs pour la détection V23 (1300/2100 Hz), donc une horloge timer à Fx/4/2 = 2 µs @4MHz

On enchaîne 3 routines

1. La première traite la demi période, et positionne la sortie BITLU, qui peut alors être lue par un PC réglé sur 1200 bauds.
2. Si on vient de passer un baud (833 µs), la deuxième routine compte le nombre de bits alternés successifs, séquence typique d'une série de caractère des synchro UUUUU. Si on en détecte suffisamment, la sortie SYNCHRO est validée

Cette version marche bien pour générer un train série qui sera lu par un PC par exemple.

2.3.4.2 Version étendu avec interruption

En utilisant un PIC16F627/628 doté d'un timer 16 bits à capture on dispose d'un outil puissant pour mesurer les demi périodes.
Sur interruption CCP1 il faut faire :

- sauver le registre de capture
- calculer la demi période terminée,
- calculer la moyenne des 2 dernières demi périodes
- inverser le front de capture
- si la demi période est une V23…
    - si les 2 demi périodes sont autour de 1300 Hz, c'est un bit UN
    - si les 2 demi périodes sont autour de 2100 Hz, c'est un bit ZERO
        - si un octet n'est pas en cours
            - démarrage timer 2
            - octet en cours = 1
- si les 2 demi périodes sont une 440 Hz…
- si les 2 demi périodes sont une 330 Hz…
- si les 2 demi périodes sont une 50 Hz…
- sinon, raz de toutes les variables
- raz le drapeau d'interruption et retour interruption

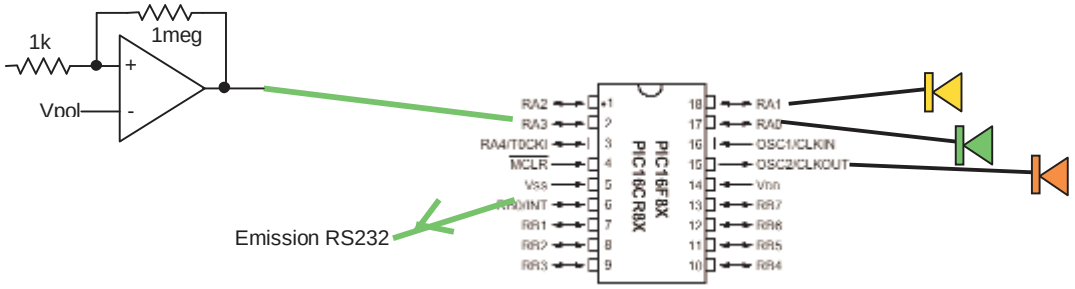
Les valeurs par défaut doivent être initialisée au seuil, afin de détecter les niveaux corrects dès la première demi période.

Le timer 2, associé à un registre de remise à zéro, permettra de cadencer l'échantillonnage du train série à 1200 bauds. Il est activé sur détection du bit start.
Sur interruption TMR2 il faut faire :

- échantillonner le dernier bit reçu
- le ranger (shift right car LSB en premier)
- si c'est le 9ème (bit MSB de la donnée vient de rentrer, start vient de sortir)
    - ranger l'octet reçu et monte le drapeau de mise à disposition pour le traitement à faible vitesse
    - arrêter le timer 2
    - octet en cours = 0

3 PROTOTYPAGE

3.1 version simple (V23 seul)



On utilise un signal CID capturé et recalibré avec Audacity, qu'on réinjecte dans le montage.
Captures : 50mV crete, seuils 95, 143, 173, 211 ; 2 fois identiques hors glitch

```
EF 60 00 C0 00 80 C0 00                          glitch
44 55 15 55 55 44 55 15 55 55 55 55 15 55 55     29 caracteres 'U'
55 55 54 55 15 55 55 44 55 15 55 55 44 D5
80 16                                            entête et longueur totale
01 08 30 32 31 36 70 38 32 32                    date et heure 12160822
02 0A 20 36 xx xx xx xx xx xx xx xx      numéro de appelant 06xx xxxx
C8                                               check sum
FF 1E 1C 03 FF F8 00 06 00                       glitch
```

Constatations :

- Le champ des dates/heures n'est pas correct, mmjjhhmm, 0216p822 n'est pas correct, c'était en décembre pour commencer
- Et donc, le check sum est faux (somme totale = 0x052F au lieu de 0), on calcule 0x0467 soit un check sum de 0x099 ; il y a 3 erreurs en tout.
- On lit que 29 caractères à partir du 44, dans l'entete
- Dans ces 29 caractères, on voit des erreurs simples plutôt de collage à 0 ; 44 ou 15 au lieu de 55.
- Le dernier des 29 caractères est D5 au lieu de 55, un collage à 1 sur le bit de poids fort, ce qui laisse penser qu'on est décalé de 2 bits sur toute la séquence, et que donc on a
- 14 erreurs collées à 0, et 1 erreur collée à 1
- Au scope, les périodes de 1300 Hz sont un peu amplifiées par rapport au 2100 Hz. L'injection est faite par la sortie casque de l'ampli, on suspecte que les correcteurs de tonalités détériore le signal généré.

Hypothèse : l'état 1 semble fragile ! En se connectant directement en sortie ligne, sans correction de tonalité possible et en élargissant un peu la gamme de décision de 10% à 20% sur le 1300 Hz.

Captures : 50mV crête, seuils 95, 143, 154, 231 : 2 fois identiques hors glitch

```
FC 98 44 EE FE FE C4 FF 08 FC FE 71               glitch
55 55 55 55 55 55 55 55 55 55 55 55 55 55         28 caracteres 'U'
55 55 55 55 55 55 55 55 55 55 55 55 55 55
80 16                                             entête et longueur totale : 22
01 08 31 32 31 36 20 38 32 32                     date et heure : 12160822
02 0A 30 36 xx xx xx xx xx xx xx xx       numéro de appelant : 06xx xxxx
C8                                                check sum
EE 83 BF 5E FE FE EE BE….                         glitch
```

Constatations :

- On voit nettement 28 octets à 55

L'heure et le numéro sont bons, le check sum est correct.

Les essais à 100mV crête, 200mV sont bons sans erreurs, mais à 300 et 400 mV crête, on observe une erreur collage à 1 (76 au lieu de 36)

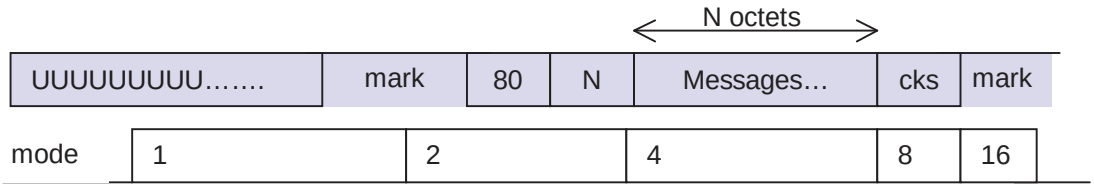
Après réglage des paramètres, on arrive à une détection qui fonctionne
Le logiciel développé comporte un espion qui envoie des octets lors d'évènements remarquables :

Hexa	ASCII	
53 28 73 55 5F 80 16 01 08 30 36 30 32 30 38 35 33 02 0A 30 36 3x 3x 3x 3x 3x 3x 3x 3x Cx 73 73 73 73 29 0D 0A	S(sU_€...06020853..06xxxxxxxÆss ss)..	Sonnerie/inversion ligne Sonnerie V23 synchro V23 Mark V23 message V23 contrôle Fin de temporisation
44 28 54 30 35 36 3x 3x 3x 3x 3x 3x 3x 54 54 54 29 0D 0A	D(T056xxxxxxTTT)..	Décroché Tonalité 440Hz Numéro composé Tonalité 440Hz Tonalité 440Hz Tonalité 440Hz Raccroché

4 DECODAGE DU MESSAGE SERIE

4.1 Premier niveau

La séquence V23 est pré-décodée grâce à un algorithme à états, activé chaque fois qu'un octet est disponible.



Mode = 0	Attente d'une synchro, soit N caractères U successifs Si N > seuil, alors Mode = 1 Reset V23stack (piles des derniers bits V23 reçus) Sinon, on boucle
Mode=1	Attente de la séquence mark, N bits à UN Si on a détecté MARK, alors Mode = 2 Clear checksum Index = V23message Sinon, on boucle
Mode=2	Si on a un octet valide, alors Checksum = + octet index++ @index = byte Si index=2, alors Noctets = byte Mode = 3 Sinon, on boucle Sinon, on boucle
Mode=4	Si on a un octet valide, alors Checksum = + octet index++ @index = byte Noctets = Noctets - 1 Si Noctets = 0, alors Mode=4 Sinon, on boucle Sinon, on boucle
Mode=8	Si on a un octet valide, alors Checksum = + octet index++ @index = byte Si Checksum <> 0, erreur de message, sinon message valide Mode = 5 Sinon, on boucle
Mode=16	Enregistrement du message : V23id,message Reset V23
Erreur V23	Si mode = 0 ou 1 alors reset V23 Si mode = 2,4,8 alors, mode=16

4.2 Deuxième niveau

Une fois qu'un message valide (somme de contrôle correcte), on peut le décoder et chercher les messages.

- cherche le paramètre 0108 mois-jour-heure-minute
- cherche le parametre 02xx numéro de l'appelant
- cherche le parametre 040150 raison de l'absence du numéro (O/P)
- cherche le parametre 07xx nom
- cherche le parametre 080150 raison de l'absence du nom (O/P)

5 REFERENCES

- **ETSI TS 103 021 :** Access and Terminals (AT); Harmonized basic attachment requirements for Terminals for connection to analogue interfaces of the Telephone Networks; Update of the technical contents of TBR 021, EN 301 437, TBR 015, TBR 017.
  Document en 3 parties Possible à downloader en donnant son email.
- **STI** Spécification Technique d'Interface France Télécom
  - STI1 : caractéristiques de l'interface d'abonné analogique
  - STI2 : sonneries, tonalités et numérotation sur les lignes analogiques
  - STI4 : caractéristiques de l'interface usager-réseau pour la transmission de données modem (V23)
- **Projets perso V23/CID**
  - Projet RAT (Remote Applications Terminal) de Ken Boak , très intéressant malgré quelques manques de rigueur dans le design, qui amène une reproductibilité non garantie.
  - Projet de Matthieu Benoit, décodeur CID sur 68HC11, superbe étude basée sur un MC145447P avec beaucoup de références
  - Projet de Danieljm, décodeur CID sur 89C2051 avec un circuit modem MC145447P
- **Circuits téléphoniques**
  - Charles Wenzel's TECHLIB
  - Tomi Engdahl's Telephone line audio interface circuits
  - Tomi Engdahl's Telephone line surge protection
  - Tomi Engdahl's Telephone ringing circuits
  - Epanorama links on telephone
  - Atexa interface téléphone pour un µcontrolleur DTMF in/out