# TD#5 – Intégration d'un Modem V.21 sur DSP BF 706

Électronique programmée : Processeurs DSP – Richard Salvetat
Simon Lignac, Steeve Mbock | Mai 2026 | Aéro4 – IPSA Toulouse

## 1 Contexte et normalisation V.21

La recommandation V.21 de l'UIT-T définit un modem full-duplex à 300 bauds en modulation FSK sur le réseau téléphonique commuté. Les deux voies utilisent des fréquences disjointes :

TABLE 1 – Fréquences V.21

| Voie | $f_c$ | F1 (bit = 1) | F0 (bit = 0) |
|---|---|---|---|
| 1 – appel | 1080 Hz | 980 Hz | 1180 Hz |
| 2 – réponse | 1750 Hz | 1650 Hz | 1850 Hz |

La déviation est $\Delta f = \pm 100\,\text{Hz}$. Les tolérances normatives sont $\pm 6\,\text{Hz}$ en émission et $\pm 12\,\text{Hz}$ en réception. Ce TD porte sur la voie 1 (appel).

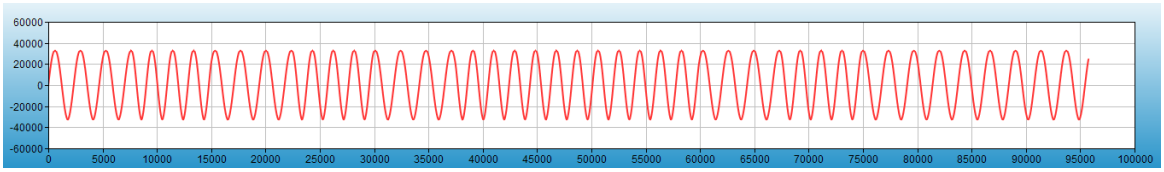


FIGURE 1 – Spectre de la modulation FSK V.21 – voie appel

## 2 Principe du démodulateur FSK

### 2.1 Signal reçu

En présence de phase inconnue $\varphi$ et de bruit gaussien additif $n(t)$ :

$$e(t) = A_0 \cos\!\big(2\pi f t + 2\pi m\,\Delta f\, t + \varphi\big) + n(t), \quad m \in \{0,1\} \tag{1}$$

La phase $\varphi$ étant inconnue, la détection est non cohérente.

### 2.2 Corrélateurs en quadrature

Pour chaque fréquence $f_k \in \{F_1, F_0\}$, on calcule les projections du signal sur $\cos(2\pi f_k t)$ et $\sin(2\pi f_k t)$ sur une période symbole $T = 1/300\,\text{s}$ :

$$r_{kc} = \sum_{n=0}^{N_T-1} x[n] \cos\tfrac{2\pi f_k n}{F_e}, \quad r_{ks} = \sum_{n=0}^{N_T-1} x[n] \sin\tfrac{2\pi f_k n}{F_e} \tag{2}$$

L'énergie de branche $R_k = r_{kc}^2 + r_{ks}^2$ est indépendante de $\varphi$ (identité $\cos^2 + \sin^2 = 1$). Pour la bonne branche ($k = m$) : $R_m \approx (A_0 N_T/2)^2$ ; pour la mauvaise ($k \neq m$) : $R_k \approx 0$ par quasi-orthogonalité des deux fréquences.

### 2.3 Règle de décision

$$\text{bit} = \begin{cases} 1\ (\texttt{0x7F}) & \text{si } R_1 > R_2 \\ 0\ (\texttt{0x80}) & \text{sinon} \end{cases} \quad \Longleftrightarrow \quad R_1 \underset{H_0}{\overset{H_1}{\gtrless}} R_2 \tag{3}$$

En canal AWGN, $\sqrt{R_1}$ suit une loi de Rice (signal + bruit) et $\sqrt{R_0}$ une loi de Rayleigh (bruit seul). La probabilité d'erreur théorique est $P_e = \frac{1}{2}\exp\!\big(-E_b/2N_0\big)$.

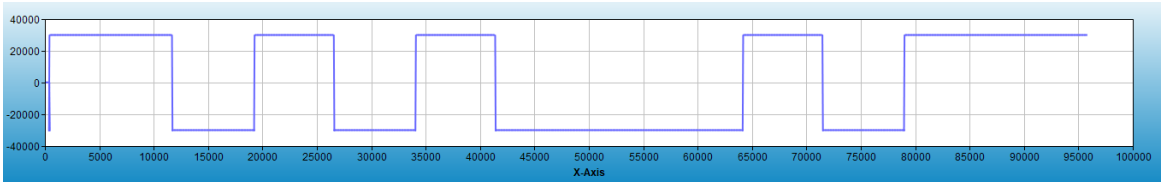


FIGURE 2 – Architecture du démodulateur FSK binaire

## 3 Implémentation sur DSP BF 706

### 3.1 Paramètres numériques

Projet développé sous CrossCore Embedded Studio ($F_e = 48\,000$ Hz, LUT sinus de 1600 points). Le cosinus s'obtient par décalage de 400 points dans la LUT.

TABLE 2 – Paramètres numériques

| Paramètre | Signification | Valeur |
|---|---|---|
| `N` | Taille buffer | 2048 éch. |
| `BAUD` | Éch. par bit | 160 |
| `pas_F1` | Pas LUT pour F1 | 32 → 960 Hz |
| `pas_F0` | Pas LUT pour F0 | 39 → 1170 Hz |

### 3.2 Algorithme et trame UART

`SimDemodulatorFsk` traite `gTableau[N]` comme : calcul des quatre corrélateurs, comparaison $R_1$ vs $R_2$, écriture de `0x7F`/`0x80` dans `gTableauOut[N]`.

```
void SimDemodulatorFsk(volatile short* pDataIn,
                       volatile short* pDataOut,
                       short pasf1,
                       short pasf0)
{
    int index_cosf0 = 0;
    int index_sinf0 = N_LUT / 4;

    int index_cosf1 = 0;
    int index_sinf1 = N_LUT / 4;

    int index_rc0 = 0;
    int index_rs0 = 0;
    int index_rc1 = 0;
    int index_rs1 = 0;

    float rc0 = 0.0f;
    float rs0 = 0.0f;
    float rc1 = 0.0f;
    float rs1 = 0.0f;

    float outk = 0.0f;
    short ek = 0;

    for (int k = 0; k < N; k++)
    {
        ek = pDataIn[k];

        // cos f0
        buf_rc0[index_rc0] = (ek * LUT[index_cosf0]) / 65536.0f;
        index_cosf0 = (index_cosf0 + pasf0) % N_LUT;
        index_rc0 = (index_rc0 + 1) % T;

        rc0 = 0.0f;
```

```
        for (int i = 0; i < T; i++) rc0 += buf_rc0[i];
        rc0 = (rc0 * rc0) / 65536.0f;

        // sin f0
        buf_rs0[index_rs0] = (ek * LUT[index_sinf0]) / 65536.0f;
        index_sinf0 = (index_sinf0 + pasf0) % N_LUT;
        index_rs0 = (index_rs0 + 1) % T;

        rs0 = 0.0f;
        for (int i = 0; i < T; i++) rs0 += buf_rs0[i];
        rs0 = (rs0 * rs0) / 65536.0f;

        // cos f1
        buf_rc1[index_rc1] = (ek * LUT[index_cosf1]) / 65536.0f;
        index_cosf1 = (index_cosf1 + pasf1) % N_LUT;
        index_rc1 = (index_rc1 + 1) % T;

        rc1 = 0.0f;
        for (int i = 0; i < T; i++) rc1 += buf_rc1[i];
        rc1 = (rc1 * rc1) / 65536.0f;

        // sin f1
        buf_rs1[index_rs1] = (ek * LUT[index_sinf1]) / 65536.0f;
        index_sinf1 = (index_sinf1 + pasf1) % N_LUT;
        index_rs1 = (index_rs1 + 1) % T;

        rs1 = 0.0f;
        for (int i = 0; i < T; i++) rs1 += buf_rs1[i];
        rs1 = (rs1 * rs1) / 65536.0f;

        // Sortie analogique
        outk = ((rc1 + rs1) - (rc0 + rs0)) / T;

        // Binarisation
        pDataOut[k] = (outk > 0) ? 30000 : -30000;
    }
}
```

Listing 1 – `SimDemodulatorFsk`

La trame UART (1 idle + 1 start + 8 data + 1 stop) occupe $11 \times 160 = 1760$ échantillons. Les 8 bits utiles sont extraits par : $\texttt{gOutBinaire}[i] = \texttt{gTableauOut}[(i+2) \times \texttt{BAUD}]$, $i \in [\![0, 7]\!]$.

## 4 Résultats et validation

### 4.1 Caractère de référence

Pour `gDataTx = 0xA2` ($= 1010\,0010_2$), la trame démodulée attendue est :

| Idle | Start | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Stop |
|---|---|---|---|---|---|---|---|---|---|---|
| 7F | 80 | 7F | 80 | 7F | 80 | 80 | 80 | 7F | 80 | 7F |

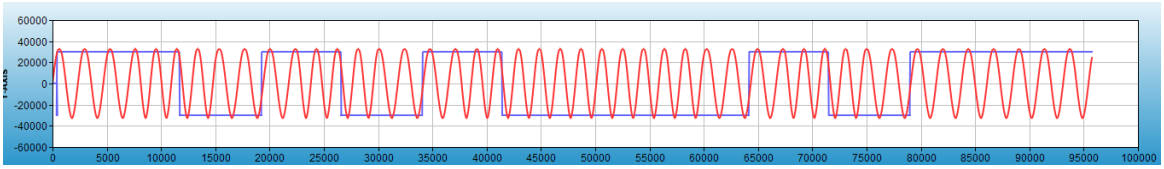


FIGURE 3 – Signal modulé et démodulé pour `gDataTx = 0xA2`

### 4.2 Discussion

Les corrélateurs en quadrature rendent le démodulateur insensible à la phase $\varphi$. Les fréquences effectives (960 Hz / 1170 Hz) s'écartent des cibles V.21 à cause de la quantification LUT, mais modulateur et démodulateur partagent les mêmes pas, donc la corrélation reste accordée. L'absence d'orthogonalité stricte entre $F_0$ et $F_1$ est négligeable en simulation sans bruit.