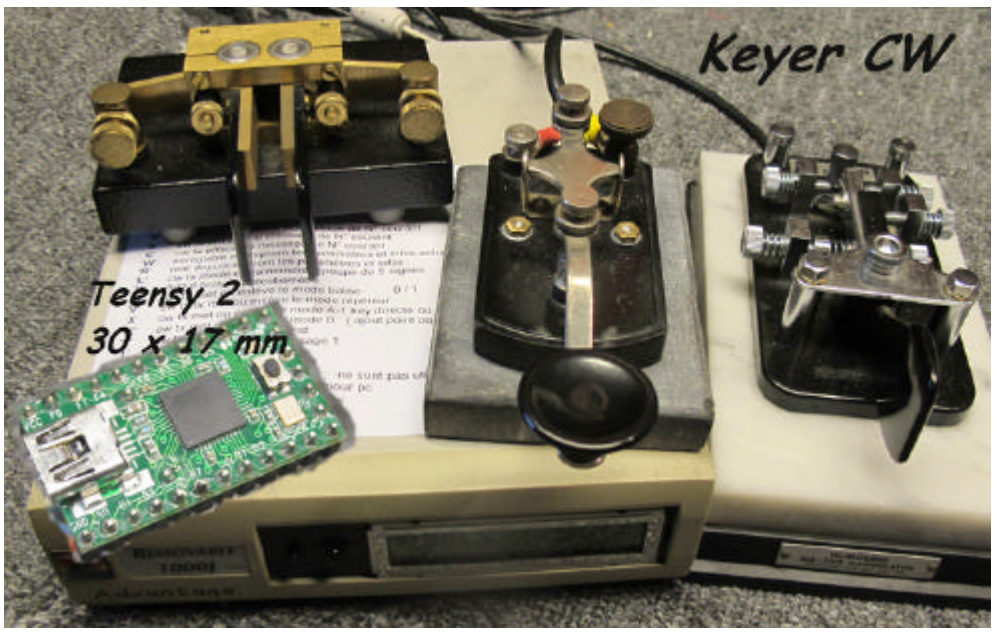
A-MORSE par F5BEZ 2011

Cette réalisation propose des fonctions CW « morse » avancées autonomes avec un microprocesseur atmega32u4 ‘ Teensy 2 ’



Caractéristiques

- Teensy 2, autonome, LCD 1 ligne de 16 caractères, PC.

- Fonctions :

- RX CW
- **Manipulateur classique et interface**
- **Entrée audio, directe ou micro. Filtre étroit à PLL**
- **Décodage CW auto adaptatif *, choix d'algorithmes ***
- **Affichages sur LCD et sortie USB pour PC**
- **Enregistrement de messages**

- TX CW « Keyer »
- **Manipulateur Iambique modes A et B**
- **Affichage décodage sur LCD**
- **Vitesse ajustable**
- **Intervalle lettres réglable**
- **Sortie de messages**
- **Sorties LED , Buzzer , positive, négative**
- **Sorties cw audio, TX- PTT**

- Annexes
- **mémo en eeprom intégrée**
- **mode entraînement CW**
- **mode balise simple, avec paramètres**
- **mode répondeur simple * , report S-mètre**

- Commandes
- **Un seul bouton : Smart Key ***
- **Gestion depuis logiciel Hyperterminal sur PC**
- **Gestion depuis logiciel « A-MORSE » sur PC**

Décodage CW

Principe et méthode

Retour sur un décodage CW auto adaptatif …

Le décodage informatisé d'un signal morse semble être un exercice de traitement assez simple, si l'on considère un signal net, des rapports de temps parfaits et une vitesse stable… Un signal morse nominal est une suite de points et de traits dans une séquence de temps précise théorique. Un trait fait 3 fois la durée d'un point. Un espace fait la durée d'un point. Un espace triple termine la lettre, et un espace quintuple (septuple recommandé) termine un mot. Mais en pratique cette séquence n'est pas parfaite, et les irrégularités nombreuses sont dans l'ordre : Une manipulation manuelle, avec une clé classique, est à l'image de l'opérateur. Nous avons alors du séquençage trait - point ' personnalisé ', des espacements et traits irréguliers, une vitesse variable et des lettres collées !… Une manipulation manuelle avec une clé iambique est déjà plus respectueuse des temps dans la lettre, mais les espacements, et donc la vitesse, restent irréguliers. Une séquence générée par logiciel serait plus rigoureuse. De plus nous aurons quantités d'abréviations, de signes et de groupes, ce qui revient en pratique, à allonger l'alphabet morse. Enfin nous aurons une étape de mise en forme du signal reçu, point déterminant à lui seul. Voir à 'réception'. *Un système informatisé ne pourra pas rivaliser avec la concentration d'un humain entraîné capable de dissocier un petit signal faible, non conforme en timing, dans le qrm , le qsb et près d'autres cw qro…*
(REF 1994 f5bez) Mais c'est un challenge intéressant !

Le sujet a été abordé régulièrement, ici on ne reprend donc que les base générales de descriptions anciennes, du point de vue micro-informatique, mais toujours d'actualité *(QST de 1971 , I/0 de 1977 W9KPT, W6PRO)* qui m'ont servi de base, en 1979, pour réaliser une version, en assembleur, sur microprocesseur Intel 8080, que j'étudiais alors. On repart du ' flowchart ' pour une adaptation actualisée en langage ' C ' simplifié sur un petit microprocesseur Amtel. Il peut être exploité par interruption, par exemple une analyse de compteurs chaque mS par un ' timer ', ou plus simplement dans une boucle, elle-même cadencée par pas de 1ms. Dans ce contexte, et pour démarrer les explications, nous prendrons, pour commencer, le cas d'une suite parfaite normale, puis nous verrons comment le processus s'adapte aux variations des points et traits et de vitesse.

Une affaire de temps

Si nous prenons notre manipulateur (Key) nous avons 2 états : repos (Up) et bas (Down) dont les durées sont à mesurer. Avec un pas de 1ms, nous pouvons ' compter ' du temps. Pour une cadence donnée, si on compare les compteurs du temps ' clé basse ' écoulé (ELapsed) avec un temps de référence, on peut déterminer s'il s'agit d'un point (dot) ou d'un trait (dash). De même, ' clé haute ' , s'il s'agit d'un espace dans une lettre ou une attente (wait) identifiant la fin d'une lettre, ou si plus d'un mot.

Initialisations
Pour débuter il faut pré positionner ces 2 temps de référence : la durée relative dash dot (DD) et la durée relative Lettre eSpacement (LS) ainsi que les 2 tables des moyennes, correspondant à une vitesse arbitraire de 10 mots minute environ.

Début (espace)
Une attente de fin d'un mot, inscrit un **espace** de séparation. Au début, tant que la clé n'est pas activée, on tourne dans une boucle qui appelle la fonction ' **status** ' laquelle compte le temps écoulé EL et positionnera 2 drapeaux (flag).

Key basse
On commence par mettre à 0 le compteur EL. C'est donc le début d'un **point** ou d'un **trait**. Quand la clé est relevé, le flag DD est testé et selon le cas on aiguille vers le bloc ' point ' ou ' trait ' pour continuer à ' **clé haute** '.

Point
Le temps écoulé EL serait la durée du point actuel ? Alors le nouveau temps DD est calculé comme étant son double. Ce qui détermine la frontière à 2/3 du trait qui serait 3 fois celle du point… Le code en cours de lettre est doublé.

Trait
Le temps écoulé EL serait la durée du trait actuel ? Alors le nouveau temps LS est calculé comme étant la moitié de ce temps EL plus le dernier DD. Ce qui détermine la frontière à un peu plus des 2/3 (5/6) d'un trait pour la fin d'une lettre. Le nouveau temps DD est positionné à la moitié de ce dernier trait. Voir à 'algorithme' plus loin.
Le code en cours de lettre est doublé et incrémenté de 1.

Key haute (cycle dans une lettre)
Les 2 fonctions précédentes ont fait progresser le ' code ' , il faut donc tester qu'il ne déborde pas des possibilités de la table et le diminuer alors arbitrairement de moitié.
Puisque la clé est haute, on remet le compteur EL à 0, et on boucle en attente (wait). Maintenant, soit on continue dans la lettre morse en cours, et on repart à 'clé basse ', soit le flag LS est vrai et le temps key haute a atteint une fin de lettre.

Key haute (cycle fin de lettre)
On exploite le code et on affiche le **caractère** décodé. Le temps EL est remis à 0 et on boucle en attente encore. Maintenant soit on continue avec le début d'une nouvelle lettre, et on repart à clé basse, soit le flag LS est vrai à nouveau. Le temps key haute a atteint le **double**, on déduit une fin de mot et on repart avec un espace de séparation en début de boucle.

Status

Lit l'état de la clé et compte le temps par pas de 1ms dans le compteur de temps écoulé EL . Ce comptage est limité à une valeur maximum. Reset les 2 drapeaux (flags) et les positionne lorsque ce temps EL dépasse les temps DD et LS respectivement. Le signal bas réel de la clé est compté et n'est pris en compte qu'après quelques cycles pour intégrer les rebonds. La LED interne au chip répercute le signal et une sortie monitor peut commander un buzzer piezo à auto – oscillateur intégré. Les codes ascii CR-LF sont envoyés s'il y a plus de n caractères.

Délai

C'est une boucle ou une fonction intégrée qui compte un temps mort de 1ms.
Dans une version par interruption, le processeur peut faire autre chose comme analyser directement une entrée audio sur une voie analogique, et appeler chaque ms nos fonctions. Ici il ne fait rien d'autre que du e-jogging…

Key

Lit l'état physique de l'entrée F0 d'un manipulateur classique ou d'un module de mise en forme audio. Lit l'état physique des entrées F1 point et F4 trait d'un ' keyer '.

Manipulateurs à 1 ou 2 poignées ('paddles')

Type alternatif, contacts simultanés impossible, sans pincement, dit non-iambique
Type à 2 poignées et contacts simultanés possibles par ' pincement ' : iambique
Le trait est à gauche et le point à droite, mais les entrées peuvent être inversées !
Documentations générales (annexe 1) et observations.

En mode A-1 l'état direct des appels sont répercuté, mode simple d'entraînement.
En mode A-2 par front (trigger) les appels impulsionnels instantanés décalés sont mémorisé, c'est à dire qu'un pulse sur le point pendant la durée du trait en cours sera traité à la suite : 'N' . Mode normal rapide (Cde 'X')

Un appel impulsionnel simultané des 2 entrées (pincement) génère d'abord un trait puis un point 'N' et si maintient un 'C' ou ' ;'. Au relâchement pendant le 2nd trait
En mode A un 'K' , (fin)
En mode B un 'C' (un point est rajouté) (Cde 'K')
ou inversement un trait si relâchement sur un point.

La vitesse est prédéfinie au départ mais peut être modifiée simplement par un message CQ sur la key simple, où la durée moyenne du point sera prise en compte comme nouvelle référence de cadence. (si mode 'F' à 2, sinon voir Cde 'B' 'P' 'J')

Monitor

Sort un état physique suiveur de key, pour un contrôle LED et auditif sur buzzer, via un pot de niveau bf éventuellement.

Décodage

En entrée le ‘ code ‘ calculé qui servira d’index, en sortie le caractère ascii
pointé par cet index dans une table pré chargée. Le code est remis à 1 pour le traitement de la lettre suivante.

1 initialement **1** *(méthode w9kpt)*
2 'E', si 1 dot 1 * 2 = **2**
3 'T', (si 1 dash 1 * 2 + 1 = 3)
4 'I', seconde passe + 1 dot **2** * 2 = 4
5 'A', seconde passe + 1 dash **2** * 2 + 1 = 5 etc

Cette méthode intègre les ponctuations, les abréviations et les groupes particuliers de l’art de la CW… Les pseudo caractères déduits seront transposés avec une autre table et affiché entre crochets. expl [AR] *(f1bez)* (voie USB PC seulement)

Print

Sort le caractère courant vers le PC et - ou vers l’afficheur LCD.

Auto adaptation

Maintenant vous avez compris comment, dans les blocs point et trait, on recalcule dynamiquement DD et LS sur la base des derniers points et traits. Cela ne suffit pas.
Les notes de W9KPT rappellent ceci :
La tendance, en manipulation manuelle, porte surtout sur la durée du trait et apparaît usuellement à la fin d’un mot. Cela précède souvent une pause plus longue où l’opérateur collecte ses pensées pour la suite. La décision lettre espacement est très influencée par la durée des traits. Les effets des espacements longs sont atténués.

Le module fonctionne correctement si l’on reste dans une manip calme et nette.
Au départ, le programme, peut dérouter en affichant nombres de E et de T. Il faut qu’il reçoive des L, P,J .. pour qu’il ‘ cale ‘ bien ses premiers rapports point trait DD.
Une manip avec erreurs courtes ou parasitée tends à accélérer ce rapport DD et donc la vitesse relative, et à rythme égal, on affichera encore des E et T !!!
Mes expérimentations visent à moyenner les temps DD et LS avec les EL des n derniers points et traits reçus pour ‘ adoucir ‘ les variations instantanées de DD et LS.
Le taux de complexité rajouté améliore un peu… *(f1bez 1979)*

Voir les commandes Z P J F et les paramètres
cw-adaptative, type-algo, nbre-rebonds,

Algorithmes

L'algo 1 est de type classique et prends **DD** et **LS** comme étant la moitié de la moyenne des n derniers traits. Le 'QRQ' s'adapte mais le 'QRS ' est moins efficace.
L'algo 2 effectue une moyenne (*W6PRO) ,* avec ou sans le moyennage traits
Les algos 3 et 5 sont des tests d'optimisation , dont ¾ dash *(Petit)*
L'algo 4, détermine une plus faible valeur DD et LS et tient compte du point.
((EL / 2) + DD) *(W9KPT)*
Tous les algos n'utilisent pas le point moyen , mais que le trait moyen
C'est ouvert pour tester 'vos' méthodes DD LS et LS fin de mot…(src N° Algo > 5)

Prise en compte

Algos 1 à 9 pour prise en compte immédiate sur trait, sur point ou les 2 …
Algos 11 à 19 ce sont les mêmes +10 mais placés à keystart à l'espace de mot

Validation et contrôles

Avec les manipulateurs classiques 'pioche' et à paddles ..

Pour la ' réception ', avant de tester une sortie bf Rx décamétrique, un PC auxiliaire avec CW-PLAYER de FDQM permet de générer en boucle de la CW ' calibrée '.
Les réglages et les paramètres vont nous permettent de tester AMORSE.
Relier la sortie casque du PC à l'entrée PLL.
Options / texte / rejouer en boucle mettre 10 ' E ' à 20 mots / mn
Choisir une note à 900Hz , un niveau à –12 . Un scope synchro voie 1 sur entrée audio HP et voie 2 sur la sortie Key permet de caler et d'optimiser le PLL NE 567.
Ensuite ' jouer ' un texte ou les séries aléatoires et vérifier le décodage…
Les options vitesse, espacement caractères, poids traits-points, et les choix d'algo permettent ensuite de tester les limites de AMORSE…
La qualité en phase du filtre influencera la vitesse cw maxi décodée… 30..35
Le mode sans ' moyennage' permet de borner ces limites pour essais et exercices.

Encodage CW

Voir diagramme CW TX

Principe

En entrée un caractère **ASCII** , en sortie les signaux ci-dessous.
ascii 48d 30h '0' , 65d 41h 'A' , 84d 54h 'T' etc
La table contient les ponctuations standards, les nombres 0 .. 9, les lettres A .. Z
et à la suite certaines abréviations AR VA AS AT(@)

On recherche de la gauche MSB vers la droite LSB le le **1er 0** : soit le bit de départ :
S'il s'agit d'un 1 on boucle, s'il s'agit d'un 0 on positionne un drapeau indiquant que les bits suivants sont alors **1** pour un **point** et **0** pour un **trait**. *(f1bez 1979)*
Ceci permet de coder (de droite à gauche) les caractères courants (6 signes cw)

Exemples :
0xFD = **1111101** 1 = **.** (E) , 0xFC = **11111100** 0 = **-** (T) , 0xC0 = **11000000** **-----** (0)
Un point (1 unité) ou un trait (3 unités) sont suivi de fait d'une attente (1 unité).
Une fin de lettre est suivie d'une attente de 2 unités , soit au total 3 unités. (cf trait)
La fonction 'envoie message' appelle cette fonction 'encode' itérativement.
Un espace appelle la fonction 'attente' pour 4 unités, soit au total 5, qui est la séparation minimum de fin de mot.

Signaux en émission CW

Sortie sur 1 pour LED interne Teensy 2 (D6) voir liste Inp/Out
Sortie sur 1 pour LED et Buzzer avec auto-oscillateur (via un pot ..))
Sortie sur 0 pour action Key sur Tx avec Break-In
Facultatif pour Tx mais prévu pour d'autres applications (balise) :
Sortie Audio signal carré de 1khz vers un filtre passe-bas R-C
Sortie Mox sur 0 PTT, pour passage Tx avec un temps post-délai cw
et un temps de fin lettre inclus.

Manipulateur ‘keyer’

Postulats

Manipulateur à **2** contacts à 1 ou 2 poignées Key no-iambic ou iambic (annexe 1)
Base : ‘**point**’ à droite (le pouce) et ‘**trait**’ à gauche (l’index) pour droitier.
Si un contact est fermé (impulsion) : le point (ou le trait) part.
Si un contact est maintenu : répétition du point (ou du trait).
Si un contact est changé pendant l’envoi du trait (ou du point) :
Mémorisation de l’action et envoi du point (ou du trait) à la suite. Mode A - 2
Si les 2 contacts sont fermés ensemble, le trait prévaut (envoi de ‘N’ et non ‘A’).
Si le pincement est maintenu, la séquence continue (envoi de ‘C’ , et de ‘ ; ’)
Si le pincement est relâché sur un point ou un trait : fin de la lettre en mode A
Si le pincement est relâché sur un point : ajout d’un trait (ou inversement) en mode B
Le timing respecte 1 unité de temps pour le point et une attente d’autant.
et 3 unités de temps pour le trait et une attente d’une unité également.
Un temps d’une unité à la suite (soit 2 unités) permet la détermination
soit de continuer dans la lettre, soit d’indiquer une fin de lettre.
Si un contact est fermé dans ce temps on continue la lettre.
Si c’est après ce temps, on mémorise et diffère la suite dans 1 unité.
Ceci permet d’avoir une séparation synchrone de 3 unités (trait).
Un temps plus long, de fin de mot, fait retomber le drapeau mox-ptt.

Principe

voir diagramme temporel

Cette partie utilise des séquences combinatoires où l’on compte des cycles de temps et où l’on positionne et teste différents drapeaux (flags) qui produiront la CW cadencée selon les règles indiquées.

L’unité de temps est ajustable par la vitesse CW : voir commandes ‘P’ ‘J’ et ‘B’

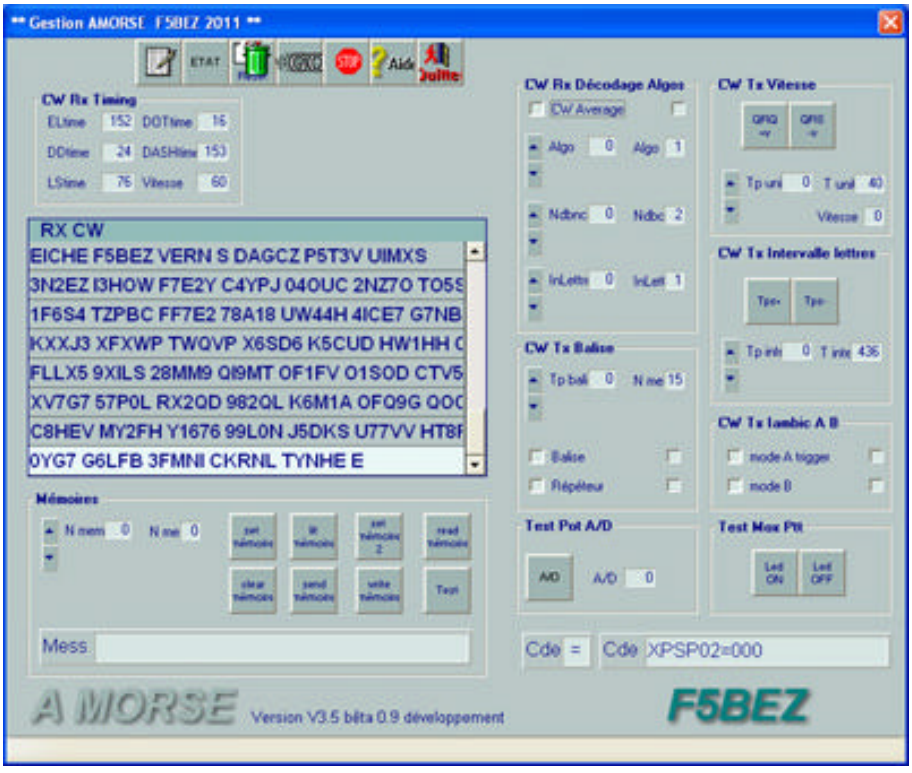
Affichages et mémoires

Affichage sur LCD

Basé sur un module standard 1 ligne de **16** caractères
de type HD44780 et de fonctions auteur réécrites pour le Teensy
NB un module 2 x 16 compatible peut-être utilisé. src
NB il existe aussi, sur le site, une bibliothèque pour ce modèle LCD, non testée.
La mémoire **1** garde les 32 derniers signes.
La commande N permet de refaire défiler les 32 derniers caractères.
La mémoire **2** est prévue pour un message d'appel répétitif Cde S
La mémoire **3** est prévu pour la fonction 'balise' Cde Q
La mémoire **4** est prévue pour la fonction 'répéteur' Cde V

Affichage sur PC

Utilise les 2 fonctions disponibles en 'define' dans 'print.h' pour les sorties.
La liaison USB voie série virtuelle est testée avec « Hyperterminal »
Un logiciel d'affichage et de gestion A-MORSE complètera ce projet
Les 2 cas permettent un enregistrement continu.



A MORSE . EXE prototype…

Commandes

smart button

On utilise un seul bouton ‘ intelligent ’ Key-cde plutôt que 36 boutons !
Comme on dispose d’une ressource de type ‘Key’ et d’un décodage de séquences, alors utilisons nos lettres Morse pour les commandes de ce logiciel…

Les lettres ‘E’ et ‘T’ sont écartées :
Une commande impulsion (‘E’ ou ‘T’) arrête certaines commandes.
Les séquences courtes ‘I’ , ‘A’ , ‘M’ ne sont pas utilisées.
Un maintien prolongé stoppe momentanément le défilement (voir ‘N’).
Les autres lettres et chiffres CW sont exploités pour les fonctionnalités.
Les accentuations sont utilisées comme commandes par le programme externe.
Les commandes du bouton Key-cde sont basées sur la vitesse de 10 M/mn, indépendamment de la vitesse CW en cours, de façon à rester accessibles.
Une fiche sur le coffret rappellera ces commandes…

Changement vitesse CW Commandes **‘Z’ ‘B’ ‘P’ ‘J’**

Affichage de la vitesse courante Commande **‘Z’**
Appelé aussi au démarrage à froid
Affiche la vitesse en **Mots/Mn** (W/mn) approximativement
Base : mots de 5 caractères moyennés sur 1 mn

Réglage par potentiomètre Commande **‘B’**
Génère ‘ 73 ‘ chaque seconde et affiche la vitesse. Fin par ‘T’ sur le bt cde
NB option automatique si changement de la valeur pot de + - 10% . src

- Augmenter **QRQ : ‘P’** (Plus vite) et Diminuer **QRS : ‘J’** ( )
Affichage de la vitesse modifiée (+ - 10%) puis réaffichage de
la réception ou de la manipulation en cours.

Rangement vitesse courante en eeprom **: ‘W’** (write)
Relecture vitesse **: ‘R’** (read) et au démarrage à froid
NB : ‘W’ et ‘R’ concernent aussi les mémoires et paramètres

Lecture potentiomètre Commande **‘Y’**
Affiche la valeur relative du pot (1-1000), puis en W/mn (‘Z’)
Le potentiomètre sera gradué et ainsi étalonné à l’usage.

Key ‘pioche’ en mode ‘F’ à 2 ‘entraînement’ (non par défaut)
La manip des x derniers caractères, notamment la durée moyenne des points, est prise en compte. Vérification de l’effet par ‘Z’ et par la manip Key- iambic
NB sans ce mode le temps de fin de lettre est réglable voir ‘O’ et ‘G’

NB : les commandes du bouton Key-cde restent basées sur la vitesse de 10 M/mn indépendamment de la vitesse CW en cours, de façon à rester accessibles.

Mode entraînement CW Commande **'L'** (Learn)

Génération aléatoire de groupes de **5** caractères
Lettres, chiffres . Option dosage chiffres, (option ponctuations voir src)
Vitesse CW variable : commandes préalables 'P' (QRQ) 'J' (QRS)
Intervalle temps variable indépendamment : 'F' à 2 , commandes '0' (+) 'G' (-)
Ceci permet un entraînement QRQ avec un temps inter-lettres au choix.
Affichage LCD après la séquence, pause de **3** sec, pour lecture.
Affichage PC après la séquence, pour correction finale.
Fin par le bt-commande 'E' (ou 'T') pendant cette pause.
Une autre commande 'L' relance la même séquence.
Une action Key pioche de quelques lettres préalables lancera une autre séquence

Mode balise CW Commande **'Q'** (Qui ..)
Envoie du message **3** cycliquement chaque x minutes (voir paramètre 10)
Fin par bt-commande 'E' (ou 'T') pendant les pauses.
NB Le message peut être complété avec des paramètres (Tx , météo) src

Mode répondeur CW Commande **'V'** (Vérif..)
Envoie du message **3** sur ' réception ' d'une séquence prédéfinie
en mémoire en message **4** . Fin par bt-commande 'E' (ou 'T')
NB Le message peut-être complété avec la valeur S-mètre du Rx src
si sortie physique DC s-mètre intégrée et mémorisée sur cellule R/C…

Mode test message CW Commande **' ?'** (devinez …)
Message test CW prédéfinie ou personnalisable src

Mode espace de fin lettre Commande **'F'** (temps de Fin lettre)
'F' bascule de l'un à l'autre 1 à 2 NB options cas 3 .. 4 …

Réception
Le mode 1 '**normal**' tient compte d'un temps espace lettre nominal (3 unités de temps ou trait) au-delà , un espace de fin de mot est ajouté.
Le mode 2 '**allongé**' double ce temps et permet de maniper sans rajouter trop d'espaces dans le mot. Ce mode ' allongé ' ou d'entraînement permet aussi :
- la prise en compte de la vitesse CW par la Key-pioche.
- la prise en compte du paramètre intervalle entre lettres Cdes 'O' 'G' , pour les envoies CW 'S' , test ' ?' et entraînement 'L'

Emission en mode Keyer
Le mode 1 normal traite un intervalle de 3 unités (trait) synchrone si au moins 1 unité (point) est dépassé, sinon continu dans la lettre.
Le mode 2 permet d'ajouter un temps d'intervalle au choix
Paramètre 22 et commandes ' O ' et ' G ' .
Ce temps est synchrone, la 1ère action (point ou trait) de la lettre suivante peut être ' engagée ' mais commencera après le temps d'intervalle défini, la suite à l'oreille…

Mémoires

Mode changement de mémoire Commande **'1' '2' '3' '4' …** '9'
Changement de mémoire courante

Mode effacement mémoire Commande **'C'** (Clear)
Effacement de la mémoire courante

Mode chargement mémoires Commande **'R'** (Read)
Lecture eeprom des mémoires **2** à **4** et de la vitesse CW courante.
Appelé au démarrage à froid.

Mode écriture mémoires Commande **'W'** (Write)
Rangement eeprom des mémoires **2** à **4** et de la vitesse CW courante
Manipez votre message d'appel en 2 puis faire ce rangement

Mode contrôle mémoire Commande **'N'** ( )
Lecture contrôle (défilement LCD) de la mémoire courante
Un maintien prolongé Key-cde stoppe momentanément le défilement.
Le buffer en cours de réception ou de manipulation est réaffiché ensuite.

Mode envoie mémoire Commande **'S'** (Send)
Envoie en CW de la mémoire courante. (dont espace = 1 unité de temps)

Affectation des mémoires :
La mémoire **1** 'suit' la réception ou la manipulation, mémoire par défaut au départ.
La mémoire **2** est prévue pour un message d'appel prédéfini. Voir 'S'
La mémoire **3** est prévue pour un message balise.
La mémoire **4** est prévue pour l'identifiant en mode répondeur.
Les mémoires 3 et 4 peuvent servir comme pré-messages tout comme 2
Les mémoires 2, 3 et 4 sont initialisées au démarrage à chaud depuis l'eeprom
Les mémoires suivantes 5..9 sont 'open' libres (selon ram Teensy src).

NB les notations src indiquent des options possibles par adaptations des sources.

Protocole

Commandes à 1 lettre

en **CW** par le bouton de commande en façade (smart key)
ou par la liaison série suivi de <cr>, sous hyperterminal ou par amorse.exe

'**Z**' cw affiche la vitesse actuelle en mots/minute
'**B**' cw tx réglage vitesse par le pot env 6 .. 40 W/Mn
'**P**' cw tx plus vite de +10% QRQ
'**J**' cw tx moins vite de -10% QRS
'**O**' cw tx temps intervalle lettre +20% (si mode F à 2)
'**G**' cw tx temps intervalle lettre -20% idem
'**F**' cw rx temps décision espace de fin de lettre normal ou allongé-entraînement
'**N**' cw rx lire et afficher le message de N° courant
'**S**' cw tx envoyer le message de N° courant
'**C**' cw tx effacer le message de N° courant
'**W**' enregistre en eeprom les paramètres et infos actuels
'**R**' relit depuis eeprom les paramètres et infos
'**L**' cw tx mode entraînement : groupe de 5 signes
'**Y**' A/D-6 lecture potentiomètre
'**Q**' cw tx met ou enlève le mode balise 0 / 1
'**V**' cw rx tx met ou enlève le mode répéteur
'**X**' cw tx met ou enlève le mode A-1 key directe ou A-2 key pulse
'**K**' cw tx met ou enlève le mode B (ajout point ou trait)
'**?**' cw tx envoie un message test src
'**1**' sélectionne le N° de message 1
'**2**' '**3**' '**4**' idem 5 .. 9 src
'**E**' '**T**' ou ' impulsion ' réservés comme fin des fonctions ? B S L Q
'I' 'M' 'A' 'D' 'H' ne sont pas utilisées et affichent ' error '
'**+**' met ou enlève le mode trace pour pc
'**=**' affiche les infos actuelles vers pc **' : '** affiche la version
'/' , '.' , ' ;' , ' " ' , ' - ' , ' (' , ') ' sont utilisées en cde externes

Commandes à 3 ou 4 signes

suivies de <cr> Entrée par la liaison série USB, sous hyperterminal ou autre
Lire une entrée <lettre port A..F><numéro 0..7>< ?> **F0?<cr>** retourne 0 ou 1
Positionner une sortie <lettre port A..F><numéro 0..7><=><0> ou <1> **D3=1<cr>**

Commandes étendues

suivies de <cr> par la liaison série USB, sous 'hyperterminal'
ou gérées directement en IHM par boutons à cliquer avec amorse.exe
En-tête **X** (e**X**tented) , **P** (**P**rotocol 1)
Ordre **S** (**S**et) positionner ou **R** (**R**ead) lire
P (**P**arameter) ou **M** (**M**essage)
N° paramètre **01**..99 maxi .. ou N° message 1..4 maxi ..
? lire ou **=** mettre nnn données **000**..999 ou **TEXTE** (majuscules)

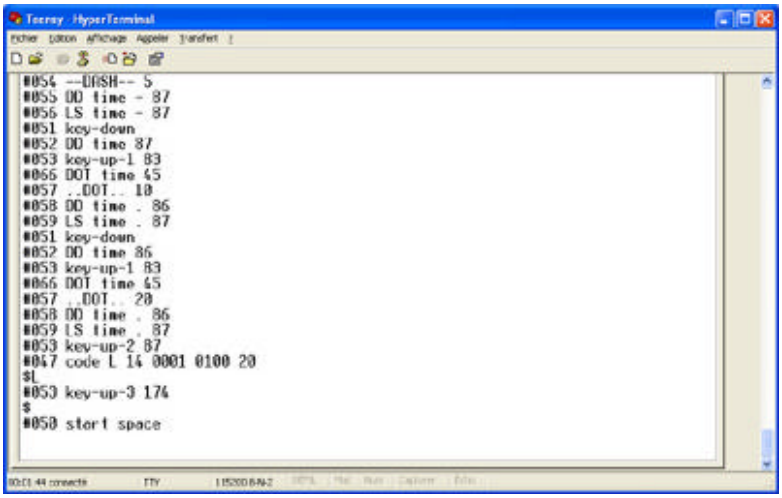
Exemples :

XPSP07=123 XPRM2? XPRP03? XPSP01=001 XPSM2=CQ CQ DE F4KIO K
Les valeurs envoyées doivent être ‘ acceptables ‘ par le programme !

Paramètres (XPSP02=002)

02	moyennage adaptatif	000.. 001		
03	type algo	001.. 005	ou 010.. 015	src
04	lissage rebonds	001.. 010		
05	fin lettre normal / allongé	001.. 002		
06	mode A simple / trigger	000.. 001		
07	mode B non / oui	000.. 001		
08	réservé 1	000.. 001		
09	réservé 2	000.. 001		
10	temps cycle balise (Mn)	001.. 060		
21	unité temps ms (point)	020.. 200		
22	tps intervalle lettres (ms)	010.. 900		

Positionner le mode éducation ‘ F ‘ à 2 (prise en compte du paramètre suivant)
Positionner un intervalle de 400ms entre lettres XPSP22=400
Positionner une vitesse à 14 M/mn XPSP21=070 ou ‘ P ‘ ou ‘ J ‘
Lancer le mode ‘ ? ‘ ou ‘ S ‘ message ou ‘ L ‘ entraînement
Positionner un message de balise XPSM3=BALISE
Modifier le temps de balise XPSP010=010 et lancer avec ‘ Q ‘



Hyperterminal

Traces

Les lettres, et les groupes de 5 signes en mode éducation L, sont envoyés directement sans en-tête. Un retour à la ligne (cr/lf) est inséré, au-delà de 30 signes, s'il y a un espace cw de fin de mot. Le départ et la commande '**+'** génèrent des traces commençant par **#** . Si la commande **+** est active, tous les évènements sont tracé. Les signes commencent alors par **$**. Les lignes finissent par **%**
Le programme externe AMORSE les utilise ainsi que leurs **numéros** repères. Ses enregistrements sont horodatés, pour une analyse différée.

Extraits

Départ

```
#020 CW F5BEZ 2011  V3.5
#080 command R     // relecture  eeprom
#022 start mode : 1
#016 trace is 1
#080 command Z
#071 speed W/mn 10
```

Manip ' E '

```
#052 key-down
#053 key-up
#058 ..DOT..
#061 EL time 88
#062 DD time 178
#063 LS time 281
#047 code E 02 0000 0010 2
$E
```

Manip ' T ' long

```
#052 key-down
#053 key-up
#054 --DASH--
#055 EL time 496    ( mS )
#056 DD time 158
#057 LS time 316
#047 code T 03 0000 0011 3
$T
```

Envoie de Z

```
#048 code Z E3 1110 0011 227
#041  -
#041  -
#042  .
#042  .
```

Test commandes

```
#034 mcde  XPSM4=ESSAI
#034 mcde  XPRM4?
#092 ESSAI
```

Hardware

Voir les plans et la liste des entrées-sorties. pour le minimum :
Câblez la ‘ **key** ‘ entre **F0 pin 21** et la masse **GND**.
Câblez la sortie open collecteur du module filtre en parallèle.
Câblez le **buzzer D1**
Et utiliser les commandes et les lectures via le mode hyperterminal.
Câblez les entrées F1 F4 pour tester le mode Iambic Keyer …
Câbler le bt ‘ smart ‘ F5
Câblez le LCD facultatif, le prg peut fonctionner sans cet affichage !
http://www.pjrc.com/teensy/td_libs_GLCD.html : exemple de brochage
LCD 7 à 14 (data) : B0 à B7, D/I 4 : D0 , (r/w à 0) , E (strobe) 6 : C6
display lcd (seiko de récup) modèle quasi standard 1 : 0v , 2 : +5v ,
3 : (0v) contraste. Chip HD44780

Software

Préalables

Vous devez télécharger et installer les outils du site :
http://www.pjrc.com/teensy/index.html
- Vous avez chargé et testé le 1er petit programme **‘ blinky .hex ‘**
- Lancer **teensyloader**(xp).exe (ou …vista.exe)
  http://www.pjrc.com/teensy/**loader**.html
- File / open HEX file **blinky.hex** . Cliquez ‘ automatic mode ‘ qui passe en vert
- Appuyer sur le petit bouton du Teensy, ce qui va ‘ charger ‘ le .hex
- Vous le tester avec **Hid_listen** (qui ne nécessite pas d’installation)
  http://www.pjrc.com/teensy/**hid_listen**.html : Rx serial_usb en fenêtre DOS

Installation simple

Si vous ne souhaitez pas installer les outils de développement, vous pouvez charger le Teensy avec **amorseV35.hex**, avec les préalables teensyloader( ).exe vu plus haut et installer **serial_install.exe** et utiliser **HyperTerminal** Windows

- Lancer **teensyloader**(xp).exe (ou …vista.exe)
  http://www.pjrc.com/teensy/loader.html
- File / open HEX file **amorseV35.hex** . Cliquez ‘ automatic mode ‘ qui passe en vert
- Appuyer sur le petit bouton du Teensy, ce qui va ‘ charger ‘ le .hex

http://www.pjrc.com/teensy/usb_serial.html : **serial_install.exe** (assez long …)
- Vérifier avec ‘ gestionnaire de périphériques ‘ . Noter le N° port **COM x** créé.

- **HyperTerminal** Windows (ou autre) propriétés / connexions , mettre **COM x** , 56800, pas de protocole, 8 bit, 2 stop. (Adaptation débit auto adaptative)
Fichier / enregistrez sous / amorseV35 (.ht) dans le dossier amorse.

Il faut lancer hyperterminal après connexion et lancement du Teensy.
La liaison USB n’est pas nécessaire si l’on reste en mode ‘ autonome ‘ LCD seul.

Déploiement

Installation outils de développement

- http://winavr.sourceforge.net/download.html : **avr-gcc** (WinAVR 20100110) 4.3.3
- Programmer's note pad 2 PN

- Créer un **dossier** : .. \ Mes applis teensy \ **amorse**
- Dézippez-y amorse**.zip** : amorse.c .h lcd_1x16.c et .h
  NB Les print.c et .h , les usb-serial.c et .h et le makefile sont personnalisé
- Editeur : **Programmer's note pad 2** PN
  Tools / options / code template C C++
  View / line numbers View / syntax highlighting
- Vérifier makefile chercher ces lignes :

```
# Target file name (without extension).
TARGET = amorse            <<<<<   nom  destination
# List C source files here. (C dependencies are automatically generated.)
SRC =         $(TARGET).c \
              print.c \
              usb_serial.c\     <<<<<<  modules C  rajouté
              lcd_1x16.c        <<<<<<  modules C  rajouté
MCU = atmega32u4       # Teensy 2.0
```

faire file / save

Développement

Pour les habitués du **C**, ça devrait aller… Pour les néophytes c'est un départ…
Il y a nombre de cours, de livres, d'exemples pour les bases et plus plus encore…
Les explications des fonctions et les diagrammes sont transcrits le plus simplement.
Les lignes C ' compressées et tarabiscotées ' sont évitées au profit de codes lisibles.
Les noms des fonctions et des variables sont au plus explicites sans être trop longs.
Les blocs de codes sont encadrés par les accolades **{ }** posées verticalement, l'une sous l'autre, et indentées, selon l'usage de l'auteur.
Les commentaires du source, complètent celles-ci.

- Editez amorse.c , compilez : Tools / [WINavr] make all
  Vous avez créé des fichiers amorse.xxx dans le dossier dont **amorse.hex**
- Lancer **teensyloader**(xp).exe (ou …vista.exe)
  http://www.pjrc.com/teensy/loader.html
- File / open HEX file amorse.hex . Cliquez ' automatic mode ' qui passe en vert
- Appuyer sur le bt loader de la puce Teensy, ce qui charge ce Hex et lance le prg.
- Maintenir le bt cde appuyé pendant ce temps *1 de chargement et lancement
- Lancer Hyperterminal juste après l'info reboot . Faire « **+** » <entrée>
- Faire « **W** » <entrée> pour enregistrer en eeprom les infos par défaut.
- Arrêter Hyperterminal, débrancher – rebrancher USB sans appuyer sur le bt cde
- Démarrage à chaud Lancer à nouveau Hyperterminal OU AMORSE.EXE
  (*1 car un load efface les ex infos amorse eeprom)

Manipez…

FAQ

Testez modifiez les paramètres… à 1 lettre avec le bt de commande
‘ = ‘ affiche des paramètres indépendamment du mode trace actif ‘ + ‘
N’oubliez pas de terminer par ‘ W ‘ pour les conserver.

Développez testez améliorez loader : ceci efface les ex infos eeprom
Mettez vos paramètres préférentiels par défaut en ‘ setting ‘ dans le source src
Démarrez ensuite avec le bt cde appuyé jusqu’au beep , puis faire ‘ W ‘

Possibilités de blocage : le ‘ stack ‘ et la place ram sont limités.
Utilisez sans le mode trace ‘ + ‘ pour limiter la charge stack ..
variables fixes : utilisez PSTR( ) src

Les smart commandes ne sont pas prises ou décalées
La prise en compte des commandes smart ou via RS ne s’applique qu’à la phase espace mot. Si on est en réception dense, il y a peu d’espace mots !

Les smart commandes ne sont pas bien prises
La manip morse doit être correcte : si vous voulez Z et tapez G E
Il y aura l’affichage G intervalle - , et le point de fin du Z est perdu

La smart commande n’agit plus
Cas de ‘ N ‘ sur une mémoire vide (espaces) donc ‘ temps ‘ !!!

La prise en compte des commandes smart ou via RS ne s’applique qu’à la phase espace mot. Si on est en réception dense, il y a peu d’espace mots ! Couper la bf.

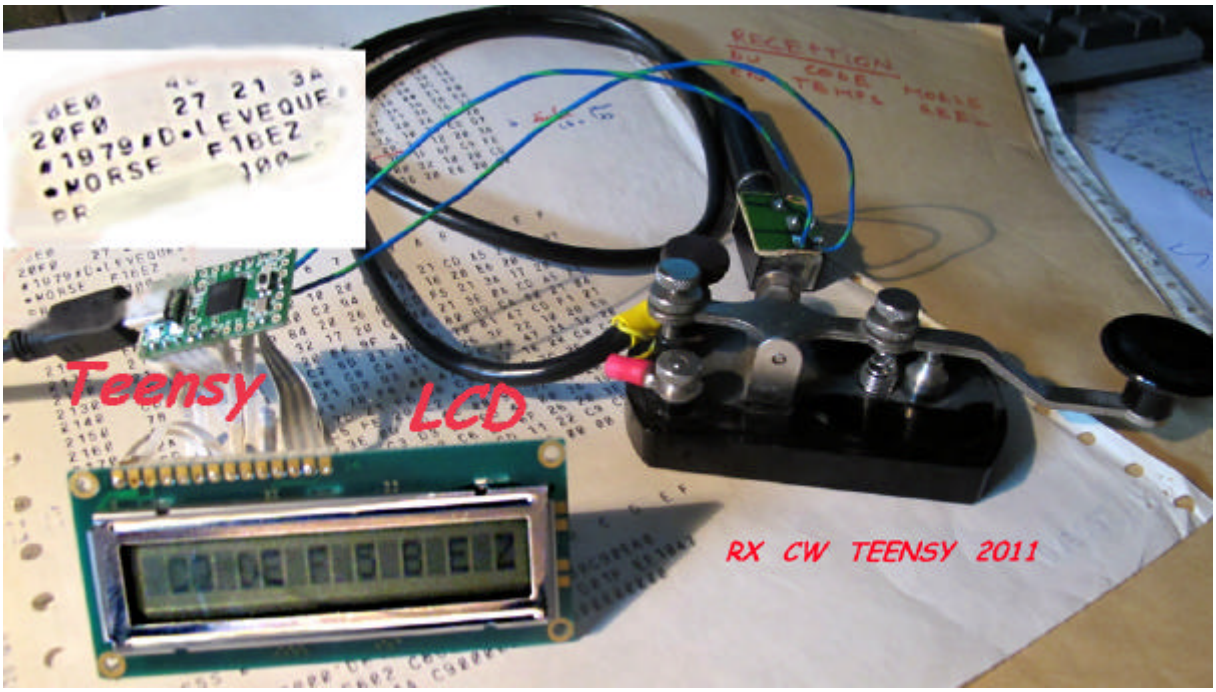
Le décodage ne suit plus…
Si l’on est en situation vitesse dense et parasites courts les DDtime et LStime deviennent très très courts ! La Led mox reste allumée. Relancer AMORSE, réception coupée…

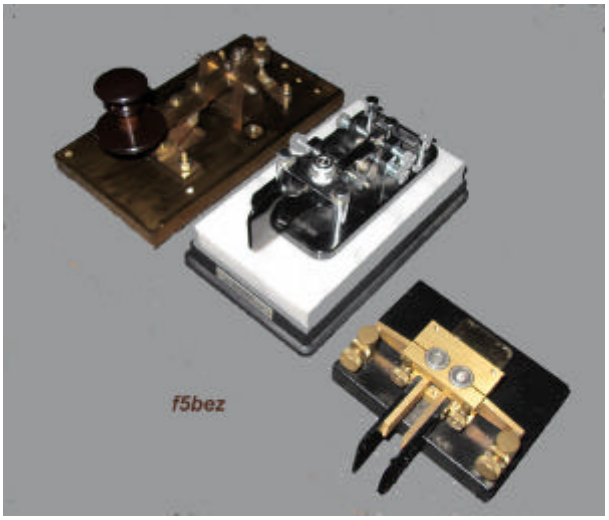
Communication perdue ou ‘ com x déjà ouvert ‘ , arrêter hyper terminal, débrancher rebrancher USB (restart chaud) , redémarrer hyper terminal

Copyright

Cette version **V3.5 15 octobre 2011** vous est proposée en open source pour le domaine radioamateur éducatif sans usage commercial.
Elle est adaptable sur d'autres modules moyennant adaptation des **I / O**
Merci de mentionner l'auteur **F5BEZ** Bonnes 'manips'

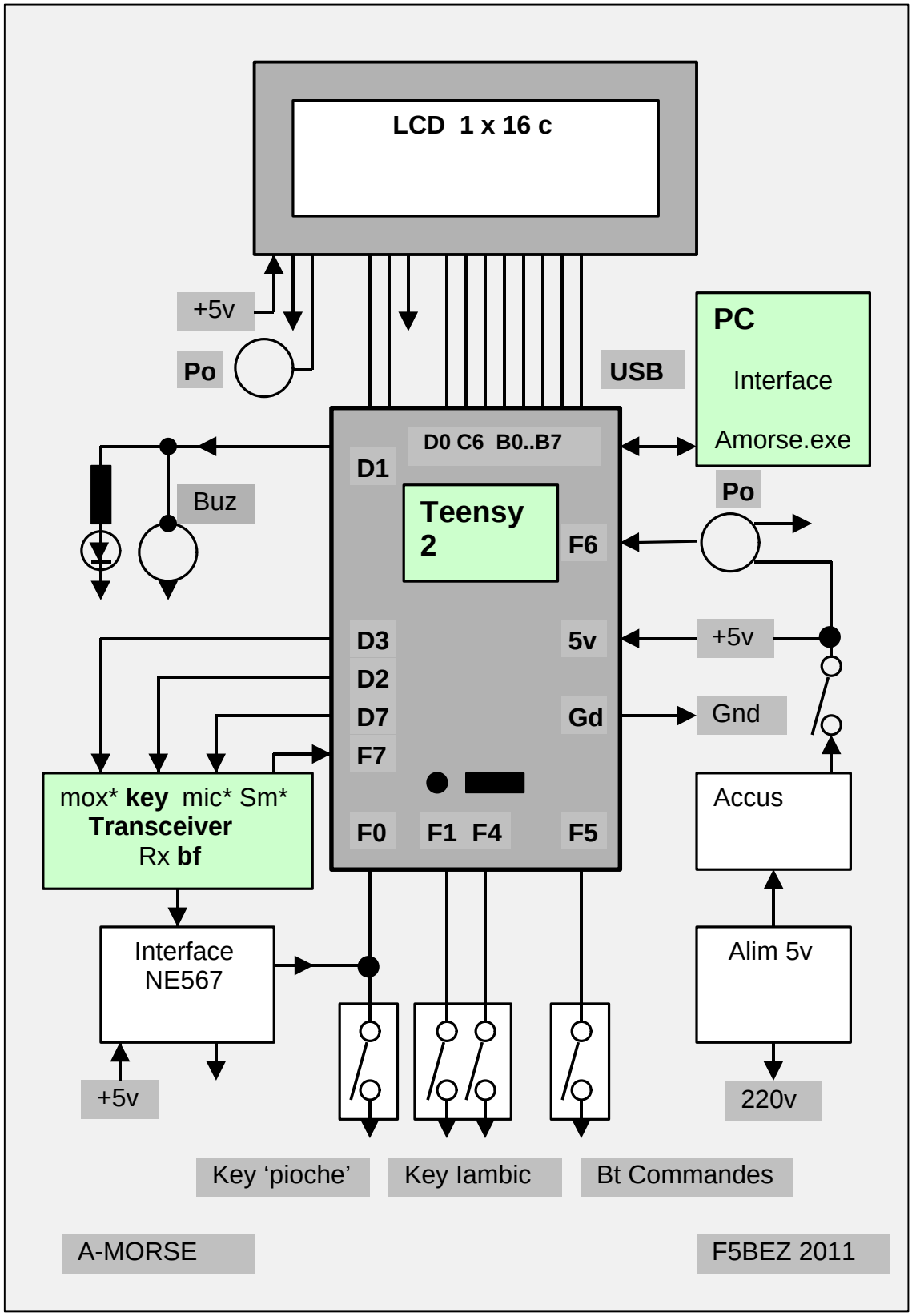
L'auteur : n'est pas un ' féru ' de la pratique CW et son trafic cw y est plus que modeste, mais il aime à tester des méthodes et créer des applis autour des thèmes radio… Merci à F8OP, ARA35, F6GQO, F5VJD, F5RKC, F8CFE, …





Key ' lame de scie '

Plan Teensy


LCD 1 x 16 c
+5v
Po
USB
PC
Interface
Amorse.exe
D0 C6 B0..B7
D1
Teensy 2
F6
Po
Buz
D3
D2
D7
F7
5v
Gd
+5v
Gnd
mox* key mic* Sm*
Transceiver
Rx bf
Accus
F0
F1 F4
F5
Interface NE567
Alim 5v
+5v
220v
Key 'pioche'
Key Iambic
Bt Commandes
A-MORSE
F5BEZ 2011

Liste Inp / Out

Num	Teensy 2.0	A morse	In/out	active
0	PIN_B0	LCD D0	O	X
1	PIN_B1	LCD D1	O	X
2	PIN_B2	LCD D2	O	X
3	PIN_B3	LCD D3	O	X
4	PIN_B7	LCD D7	O	X
5	PIN_D0	LCD D/I = 0 fct 1 data	O	X
6	PIN_D1	OUT CW LED BEEPER	O	1
7	PIN_D2	OUT CW TX	O	0
8	PIN_D3	OUT CW PTT-MOX	O	0
9	PIN_C6	LCD Strobe front 0	O	0
10	PIN_C7			
11	PIN_D6 *	OUT CW Led internal	O	1
12	PIN_D7	OUT CW audio	O	
13	PIN_B4	LCD D4	O	X
14	PIN_B5	LCD D5	O	X
15	PIN_B6	LCD D6	O	X
16	PIN_F7	INP A/D 7 S-mètre	I	X
17	PIN_F6	INP A/D 6 Pot	I	X
18	PIN_F5	INP CW Command button	I	0
19	PIN_F4	INP CW KEY Iambic dash	I	0
20	PIN_F1	INP CW KEY Iambic dot	I	0
21	PIN_F0	INP CW KEY straight	I	0
22	PIN_D4			
23	PIN_D5			
24	PIN_E6			

Coffret

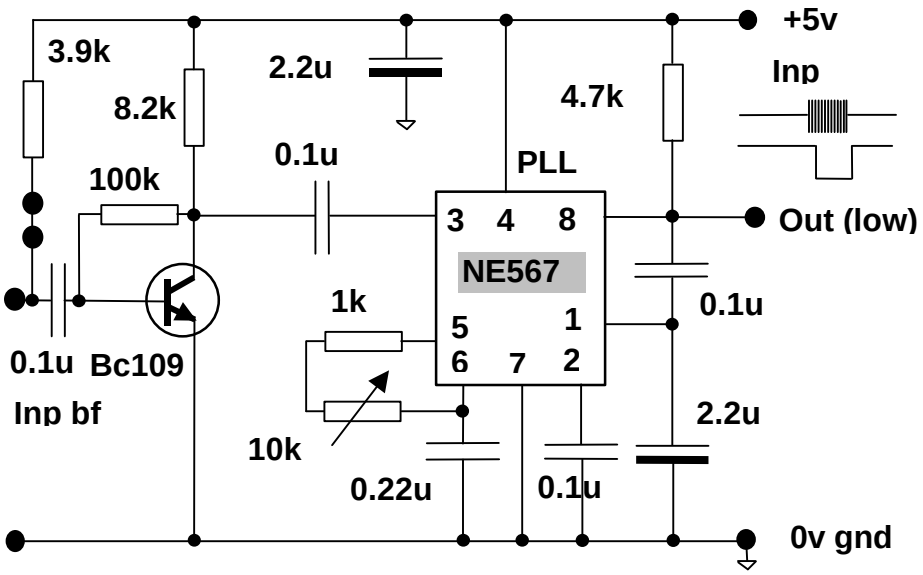
Le petit coffret métallique est un ancien lecteur de cassettes amovibles.
Le Bt de commande et la LED (D1) sont devant.

Câbles

- Secteur 220v si alim interne ou bloc 220v – 9v dc de récup et jack alim
- câble USB vers PC sert en programmation du .hex et en liaison appli.
  Il alimente concurremment le module en 5v , hors alims activées
- entrée Key normale ‘ pioche ‘ jack 6 mm
- entrée Key ‘ double contacts ‘ jack 6 mm
- entrée audio liaison Rx déca (ou PC test) entrée jack 3 mm
- entrée option s-mètre D.C. d’un Rx ou module Rx
- sortie option Key vers Tx déca (led) jack 3mm vers jack mâle 6 mm Tx
- sortie option mox ptt vers un module Tx (led) englobe un message
- sortie option audio cw vers un module Tx

Interface réception

Le sujet est vaste ! La mise en forme ‘ propre ‘ peut être faite par différentes méthodes, comme des filtres actifs, des échantillonnages avec FFT, ou des PLL …
Le montage proposé est simple mais efficace et utilise un PLL **NE567**,
La finesse est de 20hz autour de 900hz … (pot 10k)
La sortie 8 en O.C., active sur 0, est mise en parallèle avec la ‘Key’ F0.



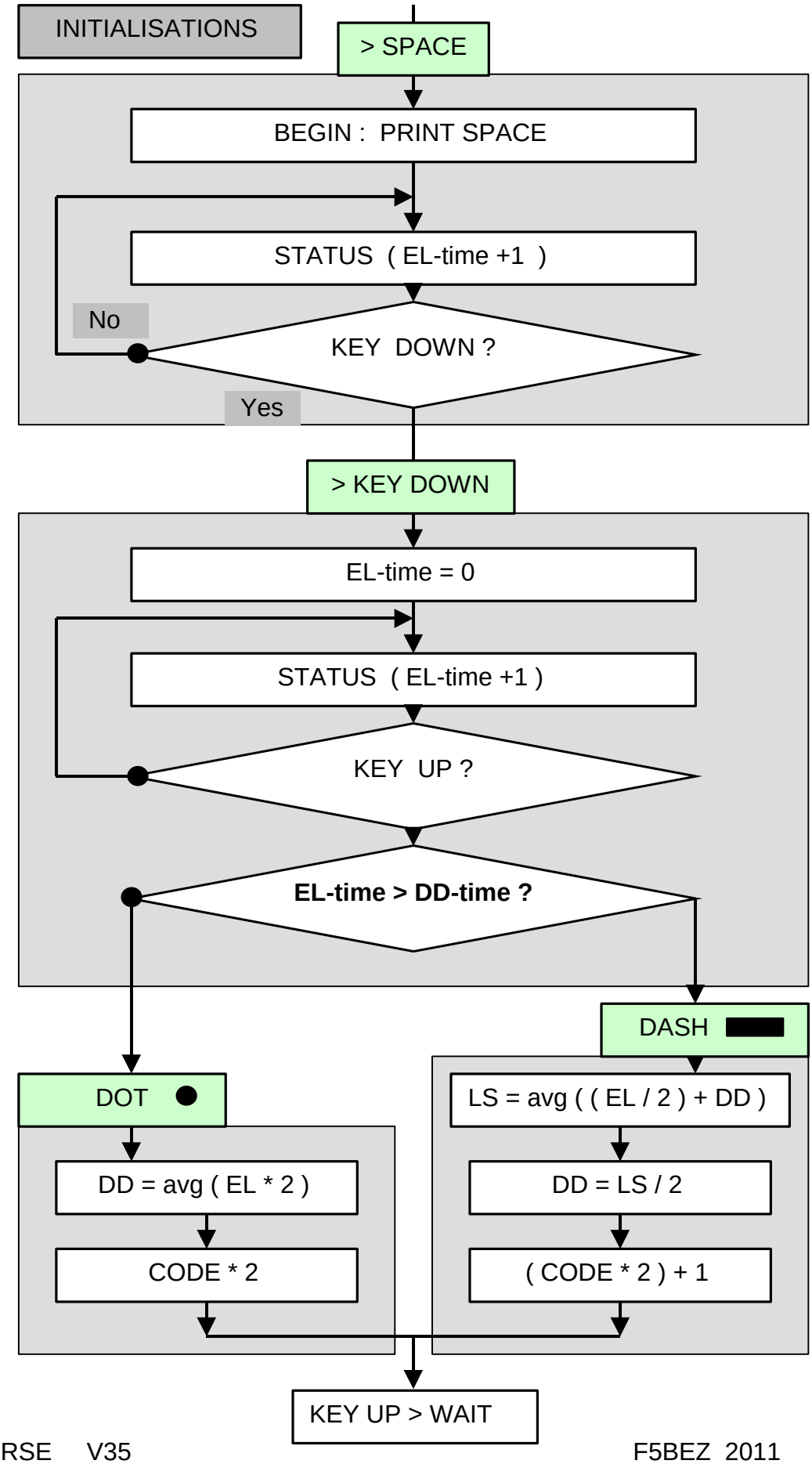
Test et mesures
Le 0,1 uF entre 8 et 1 lisse les transitions et fronts de sortie
Les composants peuvent être optimisés pour le moindre déphasage, cependant un jitter initial de 15ms + - 5 ms et de fin de 4 ms raccourcit des ‘E’ de 50ms à 20 mots/mn, à 40ms et donnera donc une limite à des vitesses cw plus hautes, selon l’algo choisit. Voir aussi le chapitre Validation et contrôles à Décodage CW

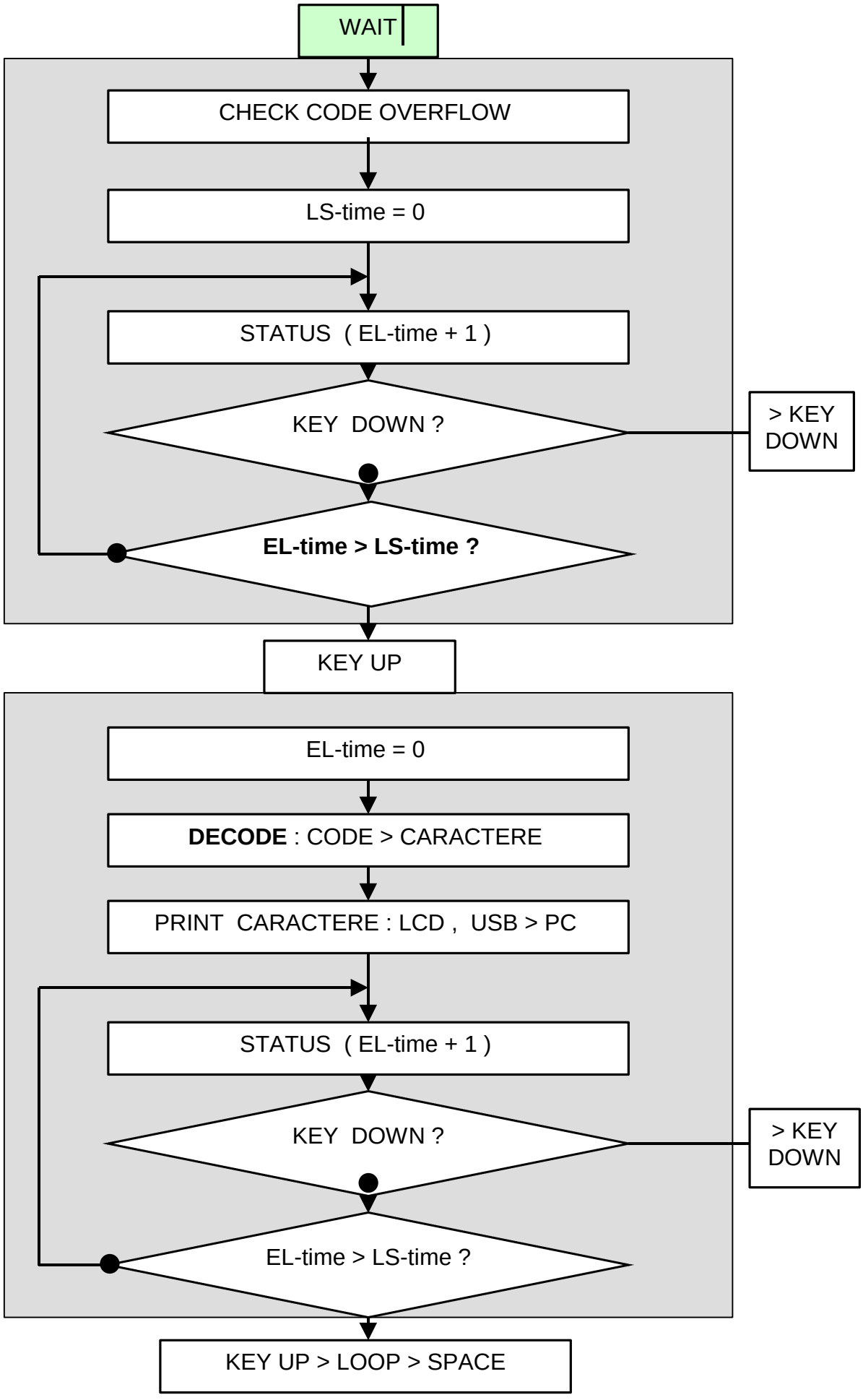
Mode Rx en déca
En mode ‘ déca ‘ 7.020Mhz.. 14.030Mhz … Le nbre de boucles ‘MAXDBNCE ‘ doit être diminué à 1. Le mode F sera sur ‘1’ (normal), l’algo sur 1 ou 2 ou …
Régler le niveau du signal audio OUT pour ne pas saturer le PLL.
Il faut jouer très lentement du VFO !… Essayez d’abord en CW BW large 2khz puis 500hz… Les algorithmes des temps DD et LS, du ‘ code ‘ et de la table des abréviations CW sont très variables et critiques (et donc améliorables…) .

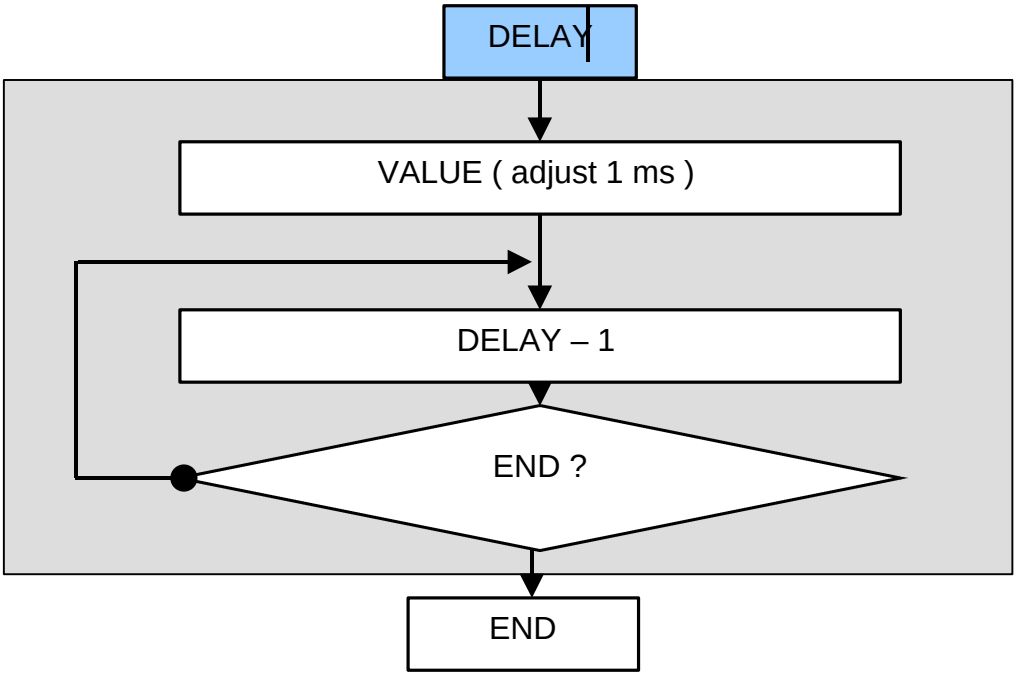
Entrée par micro
L’entrée audio du PLL peut être reliée via une 3,9k à +5v (strap) et permet ainsi de connecter un micro-électret, lequel écoute votre ‘ sifflement ‘ cw ! ou CW PLAYER, en ajustant la ‘ note ‘, ou l’ écoute décamétrique CW … *Et ça marche !* …

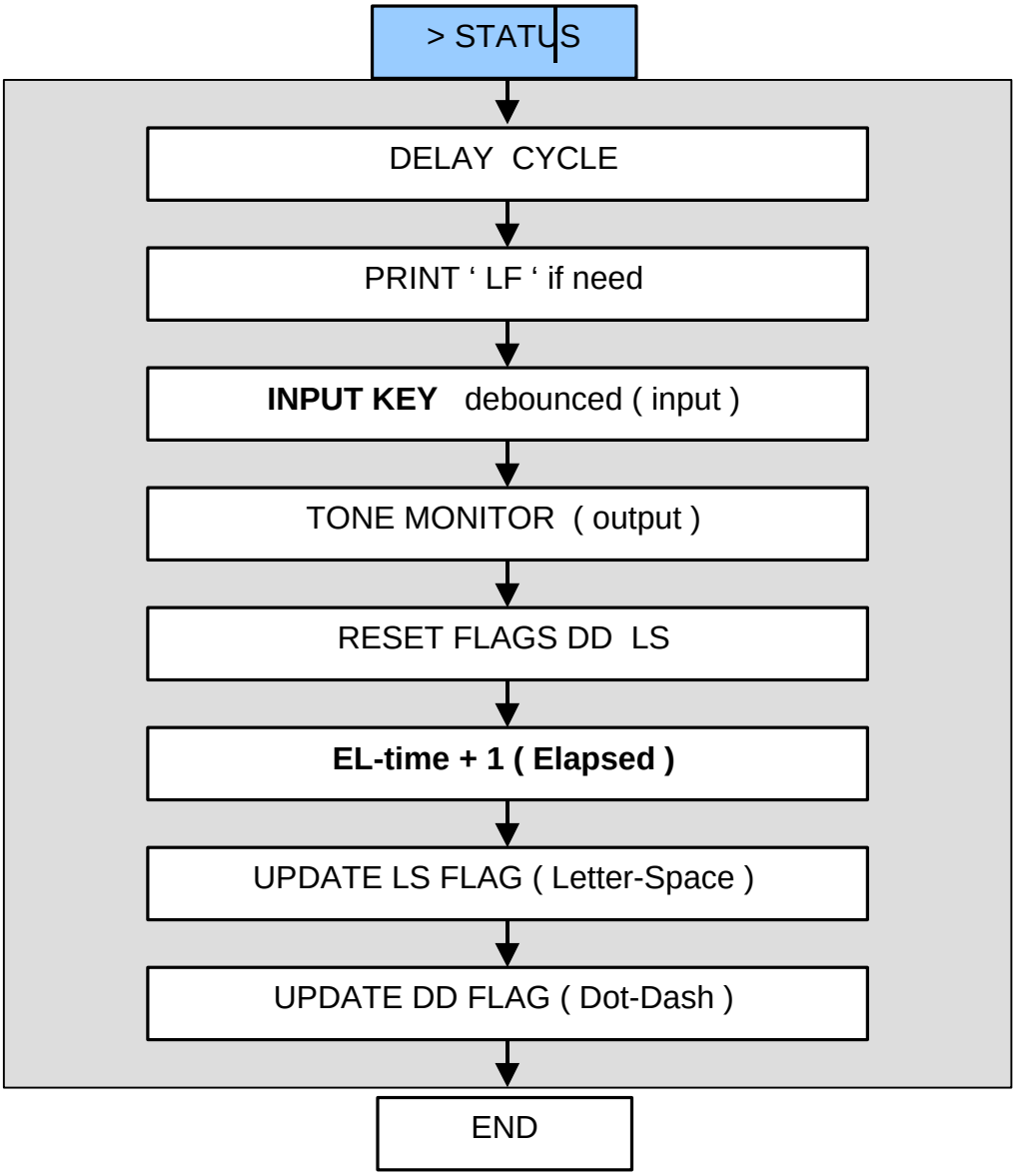
Pour plus de détails sur le NE567 :
http://www.sonelec-musique.com/electronique_realisations_decodeur_tonalite_001.html

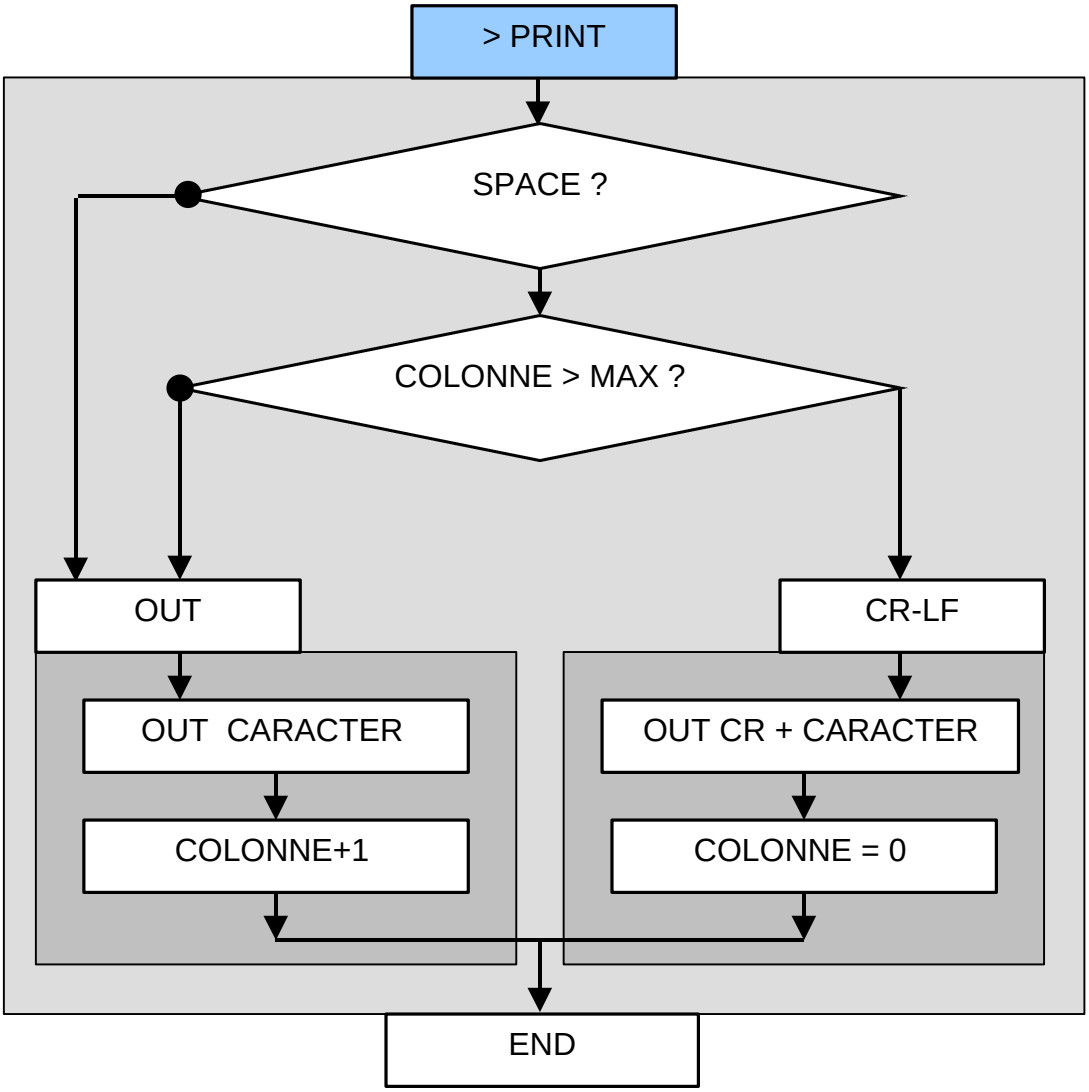
Diagramme décodage




WAIT
CHECK CODE OVERFLOW
LS-time = 0
STATUS (EL-time + 1)
KEY DOWN ?
> KEY DOWN
EL-time > LS-time ?
KEY UP
EL-time = 0
DECODE : CODE > CARACTERE
PRINT CARACTERE : LCD , USB > PC
STATUS (EL-time + 1)
KEY DOWN ?
> KEY DOWN
EL-time > LS-time ?
KEY UP > LOOP > SPACE


DELAY
VALUE (adjust 1 ms)
DELAY – 1
END ?
END


> STATUS
DELAY CYCLE
PRINT ‘ LF ‘ if need
INPUT KEY debounced (input)
TONE MONITOR (output)
RESET FLAGS DD LS
EL-time + 1 (Elapsed)
UPDATE LS FLAG (Letter-Space)
UPDATE DD FLAG (Dot-Dash)
END


> PRINT
SPACE ?
COLONNE > MAX ?
OUT
CR-LF
OUT CARACTER
OUT CR + CARACTER
COLONNE+1
COLONNE = 0
END

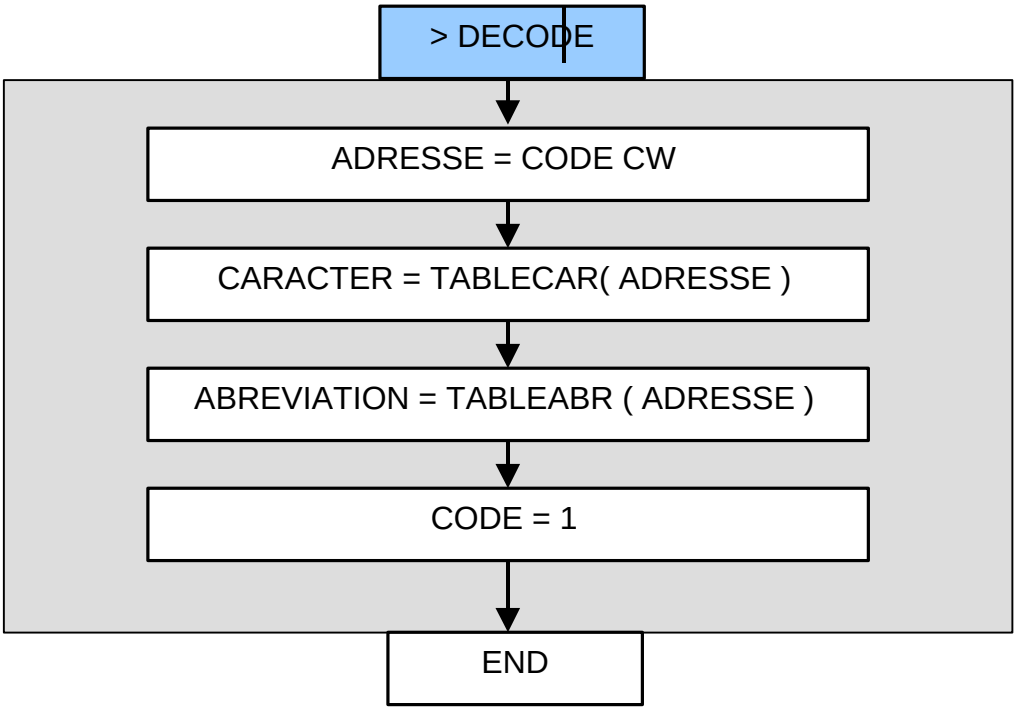
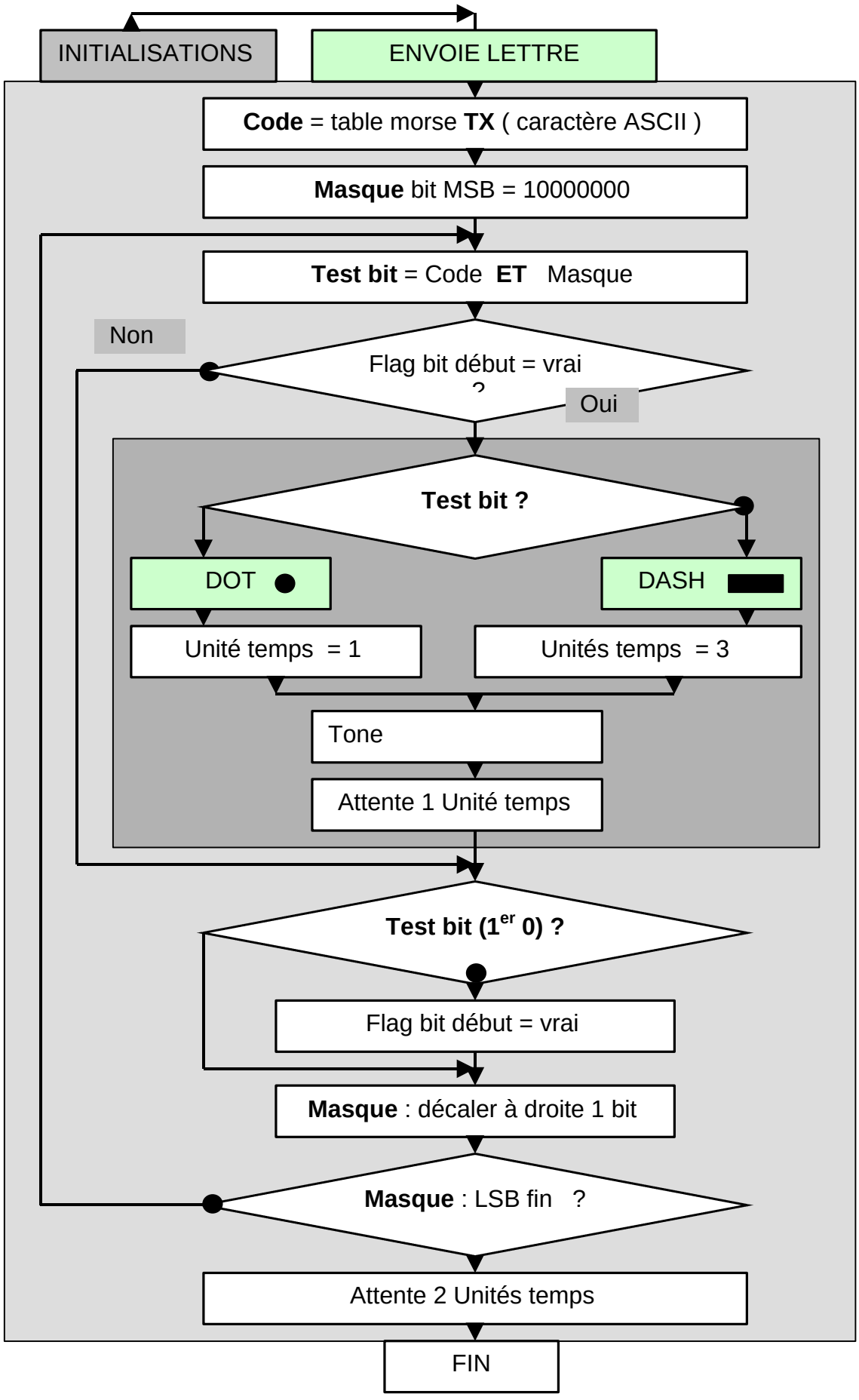
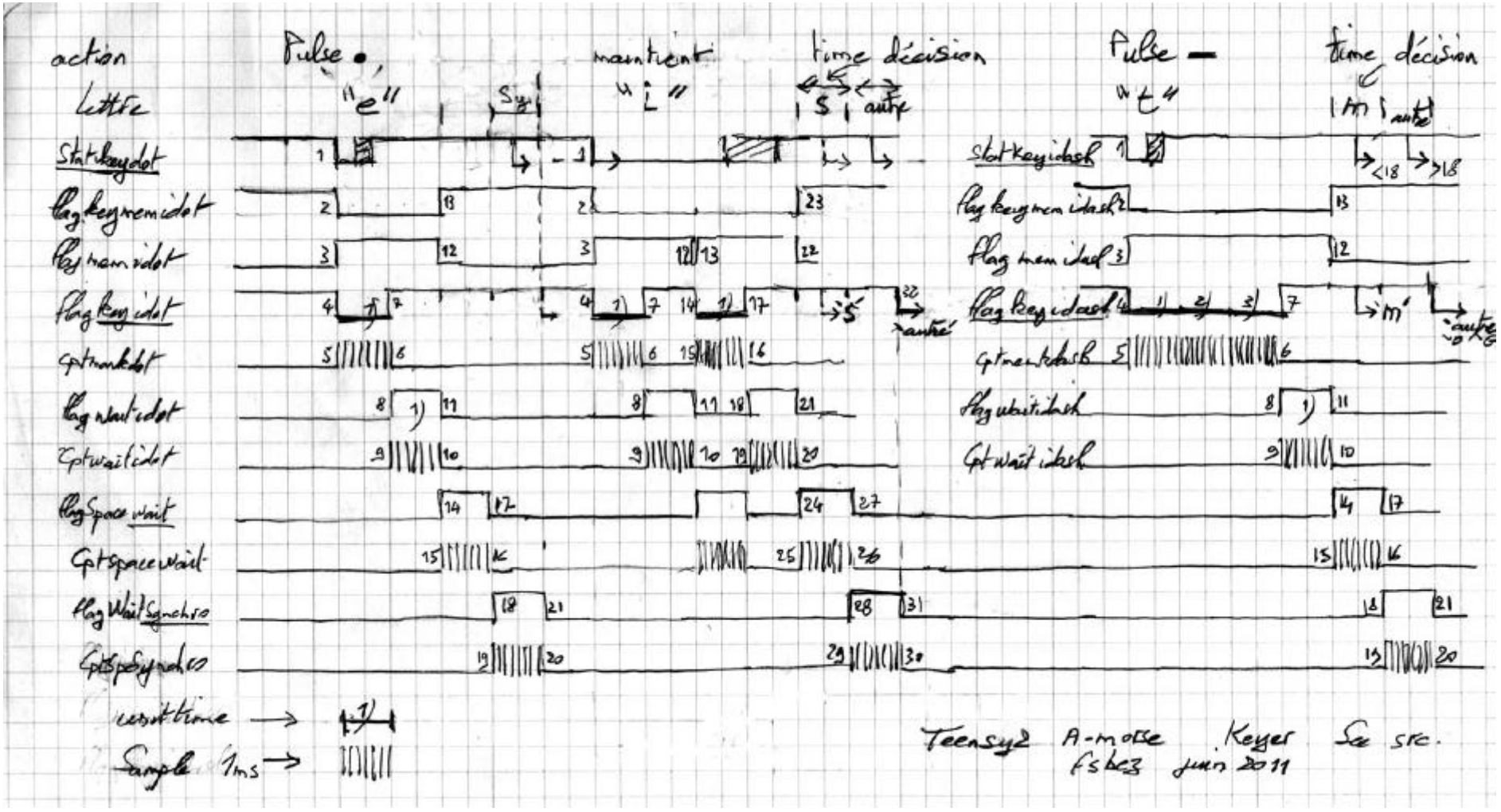

> DECODE
ADRESSE = CODE CW
CARACTER = TABLECAR(ADRESSE)
ABREVIATION = TABLEABR (ADRESSE)
CODE = 1
END

Diagramme encodage




action
Lettre
Pulse •
"e"
maintenant
"i"
time décision
S
autre
Pulse —
"t"
time décision
m
autre
Statkeyidot
flag keyremidot
flag remidot
flag keyidot
Cptremidot
flag waitidot
Cptwaitidot
flag Space wait
Cptspacewait
flag WaitSynchro
CptSpcSynchro
resttime →
Sample 1ms →
Statkeyidash
flag keyremidash
flag remidash
flag keyidash
Cptremidash
flag waitidash
Cptwaitidash
'm'
autre
Teensy2 A-motse Keyer Sa src.
F5BEZ juin 2011

LES CHRONIQUES

CW info

F6HBR, Alain Brugnon

J'espère que vous avez tous passé de bonnes vacances, et que vous êtes en pleine forme à l'approche des grands contests internationaux (avec ou sans packet-cluster !). Ce mois-ci je ne vous livre pas d'infos, mais un texte passionnant de Dominique, F5BEZ, écrit dans un style très particulier, que j'apprécie.

QSD ?... QRI ?

F5BEZ

En 1972, j'apprenais les langages de programmation de microprocesseurs. Le Morse me paraissait un excellent sujet d'exercice ! J'écrivais alors mes premiers algorithmes de décodage du Morse dans le langage de ces puces savantes, qu'il fallait bien maîtriser. Après avoir assemblé moult codes en Intel 8080, je manipulais quelques essais sur la machine de fabrication OM dont je disposais, mais elle ne mettait pas beaucoup d'enthousiasme à décoder ma CW ! Aussi fallait-il l'avis d'un spécialiste. Ma plus grande joie fut un jour de voir Georges, F8OP, champion de CW ces années-là, remplir l'écran en 2 temps et 3 mouvements avec maestria et très peu d'erreurs... Comme l'algorithme s'y retrouvait bien, on passa alors à l'étape suivante : mettre en forme exploitable pour ce programme les signaux BF reçus en décamétrique... J'ai Bidouillé et Expérimenté avec Zèle ! tous les filtres actifs à amplis opérationnels connus d'alors ou presque, mais ce n'était pas vraiment opérationnel ! Encore aujourd'hui c'est toujours une affaire délicate, et le décodage ne marche bien que sur des réceptions non entachées de QSD, QSB, QRN ou QRM, mais QSA, QRKS, c'est-à-dire en clair des CW stables, pures et aux normes ; c'est rare. Quel défi extrêmement difficile de modéliser l'aspect sélectif du cerveau ! En effet, une petite CW aigüe sera perçue et entendue dans le bruit à côté d'une autre plus puissante et plus grave que l'on n'écoutera pas, au prix d'une concentration intense. Pas simple de faire se concentrer des amplis ops ! A l'inverse, j'ai écrit aussi des programmes de génération du code Morse pour m'entraîner aux groupes aléatoires de 5 lettres à des vitesses et tonalités différentes.

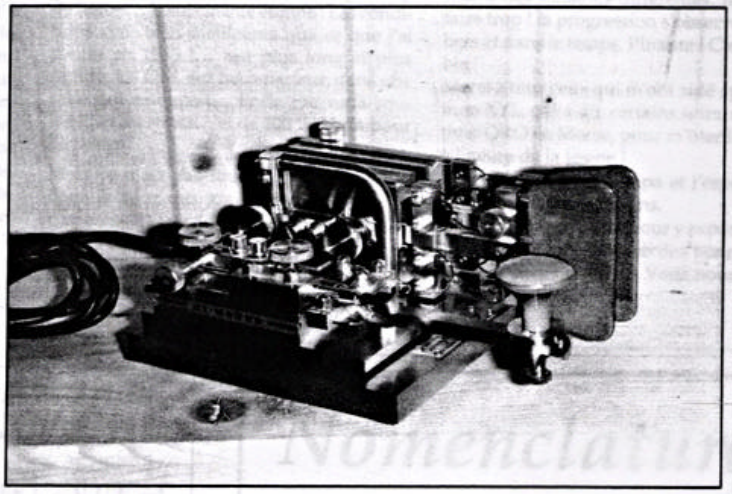
Voilà pour l'aspect technique.

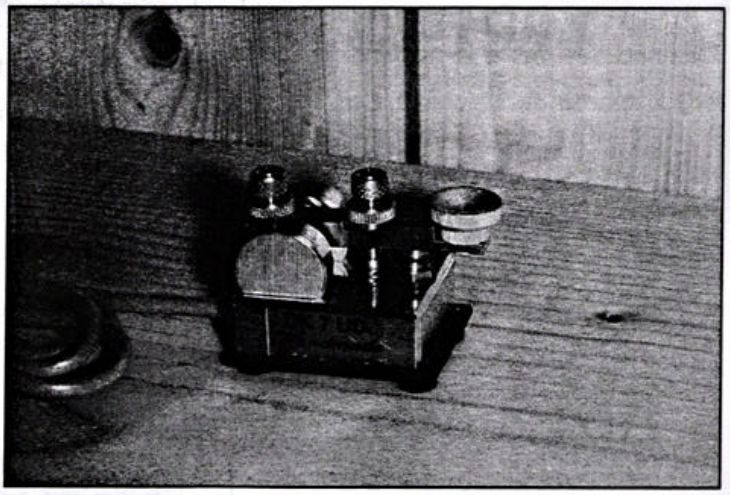
Ma motivation est revenue, un peu par défi, quand j'ai vu arriver un beau jour Jean-Yves sautant de joie avec son « D » tout neuf, et quand René sortait de sa grande boîte en bois des paquets de QSL de tous les points du monde... C'est en vacances, entre 2 randonnées à vélo, que j'ai trouvé le temps d'écouter le trafic CW de Samuel. Ensuite, comme chacun, les activités professionnelles, familiales, sportives et autres, font que l'on n'a jamais le temps... Eh bien si !... C'est décidé, je me lève un quart d'heure plus tôt pour faire une séance de télégraphie... Toutes les cassettes y sont passées et repassées. Le soir, quelques séances avec les logiciels SM ou UFT, puis l'écoute sur le 7 MHz, là c'est plus dur à cause de la vitesse élevée, du QRM, de la CW « collée »... mieux vaut aller se coucher !

Mais quand vous trouvez, dans cette cacophonie, une petite station qui va bien dans un coin calme, et quand ce que vous notez, en cachant du doigt, devient des prénoms, des villes, de la météo, des indicatifs lointains, alors là... cela devient un jeu ! Je commence à y croire ! Les QSO locaux sur le 144.050 MHz calmes, trop calmes, avec Gérard et Serge confirment cette impression.

Cent fois sur le métier, remettez votre ouvrage... et puis, un jour, Lucien trouvait que « ça allait bien ! »...

Voilà pour la préparation.





Deux manipulateurs de la collection F5SJB.

F6HBR, *Alain Brugnon* **LES CHRONIQUES**

Rendez-vous est pris et on ne pense plus qu'à ça !... Les révisions vont bon train. L'examen blanc avec Jean-Yves allait bien aussi la veille, mais il ne le savait pas ! Le centre radio-maritime est facile à trouver, il n'y a qu'à suivre les antennes ! Monsieur B... est accueillant. L'administration qu'il représente exige cette épreuve ; ainsi, j'ai l'honneur d'étrenner le nouveau logiciel d'examen télégraphique. La tonalité peut être choisie, j'ai préféré le casque au haut-parleur. Un premier texte d'essai, pour prendre conscience de ce qu'on fait là, et Monsieur B... vous ayant mis à l'aise, la tension commence à tomber. La tonalité est bonne, la CW nette, la vitesse de 10 mots/minute. On peut aller plus vite mais ne jouons pas les kamikazes ! L'examen commence avec le code Q... pas de problème ; chaque code est passé deux fois, calmement. L'excercice suivant est une série de groupes de 5 lettres, suivi d'une petite série de groupes de 5 chiffres ; ensuite, un texte en français pas très long, avec des chiffres et la ponctuation ordinaire, enfin le point et le « VA » final libérateur...

Le logiciel affiche maintenant tout cela et l'on corrige ensemble. Bon, ça va... pas trop de fautes... je suis même étonné ! Les conditions sont bien meilleures que ce que j'ai connu en 1980 !... test plus long et plus rapide, à 6 OM, sur haut-parleur, dans une salle qui résonnait... Enfin me voilà soulagé ! et du stress... et de 200 F ! Voilà pour l'examen.

Mais ce n'est pas fini ! Avant de piocher dans la cour des graphistes émérites, on commencera par des bandes calmes... Je ne m'en pensais plus capable !

En 1970, j'aurais pu être dans les F6AY... Si j'avais pris au sérieux la CW !

En 1980, je me remets à la tâche, encouragé par notre regretté Gérard, F3I... et j'aurais pu être dans les F6GZ... comme notre regretté Bernard.

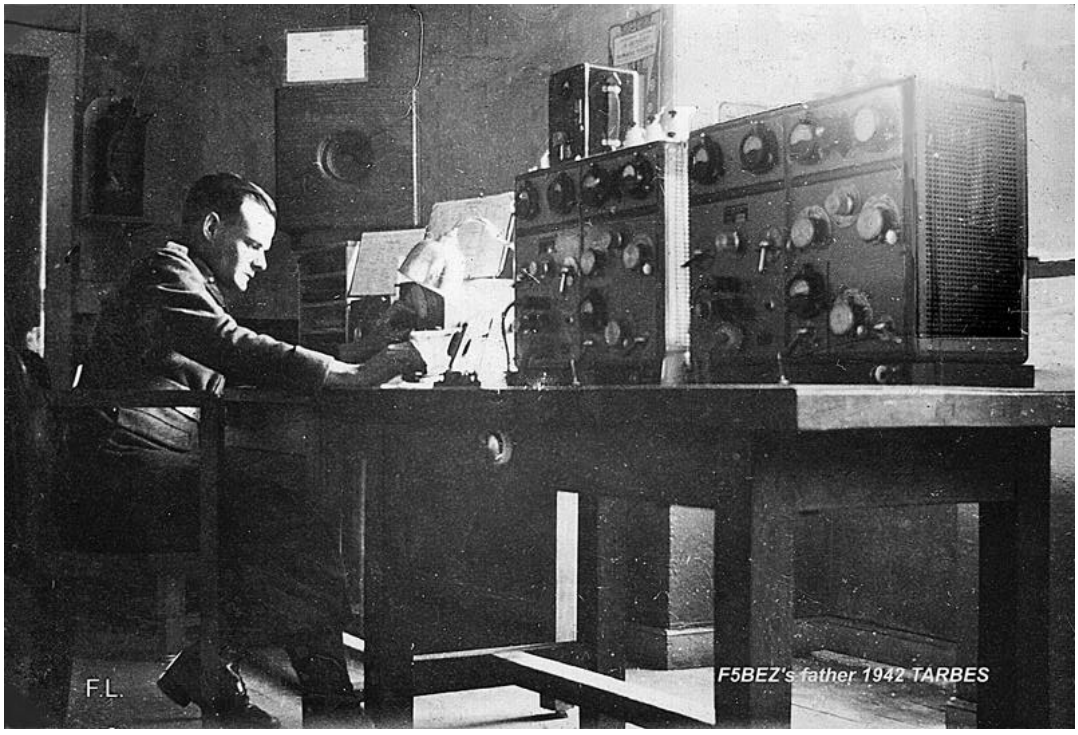
Mon récit doit motiver ceux qui hésitent. Bien sûr, il faut être motivé, il faut de la patience et de la préparation. Il faut s'habituer à des tonalités différentes, ne pas en faire trop ! la progression s'observe par paliers et dans le temps. J'insiste ! C'est possible !

Merci à tous ceux qui m'ont aidé ; pardon à mon XYL, qui a dû, certains soirs, m'invectiver QRO en Morse, pour m'ôter le casque à l'heure de la soupe !

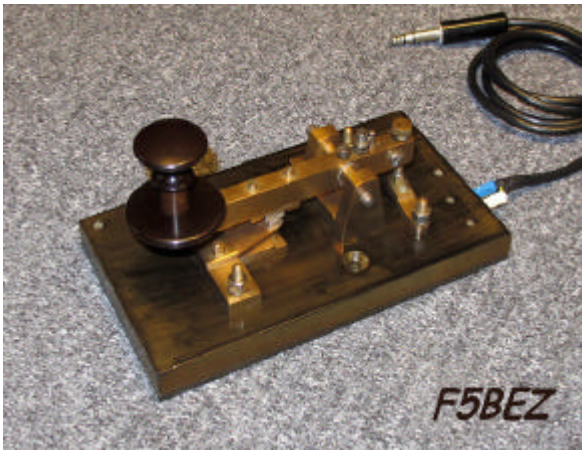
Je dédie ce 5 à mon papa et j'espère faire honneur aux plus anciens.

A bientôt sur les QRG, pour y expérimenter encore, n'y entendre que des bonnes nouvelles et parler d'amitié. Voilà pour la tranche de vie ! QRZ ?

Article REF septembre 1994



Histoire : Père de F5BEZ
2 x ER26r En 'zone libre' à la Préfecture Tarbes 1942



Manipulateur : original en laiton et bakélite provenant d'un navire inconnu. Il était fixé sur la table. Le contact conséquent, en argent, commutait un courant important sur un générateur à étincelles « spark generator »

Probable des années 20 !
les filetages ne sont pas Whitworth mais S.I.
voir site zianet où l'on trouve de semblables

Manipulateurs aviation
(extraits collection f5rkc)

La télégraphie et le Morse …

Points de départ de nombreux liens :
http://fr.wikipedia.org/wiki/Alphabet_Morse
Historique, variantes, etc
- Signes de ponctuation et symboles
- Extension aux caractères internationaux ou digrammes
- Codes spéciaux
- Radioamateurs et nombreux liens.

The Sparks Telegraph Key Review : nombreuses infos keys anciennes
http://www.zianet.com/sparks/sparkmakers2.html

Et d'autres nombreux sites !!!
Cherchez avec ces mots clés : 'straight key' 'spark generator'

doc V35a

Annexe 1 Iambic (dual-lever) Paddles

http://en.wikipedia.org/wiki/Telegraph_key

Keys offering one contact for dits (left key) and another for dahs (right key) were dubbed "iambic paddles," when each contact may be closed simultaneously. The iambic function (alternating dits and dahs) are created with an electronic keyer by squeezing the paddles together.

A single-paddle also utilizes separate contacts for dits and dahs, but there is no ability to make both contacts simultaneously by squeezing the paddles together (iambic). When a single-paddle key is used with an electronic keyer, continuous dits are created by holding the dit side. Likewise, continuous dahs are created by holding the dah contact.

Iambic keying or squeeze keying creates alternating dits and dahs. This makes sending some characters easier, like the letter "C," by merely squeezing the two paddles together. In single-paddle, non-iambic keying, the hand motion would require alternating four times for "C" (dah dit dah dit).

Iambic keyers function in one of at least two major modes. Mode A is the original iambic mode, in which alternate dots and dashes are produced as long as both paddles are depressed. When the paddles are released, the keying stops with the last dot or dash that was sent while the paddles were depressed. Mode B is the second mode, which devolved from a logic error in an early iambic keyer. In mode B, dots and dashes are produced as long as both paddles are depressed. When the paddles are released, the keying continues by sending one more element, i.e., a dot if the paddles were released during a dash, or a dash if the paddles were released during a dot. Users accustomed to one mode usually have difficulty utilizing the other, so all competent keyer designs have the capability of selecting the desired keyer mode. If forced to use a keyer with an unaccustomed mode, the user must revert to single paddle mode in which both paddles are never depressed simultaneously.