SYSTEME

Microordinateur basic à tout faire

MICRO-VON

Michel VONLANTHEN HB9AFO

RÉSUMÉ :

Le système MICRO-VON est un ensemble microordinateur d'application complet – électronique et programme – destiné à être incorporé dans un ensemble. Une fois le programme mis au point, la partie « système de développement » est débranchée et l'ordinateur fonctionne de façon indépendante. Il est programmable en BASIC, langage de haut niveau facilement assimilable par le débutant, et en langage machine. La description qui suit est destinée à permettre à chacun de composer ses propres applications. Pour en illustrer le processus, un ensemble RTTY performant (radiotélétype) pour radioamateur sera décrit. Les circuits-imprimés et les composants sont distribués par HAMCO (Suisse) et facilement obtenables.

La première partie de cette série a décrit les principes généraux du système tandis que la seconde passait en revue la carte microprocesseur VON257, « cœur » du système, et la carte de programmation VON267, utilisée pendant la période de développement du programme.

Passons maintenant aux applications en commençant par le RTTY

Nous avons étudié jusqu'à présent les buts et schémas des deux cartes formant l'ossature du microordinateur. Connectées à un display ASCII 110 Bauds, elles forment donc un microordinateur BASIC complet.

Nous entrons maintenant dans le vif du sujet au point de vue radioamateur en décrivant la marche à suivre pour en faire un SYSTÈME RTTY COMPLET. Ce sera un montage évolutif allant du plus simple au plus compliqué. Nous l'avons voulu évolutif pour trois raisons :

Primo : le radioamateur-constructeur utilise en général fort peu ce qu'il construit. Une fois la mise au point terminée et quelques liaisons effectuées, il passe à la suite. Un montage qui puisse être amélioré maintiendra le plaisir pendant plus longtemps.

Secundo : le microprocesseur est un engin merveilleusement flexible. En ce sens, il permet de suivre l'évolution de la technique en changeant simplement le programme et en rajoutant, si nécessaire, des interfaces supplémentaires. Malheureusement, les systèmes commerciaux sont figés : aucun moyen de changer le programme. Il est fixé une fois pour toutes dans ses mémoires mortes et l'acheteur ne peut pas les modifier lui-même. Plus grave encore, les vendeurs de ces engins préfèrent sortir et vendre de nouveaux modèles plutôt que d'améliorer les précédents par simple changement des mémoires.

En ce sens, le MICRO-VON est totalement différent car l'utilisateur lui-même peut, en tout temps, modifier son programme et refaire une EPROM puisque son système de développement *est* le système MICRO-VON.

Tertio : la technique évolue constamment aussi est-il bon de pouvoir s'y adapter et en tirer profit. Deux exemples de « l'ancienne méthode » :

– Lorsque la norme des fréquences RTTY a changé (passage aux fréquences basses), les possesseurs de démodulateurs équipés de tores 88mH ont du les rebobiner pour s'adapter à ce changement. Quelques-uns s'y sont attaqués mais la plupart ont changé de modem.

– Lorsque l'ASCII 110 Bauds a fait son apparition, les possesseurs de téléimprimeurs 45 Bauds n'ont pas pu les adapter. Il a fallu les remplacer.

Le système RTTY dont la description suit pourra, lui, s'adapter à toutes les situations. Dans cette optique, il va à contre-courant avec la société de consommation dans laquelle il faut « acheter-consommer-jeter ». « Ecologiquement », nous « construirons-modifierons-garderons » !...

LES OPTIONS

Le système de base comprend :

– la carte microprocesseur VON257

– la carte de programmation VON267

– le terminal série ASCII 110 Bauds par exemple le terminal RTTY tout-électronique de HB9BBN décrit dans Radio-REF ou tout autre terminal. On peut aussi utiliser un terminal mécanique type « TELETYPE ASR33 » mais cela nécessite un adaptateur boucle de courant 20mA/TTL, qui sera aussi décrit ultérieurement. On peut même utiliser un autre microordinateur programmé en terminal 110 Bd et ne considérer le système MICRO-VON que comme « boîte noire » dans laquelle entre de l'ASCII 110 Bd et de laquelle sort du Baudot 45 Bd et vice-versa. Dans cette optique n'importe-quel microordinateur individuel convient (TRS80, Apple 2, MZ80, ZX81, etc...)

Pour en faire un système RTTY, il faut lui adjoindre :

1. Un interface série (UART programmable) permettant de travailler à n'importe quelle vitesse pour autant que le terminal suive et dans n'importe quel format asynchrone.

2. Un modem comme, par exemple, le DJ6HP, HB9BBN, HAL, Dovetron ou autre. Un petit modem sera décrit pour ceux qui n'en possèdent pas encore. Il s'agit d'un générateur AFSK (manipulation par variation de fréquence BF) sans ruptures de phases et stable et d'un démodulateur à verrouillage de phase (PLL) parfait pour les VHF-UHF où le qrm ne sévit pas. Un démodulateur à filtres sera préférable sur ondes-courtes.

Tel quel, cet ensemble RTTY permettra un trafic normal avec messages pré-enregistrés (indicatifs, RY, quick brown fox..., CQ, etc...) et message permanent, par exemple les indicatifs du QSO en cours.

3. L'option suivante sera une carte permettant de changer la vitesse du terminal et de passer du 110 Bd au 300, 1200 ou même 4800 Bauds. Cela nous permettra de trafiquer à toutes les vitesses permises au radioamateur et même de décoder la télémétrie du satellite radioamateur OSCAR 9 (UOSAT) qui travaille généralement en 1200 Bd.

4. Pour ceux qui aimeraient disposer d'un ensemble RTTY aisément transportable, il sera possible de lui adjoindre un affichage à cristaux liquides, à consommation négligeable. Il est alors possible de faire tenir l'ensemble RTTY réception au complet dans un volume identique à celui d'un carton à chaussures pour bébé !

5. Il est quelquefois souhaitable de pouvoir garder une trace tangible des messages reçus. Pour cela, nous rajouterons une sortie pour imprimante parallèle type Centronics (dans notre cas MX80 d'EPSON) et la routine de programme adéquate.

Chacune de ces options fera l'objet d'une description. En pratique, il s'agit en général de rajouter un interface supplémentaire (fils wrappés) et de compléter le programme de base par l'adjonction de routines supplémentaires. Ceci se fait sur place, au moyen de la carte de programmation.

D'autre part, le principe de programmation des interfaces sera explicité en détail ce qui permettra ensuite à chacun de les utiliser selon ses propres envies et... d'en faire profiter les amis par l'intermédiaire d'une petite description dans MEGAHERTZ !...

A LA BASE

Le dessin « configuration de base » illustre les interconnections à réaliser entre les divers modules. Tel quel, nous avons un microordinateur permettant de travailler en BASIC et en langage machine, y compris la programmation de l'EPROM 2716. Les entrées-sorties se font par l'intermédiaire du clavier et de l'écran du terminal. On peut encore utiliser les sorties F2 et F3 du microprocesseur pour actionner des éléments extérieurs (lampes, relais, etc.) via un amplificateur de puissance bien sûr et aussi l'entrée SB pour entrer une donnée binaire (par exemple l'état d'un interrupteur).

Deux sorties et une entrée c'est peu pour dialoguer avec le monde extérieur. Il nous faut des interfaces. Les données RTTY étant transmises en série, il nous faut donc une carte opérant la transformation parallèle-série pour l'émission et série-parallèle pour la réception : la carte VON262. Toutes les fonctions sont réalisées par le chip 8250, *A*synchrone *C*ommunication *E*lement, et de façon entièrement programmable. Grâce à cette seule adjonction et son programme associé (qui sera décrit en détail), nous pourrons déjà travailler confortablement en Baudot 45 Bd à condition, cela va de soi, de posséder un modem pour y connecter notre logique. Si ce n'est pas le cas, cette partie devra être construite. Une description ad-hoc suivra.

Les alimentations dessinées sur le plan « configuration de base » ne seront pas décrites afin de ne pas trop rallonger cette série d'articles avec quelque chose que tout le monde connaît. Sachez simplement qu'elles existent et qu'HAMCO peut les livrer toutes montées le cas échéant pour les « paresseux » (schémas disponibles gratuitement contre une enveloppe self-adressée + 2 coupon-réponse à HAMCO).

CONNECTION DE L'INTERFACE SÉRIE

Une fois la carte montée selon la notice accompagnant le kit, exécuter le câblage suivant :

– alimentation : relier le +5V et la masse (haut du plan d'implantation) à l'alimentation 5V soit par des fils de wrap soit par des fils soudés. Dans ce dernier cas, souder les fils au ras du print sur la print à wrapper de façon à pouvoir quand même enrouler un fil sur la pin car cette dernière doit aussi être reliée à celle marquée +5 ainsi qu'à la pin qui se trouve juste en-dessous de cette dernière.

– de et vers la carte microprocesseur : tout se fait impérativement en fil à wrapper (fils enroulés) au moyen de l'outil prévu à cet effet (idsponible chez HAMCO avec le mode d'emploi). Il faut avoir sous les yeux le plan de la carte VON257 intitulé « pins wrapper » et le plan d'implantation de la carte série VON262. Le principe de base, valable aussi pour tous les autres interfaces, est de relier entre elles les pins de même nom. Par exemple, le NRDS de la VON262 va au NRDS de la 257. Les points à relier sur les deux cartes sont donc les suivants :

VON262 à VON257

NRDS	NRDS
AD2	A2
AD1	A1
AD0	A0
INT	SB
reset/	reset/
D0	D0
D1	D1
D2	D2
D3	D3
D4	D4
D5	D5
D6	D6
D7	D7
select/	X'A000
clock/	clock/
NWDS	NWDS

Note : le / qui suit certains noms signifie que ceux-ci sont actifs à 0V ce qui correspond aux noms qui sont surmontés d'un trait horizontal sur les schémas.

– du démodulateur : l'entrée SERIAL IN est à wrapper sur la sortie OUT du démodulateur. Relier les masses entre elles.

– Vers générateur AFSK : la sortie SERIAL OUT est à wrapper sur l'entrée IN (TTL) du générateur AFSK. Les masses sont à relier entre elles.

LE SCHÉMA :

Cet interface permet l'entrée-sortie en série, en mode asynchrone, c'est-à-dire que chaque caractère à transmettre est précédé d'un START et suivi d'un STOP. Tous les paramètres de ce mot sont sélectionnables par programmation.

SYSTÈME MICRO-VON
CONFIGURATION DE BASE

MICRO-VON
INTERFACE SÉRIE SCHÉMA VON262

INTERFACE ASYNCHRONE SÉRIE VON 262
PLAN D'IMPLANTATION

Ce circuit complexe au point de vue électronique est facile à utiliser si l'on se borne au mode de connection au système MICRO-VON décrit ci-dessus et aux fonctions simples que nous lui demandons et que nous allons détailler. Pour toute autre application, il est nécessaire d'étudier à fond le problème à l'aide des documents du fabricant. On peut schématiser le 8250 de cette façon :

On voit donc qu'il s'agit d'un convertisseur parallèle-série et série-parallèle ayant le générateur de vitesses (baud rate generator) commun ainsi qu'une série de registres permettant la sélection de tous les paramètres de fonctionnement par programmation ce qui donne un montage adaptable à toutes les situations de transmission asynchrone pouvant se présenter. Avant de passer à la programmation, voyons tout d'abord quels sont les signaux à émettre et à recevoir. Il s'agit donc de signaux asynchrone série, identiques à l'émission et à la réception et qui revêtent deux formes :

– *code baudot :*

C'est la norme CCITT-2 de l'UIT, utilisée pour tout le trafic TELEX international, sur ondes-courtes, par les agences de presse, météo, radioamateurs, etc. Le format est toujours le même, seule change la vitesse selon les applications : 45 Bd pour les radioamateurs, 50 Bd pour le TELEX et la plupart du trafic baudot sur ondes-courtes, 75 et 100 bd plus rarement utilisés. La forme du signal série arrivant sur SERIAL IN ou partant de SERIAL OUT est le suivant :

Au fil du temps, ce signal RTTY prend les états suivants :

– au repos il reste au +5V, ce qui correspond à l'état « mark » ;

– le start est le passage à OV (état « space ») pendant la durée d'un bit et il marque le début du caractère. C'est cette caractéristique qui différencie le mode ASYNCHRONE utilisé ici du mode SYNCHRONE (ex : AMTOR). Un code peut être envoyé à n'importe quel moment aussi faut-il savoir quand un caractère débute. Dans le cas où la transmission commence c'est simple : on se rend bien compte quand la manipulation commence. Dans le cas où les caractères sont envoyés en continu, le 8250 sait qu'un caractère commence après avoir reçu un STOP (1,5 bit) suivi d'un START (1 bit). En prenant le cas de la réception, voici ce que « pense » le 8250 lorsqu'il reçoit un signal. On prendra le cas du 50 Bd où chaque bit dure 20 millisecondes.

a) Mark (+5V) : rien ne se passe, je ne fais rien...

b) Ah on passe au OV ! C'est un START. Pendant le start je ne fais rien mais je me prépare à recevoir le caractère.

c) 20 ms ont passé. Je dois donc recevoir le premier bit. Si le signal reste à 0V, ce sera un zéro. S'il passe au +5V, ce sera un « 1 ». Je mets cet état dans la case qui correspond à la ligne D0.

d) 20 ms ont encore passé. J'arrive donc au bit 2. Même schéma que le premier et je mets ce niveau dans la case qui correspond à la ligne D1 (registre).

e) 20 ms... idem... D2

f) 20 ms... idem... D3

g) 20 ms... idem... D4, dernier bit du caractère à 5 bits.

h) 20 ms après : voyons maintenant si ce que j'ai reçu est dans le bon ordre, on doit maintenant recevoir un STOP, donc un signal qui passe à « 1 » pendant un bit et demi soit 30 ms. Il y passe et y reste 30 ms, ok, le format est donc correct. J'envoie le signal INT (interrupt) pour dire au patron (le microprocesseur) que les 5 bits reçus sont bien rangés de D0 à D4 et qu'il n'a qu'à les transférer sur son bus de données quand bon lui semblera simplement en m'envoyant le signal SELECT/ et l'impulsion NRDS (= lecture du registre de donnée). Si le format avait comporté une erreur, en plus du processus ci-dessus, je lui aurais signalé ce fait en mettant dans mon registre 5 (= line status) certains bits à 1 ce qui lui aurait indiqué que le caractère reçu est bien disponible dans le regostre 0 mais qu'il est probablement faux.

Il est à noter que l'entrée série du 8250 est indépendante de la sortie parallèle et qu'on peut recevoir un signal série pendant que le caractère précédent est transféré sur le bus de données.

Remarquons quand même l'échelle des vitesses :

durée du caractère baudot 50Bd : 150 ms

durée du transfert parallèle : 0,5 μs

soit 300'000 fois plus vite !

– *code ASCII :* (American national Standard Code for Information Interchange)

C'est la norme CCITT-5 de l'UIT. Comme le code baudot, le caractère commence par un START de 1 bit, mais suivi cette fois-ci de 7 bits de données au lieu de 5. Ici également, c'est le bit ayant le poids binaire le plus faible qui est transmis le premier (= LSB, Lowest Significant Bit) et qui correspond donc à la ligne D0 du bus de données.

Après le septième bit du caractère ASCII, nous avons le bit de parité, rarement utilisé et on le laisse en général à zéro. Notons en passant qu'en cas de parité paire, le bit de parité passe à 1 lorsque le nombre de bits à 1 du caractère ASCII est pair. Par exemple :

« A » en ASCII : X'41 = en binaire 0100 0001.

Nous avons donc 2 bits à 1 si bien que le bit de parité pair passe à 1. Il est important de noter que l'ordre de transmission des bits (c'est aussi valable pour le baudot) est inversé par rapport à l'écriture puisqu'on transmet d'abord le bit de poids faible.

Exemple : « B » en ASCII = X'42 et en binaire 0100 001 (le LSB est à droite).

Le STOP dure ici 2 bits pour la vitesse de 110 Bd et 1 bit au-dessus. En pratique, le code ASCII est rarement utilisé sur ondes-courtes par d'autres que les radioamateurs. Par contre, il règne en maître en informatique.

LES NORMES RADIOAMATEUR

Baudot : 45,45 Bd - 1 start - 5 bits de donnée - 1,5 stop bit

ASCII : 110 Bd - 1 start - 7 bits de donnée - 1 bit de parité à 0 - 2 stop bits

Vitesses : en baudot 45 Bd, ce qui signifie 45 bits/seconde, chaque caractère transmis comporte 7,5 bits ce qui donne la :

vitesse = 45Bd/7,5 bits = *6 caractères/seconde*

En ASCII 110 Bauds, nous transmettons 11 bits au total ce qui donne la :

vitesse = 110Bd/11 bits = *10 caractères/seconde*

On constate donc que l'ASCII est plus rapide et transmet aussi plus de codes différents puisqu'il a 7 bits de données ce qui fait 128 possibilités de codage alors que le baudot n'en a que 5 ce qui fait 32 possibilités (que l'on double à l'aide des codes FIGS et LTRS).

PROGRAMMATION

A l'enclenchement, le 8250 ne fonctionne pas car il n'est pas initialisé. En conséquence, il faut lui donner les informations nécessaires à son fonctionnement. Ce sont :

La vitesse : le circuit est équipé d'un oscillateur avec quartz extérieur et d'un diviseur de fréquence programmable. Dans notre cas, nous n'utilisons pas l'oscillateur mais prenons l'horloge du microprocesseur CLOCK/ de 4 MHz comme fréquence de référence. Comme sur pratiquement tous les circuits de ce genre, la fréquence finale s'obtient à l'aide de la formule :

F horloge = baud x 16

En conséquence, il nous faut diviser 4'000'000 par un certain nombre afin d'obtenir cette fréquence de cadencement (clock veut dire « horloge »). Voici les valeurs les plus courantes :

vitesse (baud)	*F horloge (Hz)*	*diviseur*	*reg X'A000*	*reg X'A001*
45,45	727,2	5501	7D	15
50	800	5000	88	13
75	1200	3333	05	0D
110	1760	2273	E1	08

Ce diviseur doit ensuite être chargé dans les registres du diviseur programmable. Pour en illustrer la méthode, prenons l'exemple du 45 Bd baudot. Il faut diviser 4'000'000 par 5501 pour obtenir la fréquence finale de 727,2 Hz. Il suffit pour cela de charger le diviseur 5501 (157D en hexadécimal) dans les

registres A000 et A001 de la façon suivante :

– indiquer au 8250 qu'on veut entrer la valeur du diviseur en mettant la valeur X'80 (dlab=1) dans le registre X'A003. En BASIC :

@≠A003=≠80

– charger la valeur du diviseur dans leurs registres respectifs. En BASIC :

@≠A000=≠7D

@≠A001=≠15

On peut constater que les registres du 8250 sont considérés comme des positions mémoire dans lesquelles on peut lire et écrire ce qui est un des avantages de ce circuit. Étant donné que le signal de sélection du 8250 est le X'A000 de la carte microprocesseur, le registre 0 du 8250 répond à l'adresse X'A000, le registre 1 à X'A001, etc... jusqu'à X'A006.

Ayant entré ces paramètres, nous pouvons contrôler que le circuit divise correctement en mesurant la fréquence d'horloge (baud × 16) sur la pin marquée MESURE de l'interface VON262. Nous devons maintenant y mesurer 727Hz.

Programmons maintenant le format du caractère

Cela se fait en entrant les valeurs appropriées dans le registre 3 ce qui, du même coup, nous sort du mode de programmation du diviseur, donc dlab=0 (il a pensé à tout le constructeur !). Ce registre répond à l'adresse X'A003 et doit contenir les informations suivantes :

bit 0 et bit 1 = sélection du nombre de bits de donnée :

nombre de bits	bit 0	bit 1
5	0	0
6	1	0
7	0	1
8	1	1

bit 2 = nombre de stop bits

si bit 2 à 0, on a : 1 stop bit
si bit 2 à 1, on a : 1,5 stop bit lorsqu'on a une donnée de 5 bits
on a : 2 stop bit lorsqu'on a une donnée de 6,7 ou 8 bits.

bit 3 = parité

si bit 3 à 0, la parité n'est ni générée ni testée.
si bit 3 à 1, la parité est générée et testée.

bit 4 = parité (seulement si bit 3=1)

si bit 4 à 0, parité paire
si bit 4 à 1, parité impaire.

bit 5 et bit 6 = pas utilisés pour nous

bit 7 = accès au diviseur (bit DLAB)

si bit 7 à 0, A000 = registre de données – A001 = registre d'autorisation des interruptions
si bit 7 à 1, A000 = partie basse du diviseur – A001 = partie haute du diviseur.

Dans le cas du baudot, nous voulons :

5 bits de donnée donc	bit 0 = 0
	bit 1 = 0
1,5 stop bit	bit 2 = 1
pas de parité	bit 3 = 0
	bit 4 = 0
	bits 5,6,7 à 0

pas de parité donc bit 3 = 0 – bit 4 = 0 – bits 5,6,7 à 0
Par conséquent, nous obtenons 0000 0100 ce qui fait X'04 qu'il nous faut mettre dans le registre 3. En BASIC :

@≠A003=4

Pour l'ASCII 110 Bd, il faut mettre 6 au lieu de 4 puisque nous voulons là 7 bits de donnée au lieu de 5. En BASIC :

@≠A003=6

C'est tout ! Notre interface série est maintenant prêt à fonctionner.

PROGRAMME DE RÉCEPTION BAUDOT 45 BD

Commençons par le plus simple des programmes, la réception seule.

Il faut bien sûr que l'entrée SERIAL IN de l'interface soit reliée à la sortie d'un démodulateur quelconque. Pour ceux qui n'en sont pas encore équipés, nous en décrirons un le mois prochain.

Le programme à réaliser est en fait simplement un convertisseur 45 Bd baudot à 110 Bd ASCII puisque l'entrée sur le terminal se fait à cette vitesse pour le moment. Il a à remplir les tâches suivantes :

– transformation du code baudot 5 bits au code ASCII 7 bits.
– changement de vitesse de 45Bd à 110Bd. Notons qu'il n'y a pas besoin de mémoire-tampon puisque la vitesse d'entrée est inférieure à celle de sortie.
– fins de lignes automatiques pour éviter que l'oubli du code RETURN par le correspondant fasse « empiler » les caractères en fins de lignes, surtout utile lorsque le terminal est un TELETYPE mécanique.

Le programme est appelé « RTTY RX SEUL 821019 HB9AFO » et tient en 8 lignes de BASIC et 9 routines en langage machine (280 bytes environ). Avant d'entrer dans le détail des explications, commençons par le charger et par le mettre en route !

Note : Nous ne pouvons pas, dans le cadre de MEGAHERTZ donner un cours entier de programmation sur le système MICRO-VON, car cela nous amènerait trop loin... surtout pour les lecteurs que la programmation n'intéresse pas ! Nous avons en préparation, à l'intention des débutants, un cours de programmation BASIC et langage machine sur système MICRO-VON – qui pourra être obtenu via HAMCO. D'autre part, la même source peut fournir aux intéressés le manuel de HB9PBM M. Magnin « Cours de tiny basic » qui détaille les commandes BASIC du 8073.

Les manuels de National, fabricant du microprocesseur sont malheureusement tous en anglais. Il y a :

« Data sheet de l'INS8250 »
« INS8073 NSC tiny basic microinterpreter »
« NSC TINY BASIC MANUAL »
« 70–SERIES MICROPROCESSOR, USERS MANUEL ».

A notre connaissance, l'adresse en France est : National Semiconductor France Expansion 10000, 28, rue de la Redoute, 92260 Fontenay-aux-Roses, Tél. 660.81.40.

Mais revenons à notre programme de réception :

Il faut commencer par entrer les routines ci-dessous dans l'EPROM au moyen de l'utility de la carte de programmation, de préférence une à une. Il n'y a pas besoin d'entrer les X'FF puisque ce sont ces valeurs qui remplissent l'EPROM lorsqu'elle est vierge, donc vide. En réalité, au lieu d'écrire des « 1 » dans une EPROM vierge remplie de « 0 », on écrit des « 0 » dans une EPROM remplie de « 1 » ! On peut tirer parti de cette particularité lorsqu'on doit corriger un bit dans un byte par exemple. Si par hasard ce bit doit être mis au « 0 », on peut le faire sans devoir tout reprogrammer.

Une fois la partie « langage machine » dans l'EPROM, et bien contrôlée, taper le programme BASIC principal dès l'adresse X'1100 (NEW≠1100 et NEW) :

```
1 CLEAR
10 REM RTTY2, RX SEUL, 821026 HB9AFO
20 LINK≠8720
30 @ ≠A001=0
35 @ ≠A005=0
40 @ ≠17FF=12
50 STAT=1
60 ON2,200
70 @ ≠A001=1
80 GOTO 50
200 LINK≠86C0
210 RETURN
```

Si un démodulateur est branché sur l'entrée SERIAL IN de l'interface série et un récepteur connecté, on peut recevoir du baudot 45Bd en faisant : RUN.

Pour que cela fonctionne, il faut évidemment être correctement calé sur son correspondant et que sa manipulation (on parle aussi de manipulation en RTTY) soit dans le bon sens. Si ce n'est pas le cas, vous lirez des caractères aléatoires sur votre écran. C'est la raison pour laquelle il vaut mieux faire son apprentissage de l'écoute RTTY sur VHF où cela se passe généralement en FM (la RTTY module l'émetteur via la prise micro). Il n'y a, par conséquent, pas de problème de calage et de QRM.

Une fois le programme testé, on peut le transférer dans l'EPROM de la carte microprocessor à l'aide de la commande « P » de l'utility. Lorsque maintenant on enclenche le système, on saute directement dans le programme de réception RTTY, sans aucune manœuvre.

HB9AFO

Note : Les « paresseux » peuvent obtenir l'EPROM RTTY programmée et testée ainsi que tout le matériel chez : HAMCO, case postale, CH-1024 Ecublens (Suisse).

PROGRAMME RTTY2 : routines en langage machine pour la réception

1. SA:8520 fins de lignes automatiques

   8520 26 F0 17 C2 0E FC OD 7C 05 C4 00 CA 00 5C 00 C2
   8530 00 F4 01 CA 00 FC 48 6C 01 5C C4 00 CA 00 18 5C

2. SA:86CO links réception

   86C0 00 00 26 FC 17 C2 00 E4 80 6C 15 00 00 20 CF 87
   86D0 20 3F 87 20 6F 87 20 EF 87 20 1F 85 5C 00 00 00
   86E0 00 00 20 CF 87 20 DF 87 20 EF 87 20 1F 85 5C FF

3. SA:8720 initialisation baudot 45bd

   8720 00 00 26 00 A0 C4 80 CA 03 C4 7D CA 00 C4 15 CA
   8730 01 C4 04 CA 03 26 FC 17 C4 00 CA 00 00 00 5C 00

4. SA:8740 aiguillage baudot-ASCII

   8740 00 00 26 FC 17 C2 01 E4 1F 7C 07 C4 00 CA 00 00
   8750 00 5C E4 1F E4 1B 6C 04 E4 1B 74 04 C4 20 CA 0
   8760 00 00 5C 00 00 00 00 00 00 00 00 00 00 00 00 00

5. SA:8770 conversion baudot-ASCII

   8770 00 00 26 FC 17 C2 01 D4 1F F2 00 F4 90 01 C4 87
   8780 01 4E C2 00 26 FE 17 CA 00 00 00 5C 00 00 00 00

6. SA:8790 table baudot-ASCII

   8790 00 45 0A 41 20 53 49 55 0D 44 52 4A 4E 46 43 4B
   87A0 54 5A 4C 57 48 59 50 51 4F 42 47 00 4D 58 56 00
   87B0 00 33 0A 2D 20 27 38 37 0D 2A 34 80 2C 00 3A 28
   87C0 35 2B 29 32 00 36 30 31 39 3F 00 00 2E 2F 3D 00

7. SA:87DO stockage du caractère reçu

   87D0 00 00 26 00 A0 C2 00 26 FD 17 CA 00 00 00 5C 00

8. SA:87EO court-circuit pour réception ASCII

   87E0 00 00 26 FD 17 C2 00 CA 01 00 00 5C 00 00 00 00

9. SA:87FO sortie série sur terminal

   87F0 26 FE 17 C2 00 17 5C FF FF FF FF FF FF FF FF FF

Toutes ces routines s'appellent à partir du BASIC grâce à l'instruction LINK≠...

DISPONIBLE

A LA

REDACTION

20FF franco