République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique
ENP
Département d'Electronique
Option : Signal et Communications

Mémoire

En vue de l'obtention du diplôme
De Magister en électronique

Présenté par : HIMA Abdelkader.
Ingénieur d'état en électronique (ENP)

Thème

Mesure de fiabilité d'une liaison numérique.

Soutenu le 17/04/2002, devant le jury composé de

M. HADDADI	*Maître de conférences à l'ENP*	*Président.*
M. MEHENNI	*Maître de conférences à l'ENP*	*Rapporteur.*
A. BELOUCHRANI	*Maître de conférences à l'ENP*	*Examinateur.*
C. LARBES	*PhD à l'ENP*	*Examinateur.*
R. ZERGUI	*Chargé de cours à l'ENP.*	*Examinateur.*
A. MOUSSAOUI	*Maître assistante.*	*Invitée.*

Remerciement :

Nous tenons à présenter nos vifs remerciements à mon promoteur Monsieur M. MEHENNI qui a proposé et dirigé soigneusement ce modeste travail.

Que Monsieur HADDADI M., Maître de conférences à l'ENP, nous apprécions votre dynamisme à l'égard des technologies avancés. Nous souhaitons que ce travail sera digne de l'honneur que vous nous témoignez en acceptant de présider ce jury.

Que Monsieur BELOUCHRANI A., Maître de conférences à l'ENP, trouve ici l'expression de notre profonde reconnaissance pour l'honneur qu'il nous fait de juger nos travaux.

Que Monsieur LARBES C., PHD à l'ENP, trouve ici l'expression de notre profonde reconnaissance pour l'honneur qu'il nous fait d'examiner ce travail.

Que Monsieur ZERGUI R., Chargé de cours à l'ENP, soit remercié pour l'honneur qu'il nous fait d'examiner ce travail.

Que Madame MOUSSAOUI A., Maître assistante à l'ENP, trouve ici l'expression de notre profonde reconnaissance pour l'honneur qu'elle nous fait d'accepter notre invitation.

Nous tenons à remercier également :

- *Le Commandant NOUASRI A., Chef chair trans-détection et guerre électronique au GET, pour son aide en nous permettant d'exploiter le laboratoire des télécommunications.*

- *Le commandant OUFAR N., Premier adjoint du chair trans-détection et guerre électronique au GET, pour son aide et ses précieuses suggestions.*

- *Le capitaine CHAOUCH A., responsable du laboratoire des télécommunications au GET, pour son aide.*

A tous ceux qui ont participé d'une manière ou d'une autre à l'aboutissement de ce travail trouvent ici toutes nos gratitudes et nos vifs remerciements.

إهداء :

- أهدي هذا البحث المتواضع إلى والديّ الكريمين أعزهما الله وزادهما بسطة في العلم والجسم،

- وإلى إخوتي الأفاضل وأخواتي الفضليات أدام الله اجتماعنا على الخير،

- وإلى ابن البنين أبقاها الله ذخرا لزوجها،

- وإلى الكتكوت عبد الرحمان، والبرعمة الصغيرة مروة قرة عيني والديها.

- وإلى قوات بحريتنا الأشاوس من القمة إلى القاعدة وأخص بالذكر:

الرائد نواصري، الرائد عوفار، الرائد رقوحة، الرائد أوهنية، الرائد قيطراوي، النقيب شاوش مع تحية خاصة، المساعد الأول بهلول، المساعد الأول منصور، والعريف مصطفى وإلى كل الضباط الصامين والضباط وصفوف ضباط شعبة الإتصالات والكشف والحرب الإلكترونية.

- وإلى جميع الزملاء والأصدقاء والأحباء وأخص بالذكر:

خ علي، ل علي، ح طلال، ب الحفناوي، ب أحمد، موسى، يوسف، محمد، رشيد...

ملخص :

يهدف هذا البحث إلى تحقيق نظام يسمح بقياس فعالية رابطة اتصال رقمية، حيث قمنا بإنجاز بطاقة تعديل FSK وبطاقة كشف FSK و من ثمَّ ربط الأولى بنهائي إرسال للمعطيات الرقمية والتالية بنهائي استقبال للمعطيات الرقمية. ثم إرسال متتالية رموز واستقبالها وإجراء مقارنة بين المتتاليتين.

الكلمات المفتاحية :

معدل FSK، كاشف FSK، احتمال الخطأ، وصلة رقمية، دارة التقرير.

Abstract :

The purpose of this work is the realization of a system permitting the measure of the reliability of a numeric communication link. We conceived an FSK modulator and demodulator. The first bind to a data transmission terminal while the second bind to a data receipt terminal. Then we send a sequence of symbols and we receive them and make a comparison between the two sequences.

Key words :

FSK modulator, FSK demodulator, error probability, numeric link, decision circuit

Résumé :

Le but de ce travail étant la réalisation d'un système permettant la mesure de la fiabilité d'une liaison de communication numérique. Pour cela, nous avons conçu un modulateur et un démodulateur FSK. Le premier étant lié à un terminal de transmission de données tandis que le deuxième est connecté à un terminal de réception de données. Ensuite nous transmettons une séquence de symboles et nous la recevons. Par suite, nous faisons une comparaison entre les deux séquences.

Mots clés :

Modulateur FSK, démodulateur FSK, probabilité d'erreur, liaison numérique, circuit de decision.

INTRODUCTION

INTRODUCTION GENERALE :

Ce travail a été fait dans le cadre d'un projet qui a comme objectif : la réalisation d'un système de mesure de fiabilité d'une liaison numérique.

Le système que nous allons étudier et réaliser comporte plusieurs parties :

- Partie micro-contrôleur : c'est l'exploitation de la carte d'interface réalisée par [1] dans le projet de fin d'étude intitulé « Réalisation d'une Interface Récepteur-Micro-ordinateur » soutenu à l'ENP en 1999,
- Partie modulateur : où nous allons étudier et réaliser un modulateur numérique FSK,
- Partie canal de communication : qui est le câble téléphonique,
- Partie démodulateur : où nous allons étudier et réaliser un démodulateur FSK,
- Partie décision : c'est l'étude et la réalisation d'un bloc de décision permettant la mise en forme du signal démodulé.

Le plan de travail de ce projet étant le suivant :

Dans le premier chapitre nous allons étudier les diverses modulations numériques pour justifier le choix de la modulation FSK.

Dans le $2^{ème}$ chapitre nous allons étudier en bref le microcontrôleur 68HC11 et sa mise en œuvre.

Dans le chapitre 3 nous allons étudier et réaliser le modulateur FSK, le démodulateur FSK et le bloc de décision.

Dans le chapitre 4 nous allons exploiter le système de communication que nous avons réalisé pour mesurer la fiabilité de la liaison numérique établie, ensuite nous terminons la thèse par une conclusion générale.

CHAPITRE I :
GENERALITES SUR LA TRANSMISSION DES DONNEES NUMERIQUES.

CHAPITRE I : GENERALITES SUR LA TRANSMISSION DES DONNEES NUMERIQUES

I-1- INTRODUCTION :

L'évolution technologique en matière de transmission peut être appréciée à travers la transmission analogique et plus récemment, la transmission numérique.

Les transmissions numériques supplantent de jour en jour les transmissions analogiques. Cette évolution s'explique principalement par les grandes possibilités offertes en traitement du signal numérisé.

La technique numérique offre la possibilité de régénérer le signal ce qui représente l'avantage le plus important d'un système de communication numérique par rapport au système de communication analogique.

Un intérêt majeur des transmissions numériques réside dans la possibilité de leur insertion harmonieuse dans les réseaux intégrés numériques qui se développent de jour en jour.

I-2- REPRESENTATION GLOBALE D'UN SYSTEME DE COMMUNICATION NUMERIQUE : [1], [4], [5]

Le schéma de base d'une chaîne de transmission numérique est représenté par la figure I-1)

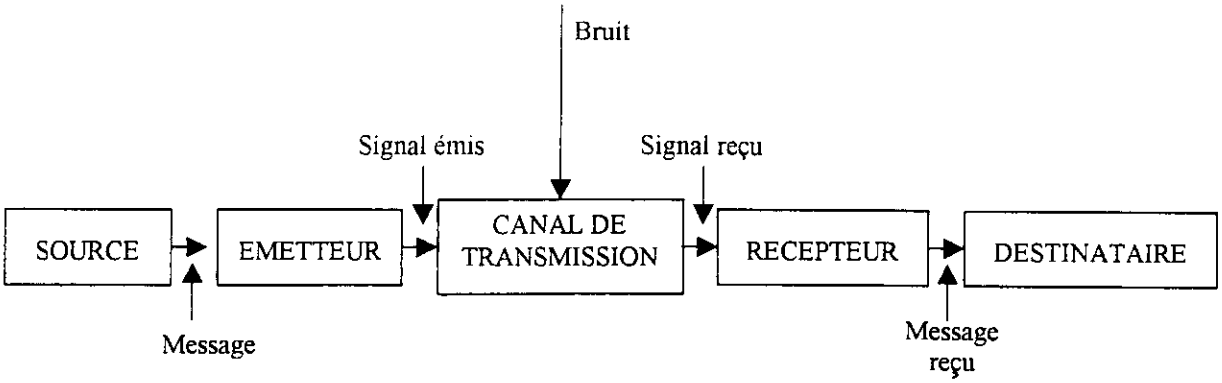


Figure I- 1) Schéma de base d'un système de transmission numérique.

Les cinq éléments qui apparaissent sont définis comme suit :

- Source : produit le message numérique à émettre.
- Emetteur : fournit le message qui doit transiter par le canal de transmission.
- Canal de transmission : constitue le lien entre l'émetteur et le récepteur.
- Récepteur : effectue l'opération inverse de l'émetteur et délivre le message au destinataire.

I-2-1- CANAL DE TRANSMISSION :

Le canal de transmission est en général modélisé par un filtre linéaire suivi d'une addition de bruit. En notant h(t) la réponse impulsionnelle du filtre linéaire, la sortie y(t) du canal est reliée à son entrée x(t) par la relation :

$$y(t) = h(t) * x(t) + w(t)$$

$$y(t) = \int_{-\infty}^{+\infty} h(\tau).x(t-\tau).d\tau + w(t)$$

Où " * " désigne le produit de convolution et w(t) le bruit additif.

- Dans le domaine fréquentiel, le filtrage du canal revient à multiplier X(f) qui représente la transformée de FOURRIER de x(t) par la fonction de transfert H(f) obtenue par la transformée de FOURRIER de h(t).

- Nous supposerons que la fonction de transfert H(f) est constante dans la bande du signal donc la seule perturbation restante est alors le bruit additif w(t).

- Donc le modèle le plus simple et le plus classique utilisé pour représenter le canal est celui du canal à bruit blanc additif Gaussien (canal Gaussien). En sortie de ce canal, le signal reçu résulte de l'addition du signal émis et du bruit Figure I-2)

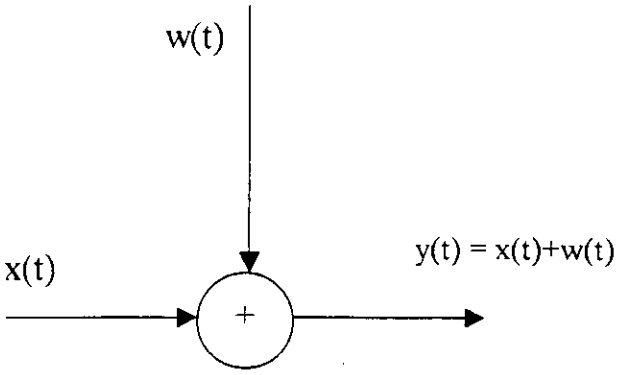


Figure I- 2) Canal à bruit blanc Gaussien.

où

x(t) est le signal émis ;

y(t) est le signal reçu ;

w(t) est un bruit blanc Gaussien de densité spectral $N_0/2$.

I-2-2- L'EMETTEUR :

Le signal émis doit être adapté aux contraintes imposées par le canal de transmission, contraintes au premier rang desquelles la nécessité de n'occuper que la bande de fréquence permise. Donc l'émetteur à une double fonction :

- Une fonction modulation, c'est-à-dire fabrication d'un signal porteur du message à transmettre dont l'énergie est centrée dans la bande de fréquence du canal.

- Une fonction filtrage, c'est-à-dire mise en forme pour répondre aux deux besoins suivants : meilleure performance possible et moins d'énergie possible émise en dehors de la bande de fréquence allouée à la transmission.

La Figure I-3). montre le schéma de principe de l'émetteur.

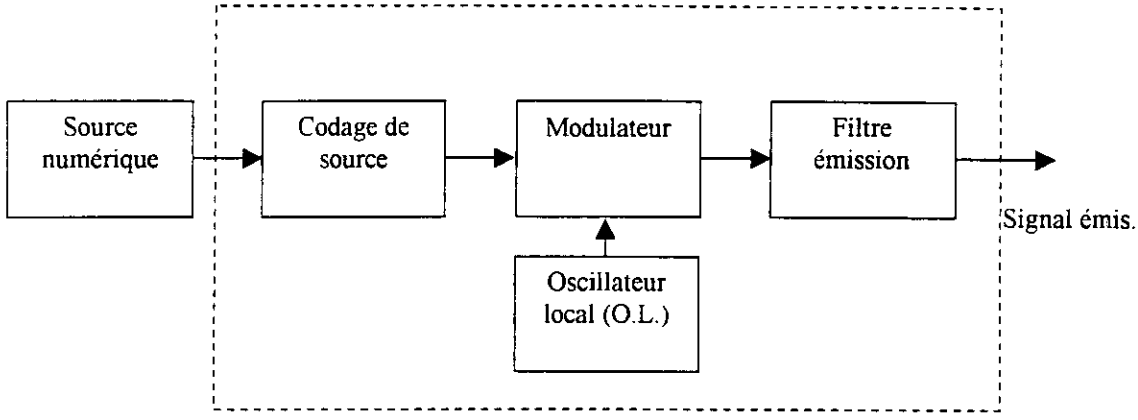


Figure I- 3) Schéma bloc d'un émetteur.

I-2-3- LE RECEPTEUR :

La Figure I-4) représente le schéma de principe d'un récepteur. La structure représentée est la plus simple et a pour but de bien mettre en évidence les diverses opérations effectuées par le récepteur.

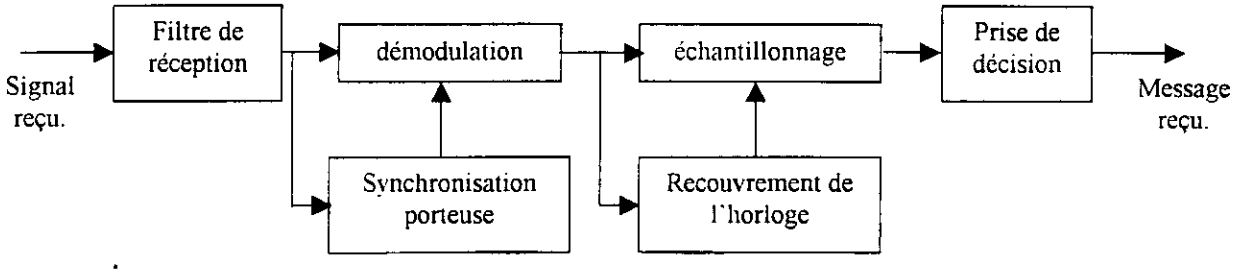


Figure I- 4) Schéma du récepteur d'une transmission sur fréquence porteuse.

I-3- ETUDE DES DIFFERENTES MODULATIONS DIGITALES : [1], [4], [5]

I-3-1- REPRESENTATION DES SIGNAUX A BANDE TRANSPOSEE (SIGNAUX MODULES) :

La fonction de modulation a pour objectif d'adapter le signal à émettre au canal de transmission. Cette opération est définie comme étant le processus de faire varier un des paramètres de la porteuse (fréquence, phase ou amplitude), proportionnellement au signal informatif (signal à bande de base).

Dans le cas d'une communication numérique, le message est donné par une séquence d'impulsions binaires représentant l'information digitale.

Il est bien connu que la porteuse est donnée par une fonction sinusoïdale et les paramètres d'une telle fonction sont : l'amplitude, la fréquence et la phase.

Le résultat de la variation de ces paramètres proportionnellement au signal de base est donné par les modulations ASK (*Amplitude Shit Keying*), FSK (*Frequency Shit Keying*) et PSK (*Phase Shit Keying*).

Toutefois il existe des variantes de ces modulations numériques et même des combinaisons de celles-ci.

A partir de la Figure I-5) on peut noter la différence existante entre les différentes modulations numériques ASK, PSK et FSK.

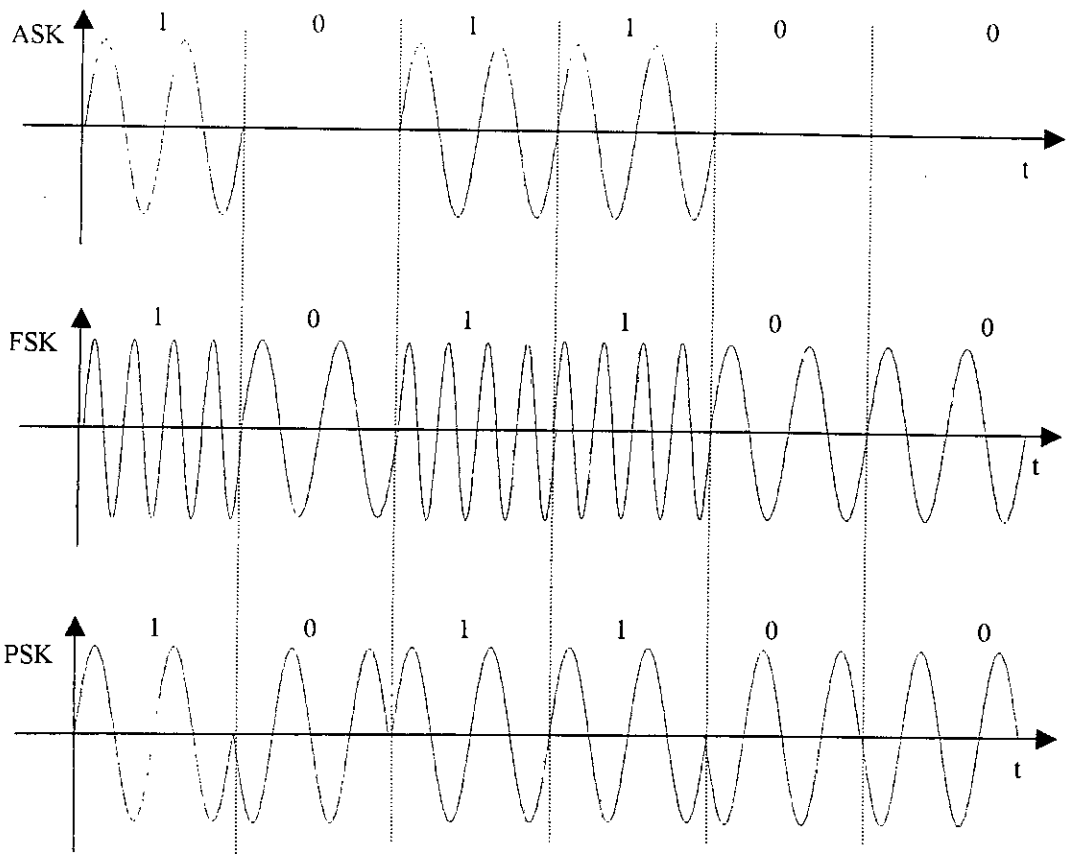


Figure I- 5) Représentation graphique de la séquence 101100 par les modulations ASK, FSK et PSK.

I-3-2- MODULATION ASK :

Le signal en bande de base module l'amplitude d'une porteuse que nous notons cos ($\omega_0 t$) sans aucune perte de généralité.

$$E(t) = \sum_k E_k \; q(t-kT_s) \; \cos(\omega_0 t)$$

Où q(t) désigne la réponse impulsionnelle du filtrage de mise en forme placé avant le modulateur et les symboles E_k sont m-aires et prennent leurs valeurs dans l'alphabet {±d, ±3d, ...,±(2m-1)d}.

La Figure I-6) montre la représentation géométrique des signaux ASK.

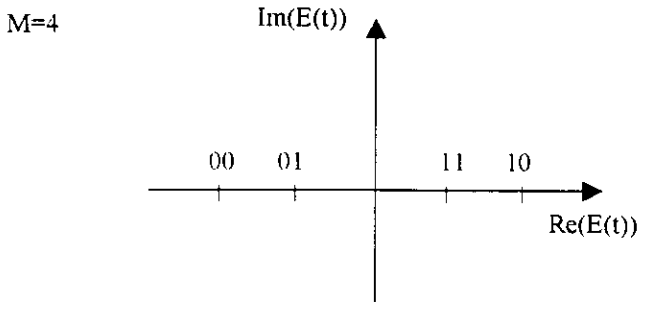


Figure I- 6) Exemple de codage de GRAY pour la modulation ASK.

I-3-2-1- SPECTRE D'AMPLITUDE :

Dans le cas de M=2, le signal ASK admet deux niveaux d'amplitude E_1 et E_2. nous pouvons définir donc :

Etat " 1 " : $E_m(t) = E_1 \cos(\omega_0 t)$

Etat " 0 " : $E_m(t) = E_2 \cos(\omega_0 t)$

Il est commode de créer le concept de la porteuse virtuelle. Cette dernière a pour expression :

$E_v(t) = E_0 \cos(\omega_0 t)$

Où $E_0 = (E_1 + E_2) / 2$

On peut définir l'indice de modulation pour un signal ASK par :

$m_a = \Delta E / E_0 = (E_1 - E_2) / (E_1 + E_2)$

Pour l'analyse du spectre de l'onde modulée, nous devons substituer le signal modulant digital par le signal de test correspondant qui est une onde carrée de période T, donc chaque état dure T/2. La pulsation de ce signal aura la valeur :

$\omega = 2 \pi / T = 2 \pi F$

Nous devons avoir $\omega_0 >> \omega$ pour que l'enveloppe reste bien définie. Les hypothèses faites permettent d'exprimer le signal ASK sous la forme :

$E_m(t) = E_0 [1+m_a\, q(t)] \cos(\omega_0 t)$

q(t) : onde carrée de modulation, variant entre ±1, de période T et qui peut s'exprimer par :

$$q(t) = \sum_{n=1}^{n=+\infty} a_n \cos(n\omega t)$$

avec

$$a_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} q(t).\cos(n\omega t).dt$$

$$= \frac{4}{n\pi}.\sin(\frac{n\pi}{2})$$

Le signal modulé s'exprime par :

$$E_m(t) = E_0 \cos(\omega_0 t) + 2\sum_{n=1}^{n=+\infty} E_0 (m_a/n\pi)\, \sin(n\pi/2)\, [\cos(\omega_0 + n\omega)t + \sin(\omega_0 - n\omega)t]$$

Finalement nous aurons dans le spectre : la porteuse plus deux bandes latérales Figure I-7)

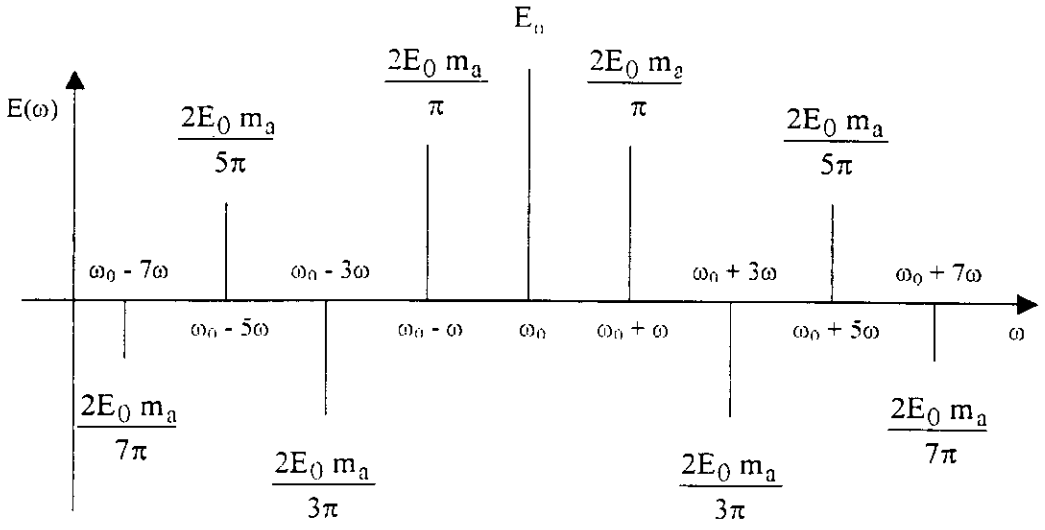


Figure I- 7) Spectre d'amplitude d'un signal ASK.

I-3-2-2- LARGEUR DE BANDE :

La largeur de bande nécessaire pour la transmission peut être déterminée en extrapolant le même critère que pour une impulsion.

$B_{min} = (\omega_0 + \omega) - (\omega_0 - \omega) = 2\omega.$

I-3-3- MODULATION FSK :

Dans le cas de la modulation FSK la fréquence de la porteuse varie au rythme de l'information. Le signal en bande de base est de la forme :

$$S(t) = \sum_{n=0}^{n=+\infty} e^{j\pi\Delta f t\, a_k}\; q(t-kT_0)$$

Le signal FSK est :

$$E_m(t) = \mathrm{Re}\left[\sum_{k=0}^{+\infty} e^{j\pi\Delta f t\, a_k}\; q(t-kT_0)\; e^{j\omega_0 t}\right]$$

$$E_m(t) = \mathrm{Re}\left[\sum_{k=0}^{+\infty} q(t-kT_0)\, e^{j(\Delta\omega a_k + \omega_0)t}\right]$$

Où ω_0 est la fréquence porteuse, $\Delta\omega = 2\pi\ \Delta f$ avec Δf l'intervalle de déplacement entre les fréquences extrêmes.

Dans le cas d'une transmission binaire : $a_k = \pm 1$.

$$E_m(t) = \sum_{k=0}^{+\infty} q(t-kT_0)\ \cos(\Delta\omega a_k + \omega_0)t$$

Lorsque $a_k = 0$, la fréquence est notée par f_1 et vaudra $f_1 = f_0 - \Delta f/2$.

Lorsque $a_k = 1$, la fréquence est notée par f_2 et vaudra $f_2 = f_0 + \Delta f/2$.

Avec $\Delta f = f_2 - f_1$.

Donc la gamme de variation de la fréquence est [$f_0 - \Delta f/2$, $f_0 + \Delta f/2$], où Δf est donnée par la relation $\Delta f = f_2 - f_1$. On choisit f_1 et f_2 de sorte que

$f_1 = n_c - 1/T_0$, $f_2 = n_c + 1/T_0$.

Où n_c est un entier fixe, et $T_0 = 1/f_0$.

Le choix de f_1 et f_2 donné ci dessus permet d'avoir une orthogonalisation entre les composants associés à 0 et à 1.

I-3-3-1- SPECTRE D'AMPLITUDE :

On peut écrire le signal FSK binaire comme suit :

- Pour le " 1 " logique $E_m(t) = E_0 \cos(\omega_1 t)$

- Pour le " 0 " logique $E_m(t) = E_0 \cos(\omega_2 t)$

En supposant $\omega_1 > \omega_2$ nous pouvons définir la porteuse virtuelle par :

$E_v(t) = E_0 \cos(\omega_0 t)$

Où $\omega_0 = (\omega_1+\omega_2)/2$: fréquence angulaire.

$\omega_d = (\omega_1-\omega_2)/2$: déviation de fréquence angulaire.

On peut écrire donc :

- Etat " 1 " : $E_1(t)=E_0 \cos(\omega_0+\omega_d)t$

- Etat " 0 " : $E_2(t)=E_0 \cos(\omega_0-\omega_d)t$

Nous pouvons considérer le signal FSK comme la combinaison de deux signaux E_1 et E_2. La Figure I-8) Représente le signal FSK avec les deux signaux E_1 et E_2 et le signal binaire.

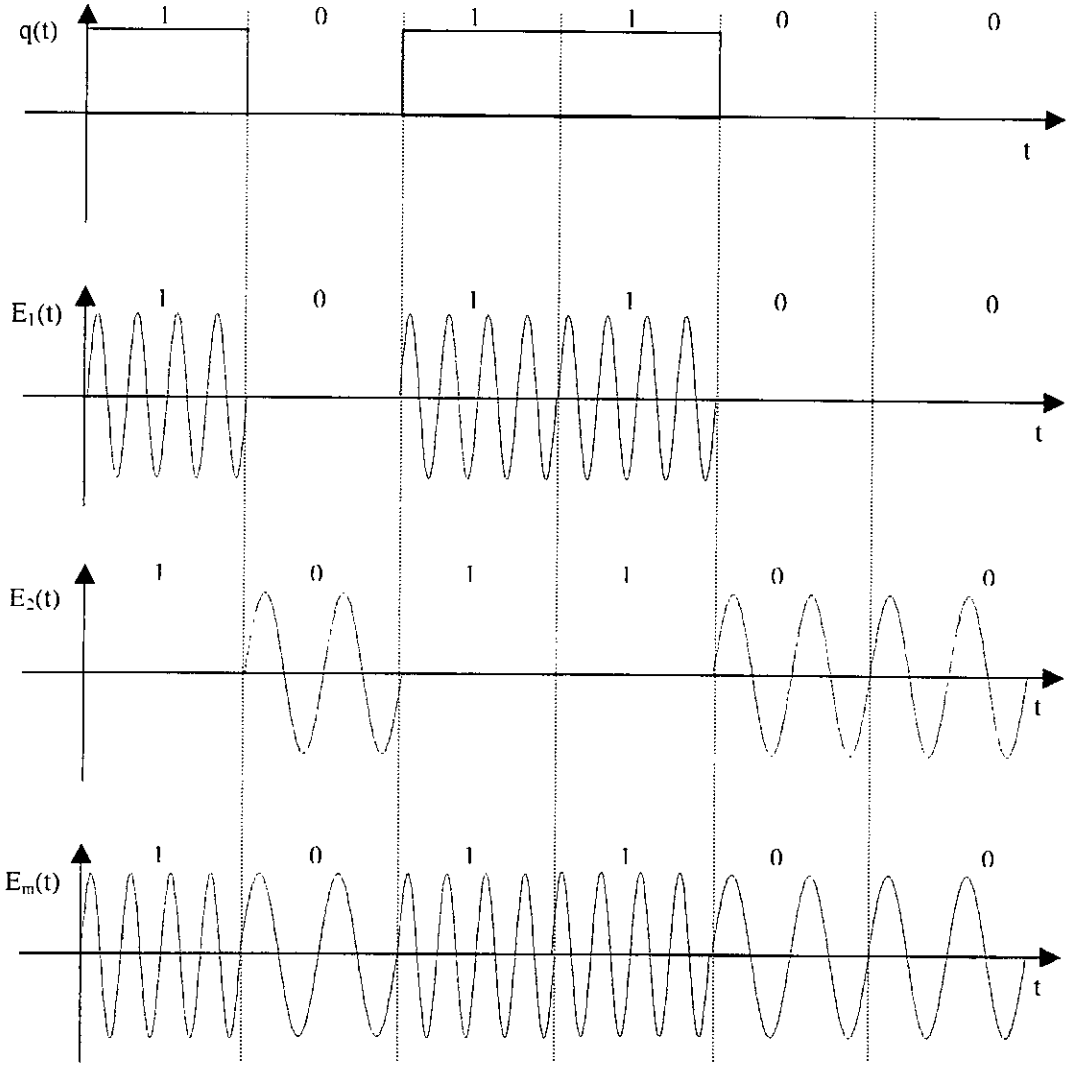


Figure I- 8) Composition d'un signal FSK en deux signaux E_1 et E_2.

Le signal FSK dans ce cas est la superposition linéaire de E_1 et E_2.

Pour le signal E_1 on a :

- Etat " 1 " :$E_{m1}(t)=E_0 \cos(\omega_1)t$.

- Etat " 0 " :$E_{m1}(t)=0$.

Auquel correspond le spectre :

$$E_{m1}(t) = E_0 / 2 \sum_{n=-\infty}^{n=+\infty} \sin(n\pi / 2) / (n\pi / 2) * \exp(j(\omega_1 + n\omega) t)$$

Pour le signal E_2 on a :

- Etat " 0 " :$E_{m2}(t)=E_0 \cos(\omega_2)t$.

- Etat " 1 " :$E_{m2}(t)=0$.

Auquel correspond le spectre :

$$E_{m2}(t) = E_0 / 2 \sum_{n=-\infty}^{n=+\infty} \exp(-jn\omega T/2) * \sin(n\pi / 2) / (n\pi / 2) * \exp(j(\omega_2 + n\omega) t)$$

où le terme de phase correspond au retard T/2 dans le domaine de temps.

Le signal résultant s'écrira :

$$E_m(t) = E_0 / 2 \sum_{n=-\infty}^{n=+\infty} \sin(n\pi / 2) / (n\pi / 2) * \exp(j(\omega_1 + n\omega) t) + E_0 / 2 \sum_{n=-\infty}^{n=+\infty} \exp(-jn\omega T/2) * \sin(n\pi / 2) / (n\pi / 2) * \exp(j(\omega_2 + n\omega) t)$$

$$E_m(t) = E_0 / 2 \sum_{n=-\infty}^{n=+\infty} \sin(n\pi / 2) / (n\pi / 2) * \exp(j(\omega_1 + n\omega) t) + E_0 / 2 \sum_{n=-\infty}^{n=+\infty} (-1)^n \sin(n\pi / 2) / (n\pi / 2) * \exp(j(\omega_2 + n\omega) t)$$

Nous avons deux spectres (Figure I-9.), l'un autour de ω_1 et l'autre autour de ω_2.

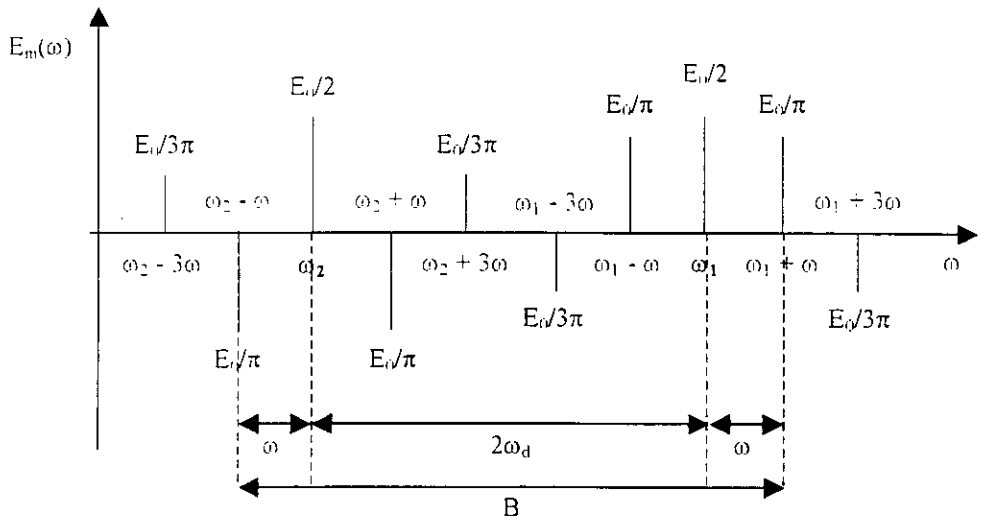


Figure I- 9) Spectre d'amplitude d'un signal FSK.

I-3-3-2- LARGEUR DE BANDE :

La largeur de bande d'un signal FSK est donnée par :

$$B = 2\omega_d + 2\omega = 2(\omega_d + \omega)$$

I-3-4- MODULATION PSK :

Dans cette technique, c'est la phase de la porteuse qui est modulée par le signal en bande de base. Dans une PSK M-aire, le signal modulé est de la forme :

$$E(t) = \mathrm{Re}\left\{ \sum_k a_k \, q(t - kT_s) \exp(j\omega_0 t) \right\}$$

Où q(t) a la même signification que précédemment et les symboles a_k sont de la forme :

$$a_k = A \exp[j(\varphi_k + \phi)]$$

Avec $\varphi_k \in \{0, 2\pi/M, 4\pi/M, \ldots, 2(M-1)\pi/M\}$ et ϕ phase arbitraire.

Il est d'usage de prendre $\phi = (M-1)\pi/M$ de façon à centrer les valeurs φ_k par rapport à la phase 0. On obtient ainsi :

$\varphi'_k = \varphi_k - \phi \in \{\pm\pi/M, \pm 2\pi/M, \ldots, =(M-1)\pi/M\}$

Quant au paramètre A dans l'expression des symboles a_k, il détermine la puissance du signal émis.

Une représentation géométrique des signaux PSK est donnée par la Figure I-10)

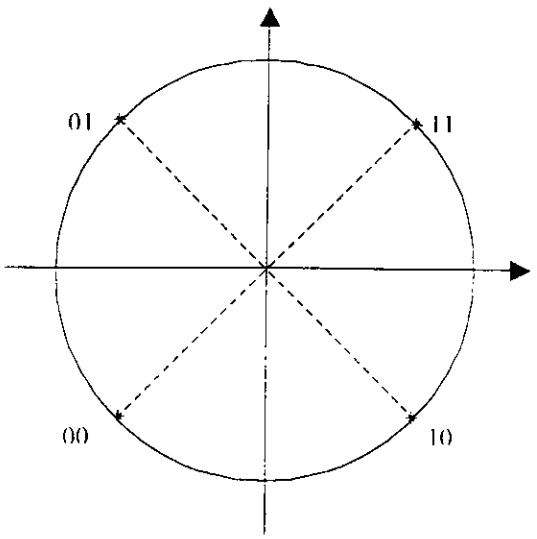


Figure I- 10) Représentation géométrique du signal PSK pour M=4.

Les points de la constellation sont ainsi régulièrement espacés sur un cercle de rayon A. comme le montre la Figure I-10), l'affectation des bits aux points de la constellation se fait en générale selon le code **Gray**.

I-3-4-1- LARGEUR DE LA BANDE :

Dans le cas où M=2 le signal PSK admettrait deux phases φ_1 et φ_2 pour les deux états informatifs donnés. Nous pouvons déterminer le spectre de cette onde PSK en utilisant le même procédé que celui de la FSK, c'est-à-dire la superposition linéaire de deux signaux. Dans ce type de modulation on trouve une bande de fréquence plus petite que celle de la modulation FSK, à savoir :

$B=2\omega$.

I-4- ANALYSE DES (S/B) EN MODULATION DIGITALE :

I-4-1- SYSTEME FSK : [1]

Il existe deux techniques pour démoduler les signaux FSK, à savoir la démodulation cohérente et la démodulation non cohérente.

- Si la porteuse est connue au niveau du récepteur on a une démodulation cohérente.

- Si la porteuse n'est pas connue, alors c'est la démodulation non cohérente.

I-4-1-1- MODULATION FSK COHERENTE :

Pour M=2, les symboles 1 et 0 sont représentés par les signaux :

$$E_i(t) = \begin{cases} \sqrt{2E_0 / T_0}\cos(\omega_i t), 0 \le t \le T_0. \\ 0, \text{ailleurs.} \end{cases}$$

On choisit $f_i=(n_c+i)/T_0$, $i=1, 2$, $n_c \in N$.

Où f_1 correspond à l'état 1 et f_2 correspond à l'état 0.

On prend comme base de représentation :

$$\varphi_i = \begin{cases} \sqrt{2 / T_0}\cos(\omega_i t), 0 \le t \le T_0, i = 1,2 \\ 0, \text{ailleurs.} \end{cases}.$$

Ce choix de la base nous permet d'avoir une base orthogonale.

On peut noter aussi que $E_1(t)$ et $E_2(t)$ sont orthogonaux.

$$E_1(t) = S_{11}\varphi_1(t) + S_{12}\varphi_2(t), S_{12} = 0.$$
$$E_2(t) = S_{21}\varphi_1(t) + S_{22}\varphi_2(t), S_{21} = 0.$$

$$\text{Avec } S_{ij} = \int_0^{T_0} S_i(t)\,\varphi_i\,dt = \begin{cases} \sqrt{E_0}, \text{si } i = j \\ 0, \text{si } i \ne j. \end{cases}$$

Dans ce cas nous avons un espace de dimension n=2 avec deux points message.

$$S_1 = (\sqrt{E_0}, 0), S_2 = (0, \sqrt{E_0})$$

Ces deux points sont montrés sur la Figure I-11) ci-dessous.

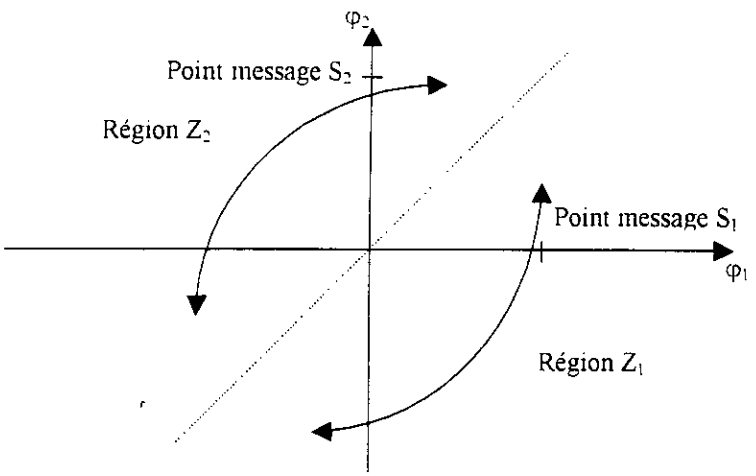


Figure I- 11) Représentation géométrique du signal FSK

Le vecteur d'observation dans ce cas possède deux composantes :

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$$

Où la réalisation de x_1 est donnée par :

$$x_1 = \int_0^{T_0} x(t)\,\varphi_1(t)\,dt.$$

Et la réalisation de x_2 est donnée par :

$$x_2 = \int_0^{T_0} x(t)\,\varphi_2(t)\,dt.$$

Où x(t) est le signal reçu donné par $x(t)=E_i(t)+ W(t)$, i=1, 2. Où E_i est le signal émis et W(t) est un bruit blanc gaussien.

On note que lorsque $x_1>x_2$, on décide en faveur de $E_1(t)$ et lorsque $x_1<x_2$, on décide en faveur de $E_2(t)$. Pour cela, on définit une variable aléatoire gaussienne L dont la réalisation est : $l=x_1-x_2$.

avec $E(L/1)=E(X_1/1)- E(X_2/1)= +\sqrt{E_0}$

$E(L/0)=E(X_1/0)- E(X_2/0)= - \sqrt{E_0}$

et $Var(L)=Var(X_1)+ Var(X_2)=N_0$ (X_1 et X_2 sont statistiquement indépendantes).

Supposant que le symbole 0 est transmis, on aura donc la fonction de densité de probabilité conditionnelle de la variable aléatoire L suivante :

$$F_L(l/0)=\frac{1}{\sqrt{2\pi N_0}} \exp\left[-(l+\sqrt{E_0})^2 / 2N_0\right]$$

La probabilité d'erreur du symbole " 0 " est :

$$\begin{aligned} P_e(0) &= P(l > 0 / 0 \text{ est émis}) \\ &= \int_0^{\infty} f_l(l/0)\, dl \\ &= 1/2 \operatorname{erfc}(\sqrt{E_0 / 2N_0}) \end{aligned}$$

et puisque la répartition est symétrique, alors :

$P_e(1)= P_e(0)$

Donc la probabilité moyenne d'erreur de la modulation FSK cohérente est :

$$\begin{aligned} P_e &= 1/2\,(P_e(0) + P_e(1)) \\ &= 1/2 \operatorname{erfc}(\sqrt{E_0 / 2N_0}) \end{aligned}$$

Les schémas de principe d'un modulateur cohérent FSK et d'un démodulateur cohérent FSK sont données par Figure I-12) et Figure. I-13)

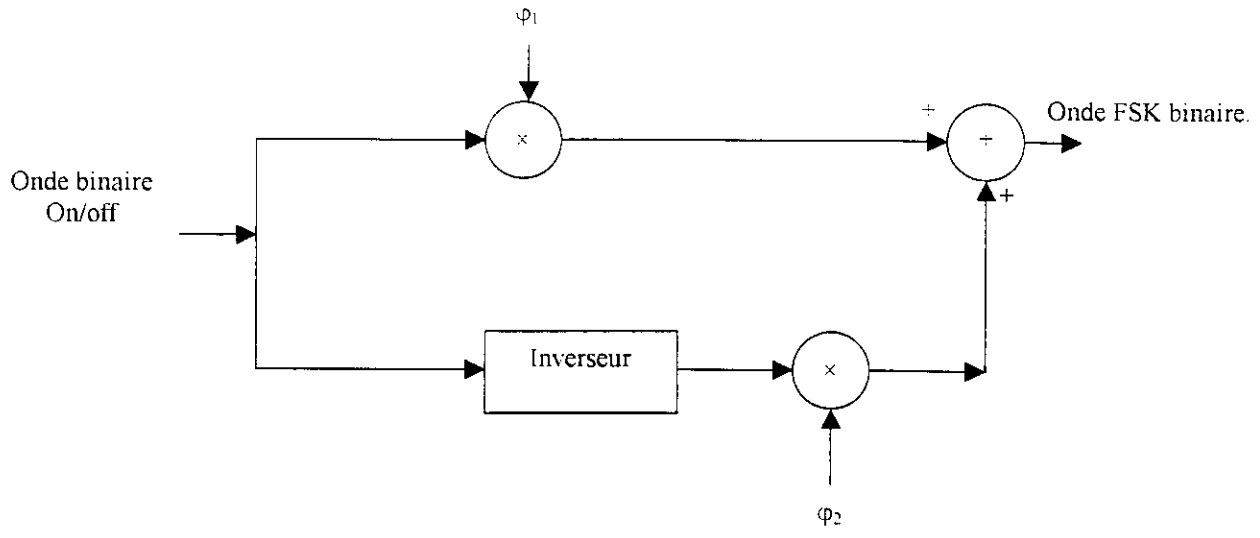


Figure I- 12) Modulateur FSK cohérent.

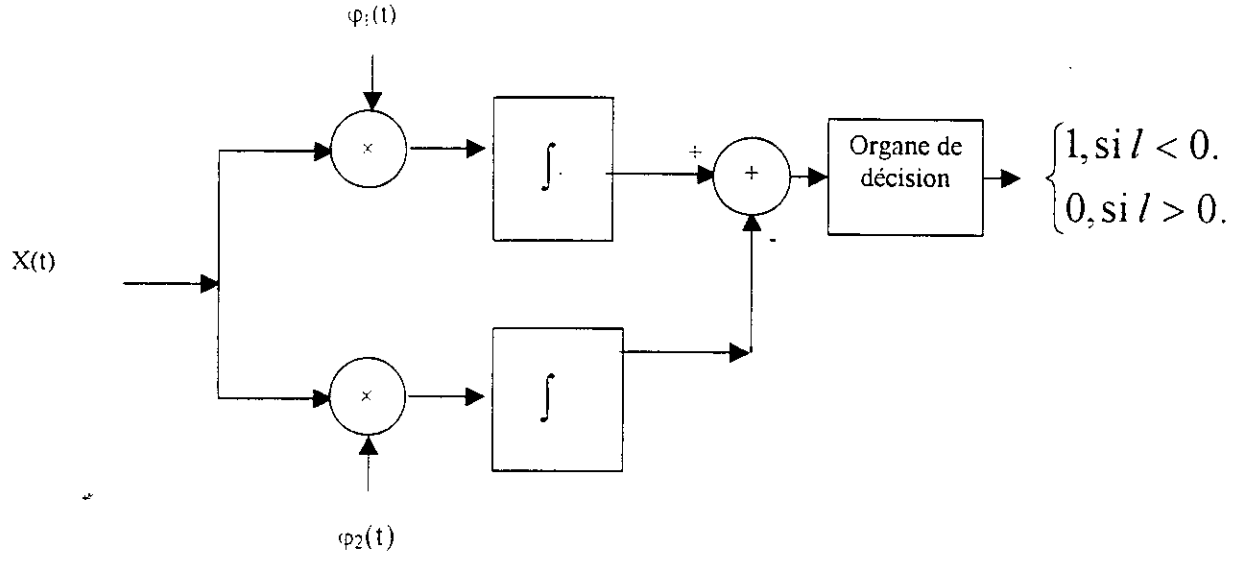


Figure I- 13) Démodulateur FSK cohérent.

Pour générer un signal modulé en FSK binaire, on peut utiliser le schéma de la Figure I-12) La séquence binaire m(t) contient les symboles 1 (représentés par une amplitude constante de $\sqrt{E_b}$ volts) et les symboles 0 (représentés par une amplitude constante de 0 volt).

Si m(t) = 1 le signal $\varphi_1(t)$ passe tandis que le signal $\varphi_2(t)$ est bloqué.

Si m(t) = 0 le signal φ_1 est bloqué tandis que le signal φ_2 passe.

A la sortie de l'additionneur on aura donc un signal modulé en FSK binaire.

Pour effectuer la démodulation d'un signal x(t) modulé en FSK binaire avec addition de bruit, on peut utiliser le schéma de la Figure I-13) Le signal x(t) est corrélé avec les deux signaux $\varphi_1(t)$ et $\varphi_2(t)$ pour fournir les signaux x_1 et x_2 respectivement. Le soustracteur fourni le signal $l=x_1 - x_2$ qui attaque l'organe de décision pour :

- décider en faveur de symbole 0, si $l < 0$,
- décider en faveur de symbole 1, si $l > 0$,

I-4-1-2- MODULATION FSK NON-COHERENTE :

Dans le cas de la modulation FSK, le signal modulé est défini en fonction de :

$$E_i = \begin{cases} \sqrt{2E_0 / T_0}\cos(\omega_i t), & 0 \le t \le T_0 \ , i = 1,2. \\ 0 \ , \text{ailleurs.} \end{cases}$$

Le signal à la réception est

$$x(t)= E_i(t) + w(t),\ i = 1,2,\ 0 \le t \le T_0.$$

où w(t) représente un bruit blanc Gaussien additif de moyenne nulle et de densité spectrale égale à $N_0/2$. Le signal $E_1(t)$ représente le symbole 1 et le signal E_2 représente le symbole 0.

La démodulation non cohérente de x(t) s'effectue à l'aide du schéma suivant :

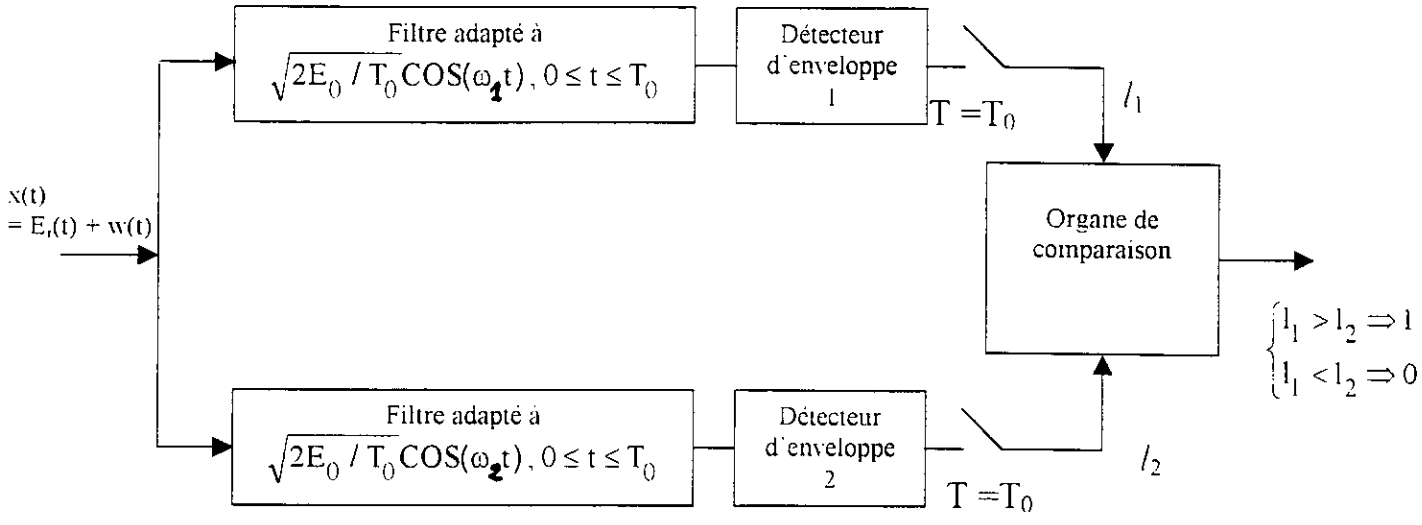


Figure I- 14) Démodulateur FSK non-cohérent.

Dans la démodulation FSK non cohérente (Figure I-14) le signal x(t) attaque, simultanément, deux filtres adaptés l'un à $\sqrt{\frac{2}{T_0}}\cos(2\pi f_1 t)$ et l'autre à $\sqrt{\frac{2}{T_0}}\cos(2\pi f_2 t)$, $0 \le t \le T_0$. La sortie de chaque filtre adapté attaque un détecteur d'enveloppe. Le signal issu du premier détecteur d'enveloppe est échantillonné à $t=T_0$ et fournit des échantillons l_1 et le signal issu du deuxième détecteur est échantillonné lui aussi à $t=T_0$ fournissant ainsi des échantillons l_2. A la sortie du comparateur on aura :

- une décision en faveur du symbole " 1 " lorsque $l_1 > l_2$.

- une décision en faveur du symbole " 0 " lorsque $l_1 < l_2$.

La démodulation FSK binaire non cohérente est un cas particulier de la démodulation orthogonale et après calcul on trouve :

$P_e = \frac{1}{2} \exp(-E_0 / 2N_0)$

Remarque :

Puisque dans le cas de la FSK cohérente la probabilité d'erreur est de la forme $\frac{1}{2}\,\mathrm{erfc}(\sqrt{E_0 / 2N_0})$, alors on peut dire que pour $E_0/N_0 >> 1$, les deux types de démodulation FSK (cohérente et non cohérente) sont similaires en performance.

I-4-2- SYSTEME PSK :

I-4-2-1- MODULATION PSK COHERENTE :

Dans la modulation PSK binaire les signaux modulés sont représentés à base des deux signaux suivants :

$E_1(t) = \sqrt{2E_0 / T_0}\,\mathrm{COS}(\omega_c t)$ représente le symbole 1.

$E_2(t) = \sqrt{2E_0 / T_0}\,\mathrm{COS}(\omega_c t + \pi)$ représente le symbole 0.

Où E_0 représente l'énergie de transmission par bit, et la fréquence $f_c = \omega_0/2\pi$ est choisie de la forme $f_c = n_c / T_0$, $n_c \in N$. (et ce pour assurer que chaque bit transmis possède une durée égale à un multiple entier de la période de la porteuse)

On prend dans ce cas

$\varphi_1(t) = \sqrt{2 / T_0}\,\mathrm{COS}(\omega_c t)$ et n=1 (dimension de l'espace), $0 \le t \le T_0$.

Donc

$E_1(t) = \sqrt{E_0}\ \varphi_1(t)$, $0 \le t \le T_0$.

$E_2(t) = -\sqrt{E_0}\ \varphi_1(t)$, $0 \le t \le T_0$.

On trouve $S_{11} = \langle E_1(t), \varphi_1(t) \rangle = E_1(t) = \sqrt{E_0}$, $0 \le t \le T_0$.

et $S_{12} = \langle E_2(t), \varphi_1(t) \rangle = E_1(t) = -\sqrt{E_0}$, $0 \le t \le T_0$.

Les points messages sont représentés sur la figure suivante (Figure. I-15).

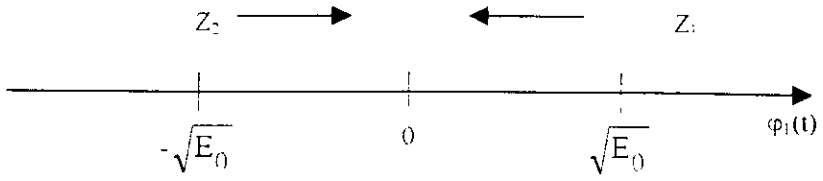


Figure I- 15) Représentation géométrique du signal PSK.

Soit x le point d'observation, lorsque $x \in Z_1$, on décide en faveur du symbole m_1 représenté par $E_1(t)$ et lorsque $x \in Z_2$, on décide en faveur du symbole m_2 représenté par $E_2(t)$.

Autrement dit, on peut écrire la règle de décision de manière plus simple :

Si $x>0 \Rightarrow \hat{m} = 1$

Si $x<0 \Rightarrow \hat{m} = 0$

Remarque :

Dans la décision précédente on a supposé que m_1 et m_2 sont équiprobables.

Dans ce cas on peut distinguer deux types d'erreur :

1) Lorsque 1 est émis et $x<0$. Donc on décide en faveur de 0.

2) Lorsque 0 est émis et $x>0$. Donc on décide en faveur de 1.

$$\text{Alors : } f_{x1}(x_1/0) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{1}{N_0}(x_1 - S_{21})^2\right)$$

$$= \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{1}{N_0}(x_1 - \sqrt{E_0})^2\right)$$

La probabilité conditionnelle du récepteur décidant en faveur de 1 alors que 0 est émis est :

$$P_e(0) = \int_0^{\infty} f_{x1}(x_1/0)\, dx_1 = \frac{1}{\sqrt{\pi N_0}} \int_0^{\infty} \exp[-\frac{1}{N_0}(x_1 + \sqrt{E_0})^2]\, dx_1$$

Posons : $z = \frac{1}{N_0}(x_1 + \sqrt{E_0})$ on aura :

$$P_e(0) = \frac{1}{\sqrt{\pi}} \int_{\sqrt{E_0/N_0}}^{\infty} \exp(-z^2)\, dz = \frac{1}{2}\operatorname{erfc}(\frac{\sqrt{E_0}}{N_0})$$

Où $\operatorname{erfc}(y) = (2/\sqrt{\pi}).\int_y^{\infty} \exp(-z^2)\, dz$ est la fonction d'erreur complémentaire.

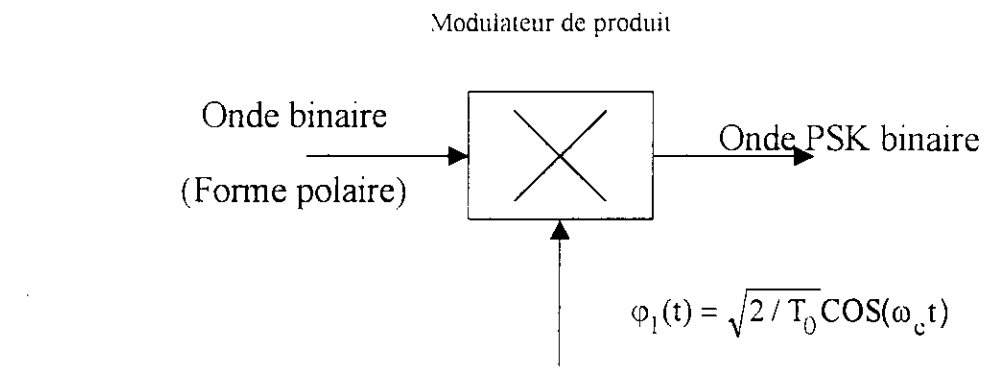
De la même manière on peut tirer $P_e(1)$ représentant la probabilité de décider en faveur de 0 alors que 1 est émis.

$$P_e(1) = 1/2\ \operatorname{erfc}(\sqrt{E_0/N_0})$$

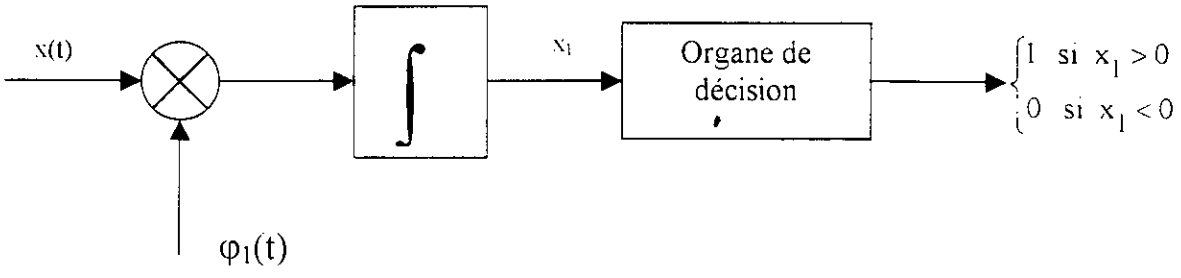
Donc la probabilité moyenne de l'erreur est :

$$P_e = 1/2\ \operatorname{erfc}(\sqrt{E_0/N_0})$$

Les schémas de base d'un modulateur et d'un démodulateur PSK sont donnés par la figure suivante :



a) Modulateur PSK.



b) Démodulation cohérente de la PSK binaire.

Figure I- 16) Schéma de base d'un modulateur et d'un démodulateur PSK cohérent.

La Figure I-16. a) représente un modulateur PSK, on note que le signal binaire en représentation polaire est constitué d'une séquence de symboles 1 et 0 représentés par des amplitudes constantes $+\sqrt{E_0}$ et $-\sqrt{E_0}$ respectivement. Cette onde binaire et une onde sinusoïdale $\varphi_1(t)$ (de fréquence $f_c = n_c/T_0$, $n_c \in N$) sont appliqués à un modulateur de produit (*product modulator*), pour fournir à sa sortie un signal modulé en PSK.

La démodulation PSK peut être mise en œuvre en utilisant le schéma de la Figure I- 6. b) où on a un signal x(t) en modulation PSK avec superposition du bruit. Les signaux x(t) et $\varphi_1(t)$ attaquent un corrélateur dont la sortie x_1 est appliquée à l'entrée du comparateur qui :

- décide en faveur de symbole 1, si $x_1 > 0$;

- décide en faveur de symbole 0, si $x_1 < 0$

I-5- COMPARAISON DES DIFFERENTES P_E POUR PSK ET FSK : [4]

La Figure. I-17) montre la probabilité d'erreur par symbole pour différents types de modulation.

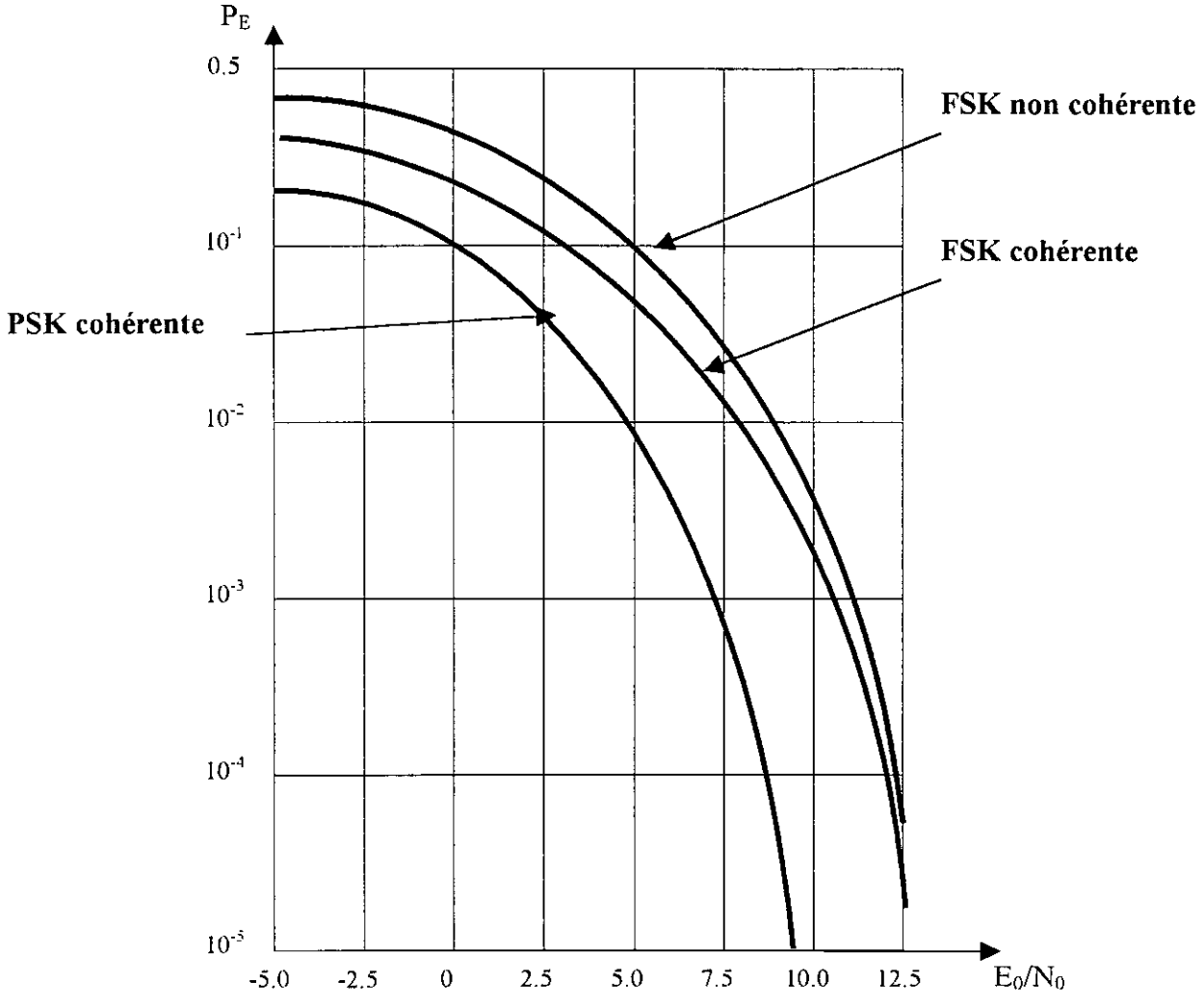


Figure I- 17) Probabilité d'erreur par symbole pour différents types de modulation.

D'après ces graphes, on peut dire que :

- Toutes les probabilités d'erreur décroissent d'une façon monotone avec l'augmentation du rapport signal sur bruit (E_0/N_0).

- Pour n'importe quelle valeur du rapport signal sur bruit (E_0/N_0), la modulation PSK présente la probabilité d'erreur la plus faible.

- La PSK est meilleure que la FSK cohérente et non cohérente.

I-6- CONCLUSION :

La modulation PSK présente la meilleure performance par rapport à la modulation FSK, mais au prix d'une circuiterie plus complexe et donc plus encombrante. La PSK est utilisée dans la transmission de données à grand débit. Dans la pratique le choix de la modulation se fait en considérant l'occupation spectrale, les performances et la complexité du couple modulateur/démodulateur. Il est à noter que la faible occupation spectrale et les performances sont deux contraintes antagonistes, ce qui nécessite un compromis lors du choix d'une modulation.

La modulation ASK est plus facile à mettre en œuvre en pratique, mais elle a un inconvénient qui est la sensibilité au bruit du canal.

La modulation FSK présente des performances meilleures que celles de la modulation ASK quant à la sensibilité au bruit, et plus facile à mettre en œuvre que la modulation PSK.

C'est pour cela que nous avons choisi pour notre application de réaliser un système de communication numérique en utilisant la modulation FSK.

CHAPITRE II :
MISE EN ŒUVRE
DU KIT DE DEVELOPPEMENT MC68HC11
DE MOTOROLA.

CHAPITRE II : MISE EN ŒUVRE DU KIT DE DEVELOPPEMENT MC68HC11 DE MOTOROLA

II-1- INTRODUCTION :

Dans ce chapitre nous allons étudier le fonctionnement du microcontrôleur 68HC11 que nous avons utilisé dans notre application.

On va s'intéresser à l'étude de l'architecture générale du microcontrôleur, ensuite on va passer à l'étude de quelques registres interne qui permettent la configuration de base du microcontrôleur, ensuite on va projeter un peut de lumière sur le fonctionnement de la liaison sérielle.

Dans une étape suivante nous allons étudier la mise en œuvre de deux logiciels qui permettent l'exploitation du microcontrôleur 68HC11 qui sont : PCBUG11 et BUFALLO.

II-2- ETUDE DU MICROCONTROLEUR MC68HC11 : [2], [6], [7], [8], [9]

II-2-1- CARACTERISTIQUES DU MICROCONTROLEUR 68HC11 :

Le processeur employé est un 68HC11A1 dont les caractéristiques essentielles sont résumées dans le schéma suivant :

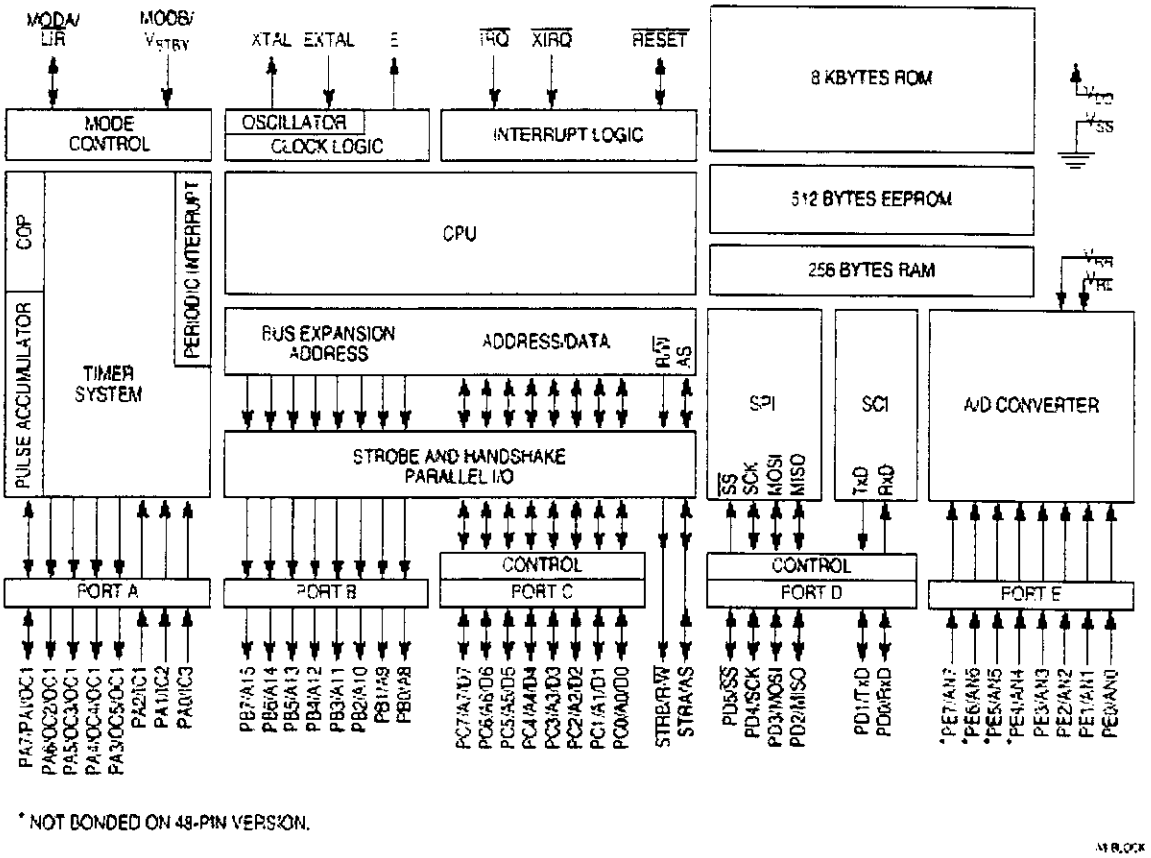


Figure II- 1)Architecture interne du 68HC11.

On peut souligner les éléments suivants :

- un CPU technologie HCMOS
- 8Ko de ROM
- 512 octets de EEPROM
- 256 octets de RAM
- un *timer* 16 bits : 3 entrées de captures, 5 sorties de comparaison
- 2 accumulateurs 8 bits
- une liaison série asynchrone (SCI)
- une liaison série synchrone (SPI)
- un convertisseur analogique - numérique 8 bits, 8 entrées
- circuit d'interruption temps réel

- circuit oscillant externe (quartz 8 MHz dans notre application)

Les modes de fonctionnement du 68HC11 sont résumés dans le tableau suivant :

MODB	MODA	MODE SELECTIONNE
1	0	*Single Chip* (Mode 0)
1	1	*Expanded multiplexed* (Mode 1)
0	0	*Special bootstrap*
0	1	*Special test*

Tableau II- 1) Modes de fonctionnement du MC 68HC11.

II-2-2- FONCTIONS REALISEES PAR LE MC68HC11 :

Un MC68HC11 peut remplir les fonctions suivantes :

- Transfert unidirectionnel ou bidirectionnel de données en parallèle (ports A, B, C et D) ;
- Transfert de données en série synchrone (SPI sur le port D) ;
- Transfert de données en série asynchrone (SCI sur le port D) ;
- Temporisation et comptage (sur le port A) permettant :
  - de générer des intervalles de temps ;
  - de mesurer des intervalles de temps ;
  - de compter des impulsions ;
  - de générer des impulsions en temps réel (horloge temps réel) ;
  - de surveiller l'horloge et le bon fonctionnement du programme (COP ou chien de garde.) ;
- Conversion analogique digitale (le convertisseur dispose de 8 entrées multiplexées) ;

Les lignes d'entrées / sorties sont polyvalentes, elle peuvent être utilisées :

- Comme des entrées sorties parallèles unidirectionnelles ou bidirectionnelles (ports B et C) ;
- Comme des lignes d'adresses et de données (port B et C) quand on utilise le micro-contrôleur en mode étendu. c'est à dire avec des périphériques externes.

II-2-3- DESCRIPTION DES CONNEXIONS :

- V_{DD} et V_{SS} : Ce sont des connexions d'alimentation du circuit. Ce circuit est réalisé en technologie HCMOS, il est alimenté avec une tension de +5 volts ce qui le rend compatible avec le circuit CMOS et TTL. Il est conseillé de bien découpler cette alimentation (commutation rapide).
- **MODA** et **MODB** : Ces deux signaux permettent de choisir (pendant la durée du RESET) le mode de fonctionnement du micro-contrôleur.
- En dehors de cette phase de RESET la ligne **MODA** devient **LIR** (*loader instruction registre* : chargement du registre d'instruction) et peut être employée pour synchroniser un appareil de mesure durant la mise au point d'un programme. **LIR** passe au niveau logique zéro pendant le premier cycle d'horloge **E** de l'exécution d'une instruction.

- **MODB** peut servir à alimenter la RAM interne dés que la tension V_{DD} devient inférieur à -0,7 volts (Figure II-2). La commutation se fait automatiquement grâce à une logique interne.

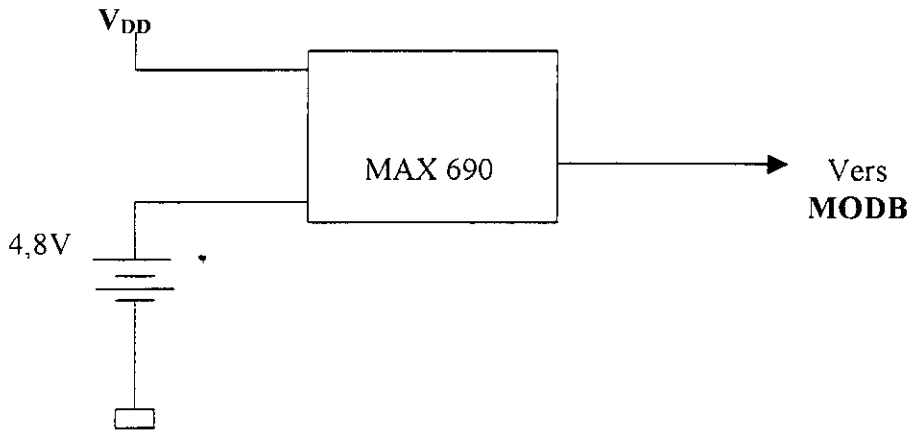


Figure II-2) Alimentation de la RAM interne

- **EXTAL** et **XTAL** : Remettent la connexion de quartz de l'horloge ou d'une horloge externe.

  Généralement on utilise un quartz de 8 Mhz mais on peut faire descendre cette fréquence bien en dessous à condition de faire attention à la limite fixée par le surveillant de l'horloge. La fréquence l'horloge de bus est égale au quart de fréquence de quartz. Ces deux connexions peuvent aussi être utilisées comme entrée d'horloge externe : **EXTAL**, ou sortie d'horloge : **XTAL** afin par exemple de synchroniser deux micro-contrôleur 68HC11 entre eux.

- **Signal E** : sortie de l'horloge bus, cette ligne est surtout utilisée en mode étendu.

- **VREFL** et **VREFH** : ces deux connexions permettent d'appliquer les tensions de référence utiles au convertisseur analogique/numérique. Ces tensions ne doivent pas dépasser les tensions d'alimentation du 68HC11 pour ne pas endommager celui-ci et l'écart entre VREFH et VREFL ne doit pas descendre en dessous de 2,5 V pour garder une bonne précision au convertisseur. Ces entrées doivent être soigneusement découplées.

II-2-4- CONFIGURATION DE LA MEMOIRE ET DES REGISTRES PAR L'UTILISATEUR :

Les emplacements des registres et des zones mémoire peuvent être modifiés par programmation de bits dans certains registres. Plusieurs registres de contrôle sont protégés en écriture sauf en certaines circonstances (modes spéciaux). Cette protection inclut la possibilité d'écrire ces bits une fois et une seule pendant les 64 cycles d'horloge **E** qui suivent un RESET.

II-2-4-1- LE REGISTRE CONFIG $103F :

Ce registre compte en réalité deux registres distincts, l'un en EEPROM qui conserve les informations en permanence et l'autre est un registre de travail normal en RAM. Après un RESET le registre en EEPROM est recopié dans le registre de travail. Une fois le micro-contrôleur en route il devient possible de modifier le registre en EEPROM par une opération particulière (programmation).

Le registre **CONFIG** est protégé par la mise à un du bit **PTCON** du registre **BPROT** sauf pour les modèles A_{XX}. La modification nécessite la même procédure d'effacement et de programmation que l'EEPROM. Ce bit n'existe pas sur tous les modèles.

CONFIG	EE3	EE2	EE1	EE0	NOSEC	NOCOP	1	EEON
single chip	1	1	1	1	P	P	1	1
bootstrap	1	1	1	1	P	P	1	1
expanded	P	P	P	P	1	P	1	P
special test	P	P	P	P	1	P	1	0

Tableau II- 2) Les différents bits du registre CONFIG selon le mode d'utilisation.

EE3..... **EE0** déterminent l'emplacement de L'EEPROM , celle-ci peut être placé n'importe où sur les 64 KO de l'espace adressable et ceci par secteur de 4 Ko. Par exemple : si on a **EE3**...**EE0**=(0000) l'EEPROM va de l'adresse $0800 jusqu'à $0FFF, et si **EE3**...**EE0** = (1111) l'EEPROM va de $F800 à $FFFF.

NOSEC : (EEPROM *security disable*) ce bit est normalement à un lors de la fabrication du circuit, ce qui veut dire la sécurité est enlevée, toutefois on peut demander au fabricant de mettre cette sécurité en service. Sécurité veut dire que le circuit ne peut être employé qu'en mode circuit seul ou en mode *boot strap*, ceci interdit la relecture de logiciel qui ne peut se faire qu'en mode étendu.

NOCOP : Ce bit permet d'enlever le système de surveillance de fonctionnement du logiciel (**COP**) appelé chien de garde.

EEON : Veut dire *EEPROM enable*, ce bit est normalement forcé à un. Si ce bit est à zéro la zone mémoire EEPROM est retirée de la carte mémoire.

II-2-4-2- LE REGISTRE INIT $103D :

Ce registre permet la modification par pas de 4KO les emplacements des registres et de la mémoire RAM. Il est protégé car il ne peut être écrit que pendant les 64 cycles d'horloge **E** qui suivent un RESET.

INIT	ram3	ram2	ram1	ram0	reg3	reg2	reg1	reg0
AU RESET	0	0	0	0	0	0	0	1

Tableau II- 3) Les affectations de déférents bits du registre INIT au moment de RESET.

RAM3...RAM0 : donnent l'emplacement de la RAM par pas de 4 KO dans l'espace adressable.

REG3...REG0 : donnent l'emplacement des registres par pas de 4 KO dans l'espace adressable

Remarque :

On peut remarquer qu'il y a un risque de chevauchement des registres sur la zone mémoire RAM. Les registres sont prioritaires sur la RAM et les octets communs sont dévalidés de la zone RAM.

II-2-4-3- LE REGISTRE OPTION $1039 :

Ce registre permet la mise en service d'un certain nombre de fonctions. Certains de ses bits sont protégés (**IRQE**, **DLY**, **CR1** et **CR0**) et ne peuvent être modifiés que pendant les 64 premiers cycles d'horloge **E** qui suivent un RESET.

option	adpu	csel	irqe	dly	cme	0	cr1	cr0
AU RESET	0	0	0	1	0	0	0	0

Tableau II- 4) Les affectations des différents bits du registre OPTION au moment de RESET.

ADPU (*A/D power up*) : Ce bit mis à un permet la mise en service de la pompe de charge du convertisseur analogique numérique (**CAN**) . Il faut attendre 100 µs avant d'utiliser le **CAN**.

CSEL (*clock select*) : Permet la sélection de l'horloge de la pompe de charge du convertisseur et du système d'écriture en EEPROM. Ce bit doit être à 1 si la fréquence d'horloge est trop faible pour la pompe de charge.

Si **CSEL**=1 :l'horloge est le signal **E**.

Si **CSEL**=0 : l'horloge est fixé par un circuit **RC** interne.

IRQE (*IRQ enable*) : Ce bit détermine le signal actif de l'entrée de demande d'interruption IRQ.

IRQE=0, l'entrée est sensible au niveau logique 0

IRQE=1, l'entrée est sensible à un front (descendant),

DLY (*enable oscillator startup delay*) délai de mise en route de l'oscillateur.

DLY=1, le temps d'attente est de 4064 cycles d'horloge.

DLY=0, le temps d'attente est de 4 cycles d'horloge.

CME (*clock monitor enable*) mise en service de la surveillance d'horloge.

CME=1, surveillance en service.

CME=0, surveillance hors service.

CR_0 ,CR_1 (*COP timer rate select*) programmation de la période d'horloge du chien de garde.

CR_1	CR_0	$E/2^{15}$ divisée par :
0	0	1
0	1	4
1	0	16
1	1	64

Tableau II- 5) Facteur de division de $E/2^{15}$ suivant les bits CR_1 CR_0 .

II-2-4-4- LE REGISTRE BPROT $1035 :

Ce registre permet la protection du registre **CONFIG** et la protection par bloc de 512 Octets de l'EEPROM.

BPROT	0	0	0	PCTON	BPRT3	BPRT2	BPRT1	BPRT0
AU RESET	0	0	0	1	1	1	1	1

Tableau II- 6) Les affectations des différents bits du registre BPROT au moment du RESET.

PTCON=1, le registre **CONFIG** ne peut être effacé ou programmé.

PTCON=0, le registre **CONFIG** peut être effacé et programmé normalement.

BPRT3...**BPRT0** : ces quatre bits sélectionnent le paquet de 512 octets qui doivent être protégés, si le bit est à 1 la protection est en service.

II-2-4-5- LE REGISTRE TMSK2 $1024 :

Les deux bits de poids faible de ce registre ne peuvent être modifiés que pendant les 64 cycles d'horloge **E** qui suivent le RESET

Ils déterminent le rapport de pré-division appliqué à l'horloge **E** pour la commande du compteur libre **TCNT**.

PR1	PR0	FACTEUR DE PREDIVISION
0	0	1
0	1	4
1	0	8
1	1	16

Tableau II- 7) Facteur de pré-division appliqué à l'horloge E suivant les bits PR1,PR0.

Remarque :

Au RESET les deux bits **PR1** et **PR0** sont mis à zéro.

II-2-4-6- CONFIGURATION DE LA MEMOIRE :

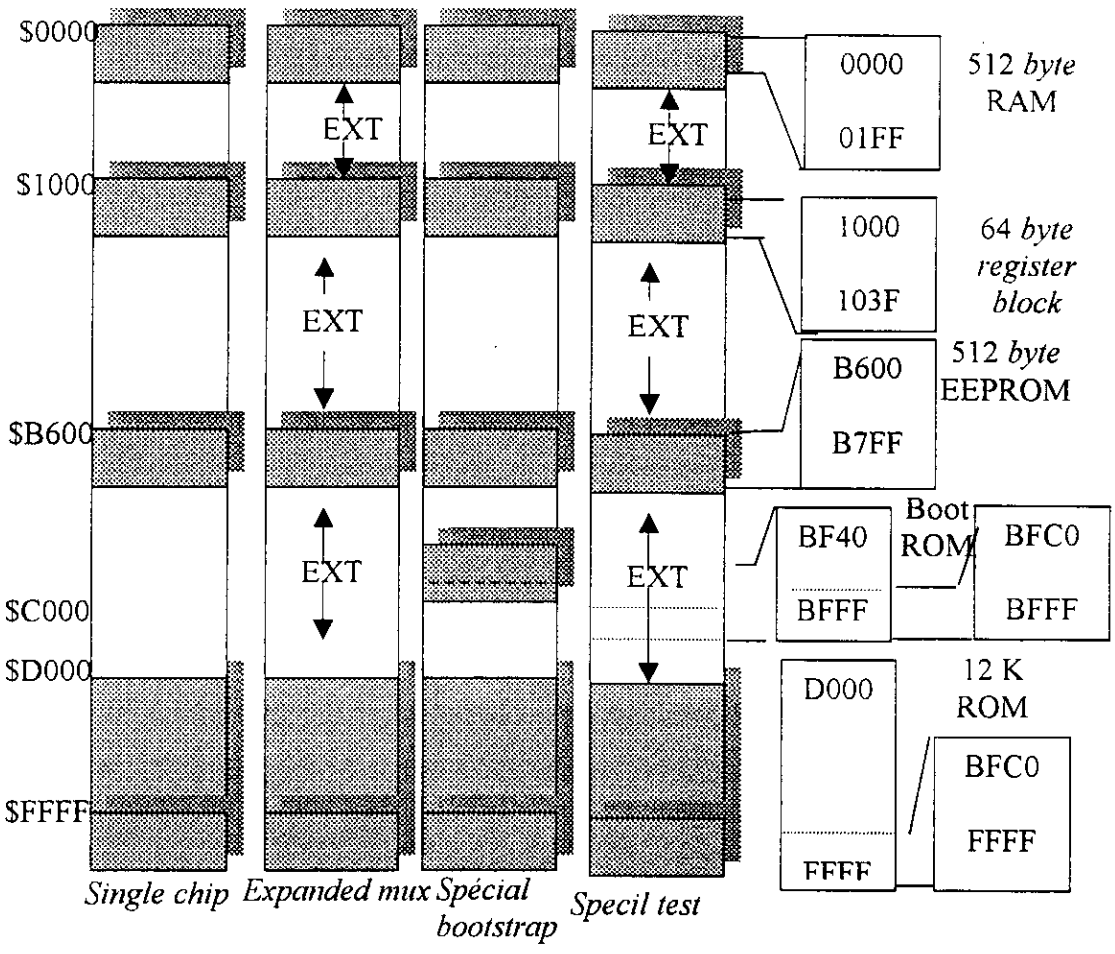


Figure II- 3) Cartographie mémoire

II-2-5- UTILISATION DE L'EEPROM :

l'EEPROM du 68HC11 peut être :

- Effacée octet par octet, ligne par ligne ou entièrement en une seule fois.
- Ecrite et programmée par une procédure particulière.
- Protégée.

La haute tension nécessaire est générée par une pompe de charge interne.

II-2-6- MODES DE PROGRAMMATION DE L'EEPROM :

La programmation de l'EEPROM est contrôlée par le registre **PPROG**.

PROG	ODD	EVEN	0	BYTE	ROW	ERASE	EELAT	EEPGM
AU RESET	0	0	0	0	0	0	0	0

Tableau II- 8) Les affectations des bits du registre PPROG au moment de RESET.

ODD EVEN : réservés au fabriquant.

BYTE ROW :

BYTE	ROW	Effacement
0	0	Effacement complet
0	1	Effacement d'une ligne
1	0	Effacement d'un octet
1	1	Effacement d'un octet

Tableau II- 9) Types d'effacement selon les bits BYTE et ROW.

ERASE :autorisation d'effacement.

ERASE=1 effacement

ERASE=0 écriture et lecture de l'EEPROM.

EELAT (*EEPROM latch control*) contrôle de verrouillage de l'EEPROM. Ce bit, quand il est à 1 permet la programmation de l'EEPROM en provoquant la configuration des données, des adresses et du temps nécessaire à l'écriture ou à l'effacement.

Si ce bit est à 0 la mémoire peut être lue.

EEPGM :(*EEPROM programming voltage enable*) mise en service du pompe de charge.

EEPGM=1 permet la création de la tension nécessaire à la programmation.

Remarque :

Le registre **CONFIG** est lui même en EEPROM il faut donc utiliser la même procédure pour l'effacer ou le modifier. Par sécurité il ne peut être effacé avec le mode effacement total.

II-2-7- PROTECTION DU CONTENU DE LA MEMOIRE :

Afin d'éviter le piratage du logiciel contenu dans la mémoire il est possible d'empêcher la relecture des mémoire RAM et EEPROM. Si l'option sécurité est en place grâce à la mise en place du logiciel approprié à la fabrication le bit **NOSEC** du registre **OPTION**, mis à zéro interdit la lecture et la copie des données en mémoire en interdisant l'utilisation du mode étendu. Le *bootloader* est utilisé pour supprimer l'option sécurité.

II-2-8- LA TRANSMISSION SERIE D'INFORMATION :

II-2-8-1- CARACRERISTIQUES DE LA TRANSMISSION SERIE ASYNCHRONE DE 68HC11 :

- La ligne en attente est à l'état haut ;
- Le bit de *start* est au niveau bas ;
- Le format de données est de 8 ou 9 bits ;
- Le bit de poids faible est transmit le premier ;
- 1 bit de **STOP** au niveau haut indique la fin de transmission ;
- signal de **BREAK** sépare les trames (données)
- l'émetteur et le récepteur sont indépendant donc permettant la transmission en FULL duplex
- L'interface permet le réveil automatique dés qu'elle reçoit des signaux valides.

II-2-8-2- UTILISATION DE LA LIAISON SERIE ASYNCHRONE (SCI :SERIAL COMMUNICATION INTERFACE) :

L'interface série asynchrone utilise la première et la deuxième ligne du port **D** : **Rx=PD_0** et **Tx=PD_1**

Adresses des registres utilisés :

- $102B **BAUD** : registre de débit.
- $102C **SCCR1** : registre de contrôle 1
- $102D **SCCR2** : registre de contrôle 2
- $102E **SCSR** : registre d'état
- $102f **SCDR** : registre de données(DTR ou RDR)

II-2-8-3- L'EMISSION :

Il suffit d'écrire dans le registre **SCDR**, l'octet est ensuite transféré dans le registre à décalage, qui lui transmet sur la ligne, les bits les uns après les autres. Ce transfert met à 1 le bit **DTRE** du registre d'état **SCSR**, ceci peut alors générer une demande d'interruption si elle est autorisée, à la fin de transmission le bit de transmission complète (**TC**) passe à 1 et peut aussi générer une demande d'interruption si elle est autorisée.

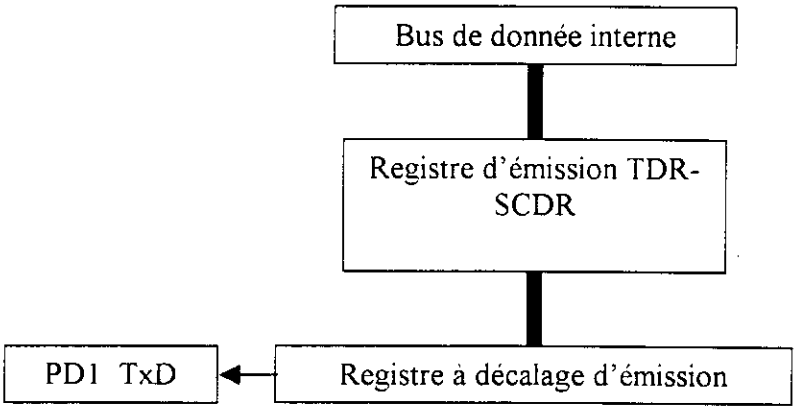


Figure II- 4) Schéma de l'interface d'émission

II-2-8-4- LA RECEPTION :

La lecture du registre **SCDR** suffit car celui-ci contient la dernière donnée reçue si le récepteur est validé. Le bit **RDRF** du registre d'état **SCSR** est mis à 1 pour indiquer que la donnée a été transférée dans le registre à décalage **SCDR**. Une interruption peut être générée (donnée reçue) si elle est autorisée.

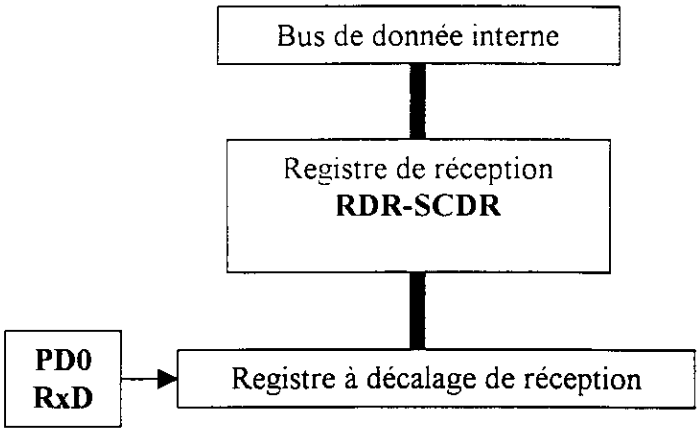


Figure II- 5) Schéma de l'interface de réception.

Le principe du réveil :

Il est possible de monter plusieurs récepteurs en réseau, le récepteur doit donc pouvoir détecter qu'un message va lui être envoyé, il est en sommeil jusqu'à ce qu'un message le réveille. Le réveil consiste à détecter le début du message qui contient l'adresse du destinataire, le logiciel dans ce type de récepteur évalue le premier caractère d'un message, si le message est prévu pour un autre récepteur il se remet en sommeil. Il existe deux possibilités de réveil, choisies par le bit **WAKE** du registre de contrôle 2 (**SCCR2**).

1- Le mode attente :

Le mode attente est caractérisé par la mise à l'état 1 de la ligne **RxD** pendant au moins la durée d'un caractère (10 ou 11 bits).

Le récepteur se réveille à chaque fois qu'un caractère est reçu, quand la ligne **RxD** est en attente. Le système doit donc prévoir une telle durée entre 2 messages mais pas entre les caractères d'un message.

2- Le mode ADRESS MARK :

Le récepteur se réveille quand il reçoit une information d'adresse. Le bit le plus significatif du mot transmis est utilisé pour indiquer si le caractère est une adresse

(**MSB**=1) ou une donnée (**MSB**=0), ensuite chaque récepteur doit déterminer si le message lui est destiné.

II-2-8-5- ETUDE DES REGISTRES :

1-Le registre de contrôle 1 : SCCR $102C :

SCCR1	R8	T8	0	M	WAKE	0	0	0
AU reset	-	-	0	0	0	0	0	0

Tableau II- 10) Etat du registre SCCR1 au RESET.

R8 : si **M**=1 le neuvième bit du mot reçu est mis dans **R8**.

T8 : si **M**=1 T8 contient le neuvième bit du caractère à transmettre.

M : longueur de caractère

M=0 8 bits de donnée ;

M=1 9 bits de donnée ;

WAKE : sélection du mode de réveil

WAKE=0 mode attente ;

WAKE=1 mode **ADRESS MARK** ;

2-Le registre de contrôle 2 SCCR2 $102D :

SCCR2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
AU reset	0	0	0	0	0	0	0	0

Tableau II- 11) Etat du registre SCCR2 au RESET.

TIE : interruption autorisée pour le transmetteur vide :

TIE=0 : interruption inhibée ;

TIE=1 :interruption autorisée ;

TCIE : interruption autorisée pour transmission terminée :

TCIE=0 : interruption inhibée ;

TCIE=1 : interruption validée ;

RIE : interruption autorisée pour récepteur plein :

RIE=0 : interruption inhibée ;

RIE=1 :interruption autorisée

ILIE : : interruption autorisée en mode attente :

ILIE=0 : interruption inhibée ;

ILIE=1 : interruption validée ;

TE : Transmetteur prêt :

TE=1 la sortie du registre à décalage est reliée à **TxD** pour l'émission ;

TE=0 la ligne dépend du **DDRD1** ;

RE : récepteur prêt :

RE=1 la ligne **PD0=RxD** est forcée en entrée ;

RE=0 la ligne dépend du **DDRD0** ;

RWU :mis en fonction du réveil .

SBK : envoi d'un signal **BREAK** : si se bit est à 1 puis à 0 le transmetteur émet 10 ou 11 bits au niveau 0.

Si **SBK** reste à 1 : le émetteur envoie continuellement des blocs de 0 (signal **BREAK**) jusqu'à ce que **SBK** repasse à 0.

3- Le registre d'état SCSR($102E) :

SCSR	TDRE	TC	RDRF	IDLE	OR	NF	FE	0
AU reset	1	1	0	0	0	0	0	0

Tableau II- 12) Etat du registre d'état au RESET.

TDRE :registre de transmission vide :

Mis à 1quand le registre de transmission a été transféré dans le registre à décalage. Et remis à 0 par une lecture de registre **SCSR** suivie d'une écriture dans le registre **SCDR.**

TC transmission terminée : mis à 1 à la fin d'une transmission et remis à 0 par une lecture du registre d'état suivie d'une écriture dans le registre de transmission.

RDRF : registre de réception plein : ce bit est mis à 1 quand le registre à décalage a été transféré dans le **SCDR** il est remis à 0 par une lecture de registre d'état suivie d'une lecture de registre de réception (**SCDR**)

IDLE : détection de mode attente : mis à 1 par la détection du mode attente sur la ligne (niveau haut pendant une durée d'au moins un caractère : 10 ou 11 bits) ;il est remis à 0 par la lecture du registre d'état suivie par la lecture du **SCDR**.

OR : (*over run error*) erreur d'écrasement : dans ce cas la donnée qui arrive dans le registre est perdue mais la donnée dans le SCDR qui n'a pas encore été lue n'est pas modifiée.

FE : erreur de format : mis à 1 si le bit **STOP** n'est pas détecté, ce bit est remis à 0 par une double lecture.

4- Le registre de débit : BAUD ($102B) :

Le registre **BAUD** permet de programmer la vitesse de transfert des informations.

BAUD	TCLR	0	SCP1	SCP0	RCBK	SCR2	SCR1	SCR0
AU reset	0	0	0	0	0	U	U	U

Tableau II- 13) Etat du registre BAUD au RESET.

Note : U signifie que le bit pas affecté par le RESET

TCLR:(*clear baud rate counter*): utilisé uniquement en mode test par le fabriquant pour remettre à 0 le compteur de gamme de débit.

RSKB :utilisé en mode test pour établir un ou exclusif entre les horloges de réception et d'émission.

TCLR :utilisé en mode test pour remettre à 0 les bits de sélection de la vitesse de transmission.

SCP1-SCP0 : pré-diviseur (à partir de l'horloge ϕ_2 : même signal que **E** mais déphasé de 90^0).

SCP1	SCP	TAUX	VITESSE MAXI AVEC UN QUARTZ DE 8MHZ
0	0	1	125000 BAUDS
0	1	3	41667 BAUDS
1	0	4	31250 BAUDS
1	1	3	9600 BAUDS

Tableau II- 14) Pré-division de ϕ_2 .

SCR2-SCR1-SCR0 : sélection de débit : Ces bits ne sont pas affectés par le RESET. Le tableau suivant donne les vitesses pour une prédivision maximale (9600 bauds)

SCR2	SCR1	SCR0	TAUX	VITESSE
0	0	0	1	9600
0	0	1	2	4800
0	1	0	4	2400
0	1	1	8	1200
1	0	0	16	600
1	0	1	32	300
1	1	0	64	150
1	1	1	128	075

Tableau II- 15) Sélection de débit.

II-3- UTILISATION DU LOGICIEL PCBUG11 : [7], [8], [9]

La façon la plus simple pour la programmation du 68HC11 est l'utilisation du logiciel PCBUG11 (ver3.42) de MOTOROLA. Cela permet la programmation de l'EEPROM du microcontôleur, l'exécution du programme téléchargé et la correction d'éventuelles erreurs.

II-3-1- LE FONCTIONNEMENT DE PCBUG11 :

Le logiciel PCBUG11 fonctionne uniquement en mode *bootstrap*. En effet, en mode *bootstrap*, il apparaît une "*boot* ROM " en $BF40-$BFFF, zone de mémoire morte à laquelle le processeur saute après le RESET. Cette mémoire de *boot*, présente dans la cartographie uniquement en mode *bootstrap* et *special test*, contient une petite routine qui autorise le téléchargement de 256 octets depuis le port série vers la RAM à partir de l'adresse $0000. En effet, ce programme en *boot* ROM configure la liaison

série à 9600 bps (bits par seconde), 1 bit de *start*, 8 bits de données, 1 bit de *stop* (pas de contrôle ni de parité ni de flux). Le programme de *boot* offre également la possibilité de télécharger les 256 octets à une vitesse de 1200 bps par l'intermédiaire d'une routine interne de la ROM appelée *autobaud* qui détecte la vitesse du téléchargement grâce à la réception du caractère $FF. C'est donc à ce moment là que le logiciel PCBUG11 envoie un petit programme de 192 octets environ qui est appelé *TALKER*. Comme son nom l'indique, ce petit programme va permettre de communiquer avec l'ordinateur hôte et ainsi d'exécuter les commandes de PCBUG11.

II-3-2- PRESENTATION DE L'INTERFACE GRAPHIQUE DU LOGICIEL PCBUG11 :

L'interface graphique du logiciel PCBUG11 se présente par le capture d'écran suivant :

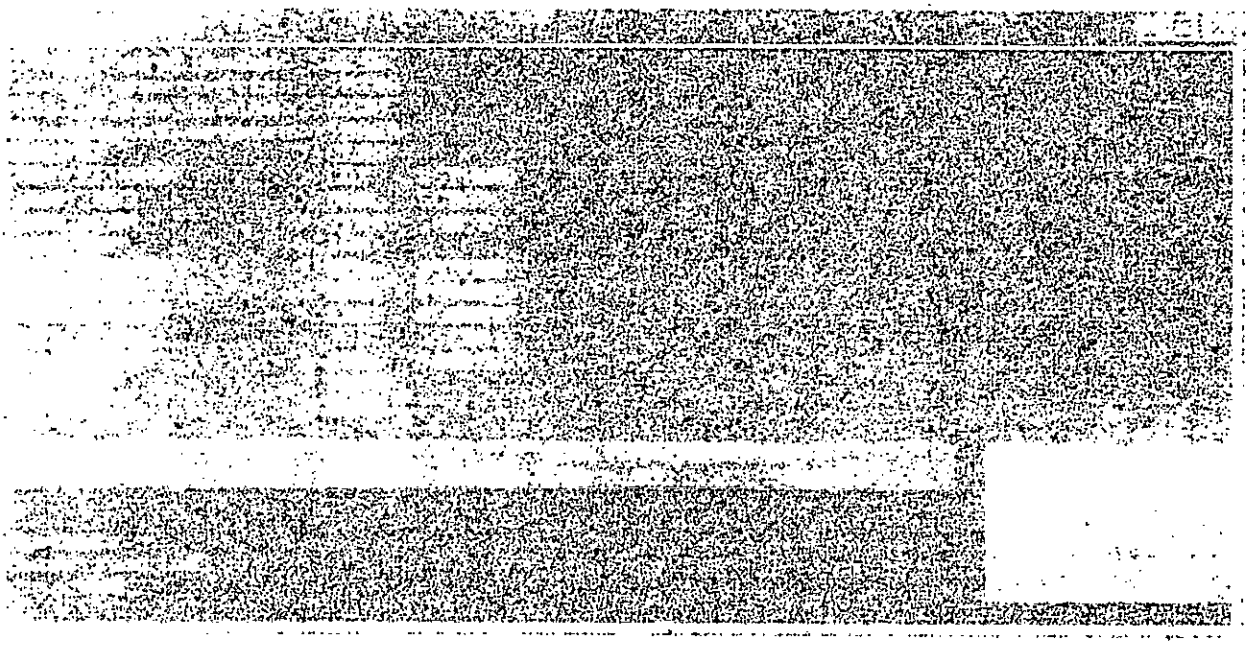


Figure II- 6) Le logiciel PCBUG11 (v3.42) en cours d'utilisation.

Cet écran est décomposé en quatre zones :

1. zone de commandes (noir) : située en bas à gauche de l'écran, elle permet de taper les commandes utilisateur.
2. zone d'état (violet) : située en bas à droite de l'écran, elle montre le type de processeur, le mode de fonctionnement de PCBUG11 (*Running*, *Stopped*, *Trace*) et l'état des lignes de communication.

3. zone de l'état des registres (rouge) : située au milieu de l'écran, elle permet la visualisation de l'état actuel des registres internes du processeur.
4. zone principale (bleu) : située en haut de l'écran, elle permet l'affichage des résultats des opérations lancées par l'utilisateur.

II-3-3- DESCRIPTION DES COMMANDES DE PCBUG11 :

Dans la zone des commandes on peut utiliser les principaux commandes suivantes :

ASM *addr* : permet la décompilation et l'édition en ligne du code téléchargé en mémoire à l'adresse *addr*.

BF *addr1 addr2 byte|word* : permet de remplir un bloc de mémoire commençant à *addr1* et terminant par *addr2* avec la valeur *byte word*.

R *addr [macroname]* : Cette fonction permet de placer des points d'arrêt permettant ainsi à l'utilisateur de débugger son programme. Cette fonction est utilisable seulement si on a répondu **NO** à la question "*do you wish use the XIRQ interrupt ?*" Au lancement de PCBUG11. Le programme doit être également implanté dans une zone de mémoire accessible en écriture puisque PCBUG11 place une instruction **SWI** (interruption logicielle) à l'adresse du point d'arrêt décalant ainsi tout le reste du programme en mémoire. Le paramètre *[macroname]* est facultatif, il permet d'exécuter une macro préalablement chargée lors d'un point d'arrêt.

ROM *0|addr1 addr2* : permet de supprimer une plage EEPROM (paramètre 0) ou bien de la configurer en spécifiant l'adresse de départ *addr1* et d'arrivée *addr2*.

EEPROM *ERASE [bulk]* : permet d'effacer partiellement ou dans son intégralité l'EEPROM dont l'étendue est définie grâce à **EEPROM** *addr1 addr2*.

G *addr* : exécute le programme à l'adresse *addr*.

HELP : affiche toutes les fonctions de PCBUG11 ainsi qu'un commentaire.

LOADS *filename* : permet de charger un fichier S19 en mémoire.

MD *startaddr [endaddr]* : visualise la plage de mémoire de *startaddr* à *endaddr*.

MM *addr* : modifie le contenu de la case mémoire à l'adresse *addr*.

NOBR *addr* : supprime le point d'arrêt placé à l'adresse *addr*.

QUIT *Y* : quitte le logiciel PCBUG11.

RD : rafraîchit l'affichage des registres dans la fenêtre centrale.

RESTART : permet de relancer PCBUG11.

La liste complète de toutes les instructions ainsi que leur mode d'emploi est disponible dans le manuel de PCBUG11.

II-3-4- CHARGEMENT D'UN PROGRAMME AVEC PCBUG11 :

Lorsque l'utilisateur désire charger un programme en EEPROM il doit procéder comme suit :

- effectuer un RESET sur la carte cible en mode *bootstrap*.
- exécuter **PCBUG11** *-A port=2* - (cas d'un 68HC11Ax relié au port série COM2 de l'ordinateur).

Remarque:

Le logiciel doit se lancer normalement et ne doit pas indiquer de messages d'erreurs. Si une erreur apparaît à ce moment là, il y a plusieurs sources de problèmes possibles :

* le 68HC11 est mal configuré (non *bootstrap*),

* le MAX232 n'est pas opérationnel (niveaux de 10 V inexistants),

* PCBUG11 ne supporte pas un ordinateur de plus de 200MHz (génération de l'erreur "*runtime error*",

* si PCBUG11 est exécuté sous windows 95/98, il peut être nécessaire de diminuer la sensibilité d'attente de PCBUG11.EXE dans propriétés-> divers.

- taper **EEPROM** *SB600 SB7FF*, si l'EEPROM est située de $B600 à $B7FF.

- taper **EEPROM** *ERASE BULK*, qui permet d'effacer l'EEPROM. Nous pouvons ainsi vérifier si l'opération a été réussie en tapant **MD** *SB600 SB7FF* : on ne doit voir que des octets $FF.

Remarque importante :

Si la programmation de la mémoire EEPROM n'est pas correcte, il est possible soit qu'elle ne se trouve pas aux adresses $B600 à $B7FF soit que le registre de protection en écriture BPROT (présent sur les 68HC11Ex, F1...) ait été configuré pour protéger l'EEPROM. Pour résoudre ce problème, il faut, après avoir lancé PCBUG11, taper **MM** *S1035 S10* où *S1035* est l'adresse du registre BPROT.

Si l'EEPROM n'est pas présente dans la cartographie mémoire, c'est que le "CONFIG *register*" est mal configuré : attention, ce registre se programme comme une case de mémoire EEPROM. Le bit concerné est nommé EEON et permet d'activer ou non l'EEPROM.

- taper ensuite **LOADS** *filename* où *filename* est le nom du fichier compilé (sans l'extension) au format S19 de MOTOROLA qui se trouve alors dans le même répertoire que PCBUG11. Les fichiers S19 sont créés par exemple par le compilateur assembleur (asmhc11.exe ou as11.exe).

- il ne reste plus qu'à vérifier que le programme est bien implanté en mémoire grâce à l'instruction **ASM** *SB600* si le programme démarre en *SB600*.

- si le programme est bien chargé (pas de message d'erreurs) alors l'exécution du programme se fait par **G** *SB600* et l'arrêt par **S**. Lorsqu'on stoppe le programme, le 68HC11 revient au *TALKER*.

II-4- LE MONITEUR BUFFALO :

II-4-1- QU'EST CE QU'UN MONITEUR ?

Nous avons vu que le logiciel PCBUG11 télécharge un petit programme de 192 octets environ en mémoire RAM, un moniteur est un programme beaucoup plus gros (8Ko environ) que l'on programme dans une EPROM externe et qui permet donc de remplacer le *TALKER* de PCBUG11. Les avantages sont multiples puisque le moniteur

occupe très peu de RAM (uniquement pour quelques variables), il permet de débuguer le programme utilisateur grâce à des commandes intégrées que nous verrons plus loin et le moniteur dialogue avec l'utilisateur par l'intermédiaire d'un simple terminal (*l'hyperterminal* par exemple).

L'exécution du moniteur se fait lors du RESET en mode étendu et rend libre l'utilisation du port série après le lancement du programme utilisateur. Pour reprendre la main, il suffit de réinitialiser le microcontrôleur.

Le moniteur BUFFALO 3.4 pour 68HC11 occupe un espace mémoire de 8Ko en EPROM. Le vecteur de RESET aux adresses $FFFE et $FFFF doit être configuré avec $8000.

II-4-2- UTILISATION DU MONITEUR BUFFALO 3.4 :

Les commandes du moniteur BUFFALO 3.4 sont similaires à celles utilisées par PCBUG11 et sont accessibles dès la mise en route du moniteur en tapant entrée ou **HELP** : la capture d'écran suivante montre l'ensemble des commandes dont le rôle est assez intuitif.

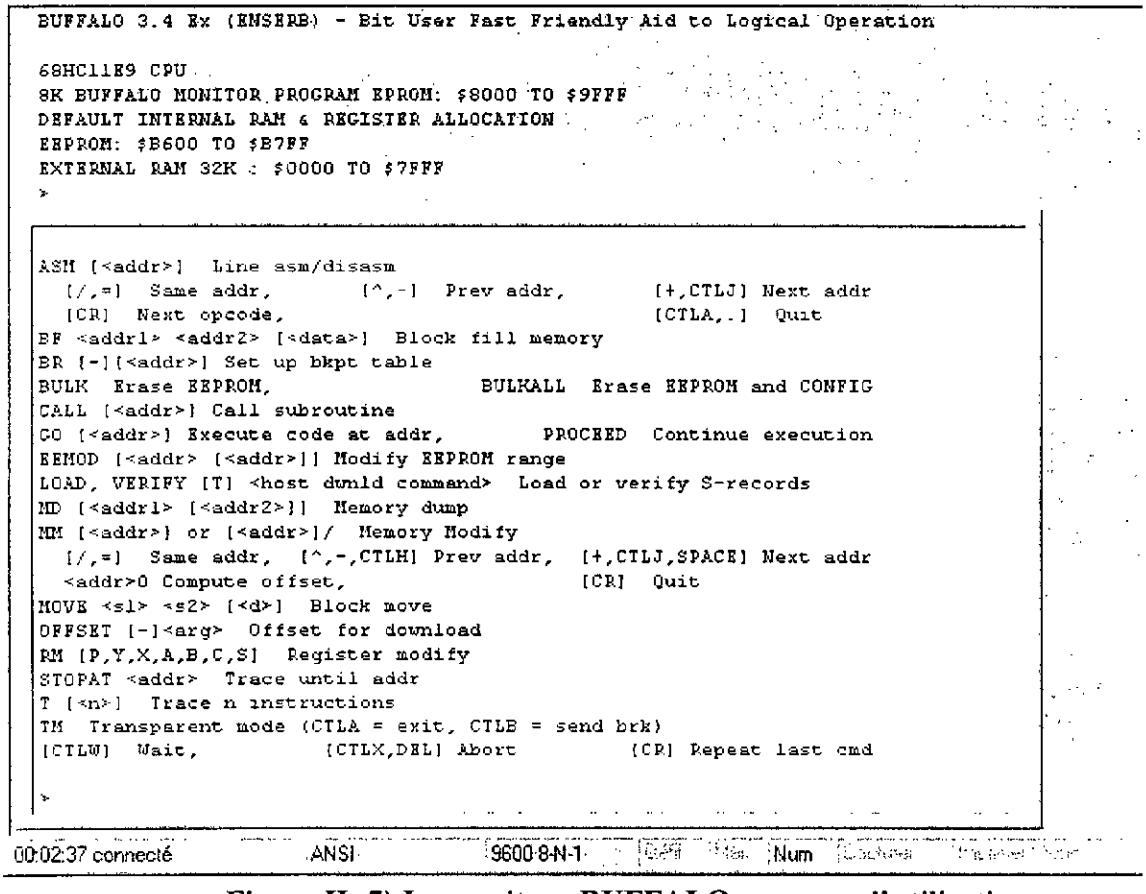


Figure II- 7) Le moniteur BUFFALO en cours d'utilisation.

Le chargement d'un programme utilisateur par le moniteur BUFFALO se fait grâce à la commande **LOAD** *T*. Après avoir tapé la commande **LOAD** *T*, le moniteur attend le téléchargement d'un fichier S19. On utilise pour cela la fonction de transfert de fichiers du terminal disponible dans le menu Transferts > Envoyer un fichier texte > fichier.s19.

Remarque importante :

Si on charge un programme en EEPROM (à partir de l'adresse $B600), il faut ralentir le transfert de caractères en attribuant par exemple une durée de 1/10s par octet.

Le téléchargement terminé, le moniteur renvoie alors le message "*done*". L'exécution du programme se fait alors grâce à l'instruction **G** *addr*. Lors des tests, le programme en format S19 doit être chargé en $C000 (RAM) puis en $B600 (EEPROM). L'exécution s'est faite par **G** *$C000* (ou $B600 selon le cas) depuis la ligne de commande du moniteur. Le retour à la ligne de commande du moniteur se fait par pression sur le bouton RESET de la carte cible (à condition d'avoir bien spécifié en $FFFE-FF l'adresse de départ du moniteur qui est ici $8000).

Pour des raisons de fiabilité, le moniteur peut être implanté en EPROM comme c'est le cas de toutes les cartes avec moniteur disponible chez MOTOROLA.

II-5- CONCLUSION :

Le microcontrôleur 68HC11 représente un choix judicieux pour notre application car grâce à lui et les logiciels PCBUG11 ou BUFALLO nous pouvons effectuer les tâches suivantes :

- écrire du code et le télécharger en EEPROM.
- débuguer et mettre en point le programme téléchargé.
- exploiter la liaison série asynchrone pour émettre des symboles au micro-ordinateur.
- choisir entre plusieurs vitesses de transfert.

Cette dernière tâche à une grande importance dans la partie expérimentale de notre application car grâce à elle nous peut changer la fréquence du signal informatif.

CHAPITRE III :
ETUDE ET REALISATION D'UN EMETTEUR RECEPTEUR FSK.

CHAPITRE III : ETUDE ET REALISATION D'UN EMETTEUR ET D'UN RECEPTEUR FSK

III-1- INTRODUCTION :

La transmission de signaux numériques s'impose de plus en plus dans de nombreuses applications industrielles, en particulier la transmission par modulation FSK et PSK.

Cependant le montage proposé ici, s'il permet de mettre en évidence les caractéristiques principales de la modulation FSK, il n'est pas le dispositif utilisé industriellement.

Par conséquent, les fréquences des porteuses et du signal modulant seront choisies de façon à ce que l'on puisse en faire une observation aisée à l'oscilloscope que l'on dispose en laboratoire.

III-2- QUELQUES CONSIDERATIONS THEORIQUES : [3], [4]

III-2-1- DEFINITIONS :

T_b : Temps bit exprimé en seconde (durée d'un bit).

$D=1/T_b$: Débit binaire exprimé en bits par seconde.

III-2-2- FORMAT DES SIGNAUX NUMERIQUES :

Un signal numérique binaire peut se présenter sous divers formats. Voici quelques-uns des formats couramment utilisés :

- Unipolaire RZ : Le niveau logique 1 correspond à la tension V_0 pendant la

première demi-période $T_b/2$ et 0 V pendant la deuxième demi période (Figure III-1). Le niveau logique 0 correspond à une tension nulle.

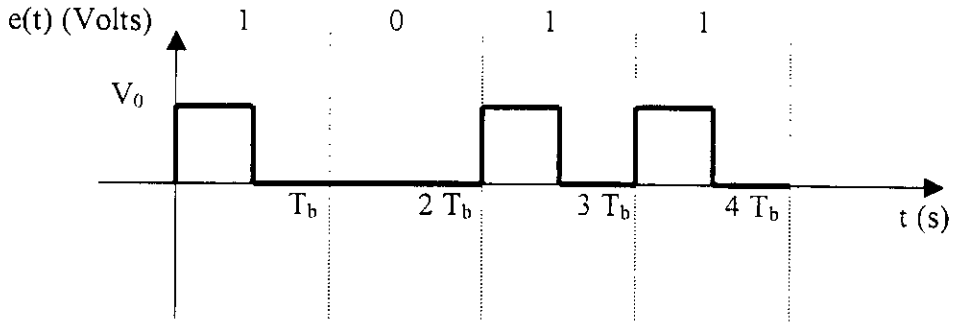


Figure III- 1) Format d'un signal binaire unipolaire RZ.

- Bipolaire RZ : Le premier 1 logique correspond à la tension V_0 pendant la première demi-période et 0 V pendant la seconde (Figure III-2). Le deuxième 1 logique correspond à la tension – V_0 pendant la première demi-période et 0 V pendant la deuxième demi-période, et ainsi de suite. Le niveau logique 0 correspond toujours à une tension nulle pendant toute la période T_b.

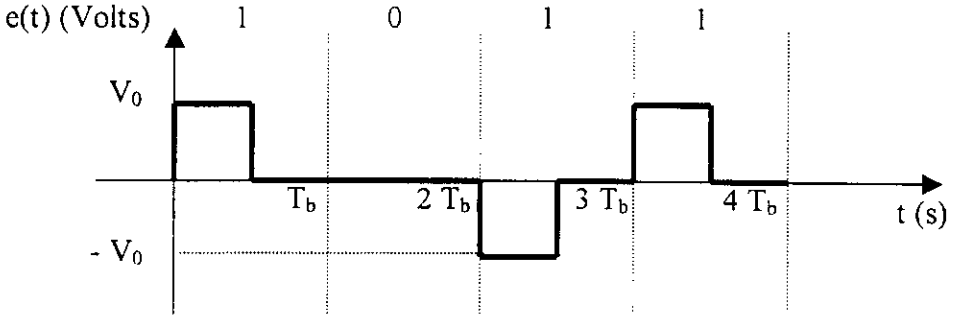


Figure III- 2) Format d'un signal binaire bipolaire RZ.

- Unipolaire NRZ : Le niveau logique 1 correspond à V_0 pendant toute la durée T_b d'une période. Le niveau logique 0 correspond à 0 V pendant toute une période T_b (Figure III-3).

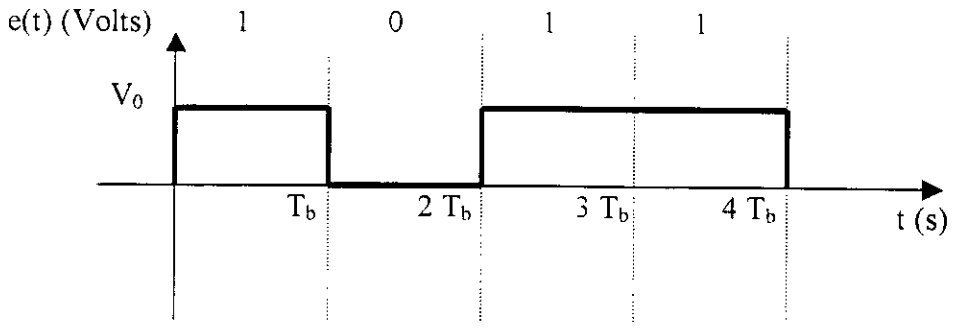


Figure III- 3) Format d'un signal binaire unipolaire NRZ.

- Bipolaire NRZ : Le niveau logique 1 correspond à la tension V_0 pendant toute une période T_b, et le niveau logique 0 correspond à la tension – V_0 pendant toute une période T_b (Figure III-4).

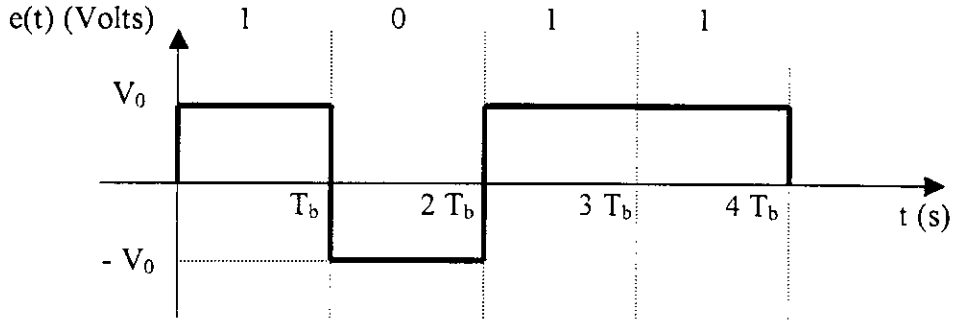


Figure III- 4) Format du signal binaire bipolaire NRZ.

Il existe d'autres formes de codage comme le code Manchester, code à haute densité binaire (utilisé pour les transmissions téléphoniques numériques), ...

Une fois le codage effectué, ces signaux binaires doivent être transmis de l'émetteur vers le récepteur.

Cette transmission peut s'effectuer sans modification des signaux. On parle alors de transmission en bande de base OOK (*On Off Keying*). Mais on peut aussi être amené à transposer le spectre du signal émis. Les modulations FSK et PSK permettent la transposition du spectre du signal à transmettre.

III-2-3- CONTRAINTES IMPOSEES PAR LE CANAL DE TRANSMISSION :

Pour la transmission par câble, le choix du câble assurant la liaison entre émetteur et récepteur dépend des conditions d'utilisation de celui-ci (câbles aériens, enterrés, ...) ainsi que du débit souhaité. En fonction de ce choix, il n'est pas toujours possible de transmettre le signal en bande de base, on doit adapter son spectre aux contraintes imposées par le support de transmission.

La transmission par onde électromagnétique nécessite aussi une transposition de spectre du signal à transmettre tant pour des raisons de dimensionnement des antennes que pour des raisons d'occupation spectrale. Ce mode de transmission ne permet pas de transmettre la composante continue, par exemple, du signal NRZ. Pour ces raisons, on est amené à utiliser les modulations FSK et PSK. Il existe d'autres types de modulation. On peut citer par exemple la modulation OOK (*On Off Keying*), dont l'application typique est la transmission par fibre optique, ainsi que la modulation QAM (Quadrature Amplitude Modulation).

III-3- SCHEMA GLOBAL DE NOTRE LIAISON NUMERIQUE FSK : [3], [4]

Le schéma bloc de notre liaison numérique FSK entre un système à microcontrôleur et le micro-ordinateur est représenté par la Figure III-5).

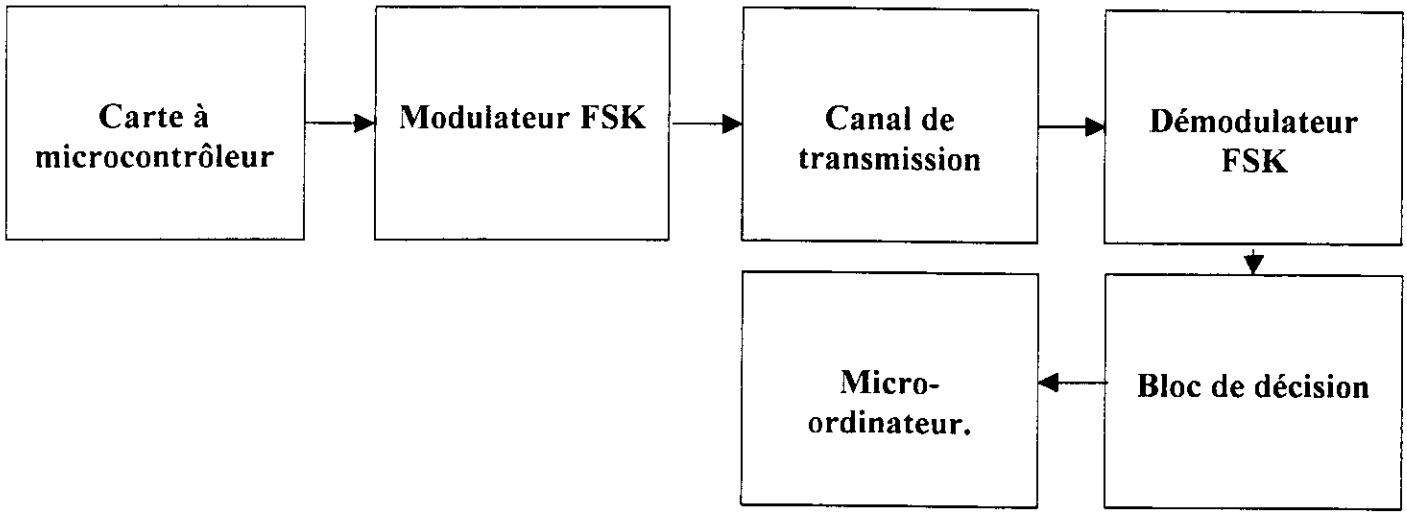


Figure III- 5) Schéma global de notre liaison numérique.

La carte à microcontrôleur étant la même réalisée par [1].

Dans les paragraphes suivants nous allons discuter la conception et la réalisation du modulateur FSK, du démodulateur FSK et du bloc de décision.

III-4- CONCEPTION D'UN MODULATEUR FSK : [3], [4]

III-4-1- SCHEMA DE PRINCIPE DU MODULATEUR FSK :

La Figure III-6) représente le schéma de principe d'un modulateur.

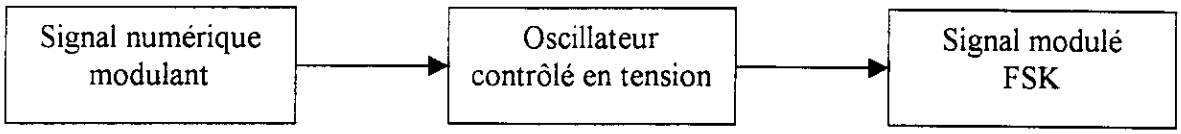


Figure III- 6) Schéma de principe du modulateur FSK.

Comme le montre le schéma de principe Figure III- 6) du modulateur FSK, nous avons notre signal binaire, en format unipolaire NRZ, qui attaque un oscillateur contrôlé en tension (VCO) pour fournir en sortie un signal binaire modulé en FSK.

III-4-2- SCHEMA DETAILLE DU CIRCUIT DU MODULATEUR FSK :

La Figure III-7) suivante représente le schéma détaillé du circuit de modulation FSK :

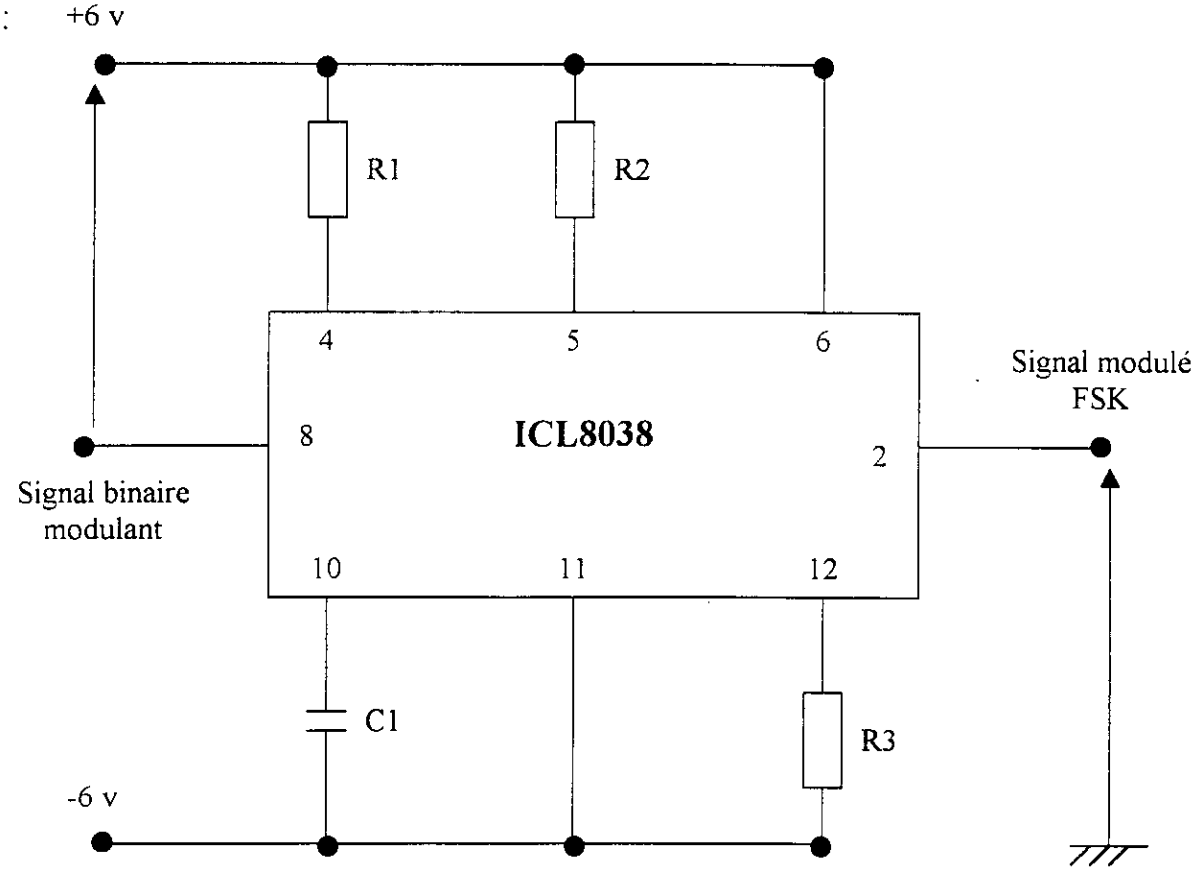


Figure III- 7) Schéma détaillé du modulateur FSK.

Choix des composants :

- Le circuit intégré ICL8038 est un VCO (oscillateur contrôlé par tension) qui constitue le corps du modulateur FSK.
- $R_3 = 82\ k\Omega$ imposée par le constructeur.
- Pour avoir un rapport cyclique de 50%, il faut $R_1 = R_2 = R$. D'autre part, d'après le constructeur, on a les relations :
- $RC_1 = 0{,}033$ ms et On choisira I = 260 m A pour répondre aux exigences du constructeur.

On obtient finalement :

$R = 10\ k\Omega$ et $C_1 = 3{,}3$ nF

III-5- CONCEPTION D'UN DEMODULATEUR FSK ET BLOC DE DECISION : [3], [4]

III-5-1- SCHEMA BLOC DU DEMODULATEUR FSK :

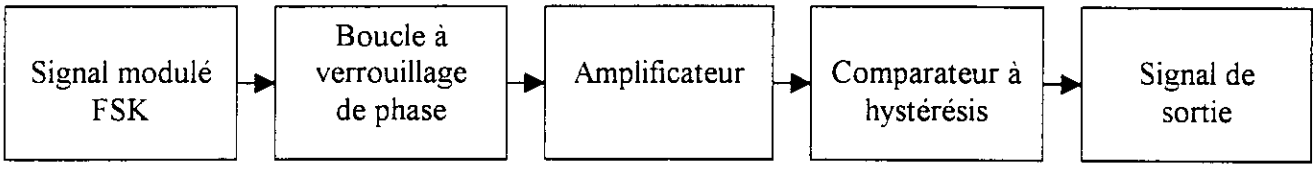


Figure III- 8) Schéma de principe du démodulateur et du bloc de décision.

Le schéma de principe de la Figure III- 8) représente un démodulateur FSK. Sur ce schéma nous avons en entrée un signal modulé en FSK qui attaque une boucle à verrouillage de phase (PLL). Comme les fréquences f_0 et f_b sont très proches, les signaux de sortie de la boucle à verrouillage de phase nécessitent une mise en forme qui est réalisée par le bloc ''décision'' composé d'un amplificateur et d'un comparateur à hystérésis.

III-5-2- SCHEMA DETAILLE DU CIRCUIT DU DEMODULATEUR FSK :

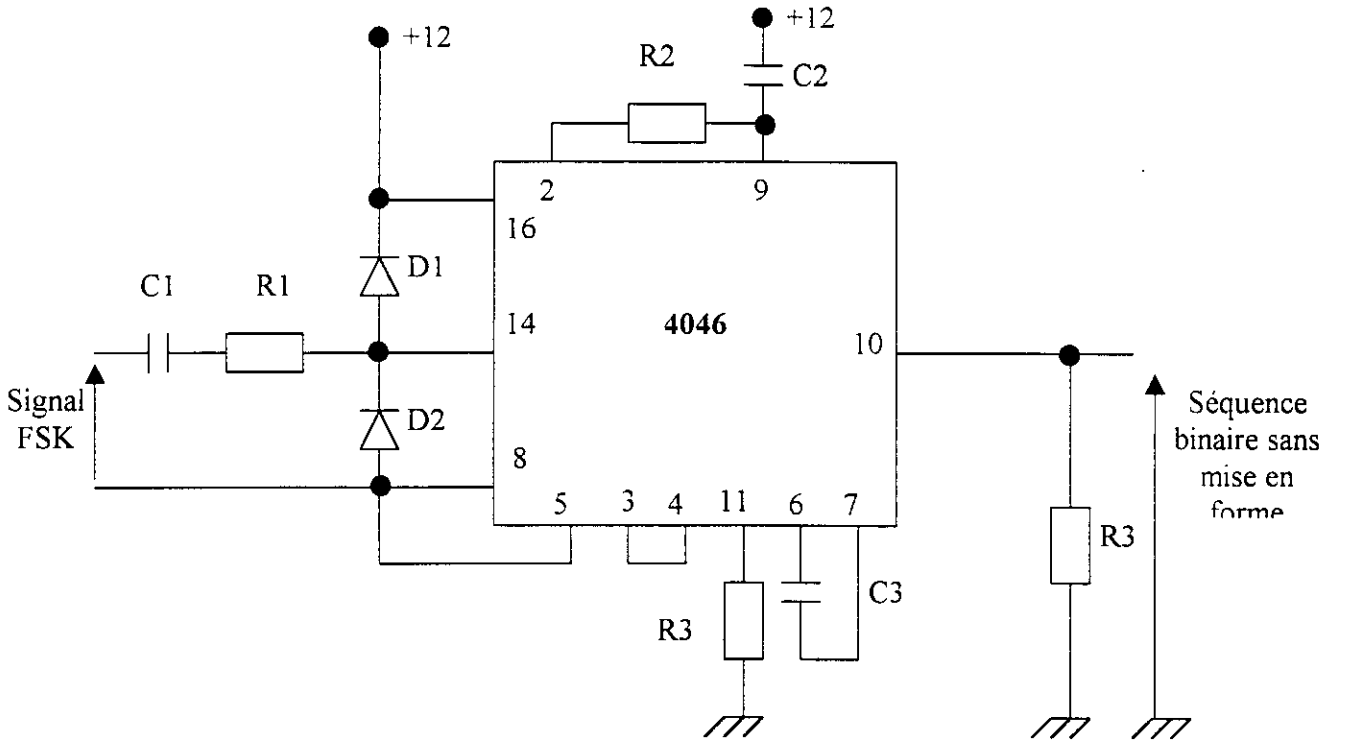


Figure III- 9) Schéma détaillé du circuit de démodulation.

Choix des composants :

- Le circuit 4046 est une PLL (boucle à verrouillage de phase).
- Le condensateur C_1 permet de supprimer une éventuelle composante continue du signal d'entrée. Nous prenons comme valeur pour C_1 :

  $C_1 = 0,1\ \mu F$

- La résistance R_1 permet avec les deux diodes (D_1, D_2) de protéger l'entrée du circuit 4046 contre des tensions négatives ou supérieures à 12 V. Nous prenons comme valeur pour R_1 :

  $R_1 = 1\ k\Omega$.

- R_3 et C_3 sont déterminées à l'aide des abaques fournis par le constructeur (voir annexes). Ils permettent de fixer la plage de verrouillage de la P.L.L.

  $R_3 = 10\ k\Omega$ et $C_3 = 10$ nF

  R_2 et C_2 composent le filtre passe-bas qui permet de commander le VCO par la

valeur moyenne de la tension de sortie du comparateur. Nous retenons les valeurs suivantes

- $R_2 = 220\ k\Omega$ et $C_2 = 1\ nF$

La fréquence de coupure du filtre est donc : $f_c = 720$ Hz ce qui permet de conserver l fondamental de fréquence f_b et le premier harmonique (à $3f_b$) du signal modulant et de couper la fréquence $2f_0$.

Du fait de la proximité de la fréquence centrale et du signal modulant, le signal démodulé est très perturbé. Le filtre passe-bas à la sortie du comparateur de la PLL n'est pas suffisamment sélectif pour reconstituer parfaitement le signal modulant.

Ce montage à toutefois l'avantage de montrer que même lorsque le signal est perturbé, on peut assez facilement reconstituer le signal d'origine, ce qui est un des avantages des signaux numériques par rapport aux signaux analogiques.

III-5-3- SCHEMA DETAILLE DU CIRCUIT DE DECISION :

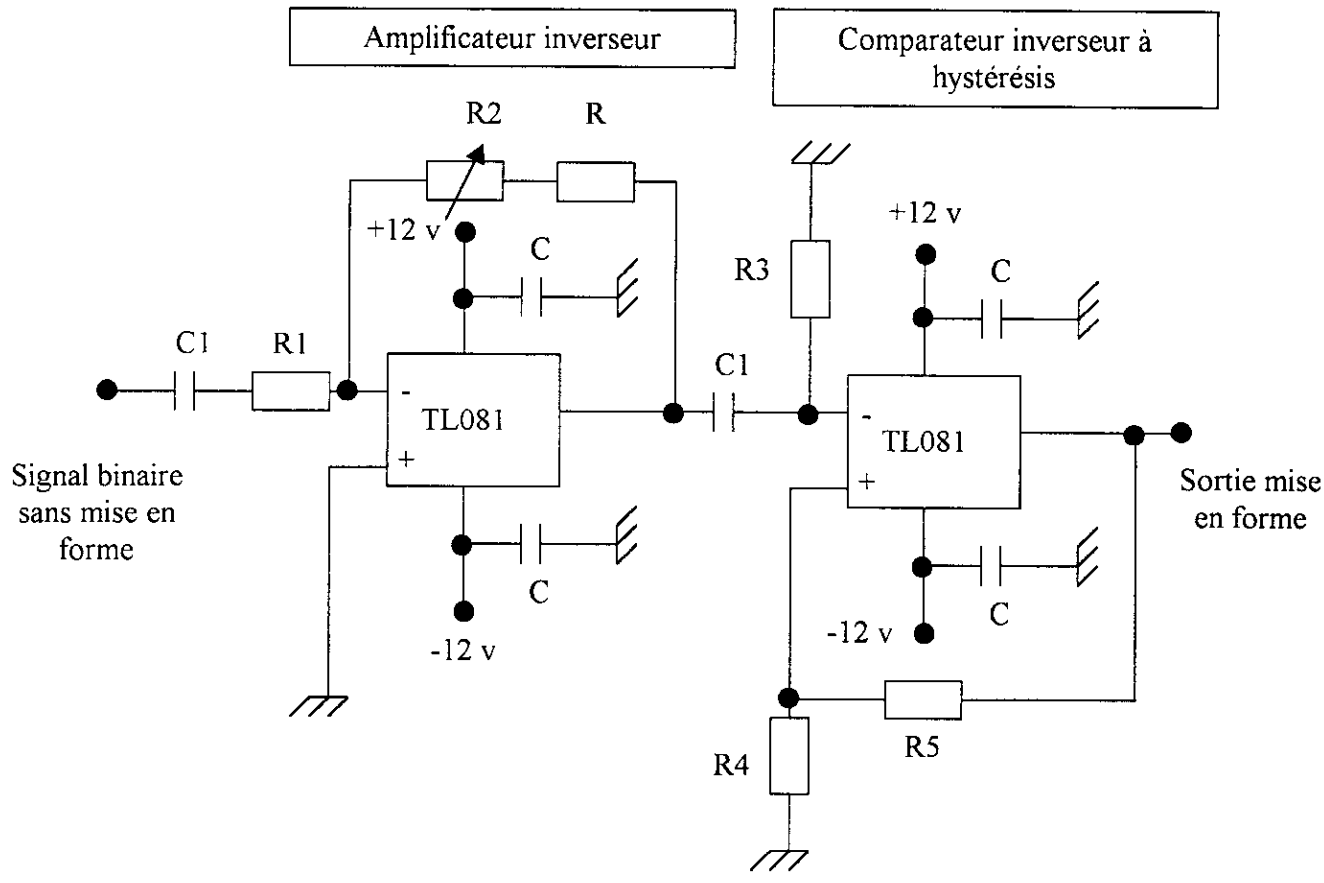


Figure III- 10)Schéma détaillé du circuit de mise en forme.

Choix des composants :

- Les amplificateurs TL081 conviennent parfaitement pour ce type d'application du fait qu'on travaille à des fréquences suffisamment faibles pour que leur slew-rate ne soit pas gênant.

- Condensateurs de découplage : C = 1 m F.

Pour l'amplificateur inverseur :

- C_1 = 0,1 µF. Ce condensateur permet de supprimer la composante continue du signal de sortie de la boucle à verrouillage de phase.

- R_1 = 10 kΩ et R_2 est un potentiomètre de 47 kΩ en série avec une résistance R de 10 kΩ .

  On a ainsi un coefficient d'amplification pouvant varier entre -1 et –5,7.

Pour le comparateur inverseur à hystérésis :

- Les tensions de basculement sont environ de +1V et –1V.

- R_4 = 1 kΩ et R_5 = 10 kΩ ; C_1 = 0,1 m F et R_3 = 1 MΩ .

III-6- REGLAGES ET ESSAIS DU MODULATEUR FSK :

Le signal numérique est à deux états (binaire) mais, dans le cas général, il peut être quaternaire voire davantage. Ici, l'information numérique est simulée à l'aide d'un générateur basse fréquence fournissant des signaux carrés.

Définition et notations :

Pour la suite, on notera :

f_0 : Fréquence centrale du signal modulé.

f_1 : Fréquence du signal modulé correspondant à l'état bas du signal modulant.

f_2 : Fréquence du signal modulé correspondant à l'état haut du signal modulant.

Indice de modulation : $m = |f_1 + f_2| / 2*f_0$

Excursion de fréquence $\Delta f = |f_1 - f_2|/2$

Intéressons-nous tout d'abord à la caractéristique du VCO ICL8038 permettant la modulation.

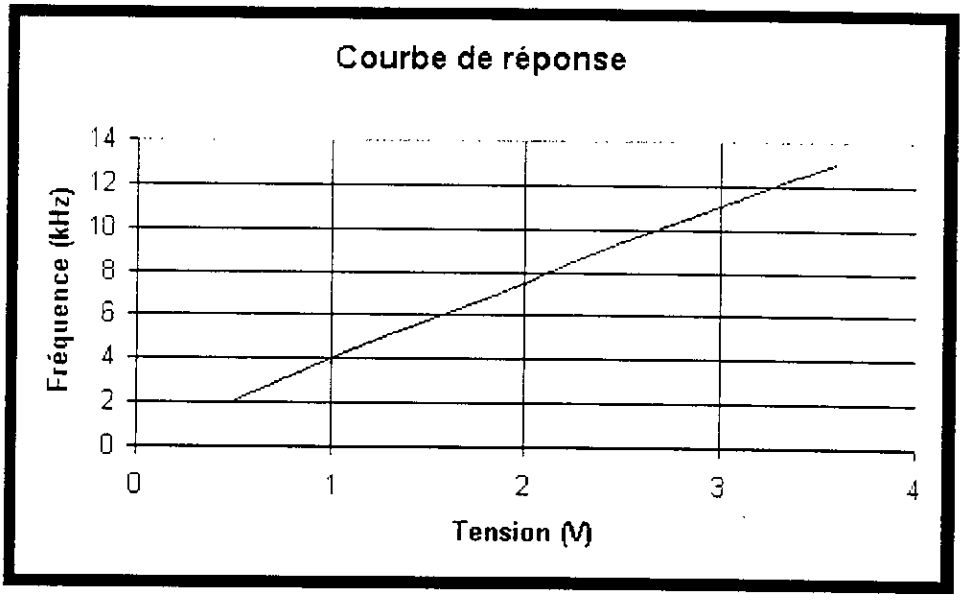


Figure III- 11) Courbe de réponse du VCO.

On choisira donc un signal modulant ayant pour valeur moyenne 2,5 V afin de fixer la fréquence centrale f_0 à environ 9,4 kHz. Cette valeur est obtenue par le réglage d'offset du GBF.

III-6-1- ALLURES DU SIGNAL MODULANT ET DU SIGNAL MODULE FSK

Voie A : (en haut) signal modulant fourni par le GBF.

Calibre : 1V/div.

Voie B : (en bas) signal modulé FSK.

Calibre :1V/div.

Base de temps : 0.5 m s/div.

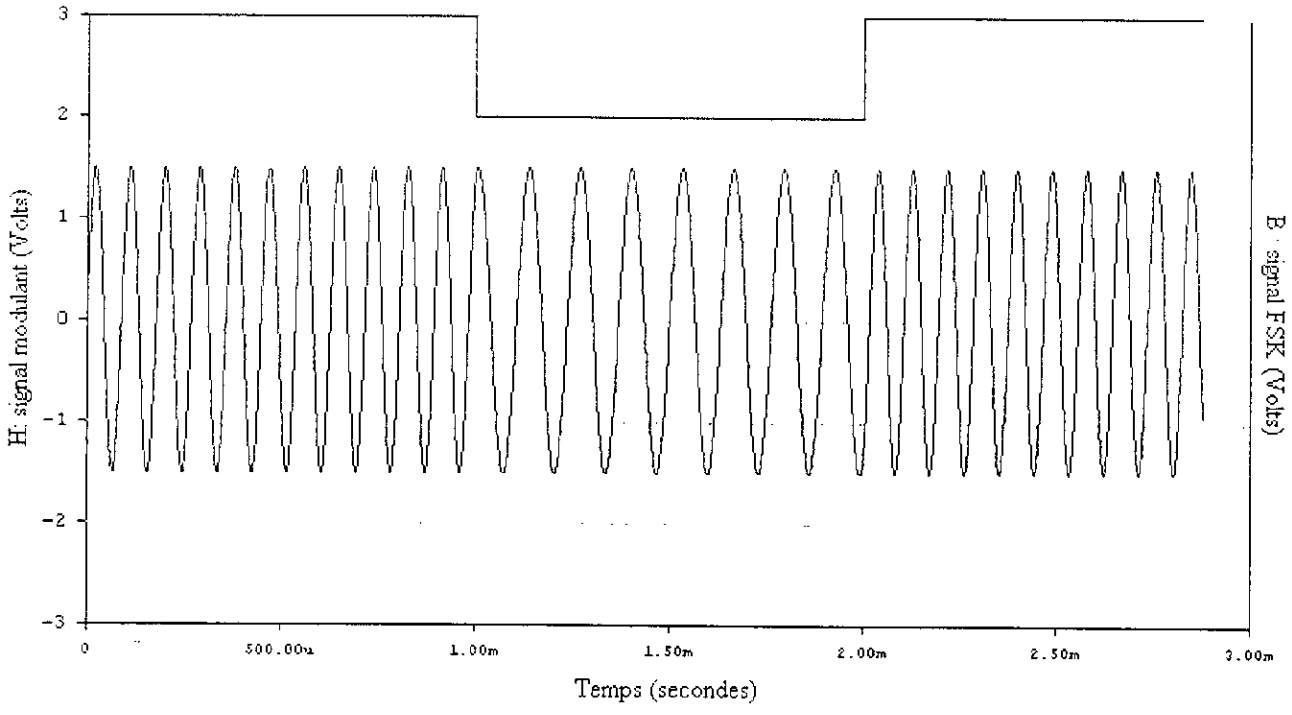


Figure III- 12) Allure du signal modulant (en haut), et du signal modulé FSK (en bas).

On remarque que le signal modulé est à phase continue (il n'y a pas de saut de phase lorsque la fréquence change). Le signal modulant est obtenu à l'aide d'un GBF. L'état bas est à 2 V et l'état haut à 3 V. Les fréquences correspondantes sont :

$f_1 = 7{,}6$ kHz

$f_2 = 11{,}2$ kHz

$f_0 = 9{,}4$ kHz

On peut écrire le signal modulé sous la forme : $S_{fsk} = A \cos(2*\pi (f_0 + k * \Delta f)*t)$

avec k= +1 ou –1, $\Delta f = (f_2 - f_1)/2$.

La Figure III-13) représente le spectre d'amplitude du signal carré modulant de fréquence f = 500 Hz.

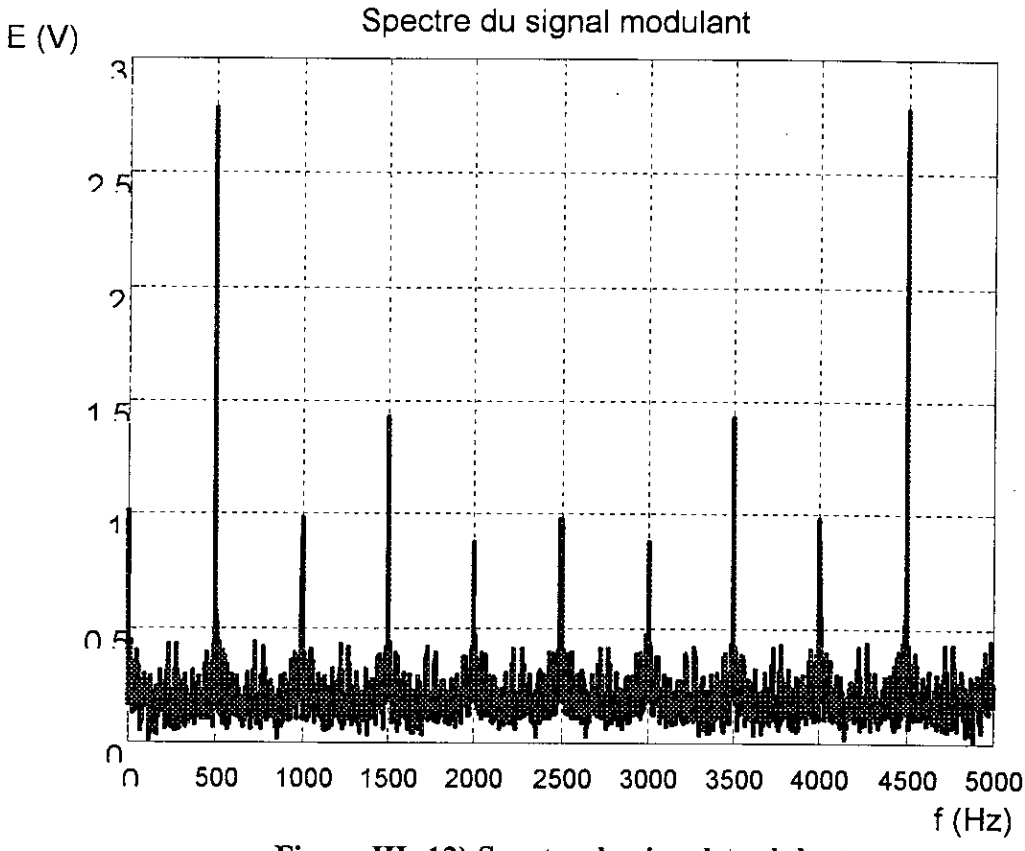


Figure III- 13) Spectre du signal modulant.

On remarque que ce spectre contient une raie fondamentale à la fréquence f = 500 Hz, et d'autres raies espacées de 500 Hz.

Ainsi, le spectre du signal modulé en FSK est représenté par la Figure III-14).

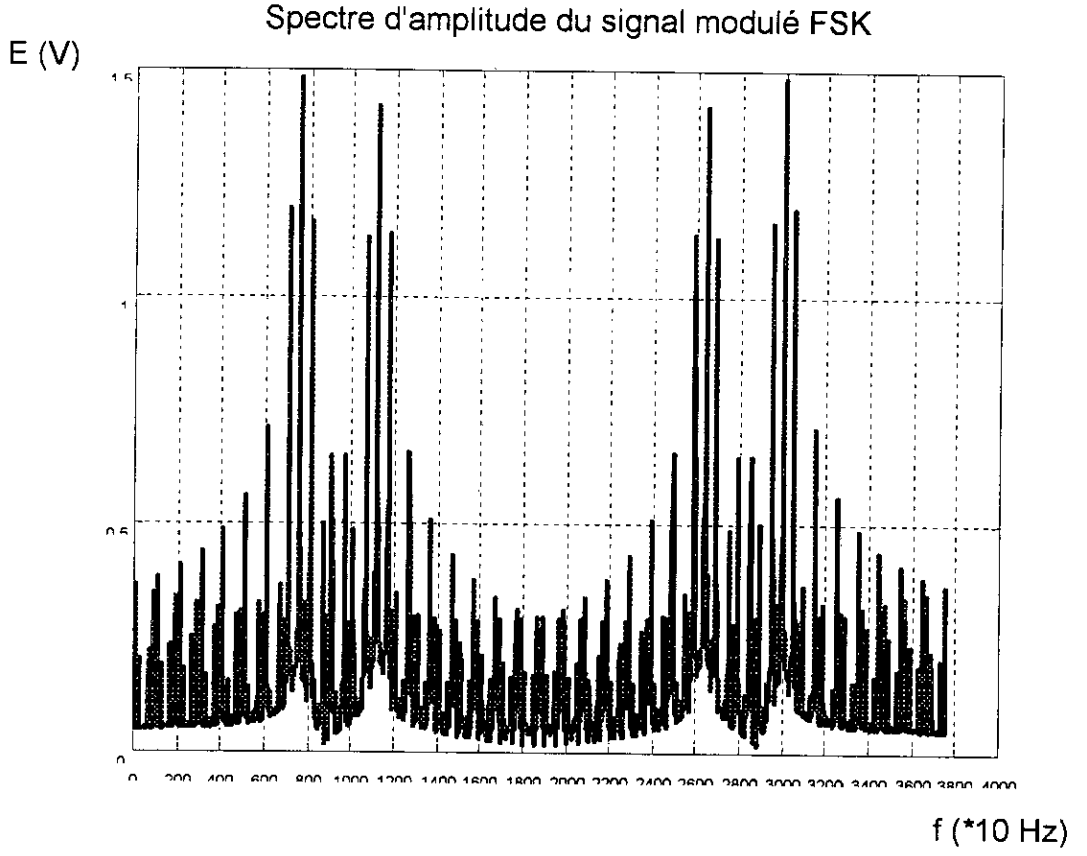


Figure III- 14) Spectre du signal modulé en FSK, pour f_1 = 7.6 kHz, f_2 = 11.2 kHz.

On remarque sur la Figure III-14) les deux raies fondamentales l'une autour de f_1 et l'autre autour de f_2. Autour de ces deux fréquences on remarque des raies espacées de 500 Hz qui correspondent aux composant du signal modulant.

II-6-1-1- SPECTRE DU SIGNAL MODULÉ :

Comme dans toutes transmissions, l'encombrement spectral du signal à transmettre revêt une importance primordiale. La détermination par le calcul du spectre d'un signal modulé en FSK à phase continue est très délicate mais, on peut, expérimentalement, observer quelques résultats importants.

A) CAS OÙ m EST IMPORTANT :

Les figures suivantes représentent le signal modulé et son spectre d'amplitude, dans le cas où : f_1 = 4 kHz ; f_2 = 14 kHz donc m=10.

Voie A : (en haut) signal modulant fourni par le GBF.

Calibre : 1V/div.

Voie B : (en bas) signal modulé FSK.

Calibre :1V/div.

Base de temps : 0.5 m s/div.

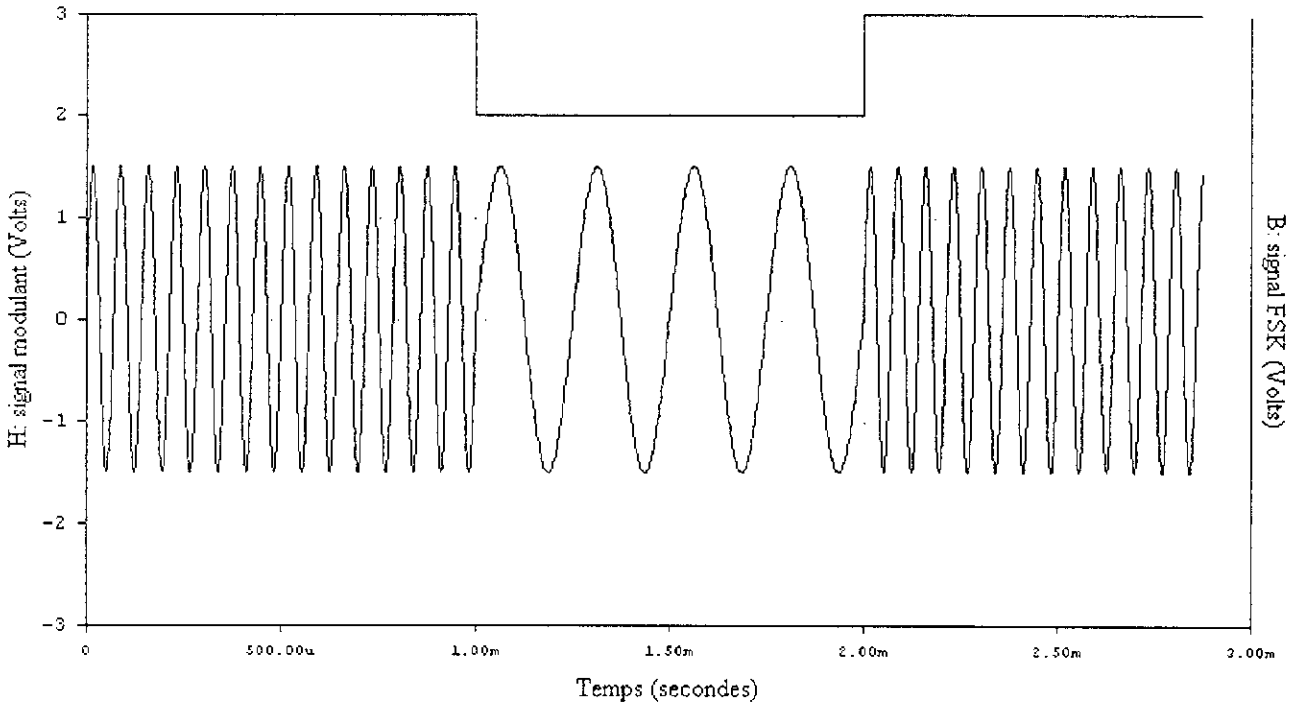


Figure III- 15) Allure du signal modulant et du signal modulé dans le cas où f_1 = 4kHz et f_2 = 14 kHz.

Le spectre d'amplitude du signal modulé représenté sur la Figure III-15) est le suivant :

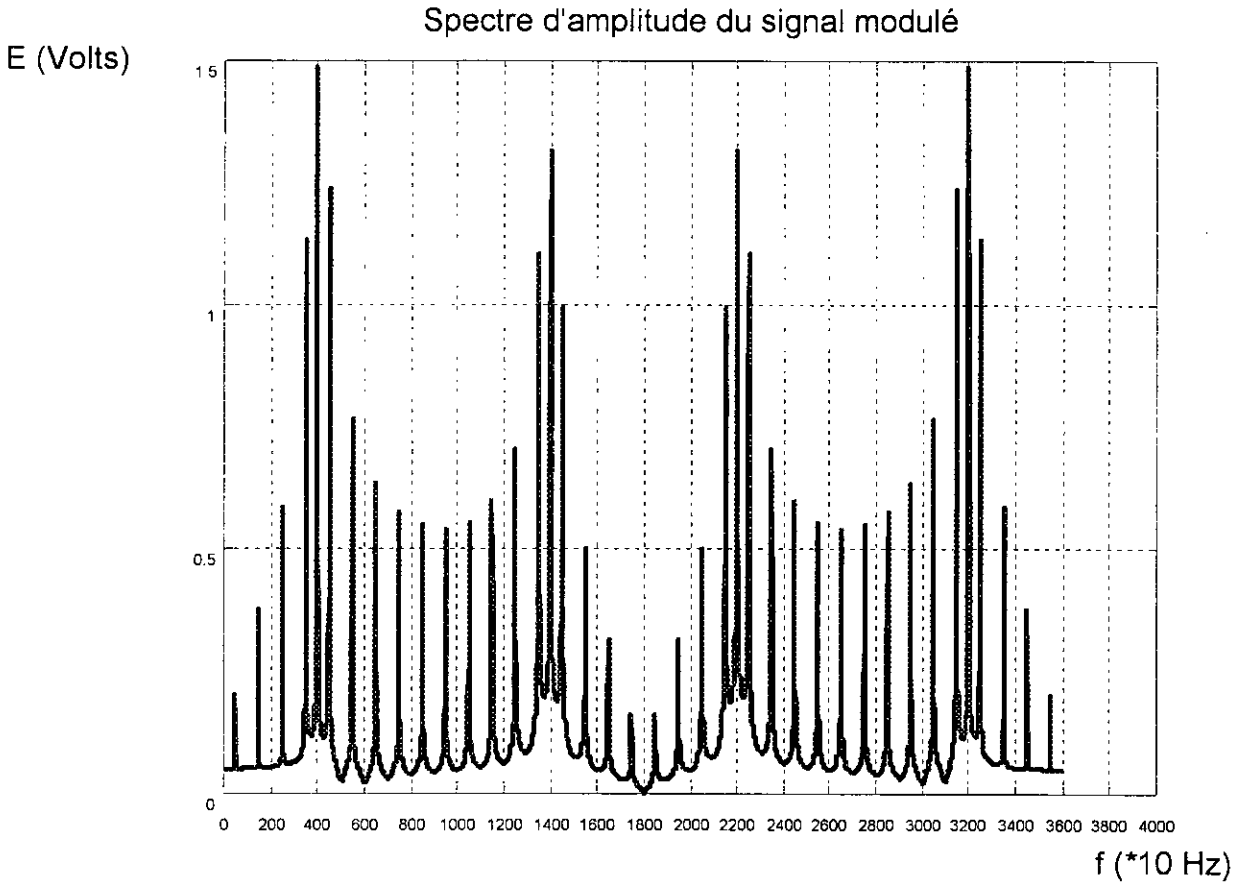


Figure III- 16) Spectre d'amplitude du signal modulé FSK dans le cas où f_1 = 4kHz et f_2 = 14 kHz.

Les deux raies de plus forte amplitude correspondent aux fréquences f_1 et f_2.

Autour de ces deux fréquences, on a des raies espacées de 500Hz, qui correspondent aux composantes du signal modulant.

Remarque :

L'amplitude des différentes raies est exprimée ici en dB. On vérifie bien que la bande de fréquence occupée est : $B = f_2 + F_M - (f_1 - F_M) = 2(\Delta f + F_M)$

où F_M est la fréquence maximale du signal modulant.

Théoriquement, cette bande est infinie car pour un signal carré F_M est infinie.

B) LORSQUE m DIMINUE :

Les figures suivantes représentent le signal modulé et son spectre d'amplitude, dans le cas où : $f_1 = 7.8$ kHz ; $f_2 = 10.8$ kHz donc m=3.

Voie A : (en haut) signal modulant fourni par le GBF.

Calibre : 1V/div.

Voie B : (en bas) signal modulé FSK.

Calibre :1V/div.

Base de temps : 0.5 m s/div.

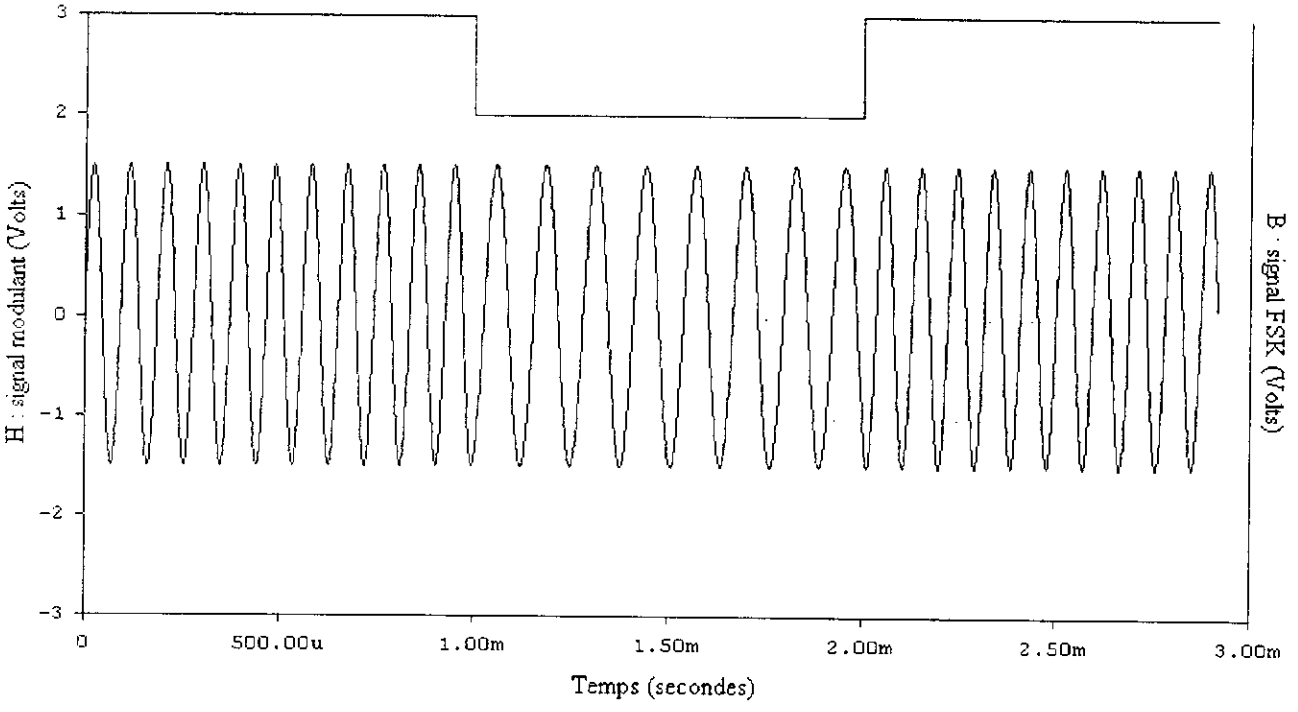


Figure III- 17) Allure du signal modulant et du signal modulé dans le cas où $f_1 = 7.8$ kHz et $f_2 = 10.8$ kHz.

Le spectre d'amplitude du signal modulé représenté sur la Figure III-17) est le suivant:

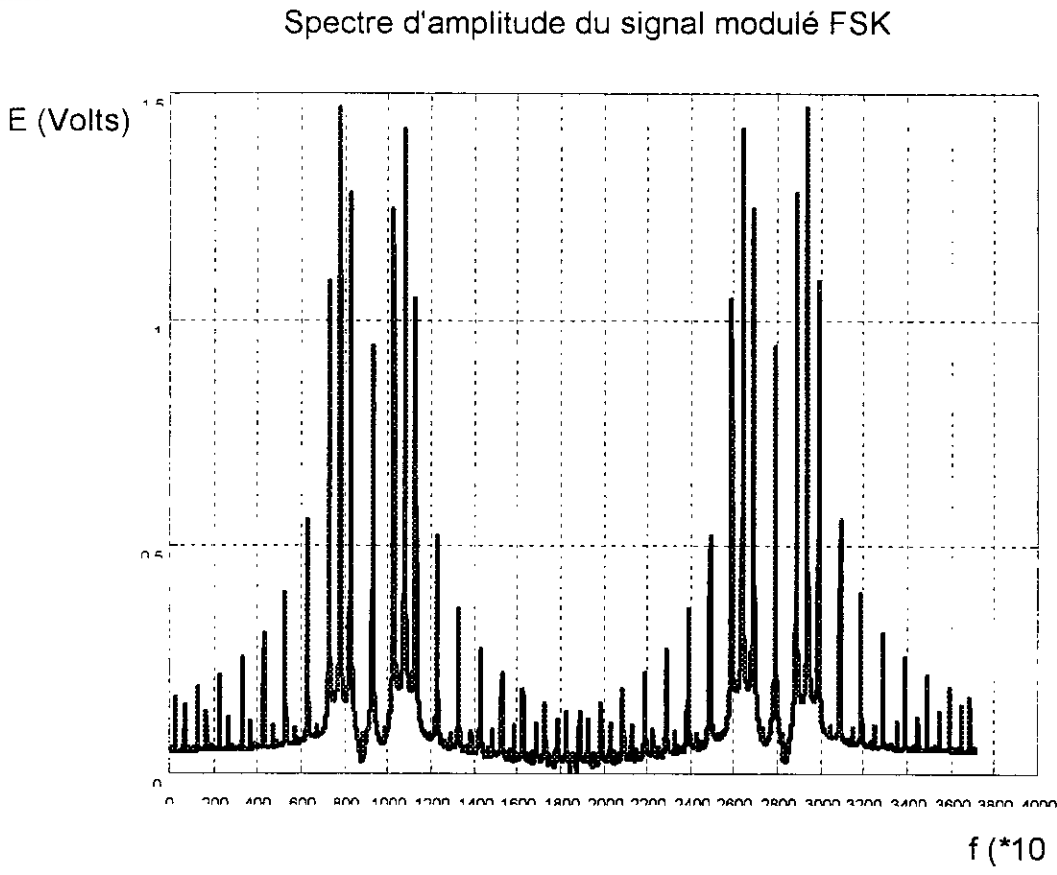


Figure III- 18) Spectre d'amplitude du signal modulé FSK dans le cas de f_1 = 7.8kHz et f_2 = 10.8 kHz.

On remarque que l'encombrement spectral est plus faible que le cas précédente où m = 10.

c) POUR m FAIBLE :

Les figures suivantes représentent le signal modulé et son spectre d'amplitude, dans le cas où : f_1 = 9.3 kHz ; f_2 = 9.8 kHz donc m=0.5.

Voie A : (en haut) signal modulant fourni par le GBF.

Calibre : 1V/div.

Voie B : (en bas) signal modulé FSK.

Calibre :1V/div.

Base de temps : 0.5 m s/div.

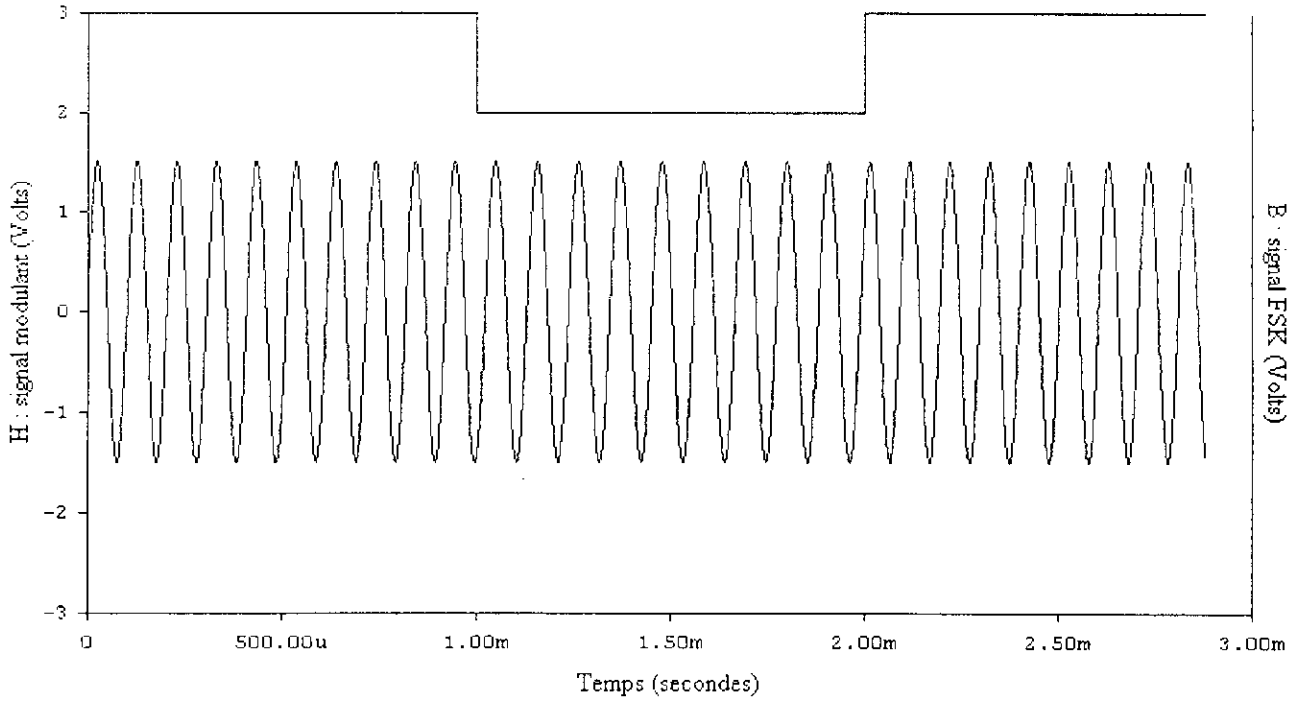


Figure III- 19) Allure du signal modulant et du signal modulé dans le cas où f_1 = 9.3kHz et f_2 = 9.8 kHz

Le spectre d'amplitude du signal modulé représenté sur la Figure III-19) est le suivant:

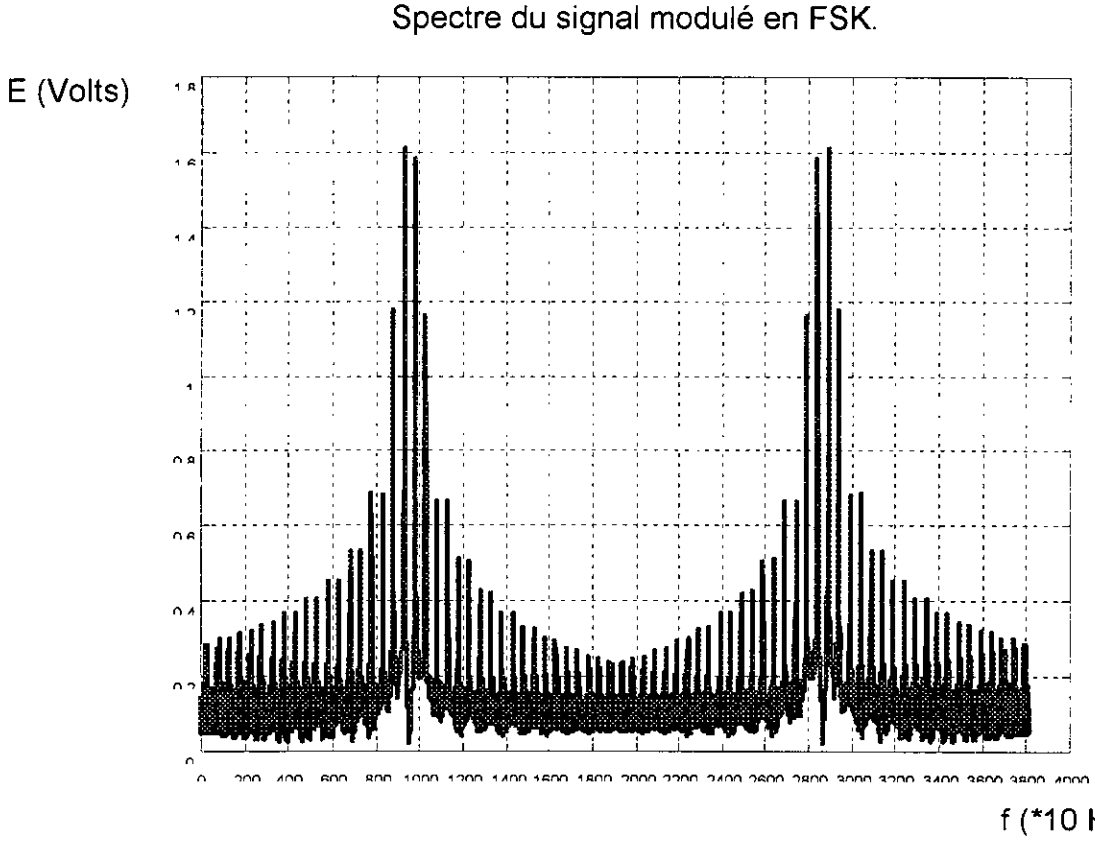


Figure III- 20) Spectre d'amplitude du signal modulé FSK dans le cas de f_1 = 9.3kHz et f_2 = 9.8 kHz.

Ici : f_1 = 9,3 kHz ; f_2 = 9,8 kHz ; m = 0,5.Nous sommes ici plus proche des applications industrielles en ce qui concerne la valeur de m.

Par contre, il est difficile, en visualisant le signal modulé, d'observer les deux fréquences différentes f_1 et f_2. Pour cette raison, nous ne nous intéresserons plus à ce cas par la suite.

Remarque :

Le spectre du signal modulé peut être plus riche en harmoniques. Voici ce que l'on peut observer : Toutes les raies sont ici espacées de 500 Hz. La justification est très délicate, mais on peut retenir qu'il existe des termes d'inter modulation qui font apparaître des raies supplémentaires.

III-7- REGLAGES ET ESSAIS DU DEMODULATEUR FSK ET DU BLOC DE DECISION :

III-7-1- STRUCTURE DE LA BOUCLE A VERROUILLAGE DE PHASE :

Nous allons étudier la manière dont s'accroche une PLL, et définir les plages de verrouillage et de maintient. Prenons l'exemple de la Figure III-21).

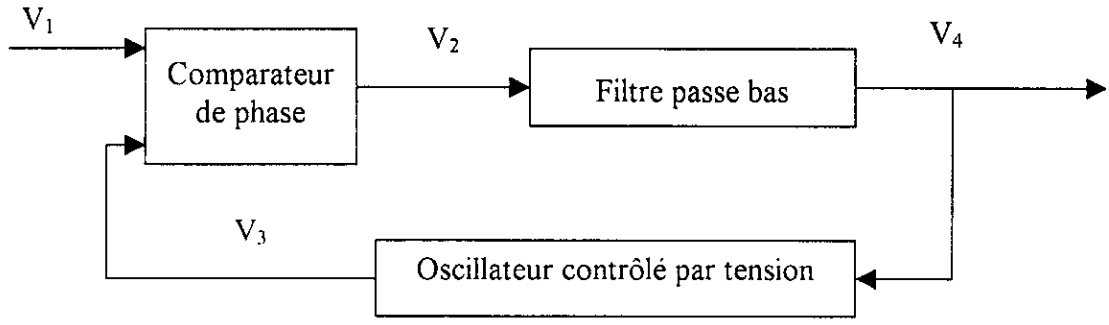


Figure III- 21) Schéma bloc d'une boucle à verrouillage de phase (PLL).

Le générateur d'entré V_1 est un GBF. On choisit le signal et le niveau de celui-ci en tenant compte de la technologie de la PLL. Une fois ce niveau réglé, il suffit de faire varier la fréquence.

Le comparateur de phase présente une caractéristique linéaire V_2 (φ) allant de $-\pi/2$ jusqu'à $\pi/2$ Figure III-22). Pour les valeurs extrêmes de déphasages, la tension d'erreur V_2 (φ) vaut respectivement –5 V et +5 V. Lorsque $\varphi = 0$, nous avons $V_2 = 0$.

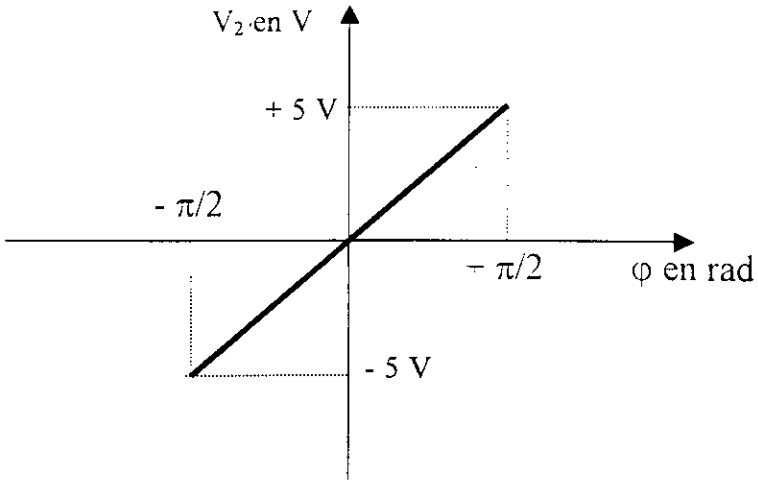


Figure III- 22) Caractéristique V_2 (φ) de la PLL.

Le signal $V_2(t)$ présente des raies spectrales à des fréquences $f_e \pm f_s$ où f_e est la fréquence du signal d'entrée V_1 et f_s est la fréquence du signal de sortie V_3 du VCO. La tension V_4 représente la tension continue ou basse fréquence que laisse passer le filtre passe bas.

Le filtre passe bas Figure III- 23) et Figure III-24) sert à éliminer la composante alternative fe + fs et tous les harmoniques de rang supérieur. Ce filtre est en générale du premier ordre (R2-C2 dans la Figure III-9.).

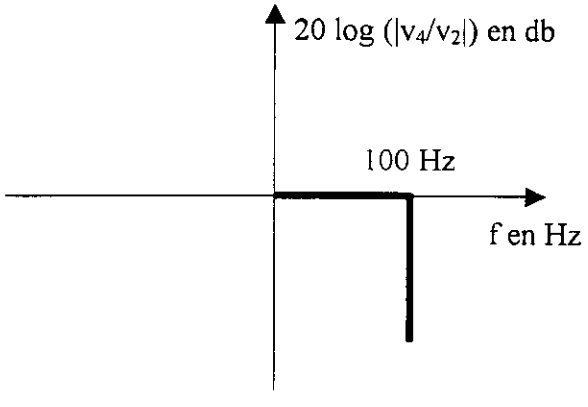


Figure III- 23) Diagramme de Bode d'un filtre idéal.

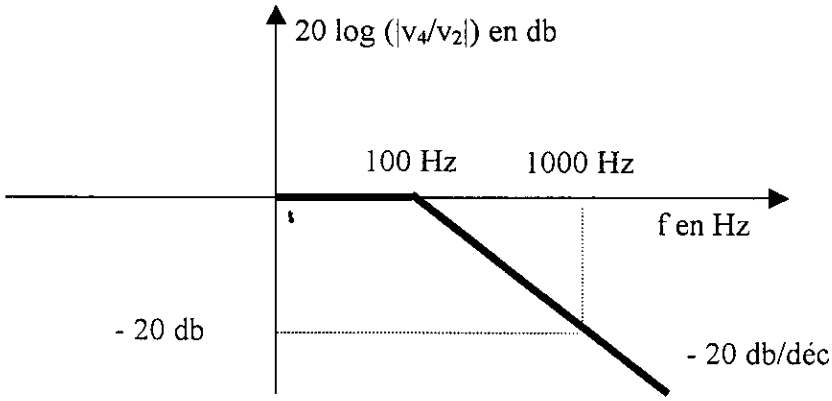


Figure III- 24) Diagramme de Bode d'un filtre réel.

Le VCO est définie par la caractéristique f_s (V_4) de la Figure III-25). La fréquence centrale pour $V_4 = 0$ est $f_0 = 1000$ Hz. La non-linéarité de ce VCO correspond aux fréquences extrêmes : fréquence max : $f_M = 1200$ Hz et fréquence minimale $f_m = 800$ Hz.

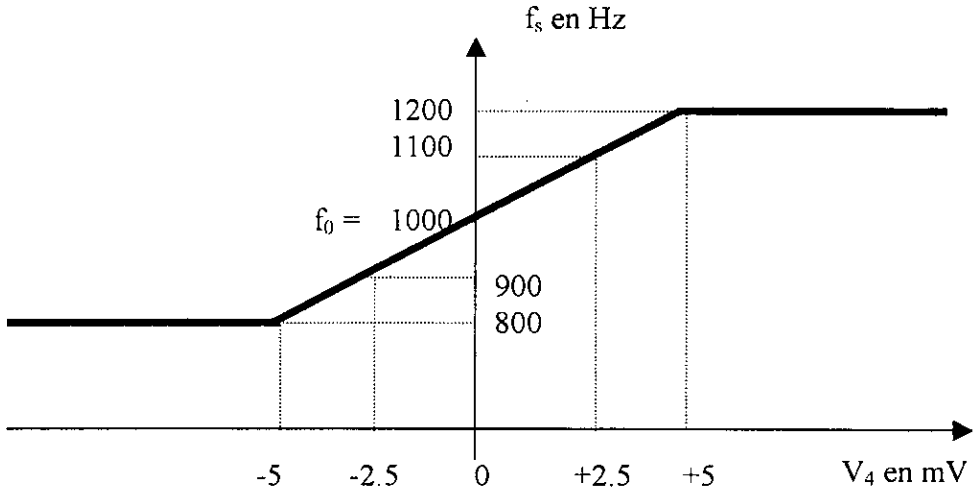


Figure III- 25) Caractéristique f_s (V_4) du VCO.

Remarque :

Un VCO est un montage pouvant travailler dans une gamme de fréquence importante. La fréquence centrale f_0 imposée par la ou les fréquences d'accrochage f_s (plage de verrouillage) est obtenue en choisissant les bonnes valeurs des éléments de R et C associés à l'oscillateur. Ces valeurs sont en général indiqués par des abaques.

Dans ce qui suit nous allons discuter le comportement de la PLL suivant le type de filtre :

Cas de filtre parfait : Figure III-23)

Si la tension d'entrée $V_1 = 0$, le VCO oscille à sa fréquence centrale $f_0 = 1000$ Hz, ainsi $V_3 = v_3 \cos(\omega_0 t)$. Le déphasage entre V_1 et V_3 est $\omega_0 t$. Le filtre passe-bas parfait élimine cette composante, donc $V_4 = 0$. Le VCO oscille bien a sa fréquence centrale $f_0 = 1000$ Hz.

Si V_1 est une composante continue non nulle, le déphasage entre V_1 et V_4 est encore $\omega_0 t$.

Donc le VCO d'une PLL bouclée à entrée nulle, oscille à sa fréquence centrale $f_s = f_0$, pour notre exemple : 1000 Hz.

Si on applique une tension à fréquence croissante on aura les cas suivants :

f_e = 100 Hz,

$V_1 = v_1 \cos(\omega_e t)$ et $V_3 = v_3 \cos(\omega_s t)$

Avec $f_e = 100$ Hz et $f_s = f_0 = 1000$Hz.

Le comparateur de phase donne un signal constitué de deux composantes :

$f_e + f_s = 1100$ Hz

$|f_e - f_s| = 900$ Hz.

Ces deux composantes sont éliminées par le filtre passe bas parfait, donc $V_4 = 0$ V.

100 Hz < f_e < 900 Hz,

La composante basse fréquence évolue de 900 Hz à une fréquence de 100 Hz. Le filtre passe bas parfait fournit toujours $V_4 = 0$ V.

f_e = 900 Hz,

La différence des fréquences $|f_e - f_s|$ donne 100 Hz. Nous somme dans la bande passante du filtre ; V_4 devient une composante variable à 100 Hz. La fréquence de sortie du VCO est modifiée de telle sorte que l'écart $|f_e - f_s|$ diminue. Ceci veut dire que f_s diminue et devient inférieur à 1000 Hz. La tension de sortie V_4 présente une fréquence

de battement qui diminue et une valeur moyenne qui devient négative. Lorsque cette valeur moyenne a atteint –2.5 Volts, la fréquence de sortie devient égale à celle de l'entrée : la boucle est verrouillée. On dit que cette fréquence de 900 Hz est la fréquence de capture ou d'accrochage. Elle est liée à la bande passante du filtre passe bas. En fait, le processus d'accrochage pour f_e = 900 Hz et un filtre parfait serait quasi instantané.

900 Hz ≤ f_e ≤ 1 200 Hz.

La boucle est verrouillée et toute augmentation de la fréquence d'entrée f_e se traduit par une fréquence de sortie f_s identique. Le déverrouillage a lieu lorsque l'on atteint la non-linéarité des éléments du VCO. La Figure III-25) nous indique que la fréquence de sortie maximum f_s est 1200 Hz.

f_e ≥ 1 200 Hz.

La boucle se déverrouille. Le VCO oscille à sa fréquence centrale f_0 de 1000 Hz. Le comparateur de phase fournit une composante à fréquence minimale de $f_e - f_0 > 200$ Hz, le filtre parfait nous donne $V_4 = 0$ ce qui confirme que $f_s = f_0 = 1000$ Hz.

La Figure III-26) nous donne l'évolution de la tension V_4 donc de la fréquence du signal de sortie f_s en fonction de la fréquence du signal d'entrée f_e pour une variation croissante de celle-ci et un filtre passe-bas parfait.

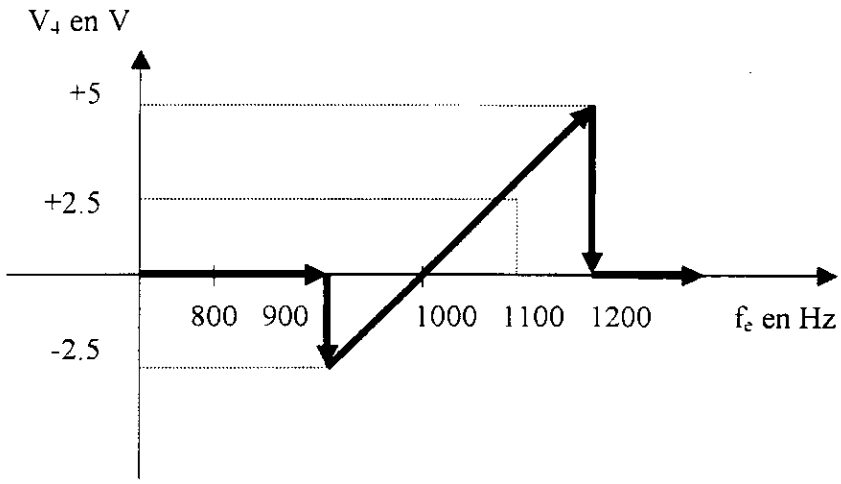


Figure III- 26) Evolution de la tension V_4 en fonction de f_e croissante (cas de filtre idéal).

Si on applique une tension à fréquence décroissante le raisonnement est identique au cas précédent. Tant que la fréquence du signal d'entrée est supérieure à 1100 Hz, la fréquence de battement est $f_0 - f_e$, valeur supérieure à 100 Hz ; donc $V_4 = 0$. Pour $f_e = 1100$ Hz, la fréquence de battement devient égale à 100 Hz et on se trouve dans la bande passante du filtre, V_4 varie tel que f_s augmente et rejoigne f_e (diminution de l'erreur) : il y a verrouillage.

Si la fréquence f_e diminue à partir de 1100 Hz, la PLL reste verrouillée jusqu'à ce que l'on atteigne la non-linéarité de ses éléments. Le déverrouillage a lieu pour 800 Hz. La Figure III-27) nous donne l'évolution de la tension V_4 donc de la fréquence du signal de sortie fs en fonction de la fréquence d'entrée fe pour une variation décroissante de celle-ci et un filtre passe-bas parfait.

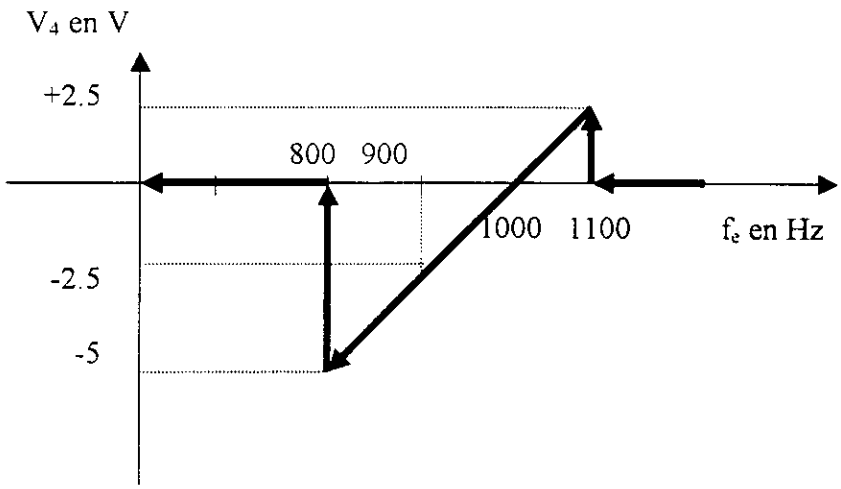


Figure III- 27) Evolution de la tension V_4 en fonction de f_e décroissante (cas de filtre idéal).

Cas de filtre réel : (Figure III-24)

Le filtre passe-bas du premier ordre présente une fréquence de cassure de 100 Hz. L'atténuation est de 20 dB à 1000 Hz.

Pour une fréquence nulle d'entrée, le signal délivré par le comparateur de phase est à une fréquence de 1000 Hz. Le filtre atténuant fortement ce signal $V_2(t)$, il en résulte une tension $V_4(t)$ voisine de 0 donc $f_s \approx f_0 = 1000$ Hz.

Au fur et à mesure que la fréquence f_e du signal d'entrée augmente, la fréquence de battement de $V_2(t)$ diminue et est de moins en moins atténuée ; $V_4(t)$ évolue alternativement avec une amplitude croissante. Il est certain que, tant que f_0 est inférieure à 800 Hz, la PLL ne peut se verrouiller, mais la fréquence du VCO s'agite.

Si l'on considère f_e = 800 Hz, la fréquence de battement est (en prenant $f_s = f_0$) de 200 Hz, soit une atténuation de 6 dB. Nous sommes, pour $V_4(t)$, à une amplitude réduite de moitié par rapport à $V_2(t)$. La PLL pourrait s'accrocher, cette opération dépend des paramètres de la boucle.

En pratique, il est rare que la fréquence d'accrochage corresponde à la limite des non-linéarités, ici 800 Hz pour des fréquences d'entrées croissantes.

La Figure III-28) représente l'évolution de $V_4(t)$ (valeurs prises par cette tension) donc l'évolution de la fréquence du signal de sortie f_s en fonction de la fréquence (croissante) du signal d'entrée. Deux remarques sont à apporter :

- La PLL se verrouille à 860 Hz et se déverrouille à la fréquence limite des non-linéarités, 1200 Hz.

- Lorsque la PLL n'est pas verrouillée, la fréquence de sortie varie, mais d'autant moins que l'on s'éloigne de la fréquence centrale f_0. Pour 800 Hz, V_4 n'atteint que : ± 1,5 V alors qu'il faudrait ±2.5 V pour avoir la condition d'accrochage.

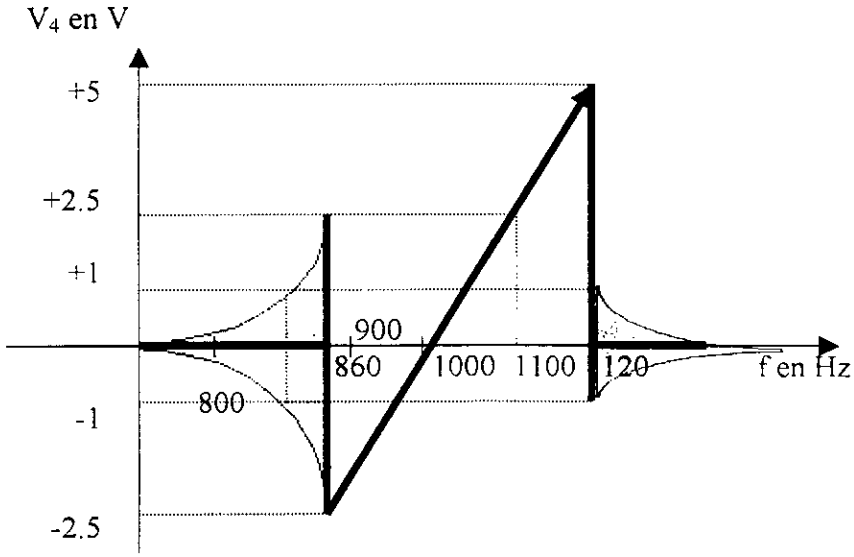


Figure III- 28) Evolution de la tension V_4 en fonction de f_e (cas de filtre réel).

Ces deux remarques sont fondamentales car elles indiquent bien que la condition d'accrochage dépend des éléments de la PLL donc du gain statique et des non-linéarités de la boucle.

Lorsque la boucle est verrouillée, les tensions V_1 et V_3 ont même fréquence. La tension V_2 qui commande l'oscillateur reconstitue donc le signal modulant. Pour que la démodulation soit effective, il faut que la boucle soit verrouillée. L'observation simultanée de V_1 et V_3 permet de connaître la plage de capture de la boucle : la PLL est verrouillée lorsque les deux signaux sont synchrones.

Avec les composants choisis dans le circuit de la Figure III-9) la capture a lieu pour $f_{min} = 4$ kHz et $f_{max} = 13$ kHz.

III-7-2- SIGNAL DEMODULE :

Le signal de sortie du bloc de démodulation est très bruité. Cela vient du fait que les fréquences f_b (du signal modulé), f_1 et f_2 sont très proches. Pour que la démodulation soit convenable, il faut effectuer une mise en forme de ce signal. Le bloc décision permet la mise en forme du signal démodulé.

A la sortie du démodulateur on reconnaît les signaux suivants :

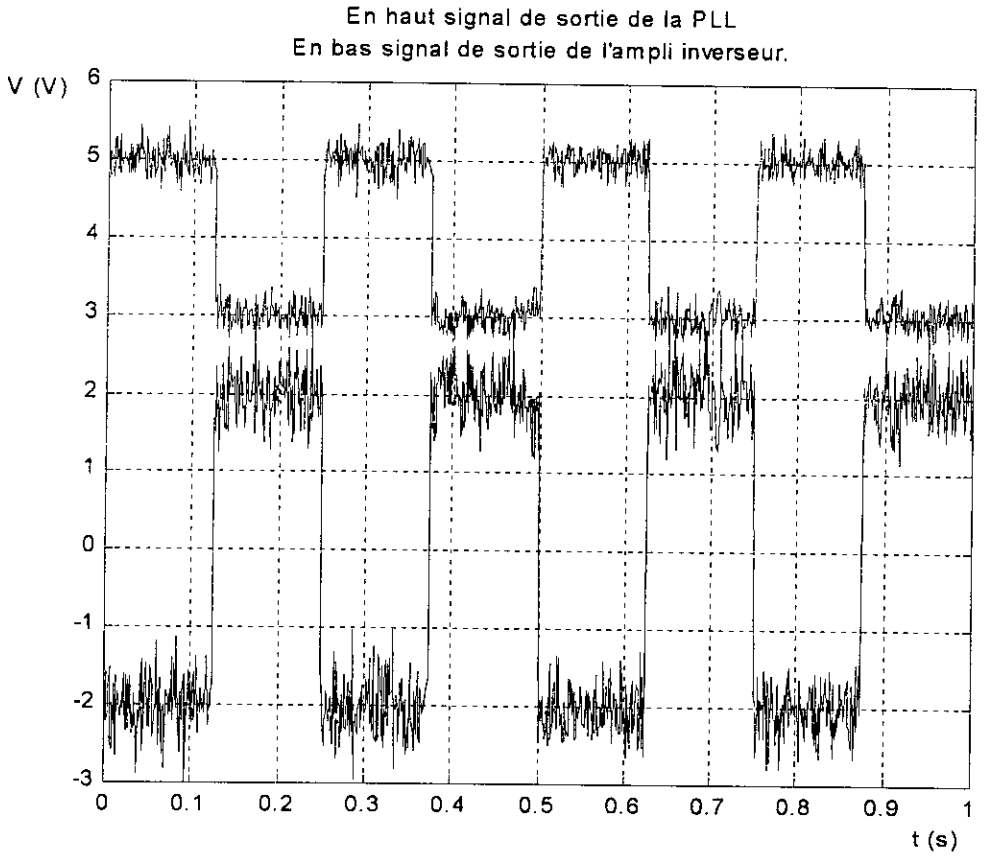


Figure III- 29) Allure du signal de sortie de la PLL (en haut) et du signal de sortie de l'amplificateur inverseur (en bas).

En haut : signal de sortie de la PLL.

Calibre : 1V/division.

En bas : signal de sortie de l'amplificateur.

Calibre : 1V/division.

Base de temps : 8 µs/déviation.

III-7-3- MISE EN FORME DU SIGNAL :

D'après l'allure du signal de sortie de la PLL, on voit qu'un simple comparateur à zéro ne conviendrait pas, puisqu'on aurait passage à 0 du signal de sortie alors que le signal modulant est à l'état 1. Pour éviter ce genre de situation, on utilise un comparateur inverseur à hystérésis dont les seuils sont –1V et +1V.

Ce comparateur est précédé d'un amplificateur inverseur à coefficient d'amplification variable. Le réglage de l'amplification permet d'adapter le signal de sortie de la PLL aux seuils de basculement du comparateur. On évite ainsi les basculements intempestifs du comparateur.

Selon les valeurs des fréquences f_1 et f_2 choisies, la variation de la tension v_2 autour de sa valeur moyenne peut être insuffisante pour permettre le basculement du comparateur. L'amplificateur permet d'augmenter la plage de fonctionnement du bloc de décision.

Signal de sortie du bloc de décision :

On reconnaît dans la courbe suivante le signal de sortie de l'amplificateur.

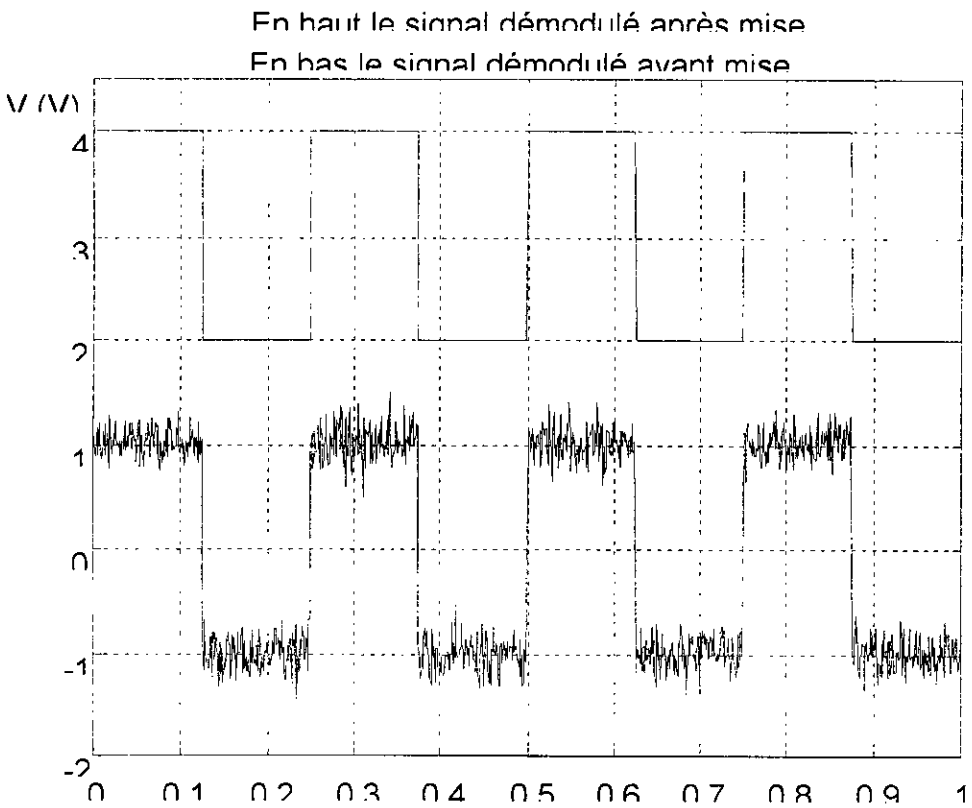


Figure III- 30) Allure du signal de sortie de la PLL (en bas) et le signal de sortie du comparateur à hystérésis (en haut).

Le signal carré est le signal de sortie du comparateur.

Calibres :

1V/division pour la tension de sortie de l'amplificateur.

1V/division pour le signal de sortie du comparateur (carré).

Base de temps : 8 µs/division.

On a donc bien reconstitué le signal modulant. La démodulation est bien effectuée.

III-8- CONCLUSION :

Les trois montages proposés dans ce chapitre à savoir : le modulateur FSK, le démodulateur FSK et le circuit de décision, permettent d'avoir un système de communication numérique de qualité moyenne, car les circuits intégrés utilisés (l'ICL8038 et le 4046) sont des circuits à usage général et non plus des circuits spécialisés dans la modulation FSK de haute qualité (par exemple le modulateur XR2228 et le démodulateur XR2212).

Nous avons choisi l'ICL8038 dans le bloc de modulation FSK car il est facile à mettre en œuvre avec des éléments externes très réduits, et il supporte des tensions d'alimentation jusqu'à ±18 Volts, ce qui permet de commander facilement la fréquence centrale de modulation f_0.

Le 4046 est une PLL très commode pour la plupart des applications utilisant la détection FM ou FSK, car il offre des performances acceptables.

Le bloc de décision permet la mise en forme du signal démodulé et ce pour avoir un signal carré plus idéalisé.

CHAPITRE IV :

MESURES ET INTERPRETATIONS.

CHAPITRE IV : MESURES ET INTERPRÉTATIONS

IV-1- INTRODUCTION :

Dans ce chapitre nous allons essayer de mettre en œuvre le micro-contrôleur 68HC11, l'émetteur FSK et le récepteur FSK dont le fonctionnement a été étudié dans les chapitres précédents.

Ce système de communication numérique (de laboratoire) va nous permettre de mesurer la fiabilité du canal de transmission numérique. Nous allons donc essayer d'émettre et de recevoir des fichiers de données, et faire des comparaisons entre les fichiers émis et les fichiers reçus et en déduire la fiabilité de la transmission.

Dans les expériences envisagées, nous allons faire varier deux paramètres essentiels : la vitesse de transmission du signal émis et la longueur du canal.

Le 68HC11 permet de transmettre des données cadencées selon plusieurs vitesses (9600, 4800, 2400 bauds, ...), ce qui donne la possibilité de varier la fréquence de transmission.

Quant à la longueur de la liaison on peut la faire varier en utilisant plusieurs câbles de longueurs différentes.

IV-2- MATÉRIEL ET LOGICIEL UTILISES :

Dans notre projet nous avons réalisé un système de communication numérique à base de microcontrôleur 68HC11 et des circuits de modulation et de démodulation FSK à savoir : l' ICL8038 (un VCO), et le 4046 (une PLL).

Concernant le logiciel utilisé, nous avons écrit trois types de programmes :

- Programme en assembleur pour le microcontrôleur 68HC11 permettant l'émission de symboles vers le modulateur FSK.
- Programme de réception au niveau de micro-ordinateur écrit en langage Turbo C++ 3.0, ce programme permet la réception et le stockage de données reçus dans un fichier.
- Programmes sous Matlab 5.3, où nous avons exploité la puissance et la facilité de mise en œuvre de celui-ci et surtout en ce qui concerne la gestion de la carte son. Ces programmes nous ont permet de visualiser les signaux reçus sur le micro-ordinateur et le calcul des probabilités d'erreur par symbol.

IV-3- ORGANIGRAMMES DES DIFFÉRENTS PROGRAMMES RÉALISÉS :

Ce projet nous a permis de manipuler trois types de programmation : la programmation linéaire, la programmation processurelle, et la programmation en assembleur.

La programmation en assembleur nous a permis de connaître et de manipuler le système au plus bas niveau. Malgré la difficulté que présente la programmation en assembleur que ce soit dans la phase de rédaction du code ou bien dans la phase de mise au point et de chargement des programmes dans le microcontrôleur, on peut dire qu'elle est très importante car elle nous permet d'étudier l'architecture de bas niveau des systèmes informatiques.

La programmation linéaire (Turbo C++ 3.0) c'est d'écrire du code qui suit un organigramme précis, où on ne peut pas intervenir au cours de l'exécution de ce programme pour faire varier quelques variables. Donc nous devons attendre que le programme nous fournis des résultats et ensuite nous les interprétons. Si ces résultats sont acceptables nous les gardons sinon nous devons réexécuter le programme à nouveau avec d'autres paramètres. Ce type de programmation est une caractéristique essentielle des logiciels qui travaillent sous DOS.

Il est à noter que la programmation sous DOS demande beaucoup d'attention et de reconnaissance, car le système d'exploitation DOS donne de vastes possibilités au programmeur pour qu'il gère le micro-ordinateur selon ces besoins, ce qui n'est pas le cas sous le système d'exploitation WINDOWS.

Dans la programmation processurelle on ne parle pas d'organigramme linéaire car il ne s'agit pas de séquences d'instructions qui s'exécutent l'une après l'autre, mais c'est plutôt des processus qui s'exécutent (virtuellement) en même temps. Dans ce type de programmation nous pouvons intervenir au cours de l'exécution du programme et faire changer quelques variables sans interrompre le fonctionnement du programme. On parle donc de la programmation WINDOWS. Dans ce type de programmation le programmeur n'a accès qu'aux fonctions délivrés par le système d'exploitation WINDOWS, car celui-ci constitue une couche logiciel entre l'utilisateur et le matériel.

Dans la programmation WINDOWS nous n'avons pas à connaître comment gérer le matériel, nous n'avons qu'à utiliser des fonctions prêtes à l'usage.

Au contraire sous DOS la connaissance du matériel à une importance primordiale ; si on ne sait pas le type du matériel, son mode de fonctionnement, ses registres internes, son adresse matériel, ..., on ne peut pas le gérer correctement.

Une petite lecture des programmes fournis en annexes fait preuve de ce que nous venons de dire, on peut citer à titre d'exemple la fonction de Matlab : **daqhwinfo** qui permet de détecter le type de la carte son, nous ne savons pas comment elle fonctionne mais nous l'utilisons seulement, et elle nous fournit directement le nombre et le type de cartes son installés dans le micro-ordinateur. Sous DOS, pour détecter le type de l'UART il nous faut comprendre le fonctionnement de celle-ci et écrire une fonction (**ser_uarttype**) plein de tests et d'instructions compliquées, et après un certains nombre d'essais et de mise au point que nous arrivons à le faire fonctionner correctement.

IV-3.1 ORGANIGRAMME D'EMISSION :

Le programme émet.asm est écrit en langage assembleur de 68HC11, il suit l'organigramme suivant :

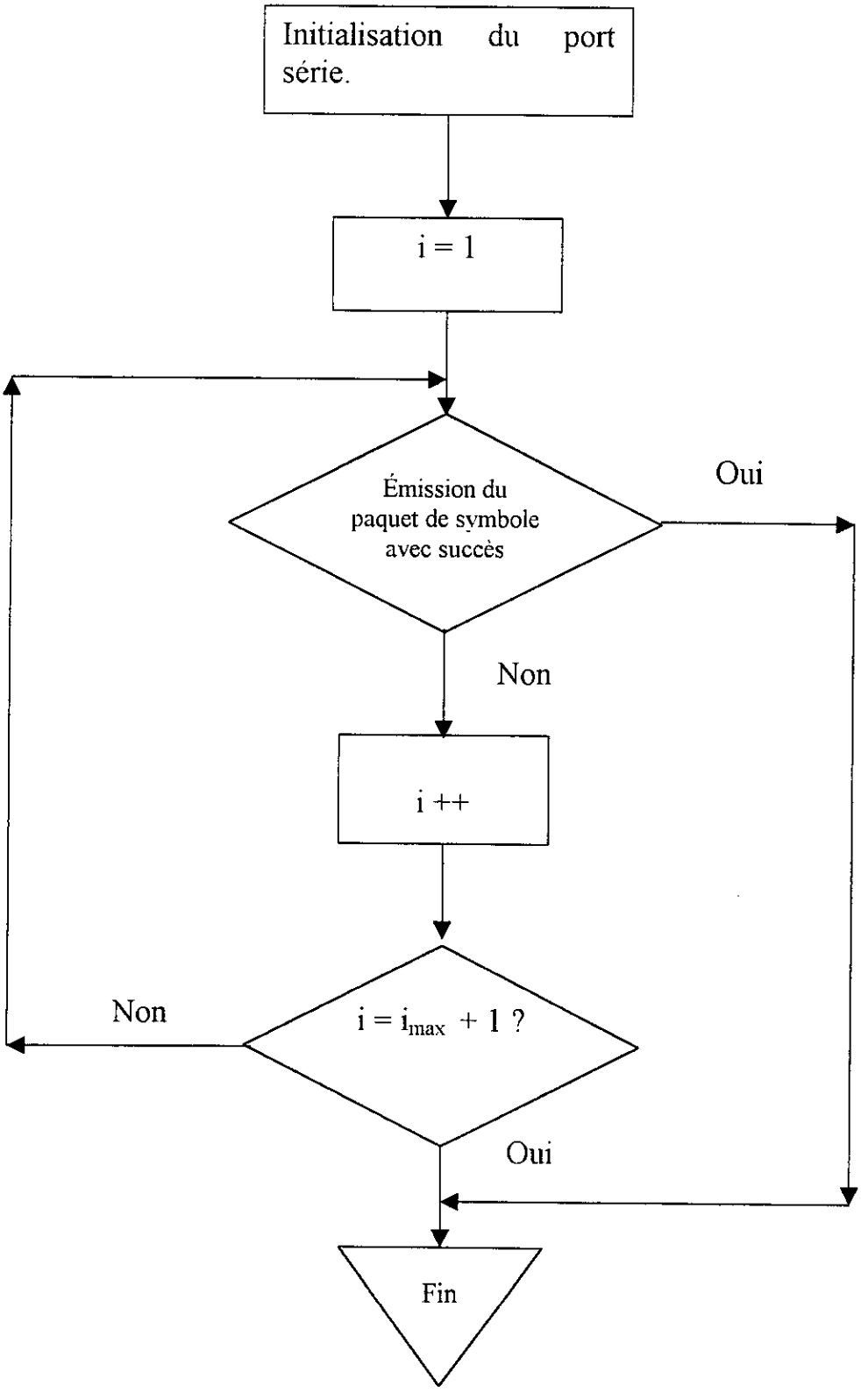


Figure IV- 1) Organigramme d'émission d'un paquet de données (côté microcontrôleur.)

Dans l'EPROM du microcontrôleur 68HC11, nous allons écrire une des séquences de symboles. Pour notre application nous avons choisi aléatoirement l'alphabet suivant :

x_1 = a, x_2 = A, x_3 = b, x_4 = R, x_5 = e, x_6 = h, x_7 = K, x_8 = m, x_9 = n, x_{10} = I.

Chaque symbole étant émis 1000 fois. Le programme emet.asm va initialiser le port série du microcontrôleur 68HC11 et ensuite envoyer les données vers le modulateur FSK.

IV-3.2 ORGANIGRAMMES DE RECEPTION :

IV-3.2.1 PROGRAMME EN TURBO C++ 3.0 :

L'organigramme suivant illustre le déroulement du programme recept.c :

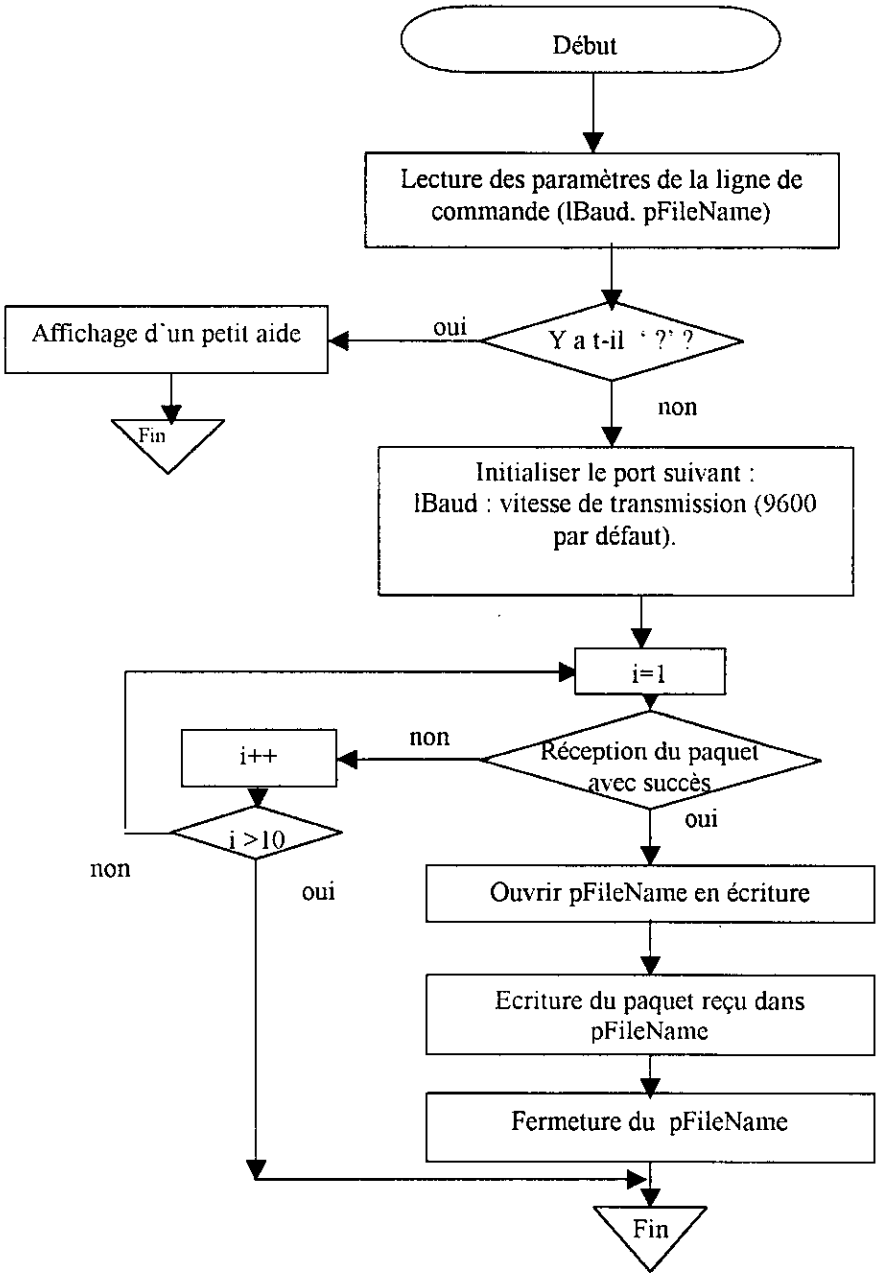


Figure IV- 2) Organigramme du programme recept.c. qui permet la réception des données à partir du port série du PC.

Ce programme s'exécute à partir de l'invite MS-DOS, en lui fournit en argument le nom de fichier où nous devons charger les données reçues et la vitesse de transmission.

Le nom de fichier contenant les informations suivantes :

Un préfixe, le caractère à recevoir, la longueur de la liaison, et la vitesse de communication, par exemple : « **sacc** ».

s est le préfixe,

a le caractère à recevoir,

c longueur 1 m,

c vitesse 600 bauds.

Le programme lit donc le nom de fichier à partir de la ligne de commande, reçoit les données et les charge dedans.

IV-3.2.2 PROGRAMMES SOUS MATLAB 5.3 :

Sous Matlab 5.3 nous avons conçu plusieurs programmes qui sont les suivants :

himfcngen.m : (Figure IV-4) ce programme permet la génération de signaux sinusoïdaux et carré que nous avons utilisés dans la phase de test de bon fonctionnement des circuits d'émission et de réception. Ce programme permet de générer des sinusoïdes et des signaux carrés dont nous pouvons modifier les paramètres : amplitude, fréquence, offset, fréquence d'échantillonnage et rapport cyclique. Les signaux générés seront véhiculés par la carte son du micro-ordinateur.

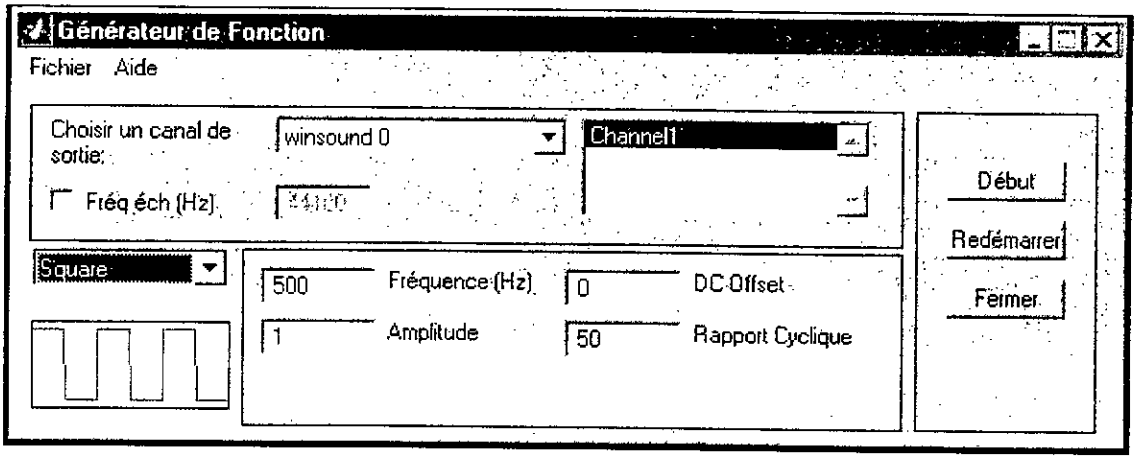


Figure IV- 3) Interface graphique du programme himfcngen.m.

himscope.m : (Figure IV-5) ce programme joue le rôle d'un oscilloscope logiciel. Il nous a servi dans la phase de mise au point des circuits d'émission et de réception. Son interface graphique permet de visualiser les signaux en provenance de la carte son du micro-ordinateur.

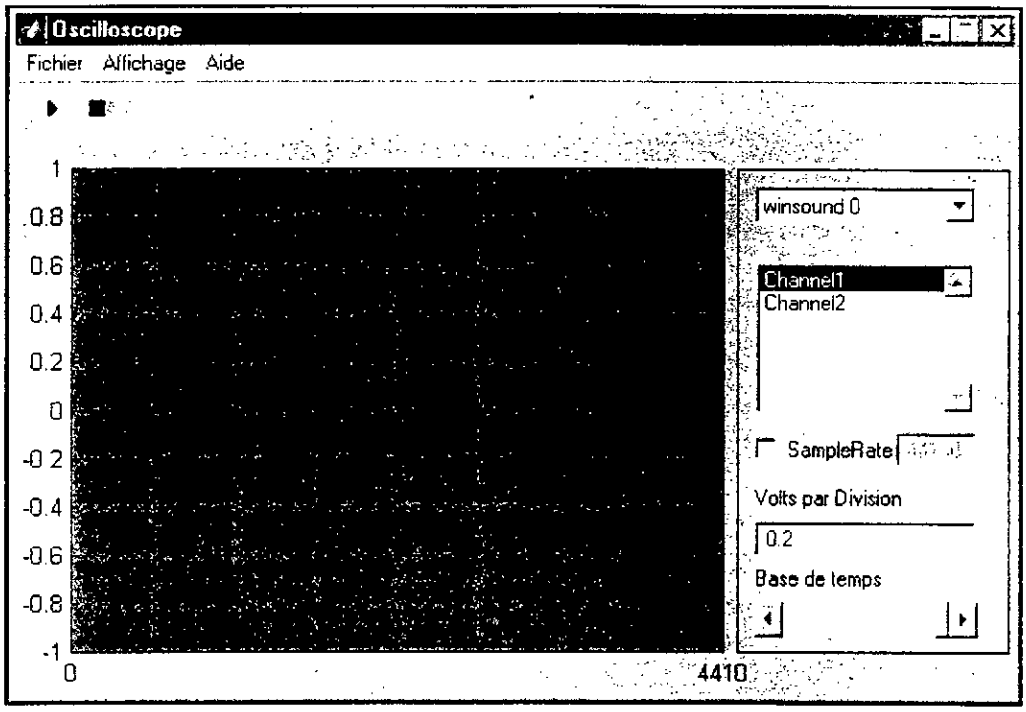


Figure IV- 4) Interface graphique du programme himscope.m.

himrecfft.m : ce programme permet l'affichage de la transformée de fourrier des signaux en provenance de la carte son, et ce en temps réel. Ce programme nous a servi dans la phase de test de bon fonctionnement des circuits de l'émetteur et le récepteur FSK, car la visualisation de signaux dans le domaine fréquentiel permet une interprétation plus claire que dans le domaine temporel.

himentropie.m : ce programme permet la mesure de la probabilité d'erreur des symboles durant la transmission car il effectue une comparaison entre le fichier émis et le fichier reçu et en déduit les résultats.

IV-4- MESURES ET INTERPRETATIONS :

Nous avons effectué des mesures et ce en jouant à chaque fois sur l'un des paramètres : la vitesse de transmission et la longueur de la liaison.

Donc nous fixons la vitesse de transmission et nous faisons varier la longueur du câble de transmission.

La procédure de mesure étant illustrée par l'organigramme suivant :

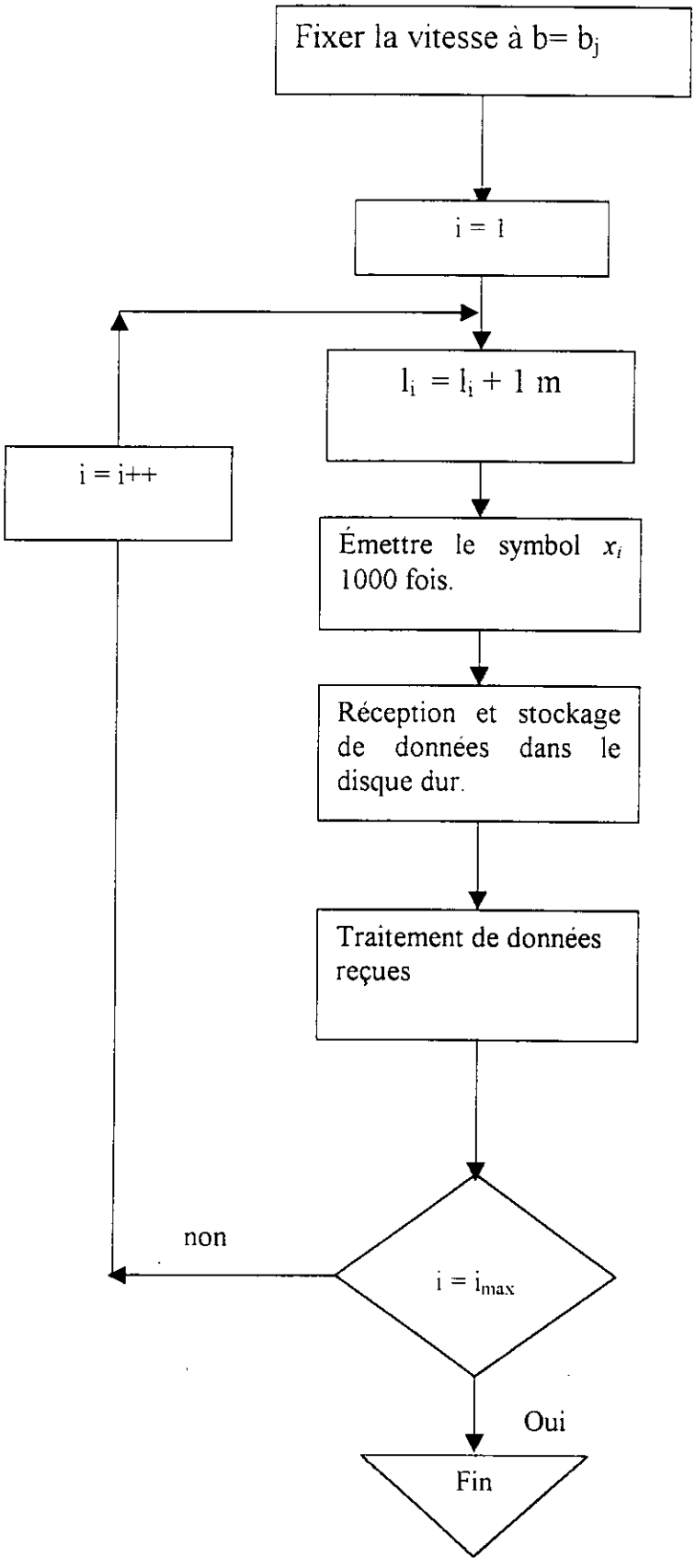


Figure IV- 5) Organigramme de procédure de mesure de la fiabilité de notre liaison numérique.

Les tableaux suivants résument les résultats obtenus pour les divers essais.

b = 600 bauds.

***l* = 1 mètre,**

Symboles	Nbe de symboles émis	Nbr de symboles erronés	Probabilité d'erreur par symbole Nbr/Nbe
x_1 = a	1000	26	0.026
x_2 = A	1000	15	0.015
x_3 = b	1000	12	0.012
x_3 = R	1000	9	0.009
x_5 = e	1000	15	0.015
x_6 = h	1000	4	0.004
x_7 = K	1000	14	0.014
x_8 = m	1000	5	0.005
x_9 = n	1000	8	0.008
x_{10} = I	1000	10	0.01

Tableau IV- 1) Résultats obtenus pour b = 600 bauds et *l* = 1 mètre.

***l* = 1.5 mètre,**

Symboles	Nbe de symboles émis	Nbr de symboles erronés	Probabilité d'erreur par symbole Nbr/Nbe
x_1 = a	1000	30	0.03
x_2 = A	1000	21	0.021
x_3 = b	1000	15	0.015
x_3 = R	1000	11	0.011
x_5 = e	1000	15	0.015
x_6 = h	1000	7	0.007
x_7 = K	1000	20	0.02
x_8 = m	1000	12	0.012
x_9 = n	1000	11	0.011
x_{10} = I	1000	13	0.013

Tableau IV- 2) Résultats obtenus pour b = 600 bauds et *l* = 1.5 mètre.

l = 2 mètres,

Symboles	Nbe de symboles emis	Nbr de symboles erronés	Probabilité d'erreur par symbole Nbr/Nbe
x_1 = a	1000	36	0.036
x_2 = A	1000	31	0.031
x_3 = b	1000	23	0.023
x_3 = R	1000	17	0.017
x_5 = e	1000	22	0.022
x_6 = h	1000	15	0.015
x_7 = K	1000	21	0.021
x_8 = m	1000	11	0.011
x_9 = n	1000	7	0.007
x_{10} = I	1000	19	0.019

Tableau IV- 3) Résultats obtenus pour b = 600 bauds et l = 2 mètres.

b = 1200 bauds.

l = 1 mètre,

Symboles	Nbe de symboles émis	Nbr de symboles erronés	Probabilité d'erreur par symbole Nbr/Nbe
x_1 = a	1000	36	0.036
x_2 = A	1000	20	0.02
x_3 = b	1000	21	0.021
x_3 = R	1000	17	0.017
x_5 = e	1000	33	0.033
x_6 = h	1000	18	0.018
x_7 = K	1000	16	0.016
x_8 = m	1000	21	0.021
x_9 = n	1000	23	0.023
x_{10} = I	1000	11	0.011

Tableau IV- 4) Résultats obtenus pour b = 1200 bauds et l = 1 mètre.

l = 1.5 mètre,

Symboles	Nbe de symboles émis	Nbr de symboles erronés	Probabilité d'erreur par symbole Nbr/Nbe
x_1 = a	1000	30	0.03
x_2 = A	1000	25	0.025
x_3 = b	1000	32	0.032
x_3 = R	1000	28	0.028
x_5 = e	1000	30	0.03
x_6 = h	1000	25	0.025
x_7 = K	1000	32	0.032
x_8 = m	1000	30	0.03
x_9 = n	1000	20	0.02
x_{10} = I	1000	25	0.025

Tableau IV- 5) Résultats obtenus pour b = 1200 bauds et l = 1.5 mètre.

l = 2 mètres,

Symboles	Nbe de symboles émis	Nbr de symboles erronés	Probabilité d'erreur par symbole Nbr/Nbe
x_1 = a	1000	35	0.035
x_2 = A	1000	33	0.033
x_3 = b	1000	30	0.03
x_3 = R	1000	31	0.031
x_5 = e	1000	25	0.025
x_6 = h	1000	34	0.034
x_7 = K	1000	40	0.04
x_8 = m	1000	34	0.034
x_9 = n	1000	25	0.025
x_{10} = I	1000	24	0.024

Tableau IV- 6) Résultats obtenus pour b = 1200 bauds et l = 2 mètres,

b = 2400 bauds.

***l* = 1 mètre,**

Symboles	Nbe de symboles émis	Nbr de symboles erronés	Probabilité d'erreur par symbole Nbr/Nbe
x_1 = a	1000	45	0.045
x_2 = A	1000	41	0.041
x_3 = b	1000	35	0.035
x_3 = R	1000	38	0.038
x_5 = e	1000	44	0.044
x_6 = h	1000	31	0.031
x_7 = K	1000	28	0.028
x_8 = m	1000	25	0.025
x_9 = n	1000	40	0.04
x_{10} = I	1000	23	0.023

Tableau IV- 7) Résultats obtenus pour b = 2400 bauds et *l* = 1 mètre.

***l* = 1.5 mètre,**

Symboles	Nbe de symboles émis	Nbr de symboles erronés	Probabilité d'erreur par symbole Nbr/Nbe
x_1 = a	1000	45	0.045
x_2 = A	1000	35	0.035
x_3 = b	1000	53	0.053
x_3 = R	1000	52	0.052
x_5 = e	1000	47	0.047
x_6 = h	1000	31	0.031
x_7 = K	1000	53	0.053
x_8 = m	1000	33	0.033
x_9 = n	1000	41	0.041
x_{10} = I	1000	54	0.054

Tableau IV- 8) Résultats obtenus pour b = 2400 bauds et *l* = 1.5 mètre.

l = 2 mètres,

Symboles	Nbe de symboles émis	Nbr de symboles erronés	Probabilité d'erreur par symbole Nbr/Nbe
x_1 = a	1000	40	0.04
x_2 = A	1000	54	0.054
x_3 = b	1000	64	0.064
x_3 = R	1000	62	0.062
x_5 = e	1000	55	0.055
x_6 = h	1000	55	0.055
x_7 = K	1000	63	0.063
x_8 = m	1000	65	0.065
x_9 = n	1000	42	0.042
x_{10} = I	1000	50	0.05

Tableau IV- 9) Résultats obtenus pour b = 2400 bauds et l = 2 mètres.

Pour une bonne interprétation de résultats il vaut mieux les présenté par des courbes :

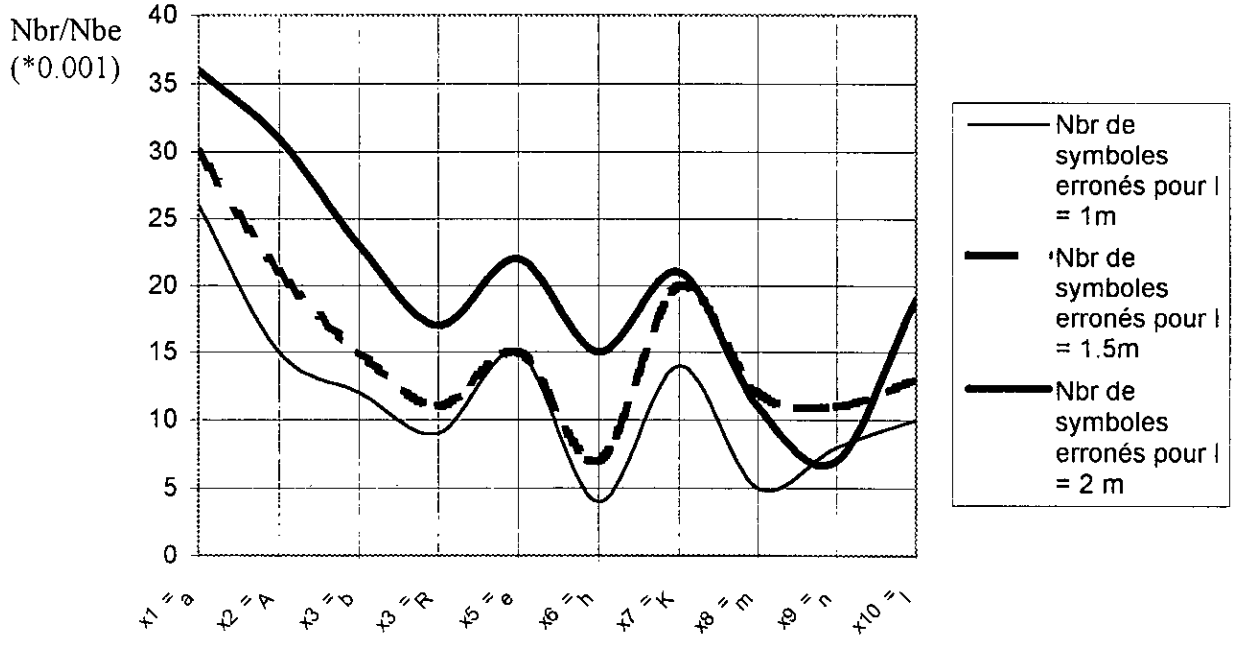


Figure IV- 6) Courbe représentant la probabilité d'erreur par symbole pour b = 600 bauds.

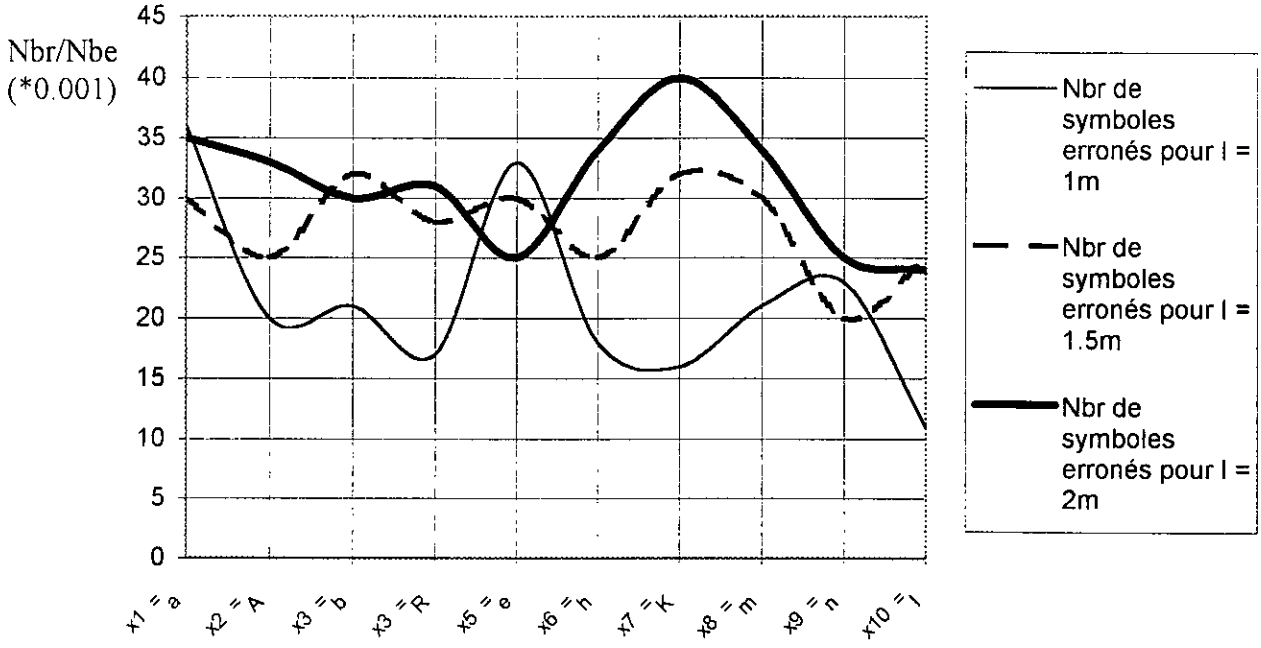


Figure IV- 7) Courbe représentant la probabilité d'erreur par symbole pour b = 1200 bauds.

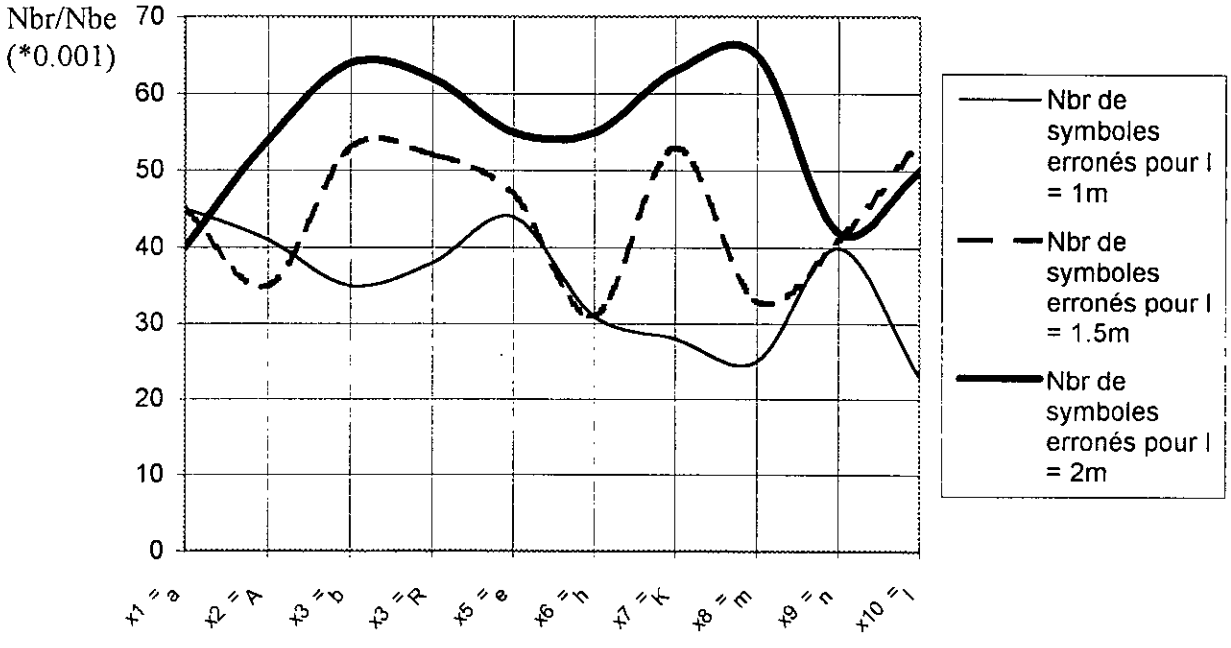


Figure IV- 8) Courbe représentant la probabilité d'erreur par symbole pour b = 2400 bauds.

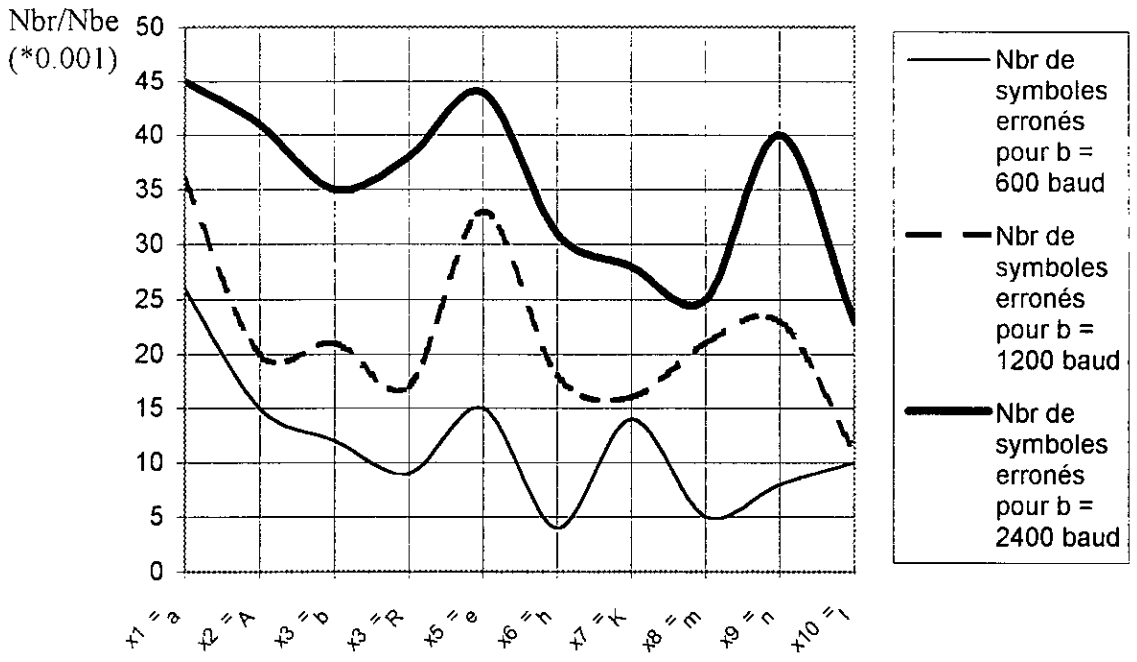


Figure IV- 9) Courbe représentant la probabilité d'erreur par symbole pour *l* = 1 m.

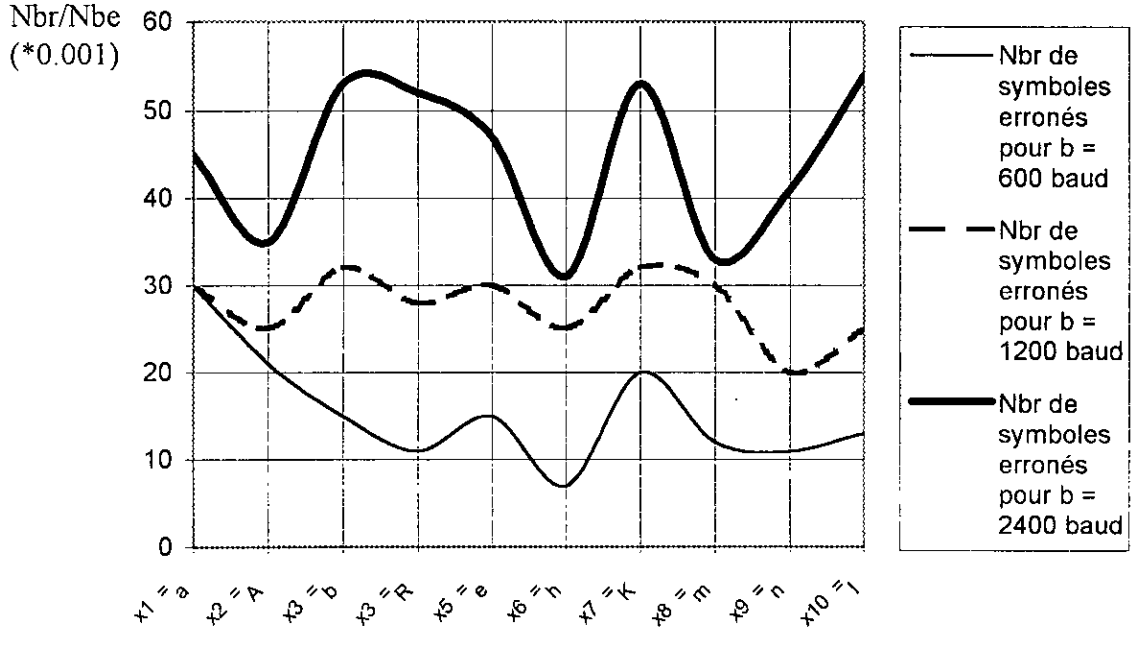


Figure IV- 10) Courbe représentant la probabilité d'erreur par symbole pour *l* = 1.5 m.

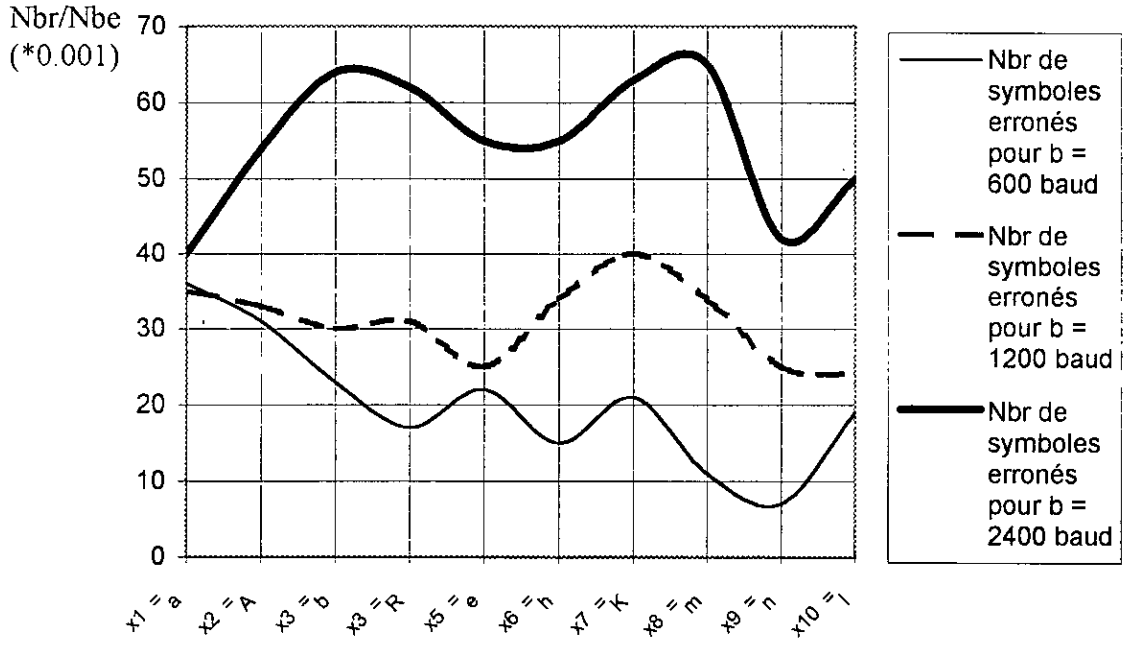


Figure IV- 11) Courbe représentant la probabilité d'erreur par symbole pour *l* = 2 m.

D'après les courbes précédentes on peut noter les remarques suivantes :

- L'augmentation de la longueur de la liaison cause une probabilité d'erreur plus grande.
- L'augmentation de la vitesse de transmission engendre une augmentation de la probabilité d'erreur.

Nous pouvons justifier la première remarque par le fait que le canal de transmission à une certaine résistivité qui fait chuter la tension, ce qui diminue la puissance du signal transmis par le canal, donc produit plus d'erreurs.

La deuxième remarque est liée aux caractéristiques du modulateur lui-même d'un côté et la bande passante du canal d'un autre côté, car l'augmentation de la vitesse de transfert implique l'augmentation de la fréquence de l'information, donc une occupation spectrale plus importante qui risque dé passer la capacité du canal, et d'induire par la suite des erreurs de communication.

On peut remarquer aussi que la probabilité d'erreur ne dépasse pas 0.065 (le symbole « m » du tableau IV-9.) ce qui permet de dire que la liaison est fiable.

IV-5- CONCLUSION :

On peut dire que la liaison numérique est relativement fiable, car la probabilité d'erreur ne dépasse pas 0.065 avec des vitesses allant de 600 à 2400 bauds et des longueurs variant entre 1 à 2 mètres.

CONCLUSION

CONCLUSION

Nous avons accompli dans notre travail la réalisation d'un système de communication numérique qui met en œuvre plusieurs cartes, à savoir : carte d'émission à base de microcontrôleur, carte de modulation, carte de démodulation et carte de décision.

Ce système a été mis en œuvre pour étudier la fiabilité d'une liaison numérique en fonction de deux paramètres essentiels qui sont : la longueur du canal de transmission et la fréquence de l'information.

Nous avons donc essayé de mesurer la probabilité d'erreur de dix caractères choisis aléatoirement, les résultats étant acceptables compte tenu du matériel disponible.

Le système que nous avons réalisé, apporte une certaine originalité résidant dans la mise en œuvre de la méthode de modulation et de démodulation FSK et la mesure de fiabilité d'une liaison numérique.

Il reste cependant, que notre travail doit faire l'objet d'amélioration sur les plans matériel et logiciel.

BIBLIOGRAPHIE

BIBLIOGRAPHIE :

[1] « Réalisation d'une interface récepteur micro-ordinateur »

Thèse de fin d'études de l'ENP,

Auteurs : DALI M. et HIMA A.

[2] « Contribution à l'étude d'une chaîne de télémesure : application aux capteurs- émetteurs alimentés par champs électromagnétique haute fréquence. »

Thèse de doctorat de l'INPL, 1991.

Auteur : MEHENNI M.

[3] « Boucles à verrouillage de phase »

Auteur : GIRARD M.

Edition : McGRAW-HILL 1991.

[4] « Advanced electronic communications systems »

Auteur : TOMASI W.

Edition : PRENTICE-HALL, 1987.

[5] « Fondements de la théorie de la transmission de l'information. »

Auteur : ALEXANDRU Spătaru.

Edition : Presses polytechniques romandes.

[6] « Microprocessor, microcontroller and pereferal data », volume II.

Auteur : Motorola,1988.

[7] HCMOS single-chip microcontroller.

Auteur : Motorola, 1988.

[8] MC 68HC11EVB, evaluation board user's manual.

Auteur : Motorola, 1986.

[9] MC 68HC11EVM, evaluation module user's manual.

Auteur : Motorola, 1989.

TABLES DES MATIERES

TABLE DES MATIERE DU CHAPITRE I :

GENERALITES SUR LA TRANSMISSION DES DONNEES NUMERIQUES

TABLE DES MATIERE DU CHAPITRE II :

MISE EN ŒUVRE DU KIT DE DEVELOPPEMENT MC68HC11 DE MOTOROLA

TABLE DES MATIERES DU CHAPITRE III :

ETUDE ET REALISATION D'UN EMETTEUR ET D'UN RECEPTEUR FSK

TABLE DES MATIÈRES DU CHAPITRE IV
MESURES ET INTERPRÉTATIONS

TABLES DES FIGURES

TABLE DES FIGURES DU CHAPITRE I:

TABLE DES FIGURES DU CHAPITRE II :

TABLE DES FIGURES DU CHAPITRE III

TABLE DES FIGURES DU CHAPITRE IV :

TABLES DES TABLEAUX

TABLE DES TABLEAUX DU CHAPITRE II :

TABLE DES TABLEAUX DU CHAPITRE IV :

ANNEXES

LE PROGRAMME EMET.ASM

```
**TEST DE LA LIAISON SCI
SCSR   EQU $2E
SCSR2  EQU $2D
SCDR   EQU $2F
BAUD   EQU $2B
REGBAS EQU $1000
BPS    EQU $1F4                ;1000 fois chaque symbole

       ORG    $ca00
       LDX    #REGBAS
       LDY    #BPS
       LDAA   #%00110100      ;vit de 600 baud : 00110100
*      LDAA   #%00110011      ;vit de 1200 baud
*      LDAA   #%00110010      ;vit de 2400 baud
       STAA   BAUD,X
       BSET   SCSR2,X %00001100
*DEB   BSR    RECEP
       LDAA   #$AA
DEB    BSR    ENVOI
       INY
       BEQ    FIN
       BRA    DEB

**RECEPTION D'UN CHAR
*RECEP LDAA   SCSR,X
*      ANDA   #$20
*      BEQ    RECEP
*      LDAA   SCDR,X
*      RTS

**EMISSION D'UN CHAR
ENVOI ASL    SCSR,X
       BCC    ENVOI
       STAA   SCDR,X
       RTS
FIN    END
```

LE PROGRAMME RECEPT.CPP

```
#include <dos.h>
#include <process.h>
#include <conio.h>
#include <stdio.h>
#include <hserie.h>
#include <recept.h>

#define SER_TIMEOUT 0x8000
#define SER_9600 1200L



/*************************************************************************/
/* SER_UARTType : D,termine le type du chip UART                         */
/*-----------------------------------------------------------------------*/
/* Entr,e : iSerPort      - Adresse de base de l'interface … tester      */
/* Sortie : 0 (NOSER)     - Chip UART introuvable                        */
/*          1 (INS8250)   - Chip INS8250 ou INS8250-B Chip               */
/*          2 (NS16450)   - INS8250A, INS82C50A, NS16450,NS16C450        */
/*          3 (NS16550A)  - Chip NS16550A                                */
/*          4 (NS16C552)  - Chip NS16C55                                 */
/*************************************************************************/
INT SER_UARTType( INT iSerPort )
{
  /*- Tester fonctionnalit,s de base ------------------------------ */

  outportb( iSerPort + SER_LINE_CONTROL, 0xAA );
                  /* Diviseur-Latch positionn, */
  if( inp( iSerPort + SER_LINE_CONTROL ) != 0xAA ) return NOSER;

  outportb( iSerPort + SER_DIVISOR_MSB, 0x55 );
                   /* Description du diviseur */
  if( inp( iSerPort + SER_DIVISOR_MSB ) != 0x55 ) return NOSER;

  outportb( iSerPort + SER_LINE_CONTROL, 0x55 );
                    /* Efface le diviseur */
  if( inp( iSerPort + SER_LINE_CONTROL ) != 0x55 ) return NOSER;

  outportb( iSerPort + SER_IRQ_ENABLE, 0x55 );
  if( inp( iSerPort + SER_IRQ_ENABLE ) != 0x05 ) return NOSER;

  outportb( iSerPort + SER_FIFO, 0 );  /* Efface FIFO et IRQ */
  outportb( iSerPort + SER_IRQ_ENABLE, 0 );
  if( inp( iSerPort + SER_IRQ_ID ) != 1 ) return NOSER;

  outportb( iSerPort + SER_MODEM_CONTROL, 0xF5 );
  if( inp( iSerPort + SER_MODEM_CONTROL ) != 0x15 ) return
NOSER;

  outportb( iSerPort + SER_MODEM_CONTROL, SER_MCR_LOOP );      /* Boucle */
  inp( iSerPort + SER_MODEM_STATUS );
  if( ( inp( iSerPort + SER_MODEM_STATUS ) & 0xF0 ) != 0 )
return NOSER;

  outportb( iSerPort + SER_MODEM_CONTROL, 0x1F );
  if( ( inp( iSerPort + SER_MODEM_STATUS ) & 0xF0 ) != 0xF0 )
      return NOSER;

  outportb( iSerPort + SER_MODEM_CONTROL, SER_MCR_DTR | SER_MCR_RTS
```

```
);

  outportb( iSerPort + SER_SCRATCH, 0x55 ); /* le registre Scratch existe?*/
  if( inp( iSerPort + SER_SCRATCH ) != 0x55 ) return INS8250;
  outportb( iSerPort + SER_SCRATCH, 0 );

  outportb( iSerPort + SER_FIFO, 0xCF );          /* le FIFO est pr,sent?  */
  if( ( inp( iSerPort + SER_IRQ_ID ) & 0xC0 ) != 0xC0 ) return NS16450;
  outportb( iSerPort + SER_FIFO, 0 );
           /* le registre Alternate-Function est pr,sent? */
  outportb( iSerPort + SER_LINE_CONTROL, SER_LCR_SETDIVISOR );
  outportb( iSerPort + SER_2FUNCTION, 0x07 );
  if( inp( iSerPort + SER_2FUNCTION ) != 0x07 )
  {
    outportb( iSerPort + SER_LINE_CONTROL, 0 );
    return NS16550A;
  }
  outportb( iSerPort + SER_LINE_CONTROL, 0 );   /* R,initialiser */
  outportb( iSerPort + SER_2FUNCTION, 0 );
  return NS16C552;
}

/*************************************************************************/
/* SER_Init : Initialiser l'interface s,rie                               */
/*                                                                        */
/*------------------------------------------------------------------------*/
/* Entr,e : iSerPort - Adresse de base de l'interface s,rie               */
/*                     qu'il faut initialiser.                            */
/*          lBaud    - Vitesse  ( von 1 - 115200 )                        */
/*          bParams  - Masque de bits des autres param□tres               */
/*                     (… voir Bits SER_LCR_...-Bits)                     */
/* Sortie :   TRUE  - initialisation de l'interface r,ussie               */
/*            FALSE - Interface introuvable                               */
/*************************************************************************/
INT SER_Init( INT  iSerPort, LONG lBaudRate, BYTE bParams )
{ WORD uDivisor;  if( SER_UARTType( iSerPort ) != NOSER )
 {                                  /* Calcul du diviseur de baud */
    uDivisor = ( WORD )( SER_MAXBAUD / lBaudRate );
    outportb( iSerPort + SER_LINE_CONTROL, /* Permet l'acc□s du diviseur */
       inportb( SER_LINE_CONTROL ) | SER_LCR_SETDIVISOR );
                /* Positionne le diviseur de baud */
    outportb( iSerPort + SER_DIVISOR_LSB, LOBYTE( uDivisor ) );
    outportb( iSerPort + SER_DIVISOR_MSB, HIBYTE( uDivisor ) );
                     /* Emp^che son acc□s */
    outportb( iSerPort + SER_LINE_CONTROL,
       inportb( SER_LINE_CONTROL ) & ~SER_LCR_SETDIVISOR);
 /*- Pour fixer les autres param□tres attendre le positionnement en  */
 /* arri□re des taux de baud Latch parce que l'op,ration efface tous */
 /*                                  les param□tres de l'interface */
      /* Fixer les param□tres de transmission - sauf pour la vitesse */
    outportb( iSerPort + SER_LINE_CONTROL, bParams );
       /* Lecture d'un octet, pour placer les erreurs ,ventuellement */
       /*                                    pr,sentes en arri□re */
    inp( iSerPort + SER_TXBUFFER );
    return TRUE;
  }
  return FALSE;
}

/*************************************************************************/
/* SER_IsWritingPossible :                                                */
```

```
/*                    L'interface, peut-elle envoyer encore un octet ?*/
/**-------------------------------------------------------------------**/
/* Entr,e : iSerPort - Port de base de l'interface … tester           */
/* Sortie : == 0 : Il ne faut pas envoyer l'octet.                    */
/*           != 0 : l'interface est pr^te … ,mettre                   */
/**------------------------------------------------------------------ **/
/*Info : Il ne faut pas utiliser une interface s,rie pour envoyer     */
/*       des octets dans les cas suivants :                           */
/*       1. Un octet re‡u n'a pas encore ,t, appel, par l'interface   */
/*       2. Une demande d',mission n'a pas encore ,t, ex,cut,e        */
/*********************************************************************/
INT SER_IsWritingPossible( INT iSerPort )
{
  return ( !inp( iSerPort + SER_LINE_STATUS ) & SER_LSR_TSREMPTY);
}

/*********************************************************************/
/* SER_IsModemStatusSet : V,rifier l',tat d'entr,e des lignes         */
/*-------------------------------------------------------------- ---*/
/* Entr,e : iSerPort    - Port de base de l'interface.                */
/*          bTestStatus - Bit ,chantillon des lignes … tester        */
/*                        (CTS, DSR, RI, CD)                         */
/*********************************************************************/
INT SER_IsModemStatusSet( INT iSerPort, BYTE bTestStatus )
{
  return ( ( BYTE )inp( iSerPort + SER_MODEM_STATUS ) &bTestStatus )
      == bTestStatus;}

/*********************************************************************/
/* SER_SetModemControl : Etablir connexions de signal pour une        */
/*                       communication avec le modem                 */
/**-----------------------------------------------------------------**/
/* Entr,e : iSerPort    - Port de base de l'interface .               */
/*          bNewControl - Etat nouveau des lignes DTR, RTS           */
/*********************************************************************/
VOID SER_SetModemControl( INT iSerPort, BYTE bNewControl )
{
  outportb( iSerPort + SER_MODEM_CONTROL, bNewControl );
}

/*********************************************************************/
/* SER_WriteByte : Envoi d'un octet                                   */
/**-----------------------------------------------------------------**/
/*Entr,e : iSerPort - Port de base de l'interface pour                */
/*                    envoyer un octet.                              */
/*         bData    - octet … envoyer                                 */
/*         uTimeOut - Nombre de passages effectu,s dans la boucle     */
/*                    avant que l',chec d'une ,mission n'est signal,  */
/*                    par une erreur TimeOut.                         */
/*                    (Si iTimeOut = 0 l'attente continue             */
/*         bSigMask - Masque de bits des lignes de signal … tester    */
/*                    (RTS, CTS, CD, RI)                              */
/*         bSigVals - Etat des lignes de signal apr□s avoir           */
/*                    d,masqu, le masque ci-dessus.                   */
/*Sortie :  = 0 - octets ont ,t, envoy,s                              */
/*          <> 0 - Erreurs                                           */
/*********************************************************************/

INT SER_WriteByte( INT iSerPort, BYTE bData, UINT uTimeOut,
          BYTE bSigMask, BYTE bSigVals )
{ if( uTimeOut )                                  /* Boucle time-out */
```

```
  {
    while( SER_IsWritingPossible( iSerPort ) && uTimeOut )
uTimeOut--;
    if( !uTimeOut ) return SER_ERRTIMEOUT;
  }
  else while( !SER_IsWritingPossible( iSerPort ) );       /* Attente! */

                  /* Test des lignes de signal */
  if( ( ( BYTE ) inp( iSerPort + SER_MODEM_STATUS ) & bSigMask
) ==
       bSigVals )
  {
      /* Transmission des octets … ,mettre vers l'interface */
    outportb( iSerPort + SER_TXBUFFER, bData );
                  /* Retourne erreur interface */
    return inp( iSerPort + SER_LINE_STATUS ) &
SER_LSR_ERRORMSK;
  }
  else return SER_ERRSIGNALS;
}

/*************************************************************************/
/*  SER_IsDataAvaiable : Les donn,es sont-elles pr^tes … la lecture ? */
/**---------------------------------------------------------------------**/
/* Entr,e : iSerPort - Port de base de l'interface … tester            */
/* Sortie : == 0 : Aucun octet n'existe pour une lecture               */
/*          != 0 : L'octet est disponible                              */
/**---------------------------------------------------------------------**/
/* Info : L'octet se transmet bit par bit. Il est de nouveau           */
/*        complet au niveau de l'interface r,ceptrice qui l'a          */
/*        r,compos,. L'action se v,rifie par cette fonction .          */
/*************************************************************************/
INT SER_IsDataAvaiable( INT iSerPort )
{
  return inp( iSerPort + SER_LINE_STATUS ) & SER_LSR_DATARECEIVED;
}


/*************************************************************************/
/* ser_FIFOLevel : Fixe la taille d'un tampon FIFO                      */
/**---------------------------------------------------------------------**/
/* Entr,e : 0 - taille du buffer FIFO = 0, Disable et Reset (1Byte)    */
/*                  SER_FIFO_TRIGGER4/8/14 - Taille = 4, 8 ou  14 Byte */
/*************************************************************************/
VOID ser_FIFOLevel( INT iSerPort, BYTE bLevel )
{
  if( bLevel ) outportb( iSerPort + SER_FIFO, bLevel |
SER_FIFO_ENABLE );
  else         outportb( iSerPort + SER_FIFO, SER_FIFO_RESETRECEIVE |
                 SER_FIFO_RESETTRANSMIT );
}


/*************************************************************************/
/* SER_ReadByte : R,ception d'un octet                                 */
/* Entr,e : iSerPort - Port de base de l'interface utilis,e            */
/*                     pour recevoir un octet                          */
/*          Data     - la variable byte accueille l'octet              */
/*                     re‡u.                                           */
/*          uTimeOut - Nombre de passages dans la boucle               */
/*                     avant qu'une erreur TimeOut ne signale          */
```

```
/*                      l',chec de la r,ception.                          */
/*                       (Si iTimeOut = 0 l'attente continue )            */
/*          bSigMask - Masque de bits des lignes de signal …              */
/*                      tester                                            */
/*                       (RTS, CTS, CD, RI)                               */
/*          bSigVals - Etat des lignes de signal apr□s                    */
/*                       avoir d,masqu, avec le masque ci-dessus.         */
/*          Sortie :  = 0 - octets ont ,t, envoy,s                        */
/*          != 0 - Erreurs                                                */
/*********************************************************************/
INT SER_ReadByte( INT iSerPort, PBYTE pData, UINT uTimeOut,
        BYTE bSigMask, BYTE bSigVals )
{ if( uTimeOut )                                  /* Boucle TimeOut */
  {
    while( SER_IsDataAvaiable( iSerPort ) && uTimeOut )
uTimeOut--;
    if( !uTimeOut ) return SER_ERRTIMEOUT;
  }
  else while( !SER_IsDataAvaiable( iSerPort ) );          /* Attente! */

                  /* Test des lignes de signaux */
  if( ( ( BYTE ) inp( iSerPort + SER_MODEM_STATUS ) & bSigMask
) ==
      bSigVals )
  {                          /* Lecture de l'octet re‡u par l'interface */
     *pData = ( BYTE )inp( iSerPort + SER_RXBUFFER );
    return inp( iSerPort + SER_LINE_STATUS ) & SER_LSR_ERRORMSK;
  }
  else return SER_ERRSIGNALS;
}

/*********************************************************************/
/* GetArg : Retourne param□tre de la ligne de commande               */
/* Entr,e : argc - Argument-Count (… voir main(INT argc, CHAR *argv[]))  */
/*          argv - Argument-Values (… voir main(INT argc, CHAR *argv[])) */
/*          pPrefix - Parameter-Prefi                                */
/*          iType - type du param□tre (_char, _int, _long, _string,  */
/*                  _none = v,rifie la pr,sence d'un param□tre)       */
/*           pVar - adresse de la variable, qui va contenir les      */
/*                  valeurs des param□tres de la ligne de commande   */
/*           iNumElements - S'il y a plusieurs valeurs s,par,s par ','*/
/*                          Enregistre les param. jusqu'… l',l,ment   */
/*                          iNumElements dans l'array indiqu, dans le */
/*                          pVar                                     */
/* Sortie : Nombre de valeurs calcul,s ou  0 s'il n'y a pas de       */
/*          param□tre de commande                                    */
/*********************************************************************/
INT GetArg( INT argc, PCHAR argv[], PCHAR pPrefix, INT iType,
         PVOID pVar, INT iNumElements )
{
  INT i, j;
  INT iLen;

  PCHAR  cPtr;
  PINT   iPtr;
  PLONG  Ptr;
  PCHAR *sPtr;

  iLen = _fstrlen( pPrefix );

  for( i = 1; i < argc; i++ )
```

```
    if( _fstrnicmp( pPrefix, ( LPCHAR )argv[ i ], iLen ) == 0 )
      switch( iType )
      {
        case _int:
        iPtr = (PINT)pVar;
          for( j = 0; ( j < iNumElements ) && argv[ i ][ iLen ]; j++ )
          {
            *iPtr = atoi( ( PCHAR )&argv[ i ][ iLen ] );
          while( argv[ i ][ iLen ] && ( argv[ i ][ iLen ] != ',' ) )
            iLen++;
          if( argv[ i ][ iLen ] == ',' ) iLen++;
          iPtr++;
        }
          return j;

        case _char:
          cPtr = (PCHAR)pVar;
        for( j = 0; ( j < iNumElements ) && argv[ i ][ iLen ]; j++ )
          {
            *cPtr = argv[ i ][ iLen ];
            while( argv[ i ][ iLen ] && ( argv[ i ][ iLen ] != ',' ) )
            iLen++;
          if( argv[ i ][ iLen ] == ',' ) iLen++;
          cPtr++;
        }
        return j;

      case _long:
          Ptr = (PLONG)pVar;
          for( j = 0; ( j < iNumElements ) && argv[ i ][ iLen ]; j++ )
        {
            *Ptr = atol( ( PCHAR )&argv[ i ][ iLen ] );
            while( argv[ i ][ iLen ] && ( argv[ i ][ iLen ] != ',' ) )
            iLen++;
          if( argv[ i ][ iLen ] == ',' ) iLen++;
          Ptr++;
        }
        return j;

        case _string:
        sPtr = ( PCHAR *)pVar;
          for( j = 0; ( j < iNumElements ) && argv[ i ][ iLen ]; j++ )
          {
          *sPtr = &argv[ i ][ iLen ];
            while( argv[ i ][ iLen ] && ( argv[ i ][ iLen ] != ',' ) )
              iLen++;
            if( argv[ i ][ iLen ] == ',' ) argv[ i ][ iLen++ ] = '\0';
          sPtr++;
        }
        return j;

      case _none:
          return TRUE;
      }
  return FALSE;
}

/*********************************************************************/
/* GetNArg:Retourne le paramètre de la ligne de commande sans pr,fixe */
/*-------------------------------------------------------------------*/
/* Entr,e : argc - Argument-Count (… voir main(INT argc, CHAR *argv[]))  */
```

```
/*          argv - Argument-Values (… voir main(INT argc, CHAR *argv[])) */
/*          pPrefix - pr,fixe des param□tres (par exemple : "-o" pour */
/*                    output) qu'il faut ,carter des param□tres       */
/*          pString - adresse  du  pointeur de String- Arrays qui     */
/*                    recueille les param□tres                        */
/*          iNumElements - S'il y a plusieurs param□tres contenus     */
/*                         dans la ligne de commande, tout au plus max*/
/*                         param□tres sont pr,serv,s dans iNumElements*/
/* Sortie : Nombre de param□tres communiqu,s, ou 0 s'il n' y a pas de */
/*          param□tres sur la ligne de commande                       */
/*****************************************************************/
INT GetNArg( INT argc, PCHAR argv[], PCHAR pPrefix,
           PCHAR *pString, INT iNumElements )
{
  INT i, j;
  INT iLen;

  iLen = _fstrlen( pPrefix );

  for( i = 1, j = 0; ( i < argc ) && ( j < iNumElements ); i++ )
    if( _fstrnicmp( pPrefix, ( PCHAR )argv[ i ], iLen ) != 0 )
      pString[ j++ ] = argv[ i ];

  return j;
}

/*****************************************************************/
/* FindString : Recherche d'un string dans un String-Array et renvoi  */
/*              de la position du string qui a ,t, trouv, dans l'array*/
/**-------------------------------------------------------------**/
/* Entr,e : pArray  - String-Array … examiner                    */
/*          pFind   - String … rechercher                        */
/*          iNum    - nombre de Strings dans le String-Array     */
/* Sortie : Position + 1 du string trouv, dans le array ou 0     */
/*           si le string n'est pas contenu dans l'array         */
/*****************************************************************/
INT FindString( PCHAR pArray[], PCHAR pFind, INT iNum )
{ INT i;
  for( i = 0; i < iNum; i++ )
   if( _fstricmp( pArray[ i ], pFind ) == 0 ) return i + 1;
  return 0;
}



/*****************************************************************/
/***********                PROGRAMME PRINCIPALE          *********/
/*****************************************************************/
void main(INT argc, PCHAR argv[])
{
  int f;
  INT iCom, i;
  LONG  lBaud;
  PBYTE recu;

  clrscr();
  if( FindString( argv, "?", argc ) )
  {
    printf("Appel:\n" );
    printf("RESEPT.EXE [Param□tres du fichier … recevoir] [-BAUD:vitesse]\n");
```

```
    printf("Les param□tres sont : Pr,fixe Symbole Vitesse Longueur");
    printf("vitesse = 50 - 115200 (d,faut: 9600)\n");
    exit(0);
  }

  if( !GetArg( argc, argv, "-BAUD:", _long, &lBaud, 1 ) )
      lBaud = 9600L;          /* Taux maximal de baud de l'UART : 8450A */
  if( lBaud > SER_MAXBAUD )
  {
    printf("Vitesse trop ,lev,e !\n");
    printf("Maximum: %ld Bd\n", SER_MAXBAUD );
  }



   clrscr();
   if ( !SER_Init( SER_COM2, lBaud ,SER_LCR_8BITS | SER_LCR_1STOPBIT |
SER_LCR_NOPARITY ) )
    {
      printf ( "Pas d'interface.....!" );
      getch();
      return;
    }

printf("Emission de l'ordre de conversion....! \nEst-ce que vous ete pret...?
\n");
while (!kbhit());
while(SER_IsWritingPossible(SER_COM2)) SER_WriteByte(SER_COM2, 'd', SER_TIMEOUT,
0,0);

i=0;
while(i!=1001)
{
   if (  !SER_ReadByte( SER_COM2, recu, SER_TIMEOUT, 0, 0 )== SER_SUCCESS  )
    {
     printf("Erreur lecture  sur le port  reception interrompu...");
     return;
    }
printf("Reception de l'%d□me symbole.\n",i++);
}

traitement(argc, argv);

}
```

ملخص :

يهدف هذا البحث إلى تحقيق نظام يسمح بقياس فعالية رابطة اتصال رقمية، حيث قمنا بإنجاز بطاقة تعديل FSK وبطاقة كشف FSK و من ثمَّ ربط الأولى بنهائي إرسال للمعطيات الرقمية والتالية بنهائي استقبال للمعطيات الرقمية. ثم إرسال متتالية رموز واستقبالها وإجراء مقارنة بين المتتاليتين.

الكلمات المفتاحية :

معدل FSK، كاشف FSK، احتمال الخطأ، وصلة رقمية، دارة التقرير.

Abstract :

The purpose of this work is the realization of a system permitting the measure of the reliability of a numeric communication link. We conceived an FSK modulator and demodulator. The first bind to a data transmission terminal while the second bind to a data receipt terminal. Then we send a sequence of symbols and we receive them and make a comparison between the two sequences.

Key words :

FSK modulator, FSK demodulator, error probability, numeric link, decision circuit

Résumé :

Le but de ce travail étant la réalisation d'un système permettant la mesure de la fiabilité d'une liaison de communication numérique. Pour cela, nous avons conçu un modulateur et un démodulateur FSK. Le premier étant lié à un terminal de transmission de données tandis que le deuxième est connecté à un terminal de réception de données. Ensuite nous transmettons une séquence de symboles et nous la recevons. Par suite, nous faisons une comparaison entre les deux séquences.

Mots clés :

Modulateur FSK, démodulateur FSK, probabilité d'erreur, liaison numérique, circuit de decision.