PICOSD

Programme de configuration pour l'incrustateur ATV conçu par F1CJN

Par Pierre COL, F8EGQ - http://col2000.free.fr/f8egq/picosd/

1°) - Présentation

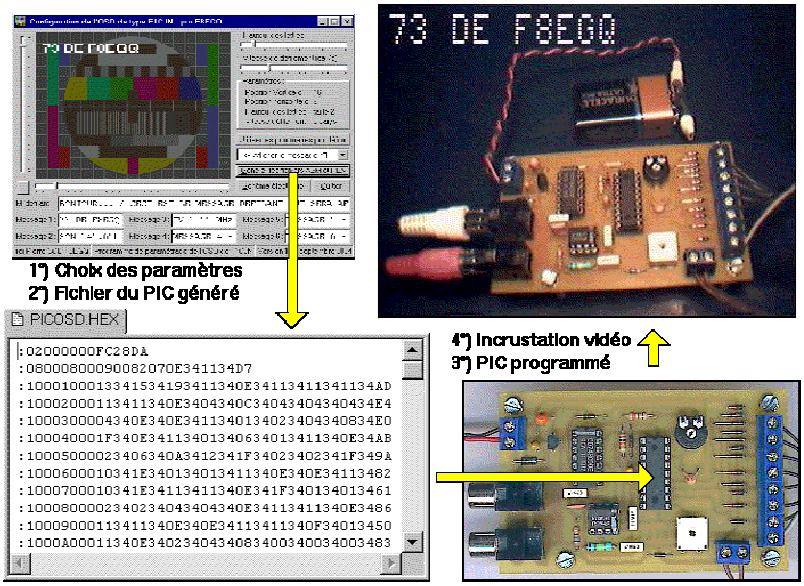
L'incrustateur vidéo :

Un système d'incrustation vidéo est un dispositif qui permet de superposer un texte (ou un logo) prédéfini sur une image vidéo ; on le désigne parfois également par son acronyme anglo-saxon "OSD" (**O**n **S**creen **D**isplay = Affichage sur l'écran). La réglementation nous impose de transmettre à intervalles rapprochés notre indicatif radioamateur ; il est bien sûr possible de le passer en phonie, ou de basculer sur une mire contenant l'indicatif, mais l'utilisation d'un incrustateur constitue, me semble-t-il, la solution la plus élégante. On peut alors en profiter pour diffuser des informations supplémentaires : locator, fréquence, etc.
Grâce à un petit montage assez génial, conçu par **Alain FORT**, **F1CJN**, la réalisation d'un incrustateur vidéo simple est maintenant accessible à tous. Celui-ci est basé sur l'utilisation d'un PIC16F84.

But de cette page :

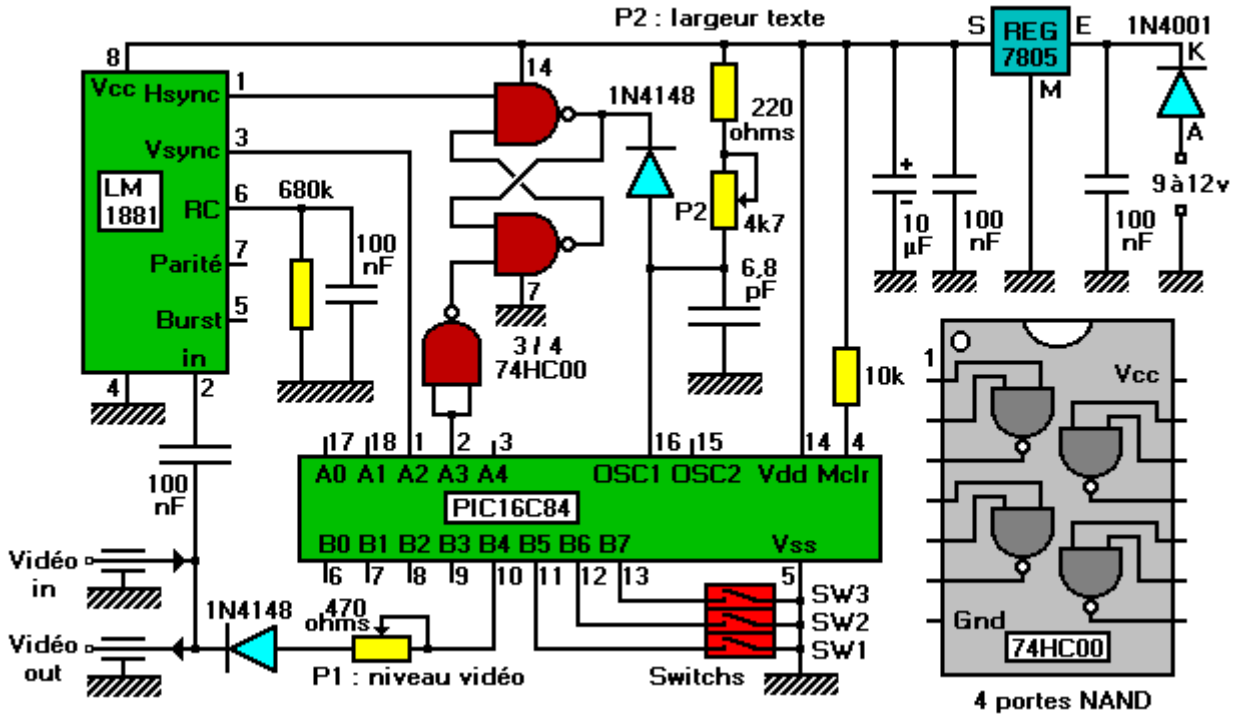
Comme je viens de le signaler, je ne suis pas l'auteur de ce montage, et le mérite en revient entièrement à Alain F1CJN. Je me suis simplement fixé pour tâche de tenter de reprendre et d'améliorer si possible le fichier source d'origine du PIC, et surtout de lui adjoindre un programme destiné à configurer aisément les messages à afficher, depuis un PC fonctionnant sous Windows ; ce programme s'appelle **PICOSD**.

Très concrètement, il suffira pour l'OM de lancer PICOSD, de modifier les messages dans les zones de saisie, de choisir la hauteur des lettres, la vitesse de défilement du message défilant, et, dans une certaine mesure, la position du texte sur l'écran, puis le programme va générer automatiquement les fichiers PICOSD.ASM (le fichier source) et PICOSD.HEX (le fichier hexa à programmer dans le PIC), qui correspondent à ces paramètres. Une fois le PIC programmé et réinséré sur l'incrustateur, la nouvelle configuration est immédiatement fonctionnelle :



L'internaute intéressé trouvera donc sur cette page, d'une part toutes les infos pour une réalisation détaillée du montage, et d'autre part le programme PICOSD qui permet de modifier facilement les différents messages à afficher.
Le schéma et le programme du PIC fournis par Alain, qui ont servi de point de départ à la réalisation de cette page, sont disponibles à la rubrique *Téléchargement* du site Internet de l'ANTA, l'Association Nationale pour la Télévision Amateur (archive Picosd.zip, 21 ko). Cette archive est aussi stockée en local sur mon site, ici.

2°) - Étude du schéma électrique :

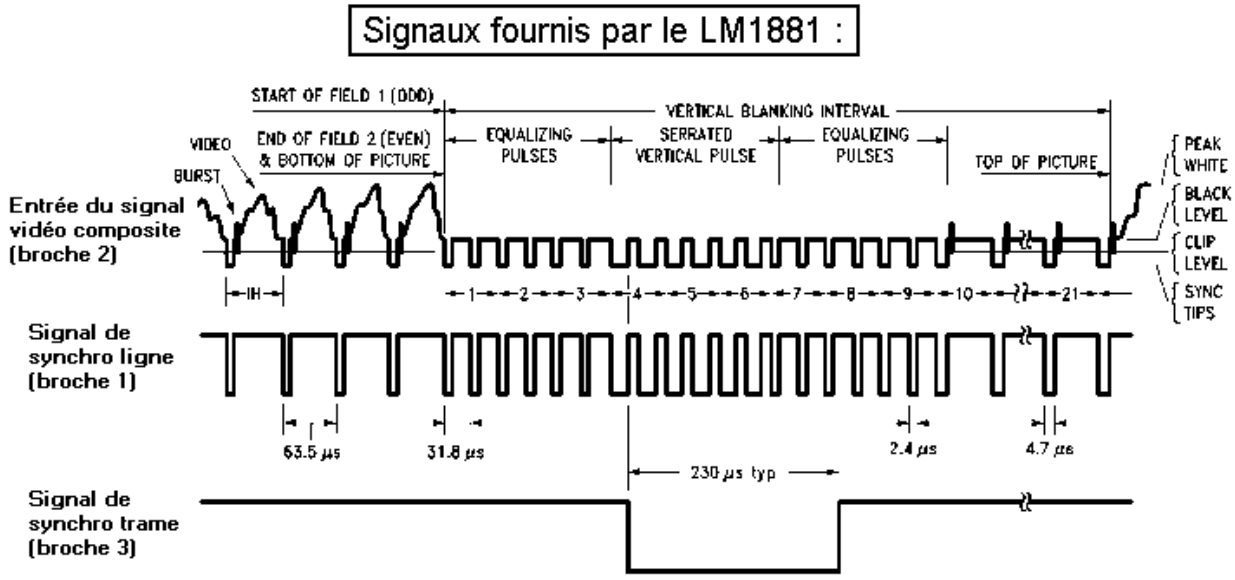


Le schéma ci-dessus correspond à celui fourni par F1CJN ; il se décompose en trois blocs fonctionnels axés autour des composants suivants : Le LM1881, qui permet d'extraire les tops de synchronisation 'lignes' et 'trames' du signal vidéo. Ces tops vont rythmer le fonctionnement du PIC, lui permettant d'exécuter son travail en parfaite synchronisation avec le signal vidéo.

- Une bascule 'RS' (constituée des deux portes NAND du haut, en rouge) qui autorisent ou non le fonctionnement de l'horloge interne du PIC, ce qui permet à celui-ci de se mettre en sommeil en attendant la survenue d'un événement précis : le début d'affichage de la prochaine ligne.
- Le PIC constitue le cerveau du dispositif : il est capable, grâce aux indications qui lui sont fournies, de détecter le début d'une image (en fait, d'une trame, donc 'une demi-image') ; il va alors attendre un certain nombre de lignes, puis va afficher un message de 11 caractères ; chaque caractère est constitué d'une 'matrice' de 5 x 7, soit 7 lignes superposées, et composées chacune de 5 pixels disposés horizontalement.

2-1) - Le LM1881 :

À partir du signal vidéo fourni sur la broche 2 (via un condensateur de liaison), il va extraire les tops de 'synchro ligne' (sortie broche 1), et de 'synchro trame' (sortie broche 3), comme indiqué ci-dessous :

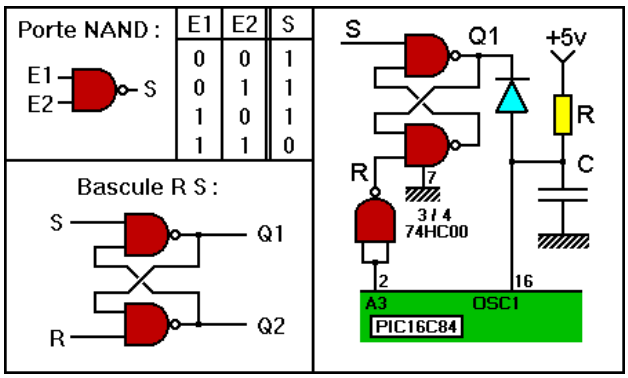


NB : la documentation étant américaine, les durées indiquées correspondent à une fréquence du secteur (ou de trame) de 60 Hz, pour des images de 525 lignes, soit une fréquence ligne de (60/2) x 525 = 15750 Hz. Une ligne dure alors : 1/15750 = 63,5 µs, contre 64 µs chez nous, notre *vieille Europe* ayant adopté F(trame) = 50 Hz, F(ligne) = 15625 Hz pour 625 lignes par image. L'ordre de grandeur ne change pas ; retenons simplement qu'un top de synchro trame dure environ 230 µs, et un top de synchro ligne un peu moins de 5 µs (sauf au moment du changement de trame). Les tops sont actifs sur un niveau logique "0".

2-2) - La bascule RS et le blocage de l'horloge :

La bascule est constituée de deux portes NAND combinées selon un montage classique : chaque sortie est rebouclée sur l'entrée de la porte opposée ; les états des sorties dépendront donc non seulement des états des entrées à un instant donné, mais également de l'état des sorties à l'instant précédant le basculement des entrées (logique 'séquentielle', par opposition à la logique 'combinatoire'). La table de vérité de la porte NAND est rappelée pour mémoire.

Regardons ensemble le fonctionnement bascule RS :



Étape 1	Supposons R=0 et S=1. R=0 implique Q2=1. S=1 et Q2=1 impliquent Q1=0. Le système est stable.
Étape 2	R passe à 1 : Q1 vaut 0, donc Q2 reste à 1. S et Q2 sont à 1, donc Q1 reste stable à 0.
Étape 3	S passe à 0 : Q1 bascule à 1. R et Q1 étant à 1, Q2 bascule à 0. Q2 valant 0, Q1 reste stable à 1.
Étape 4	S repasse à 1 : Q2 valant 0, Q1 reste à 1. Q1 et R étant à 1, Q2 reste stable à 0.
Étape 5	Dernier cas de figure susceptible de se produire : si l'on force R et S à 0 simultanément, les deux portes ayant chacune une entrée à 0, les deux sorties Q1 et Q2 sont forcées à 1 et restent dans cet état stable tant que R et S demeurent à 0.

Lorsque la sortie Q1 est à 1, le potentiel de la cathode de la diode est proche de 5v, et donc supérieur ou égal à celui de l'anode ; la diode est bloquée, et l'horloge du PIC peut osciller librement.
Quand Q1 passe à 0, la diode est passante (polarisée par R), et le potentiel de l'entrée OSC1 du PIC est fixé à environ 0,7v bloquant l'oscillation et donc le fonctionnement de l'horloge : le PIC se met en attente, plus aucune instruction n'est exécutée.

Tentons de résumer *simplement* le fonctionnement :

- Si R est à 1, et que S passe à 0, même de manière fugace (survenue d'un top de synchro ligne), Q1 bascule à 1 (ou reste à 1 si elle y était déjà) ; l'horloge est active ; le PIC est en mode 'travail'.
- Si S est à 1, et que R passe même brièvement à 0, Q1 bascule à 0 (ou reste à 0 si elle y était déjà) ; l'horloge est bloquée. Le PIC est au repos : en fait, c'est lui-même qui passe en mode 'repos' en mettant sa sortie A3 à 1.
- Si S et R sont simultanément à 1, la bascule reste stable, et l'horloge conserve son état précédent (active ou bloquée) ; celui-ci dépend donc de l'état précédent de S et R.
- Si S et R se retrouvent simultanément à 0, Q1 est à 1, donc l'horloge est active.

NB : la bascule RS se caractérise toujours par un état stable, en fonction du dernier ordre R ou S reçu, R et S étant actif sur un niveau 0. Les notations R et S correspondent respectivement à **R**eset (= mise à 0 de la sortie Q1) et **S**et (= mise à 1 de la sortie Q1).

Lorsque les entrées R et S de la bascule RS sont à 1, l'état de la sortie Q1 (et donc le fonctionnement de l'horloge du PIC) ne dépend que des états précédents de R et S :
- Si R est le dernier à avoir été à 0, la sortie Q1 est à 0 et l'horloge est bloquée.
- Si S est le dernier à avoir été à 0, la sortie Q1 est à 1 et l'horloge est active.
- Si R et S valaient 1 à la mise sous tension, Q1 est indéterminé, et on ne peut pas prédire si l'horloge du PIC est active tant qu'il n'y a pas de changement sur R ou S ; même remarque si R et S passent simultanément de 0 à 1. Mais dans ce montage, la survenue régulière du top de synchro ligne garantit contre tout blocage du système dû à une incertitude de l'état initial.

2-3) - Le comportement du PIC :

Cette partie est assez complexe, et je renvoie l'internaute intéressé consulter le fichier source ASM du programme du PIC, également généré par PICOSD, et étayé de nombreux commentaires. Les quelques lignes qui suivent ont pour but de *résumer* (?) les choses en essayant d'aller à l'essentiel.

2-3-1) Organisation de la mémoire du PIC :

Le PIC16F84 peut emmagasiner jusqu'à 1024 instructions, l'espace mémoire s'étend donc de l'adresse $000 à l'adresse $3FF (soit 0 à 1023, mais en notation hexadécimale). En se basant sur son contenu, on peut diviser celui-ci en trois zones :

2-3-1-a) - Zone 1 ($005 à $0FB), les tables de définition des caractères :

Chaque lettre doit être définie sous la forme d'un tableau de 7 rangées et 5 colonnes, pour indiquer les pixels qui seront allumés et ceux qui seront éteints. Concrètement, cela revient à définir une table de 7 octets, un par rangée, sachant que seuls les 5 derniers bits B4 à B0 seront utilisés (les bits B5, B6 et B7 seront eux ignorés). Voici l'exemple pour la lettre [A] :

A

	Caractère [A] :							
	B7	B6	B5	B4	B3	B2	B1	B0
Rangée 1 :	0	0	0	0	1	1	1	0
Rangée 2 :	0	0	0	1	0	0	0	1
Rangée 3 :	0	0	0	1	0	0	0	1
Rangée 4 :	0	0	0	1	1	1	1	1
Rangée 5 :	0	0	0	1	0	0	0	1
Rangée 6 :	0	0	0	1	0	0	0	1
Rangée 7 :	0	0	0	1	0	0	0	1

Au niveau du programme en assembleur, cela se traduira par une table de 7 octets pour chacun des caractères susceptibles d'être utilisés :

```
CarA    RETLW   B'00001110'     ; ....***.
        RETLW   B'00010001'     ; ...*...*
        RETLW   B'00010001'     ; ...*...*
        RETLW   B'00011111'     ; ...*****
        RETLW   B'00010001'     ; ...*...*
        RETLW   B'00010001'     ; ...*...*
        RETLW   B'00010001'     ; ...*...*
```

"RETLW" est l'instruction servant à constituer une table. Lorsque le programme 'saute' vers une instruction RETLW, celle-ci le renvoie à son point de départ, mais en ayant au préalable chargé la constante associée (par exemple : B'00001110') dans le registre de travail **W** du PIC. *Cela revient un peu à aller se servir dans le garde-manger.*
"CarA" est le pointeur de la table du 'A' : l'adresse de la mémoire où commence la table de définition du A. Les valeurs des 7 octets sont indiquées en binaire (notation B'00001110') pour faciliter une éventuelle modification manuelle de la table.
Les 42 caractères ainsi définis sont :

- L'alphabet en majuscule : ABCDEFGHIJKLMNOPQRSTUVWXYZ.
- Les dix chiffres : Ø123456789.
- Quelques signes supplémentaires : l'apostrophe, l'espace, le point, la barre de fraction, le Z minuscule (pour MHz !), et une tête (dans PICOSD, utilisez le caractère *étoile* [*]).

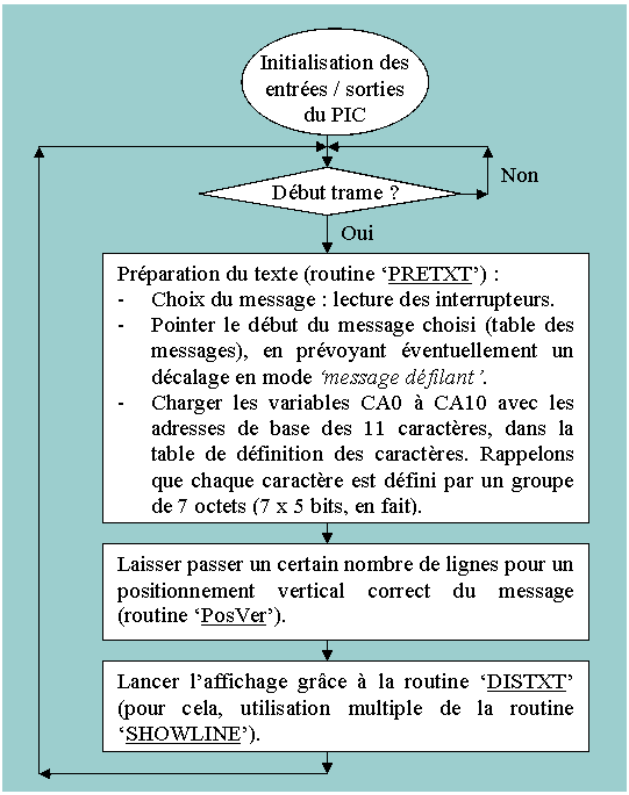
NB : La table de définition devrait occuper 42 x 7 = 294 octets, mais la syntaxe de l'assembleur (instruction RETLW) impose aux tables de ne pas sortir d'un segment de 256 octets (par exemple dans notre cas, la zone d'adresse $000 à $0FF). Une petite astuce permet de réaliser cette condition, elle consiste à faire se chevaucher les zones de définition de certains caractères, par exemple le bas du A et le haut du H, le bas du H et le haut du U, etc. Bravo au concepteur du programme !

2-3-1-b) - Zone 2 ($0FC à $24A), la partie programme à proprement parler : elle regroupe l'ensemble des routines de séquencement des opérations (analyse des tops de synchro), de choix du message, et de la gestion de son affichage. Elles seront évoquées plus loin (dans la partie 2-3-2, détaillant l'organigramme de fonctionnement).
2-3-1-c) - Zone 3 ($300 à $3F8), la table des messages :
Elle est constituée de l'ensemble des messages, c'est à dire du message défilant (jusqu'à 160 caractères suivis et précédés de 11 espaces), et des 6 messages fixes de 11 caractères. Habituellement, les chaînes de texte inscrites dans un ordinateur ou un microcontrôleur représentent la succession des valeurs ASCII des différents caractères ; cette solution pourrait être utilisée par exemple pour piloter un module d'affichage LCD standard, mais ce n'est pas le cas ici : le code ASCII ne représente rien pour le PIC, ni pour le système destiné à exploiter le signal vidéo modifié (émetteur ATV, téléviseur ou moniteur vidéo).
En fait, nous avons vu (en 2-3-1-a) que chaque table de définition d'un caractère est associé à un pointeur ("CarA", dans l'exemple), auquel nous avons donné un nom évocateur (CarA, CarB, CarC, etc.), mais cette variable virtuelle ne représente simplement que l'adresse physique de début de la table du caractère correspondant, dans l'espace mémoire du PIC. Un message de N caractères sera donc tout simplement une table de N de ces fameux pointeurs. Voici par exemple pour le message n°1 :

```
Mess1   RETLW   Car7  ; pointeur vers la table de définition du 7
        RETLW   Car3  ; pointeur vers la table du 3
        RETLW   SP    ; pointeur vers la table de l'espace
        RETLW   CarD  ; pointeur vers la table du D
        RETLW   CarE  ; etc...
        RETLW   SP    ;
        RETLW   CarF  ;
        RETLW   Car8  ;
        RETLW   CarE  ;
        RETLW   CarG  ;
        RETLW   CarQ  ;
```

Les restrictions évoquées au paragraphe 2-3-1-a s'appliquent également ; la nécessité de rester dans le segment $300 à $3FF limite donc le nombre de messages fixes, ainsi que la longueur maximale du message défilant.

2-3-2) Organigramme général :



2-3-2-1) - Supposons l'horloge active. Le PIC surveille sur son entrée A2 le top de synchro trame ; il va décider d'attendre le début d'une nouvelle trame, et scrute pour cela la fin du top de synchro (front montant sur A2).

2-3-2-2) - Puis il va ensuite lire l'état des bits PB5/PB6/PB7 ; il s'agit des entrées permettant à l'utilisateur d'indiquer le choix du message en cours, soit par trois interrupteurs (DIL ou autres), ou bien par un commutateur 7 ou 8 positions associé à une matrice à diodes, comme nous le verrons plus loin. Choix de l'affichage :

Comm.	PB7	PB6	PB5	Message sélectionné :
K7	0	0	0	message défilant
K6	0	0	1	message n°1
K5	0	1	0	message n°2
K4	0	1	1	message n°3
K3	1	0	0	message n°4
K2	1	0	1	message n°5
K1	1	1	0	message n°6
I.Off	1	1	1	pas d'affichage

À l'origine, le programme donnait le choix entre un message défilant (jusqu'à 55 caractères) ou trois messages fixes de 11 caractères ; comme vous le constatez dans le tableau ci-dessus, le nombre de messages fixes a été porté à six ; le message défilant peut mesurer jusqu'à 160 caractères.

2-3-2-3) - Il va ensuite laisser passer un certain nombre de lignes, car l'affichage ne commence jamais tout en haut de l'écran. Pour cela, il va exécuter autant de fois que nécessaire le couple d'instructions suivant :

[mise à 1 de A3] [mise à 0 de A3]

La mise à 1 de A3 (= R à 0) bloque le fonctionnement de l'horloge ; celle-ci ne redémarre que lorsque survient le top de synchro ligne (S mis à 0 par le LM1881), qui indique le début d'une nouvelle ligne. Il faut alors que la première action réalisée par le PIC soit de remettre à 0 la sortie A3, pour que l'horloge reste activée au-delà de la fin du top de synchro ligne. Le PIC patiente encore quelques µs pour être sûr que le top de synchro ligne est terminé. L'opération, réalisée N fois, permet de laisser passer N lignes, et ainsi d'obtenir le positionnement vertical voulu.

Remarque : Les N lignes de décalage dans la trame correspondent à (2 x N) lignes dans l'image, car une image est constituée de deux trames entrelacées.

2-3-2-4) - La partie affichage du message :
Que le message soit fixe ou défilant, le programme va toujours afficher 11 caractères, en utilisant la routine "DISTXT".

2-3-2-4-a) La routine "DISTXT" :

Chacun des 11 caractères est composé de 7 rangées superposées, de 5 bits ; les bits à "1" seront affichés en blanc sur l'écran ; voici par exemple les trois caractères "EGQ" (on extrapolera facilement pour onze caractères) :

EGQ	Caractère n°1 :					Caractère n°2 :					Caractère n°3 :				
	B4	B3	B2	B1	B0	B4	B3	B2	B1	B0	B4	B3	B2	B1	B0
Rangée 1 :	1	1	1	1	1	0	1	1	1	0	0	1	1	1	0
Rangée 2 :	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1
Rangée 3 :	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1
Rangée 4 :	1	1	1	0	0	1	0	0	1	1	1	0	0	0	1
Rangée 5 :	1	0	0	0	0	1	0	0	0	1	1	0	1	0	1
Rangée 6 :	1	0	0	0	0	1	0	0	0	1	1	0	0	1	0
Rangée 7 :	1	1	1	1	1	0	1	1	1	0	0	1	1	0	1

Chacune des 7 rangées va être affichées T fois (T étant la hauteur des lettres, T allant de 1 à 10), et ce grâce à la routine "SHOWLINE" : rangée n°1 (des 11 caractères !), puis rangée 2, puis rangée 3, etc. jusqu'à la rangée n°7.

2-3-2-4-b) La routine "INCLINE" :

Juste avant l'affichage de la rangée N, le programme doit aller chercher les 11 octets correspondant à la N-ième position, dans les tables de définition des 11 caractères ; elles sont stockées en interne dans les variables **Ta0** à **Ta10** du programme ; ceci prend un certain temps (environ la durée d'une ligne-écran), ce qui explique l'apparition d'une ligne noire entre deux rangées successives, ce qui n'est pas très gênant au niveau de l'aspect des lettres.

2-3-2-4-c) La routine "SHOWLINE" :

Les 11 variables "Ta0" à "Ta10" contiennent chacune les 5 bits constituant la rangée à afficher, pour chacun des 11 caractères. Voyons comment se déroule l'affichage des 5 pixels pour un caractère, par exemple le premier (bits contenus dans "Ta0") :

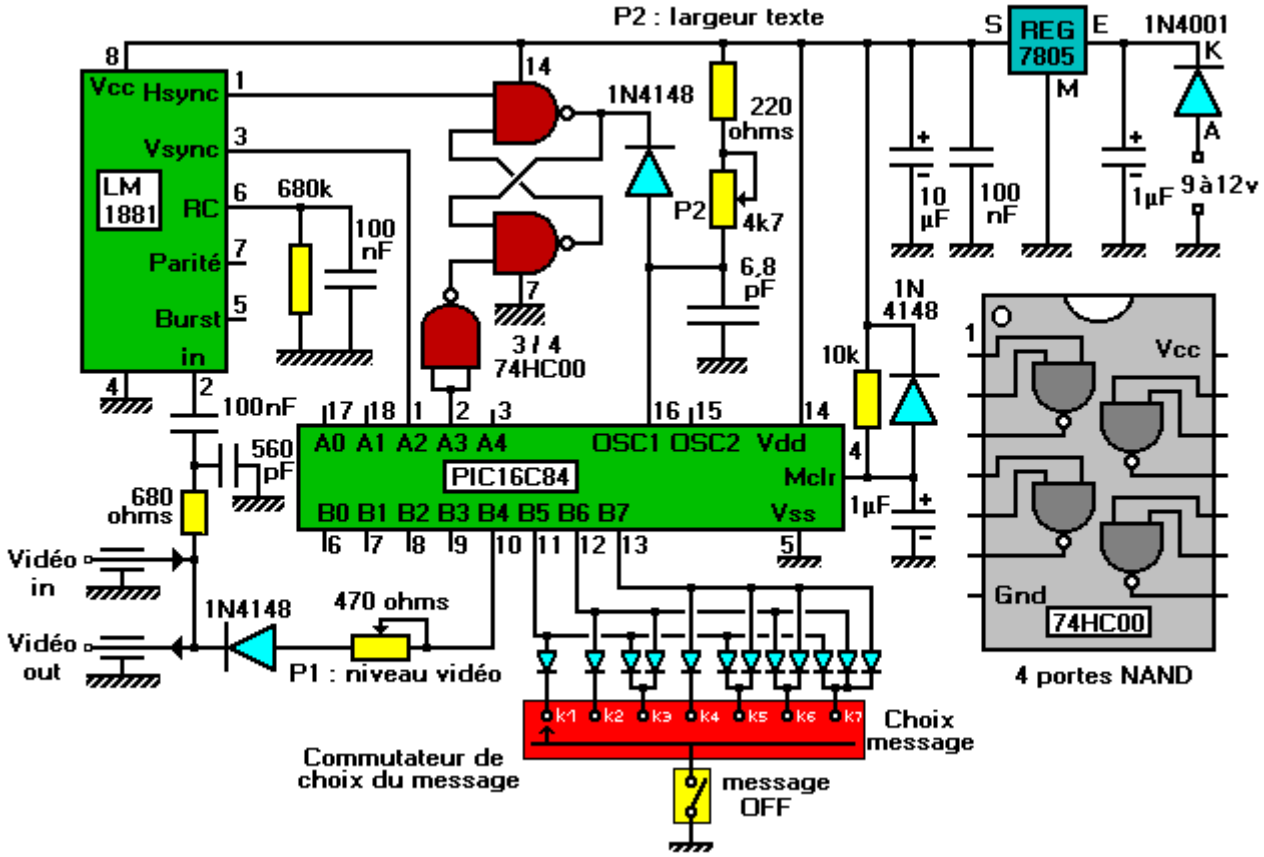
"Ta0" est une variable définie sur 8 bits (un octet), mais seuls les 5 bits de poids faibles sont significatifs et seront affichés, et ce, par ordre d'apparition : B4, B3, B2, B1, puis B0.

Le programme envoie le contenu de l'octet "Ta0" sur le port B, dont la sortie PB4 est reliée à l'entrée-sortie vidéo par une diode ; celle-ci permet de n'envoyer une information sur le signal vidéo que lorsque la sortie PB4 est à "1" ; l'ajustable 470 Ω sert à régler l'injection de courant de manière à ce que le niveau du signal vidéo atteigne, mais sans trop le dépasser, le niveau du blanc. On a donc un pixel blanc si le bit B4 est à 1. le programme va ensuite effectuer 4 fois un "décalage logique vers la gauche" du port B (instruction "RLF PortB"), ce qui signifie que les bits B3, puis B2, puis B1, puis enfin B0, vont être présents successivement sur la sortie PB4, et ainsi répercutés en temps voulu dans le signal vidéo, et donc sur l'écran du moniteur ; PB4 est ensuite remise à "0" pour éviter que le dernier des 5 pixels s'étire en longueur s'il était blanc (= si B0 valait 1). On procède de même pour les 10 caractères restants, avec les variables "Ta1" à "Ta10".

Quelques remarques :
- L'utilisation de l'instruction RLF peut sembler anodine, en fait elle est très astucieuse, car c'est la seule possibilité pour fixer une sortie du port B en un seul cycle machine, condition indispensable à un affichage assez rapide avec un PIC16F84-04. Le non-respect de cette condition entraînerait des lettres démesurément larges.
- Les broches PB5-PB6-PB7 sont configurées en entrées, elles ne sont donc pas influencées par l'instruction RLF.
- Les broches PB0-PB1-PB2-PB3, bien qu'apparemment non utilisées, doivent impérativement être laissées libres, et configurées en sorties, puisque c'est sur ces sorties que vont défiler les bits B3-B2-B1-B0 avant d'arriver sur PB4.

2-4) - Modifications du schéma électrique :

Le schéma électrique présenté en haut de cette page est une reprise du schéma d'origine fourni avec la première version du programme par F1CJN ; le schéma final utilisé pour la réalisation du prototype est le suivant :



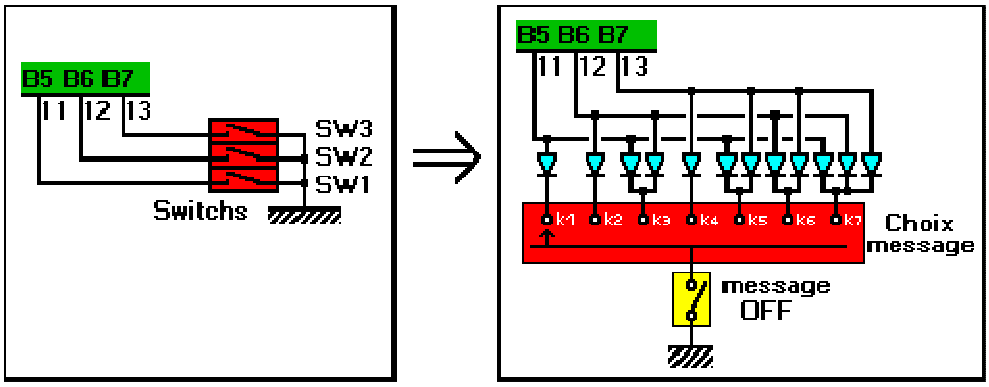
Il comporte quelques modifications mineures :

- L'ajout d'un circuit R-C (680 Ω / 560 pF) sur l'étage d'entrée du LM1881 :
  Celui-ci constitue un filtre passe-bas avec une fréquence de coupure à -3dB égale à 1/(2.π.R.C), soit 418 kHz ; cela permet de réduire notablement les sous-porteuses "chrominances" et "son" ; nous évitons ainsi qu'elles viennent se superposer au signal vidéo composite N&B classique, et perturber la détection des seuils de synchro. L'utilisation de ce circuit RC est recommandée dans la documentation du LM1881 ; j'ai simplement remplacé les valeurs préconisées (620 Ω / 510 pF) par des valeurs généralement plus disponibles ; elles sont peu critiques.
- L'ajout d'un condensateur de 1 µF et d'une diode sur l'entrée de remise à zéro du PIC (broche 4) :
  La résistance de 10 kΩ initialement présente ne servait pas à grand chose, le courant absorbé par l'entrée étant négligeable. Associée au condensateur, elle permet de maintenir l'entrée RAZ au niveau logique bas pendant une fraction de seconde (environ 10 ms) au moment de la mise sous tension : le PIC est alors inhibé le temps que la tension d'alimentation s'établisse de manière stable. La diode permet une décharge rapide du condensateur en cas de chute de tension sur l'alimentation, et donc la remise à zéro se produit bien, y compris sur une baisse de tension très impulsionnelle, qui pourrait affecter partiellement les données des registres du PIC (sans nécessairement suffire à le réinitialiser).

- Utilisation d'un PIC 16F84 ou 16C84 en version 4 MHz :
  Le concepteur prévoyait à l'origine d'utiliser un PIC en version "10 MHz", lequel devient difficile à trouver ; par ailleurs, les PICs en version "20 MHz" sont plus coûteux. Bien que la vitesse d'exécution soit assez critique, rien n'empêche de

faire fonctionner un PIC, prévu pour être cadencé à 4 MHz, à une vitesse un peu plus rapide, et c'est ce que nous avons fait dans ce montage, avec l'oscillateur en mode RC. Il suffit simplement d'être raisonnable, et d'éviter les opérations au timing critique pour le PIC, l'écriture en EEPROM notamment.
La capacité du circuit RC de l'oscillateur étant faible, le système peut être légèrement sensible à l'effet de main, il est donc préférable de laisser un espace dégagé d'un ou deux centimètres sous le circuit imprimé.

- Commutateur de choix du message :
  Comme expliqué en 2-3-2-2), l'état des entrées PB5-PB6-PB7 permet d'indiquer au PIC lequel des messages sera affiché. Signalons que les résistances de tirage au +5v internes au PIC ont été activées logiciellement, il suffit donc de mettre les entrées voulues à la masse ou de les laisser en l'air ; la combinaison "111" (avec les trois entrées en l'air) correspond à l'absence de message affiché. À l'origine, le concepteur a prévu trois interrupteurs, et j'ai moi-même utilisé des interrupteurs DIL pour mon premier prototype. Dans la nouvelle version, je vous propose de remplacer les trois interrupteurs par une matrice à diodes associée à un commutateur à sept positions, et à un interrupteur :



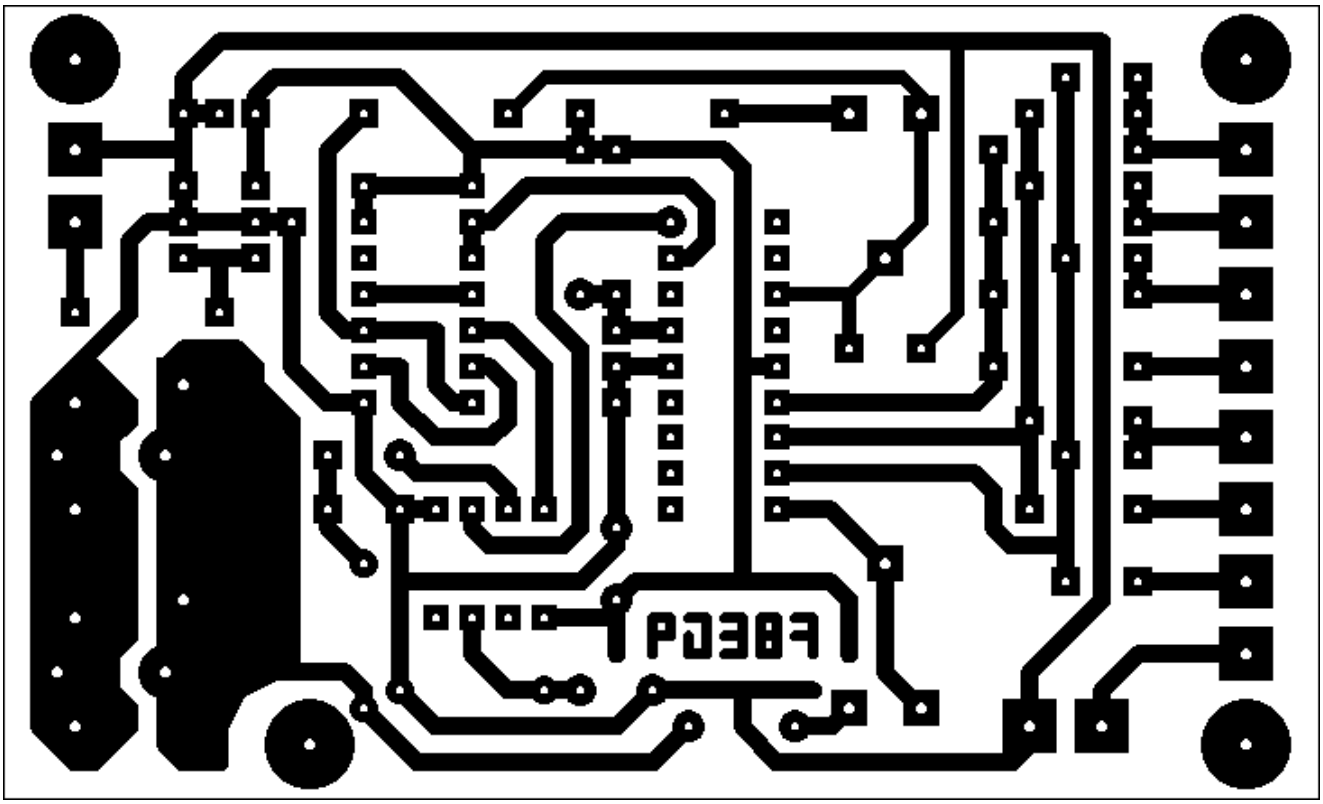
L'interrupteur permet de passer à tout moment en mode "absence de message" (inter ouvert = les trois entrées mises en l'air), ce qui permet de basculer entre deux messages extrêmes sans passer par les messages intermédiaires ; et l'utilisation d'un commutateur me semble une solution plus élégante, d'autant que le nombre de messages disponibles est plus important que dans la version d'origine. Le commutateur est un très classique "1 circuit / 12 positions" dont on peut limiter le nombre de positions grâce à une rondelle munie d'un ergot, qui est située sous l'écrou de fixation.

Remarque : le circuit existe en trois versions différentes.
La première version correspond à la version d'origine, équipée d'un triple inter DIL, sans la résistance de 10k sur l'entrée RAZ du PIC (un prototype réalisé). La seconde conserve le triple inter DIL, mais inclut le circuit RC sur l'entrée du LM1881, ainsi que le circuit de RAZ amélioré (pas de proto réalisé, mais le dessin du circuit est disponible). La troisième version utilise la matrice de diodes et le commutateur, c'est celle donc la description vous sera proposée dans les lignes qui suivent. Quelle que soit la version que vous choisirez de construire, sachez qu'elle peut accueillir indifféremment la version d'origine du programme, ou celle modifiée par mes soins et générée par le programme PICOSD. Je vous recommande tout de même de réaliser la troisième version.

3°) - Réalisation détaillée du montage (version 3) :

3-1) - Réalisation du typon :

Le dessin du circuit imprimé a été créé grâce au logiciel ARES, de la suite PROTEUS 5.2 ; il est directement issu du schéma électrique présenté plus haut ; le voici vu **côté composants** en basse résolution (200 DPI) :



PICOSDV3.LYT

Dessin du circuit imprimé et implantation des composants au format Ares 5.2 (ZIP, 9 ko).

OSDPCBv3.GIF

Dessin du circuit imprimé au format GIF (55 ko), en haute définition (600 dpi).

Si vous disposez du logiciel ARES (de la suite PROTEUS), vous pourrez imprimer directement le typon à l'échelle 1 depuis le logiciel, en chargeant le fichier "PICOSDV3.LYT". C'est la méthode la plus simple.

Dans le cas contraire, vous devrez imprimer le fichier graphique "OSDPCBv3.GIF" en respectant impérativement les dimensions, pour obtenir un typon à l'échelle 1. Celui-ci est fourni avec une résolution de 600 DPI (600 points par pouce), et il mesure 2190 pixels x 1320 pixels, ses dimensions sont donc :
- en pouces : 3.65 x 2.20
- en mm : 92.7 x 55.9
(Rappel : 1 pouce = 25.4 mm)

Si votre logiciel de traitement d'image préféré ne sait pas le faire (imprimer un fichier à des dimensions données), voici une solution simple détaillée :

1°) Téléchargez l'utilitaire Paint Shop Pro 3.12 (1,86 Mo) ; décompressez-le, puis installez-le.
2°) Lancez PSP.EXE, et chargez le fichier "OSDPCBv3.GIF" : Menu [File], puis [Open].
3°) Imprimez-le : Menu [File], puis [Print], puis [Page setup...].

Dans la boîte de dialogue "Page Setup" :
- Les dimensions sont par défaut en "pouces" (="Inches" en anglais), vous pouvez les laisser ainsi ; la case "Maintaint Aspect Ratio" doit être cochée ; la case "Use Full Page" doit être décochée ; la case "Center On Page" peut rester cochée (peu importe).
- Image Width : tapez la largeur de l'image (3.65 pouces).
- Image Height : tapez la hauteur de l'image (2.20 pouces).
- Puis validez, et quittez la boîte de dialogue : [OK].
Vous vous retrouvez dans la boîte de dialogue "Print", lancez enfin l'impression en tapant [OK].

Conseils pour l'impression du typon :

L'impression doit être impeccable, c'est-à-dire avec des zones noires bien opaques, sans coupures, sans bavures ; par conséquent, configurez votre imprimante pour une impression en noir et blanc, en haute qualité. Personnellement, j'imprime directement sur un transparent spécial pour imprimantes à jet d'encre, et je superpose deux exemplaires pour gagner en opacité. Vous pouvez également imprimer sur papier ; il faudra alors, soit rendre celui-ci perméable aux UVs (bombe "Diaphane" de KF), soit faire une photocopie laser de bonne qualité en noir et blanc sur un transparent prévu à cet effet. Quelle que soit la méthode, assurez-vous que le résultat final est bien à l'échelle 1 : le plus simple est de vérifier la coïncidence des broches et des trous en positionnant le PIC sur le typon (sans éraflure).

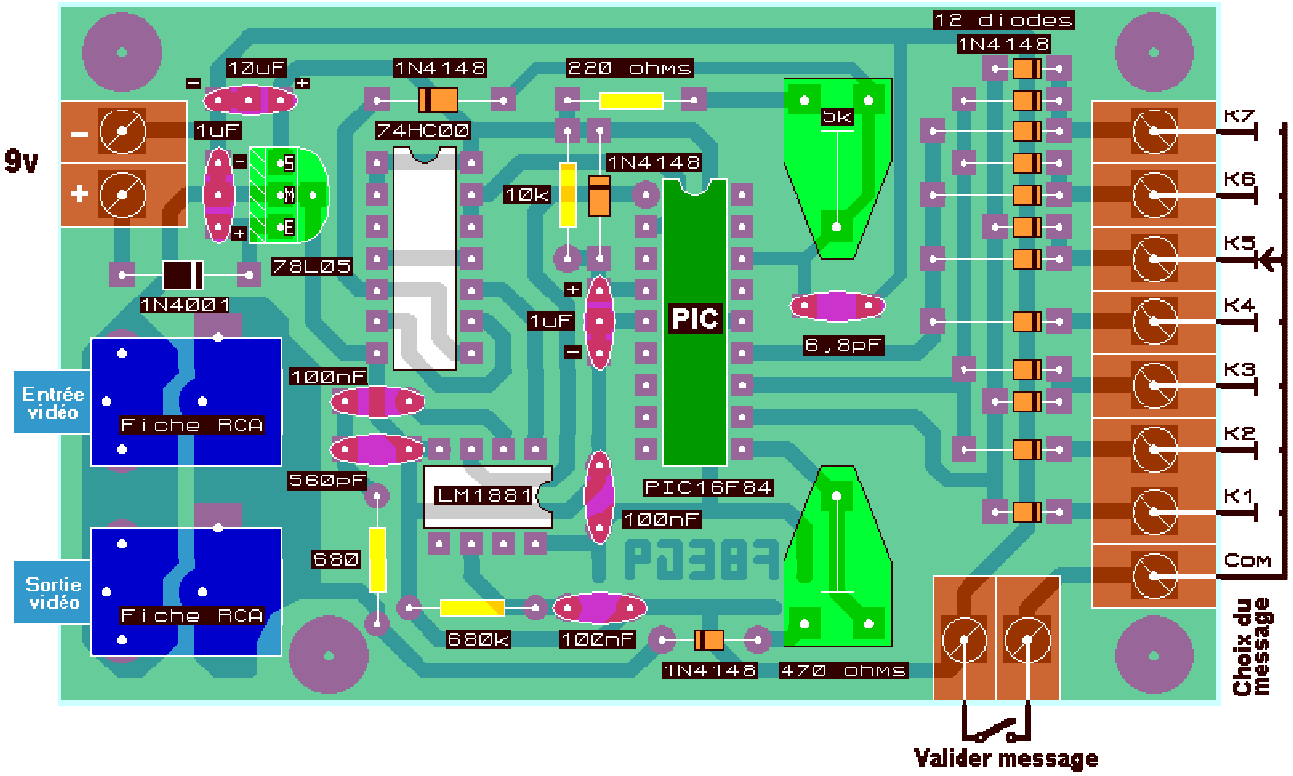
3-2) - Réalisation du circuit imprimé :

Attention ! Le dessin du circuit imprimé est vu côté composants : une fois le dessin photocopié ou imprimé sur un transparent, il faudra donc plaquer contre le cuivre la face encrée du transparent. Découpez un morceau de plaque présensibilisée d'environ 6 cm par 9,5 cm. Vous pouvez alors insoler, révéler, puis graver et percer le circuit. L'indicatif doit être lu à l'endroit, côté cuivre. Avant de percer, vérifiez tout de même les dimensions des broches des fiches "vidéo" RCA, pour adapter si nécessaire la position des trous, car il peut y avoir des différences selon les modèles.

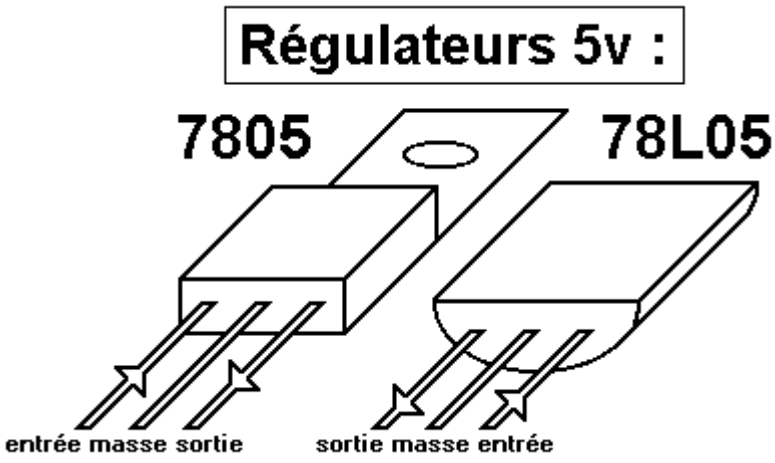
3-3) - Liste des composants :

- 1 résistance de 220 Ω.
- 1 résistance de 680 Ω.
- 1 résistance de 10 kΩ.
- 1 résistance de 680 kΩ.
- 1 ajustable petit format horizontal 470 Ω.
- 1 ajustable petit format horizontal 5 kΩ.
- 3 condensateurs 100 nF.
- 1 condensateur 6,8 pF céramique.
- 1 condensateur 560 pF céramique.
- 2 condensateurs 1 µF tantale (tension de service 25v mini).
- 1 condensateur 10 µF tantale (tension de service 6v mini).
- 15 diodes 1N4148 (ou 1N914).
- 1 diode 1N4001 (ou 4004, ou 4007).
- 1 PIC16F84 à 4 MHz.
- 1 74HC00.
- 1 LM1881.
- 1 régulateur 5v "78L05" (ou à défaut, un 7805).
- 1 commutateur 1 circuit - 12 positions (limité à 7 positions grâce à la rondelle à ergot située sous l'écrou de fixation).
- 1 interrupteur.
- 1 support "tulipe" 2 x 9 broches (18 broches, pour le PIC16F84).
- 1 support "tulipe" 2 x 7 broches (14 broches, pour le 74HC00).
- 1 support "tulipe" 2 x 4 broches (8 broches, pour le LM1881).
- 2 fiches RCA femelles, à souder sur circuit.
- 2 borniers à vis "2 plots", à souder.
- 1 bornier à vis "8 plots", à souder (ou 4 borniers "2 plots" accouplés).
- 1 clip de pile 9v et une pile 9v, ou bien une alimentation 9 à 15v *propre*.
- Divers : mèches, soudure, perchlo, révélateur, circuit imprimé, fil, etc.

3-4) - Implantation des composants :



Soudez les composants par ordre de taille, en respectant le sens de ceux qui sont polarisés : d'abord les diodes 1N4148, puis les résistances et la 1N4007, puis les supports des circuits intégrés, les résistances ajustables, puis les condensateurs et le régulateur, les borniers, et enfin les fiches RCA. Utilisez impérativement un support pour le PIC, et de préférence un support "tulipe" afin qu'il puisse supporter les multiples démontages lors des reprogrammations (changements de messages). Le régulateur 5v peut être un 7805 (arrière métallique), ou un 78L05 dont le brochage est inversé :



Dans les deux cas, attention à son sens d'implantation. Le 78L05 est largement suffisant compte tenu de la très faible consommation du montage (environ 11 mA, sur mon prototype) ; la face plate comportant le marquage sera alors orientée du côté du bornier d'alimentation. En revanche, si vous montez un 7805, ce sera alors le dos métallique qui sera orienté vers ce même bornier.

Photo du prototype vu côté composants :

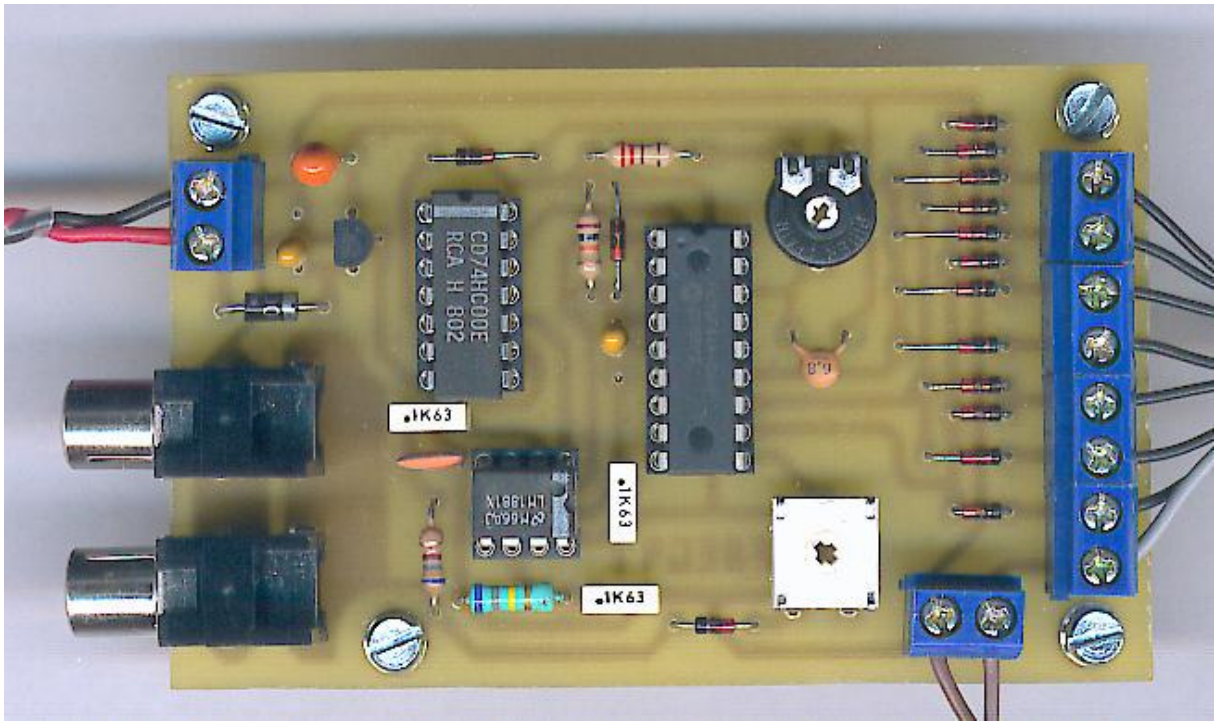
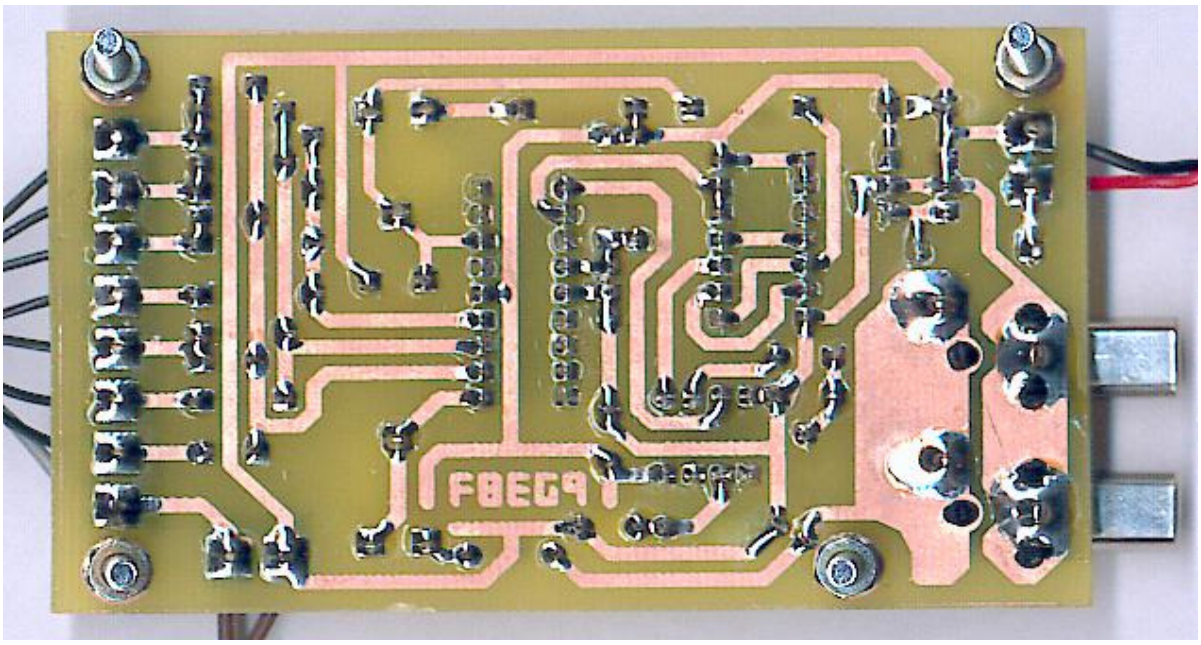
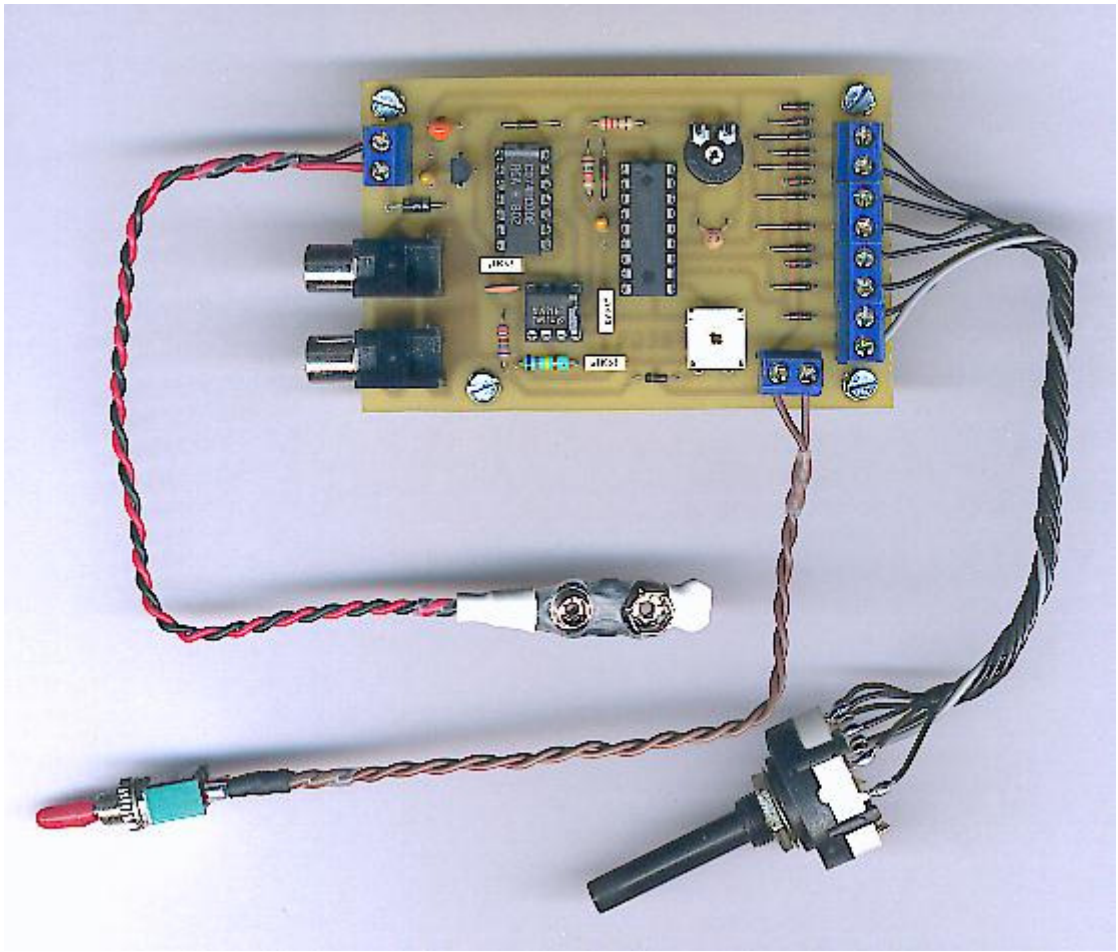


Photo du prototype vu côté soudures :



Vue d'ensemble du montage, avec l'interrupteur,
le commutateur et le fil d'alimentation (pile 9v) :



À droite de la photo, se trouve le bornier à huit plots sur lequel viennent se connecter les huit fils du commutateur de choix du message ; le fil gris est le commun ; les sept fils noirs sont respectivement (de haut en bas) :

Commutateur	Message sélectionné :
K7	message défilant
K6	message n°1
K5	message n°2
K4	message n°3
K3	message n°4
K2	message n°5
K1	message n°6

Le commutateur est bridé à sept positions grâce à la rondelle munie d'un ergot qui est située sous l'écrou de fixation. Lorsque l'interrupteur est ouvert (= non passant), aucun message n'est affiché ; il en va de même si le commutateur se trouve en dehors de l'une des sept positions prévues (on peut donc en laisser une huitième à cet effet).

3-5) - Vérification du montage :

1°) Avant de brancher l'ensemble, commencez par faire un contrôle général du montage :
- Vérifiez les valeurs de chaque composant, leurs sens de branchement pour ceux qui sont polarisés (circuits intégrés, condensateurs tantale, diodes, régulateur).
- Vérifiez l'absence de micro-coupures sur les pistes de la carte.
- Vérifiez que toutes les soudures ont bien été faites, qu'aucune ne déborde sur une autre piste, pastille ou soudure, qu'il n'y a pas de soudures "sèches".

BinPIC.HEX
(archive ZIP de 1,5 Ko)

2°) Téléchargez le fichier "BinPIC.ZIP" en cliquant sur l'icône de gauche ; vous allez alors en extraire le fichier de test "BinPIC.HEX" qui contient les messages, la taille et la position par défaut. Vous pouvez le programmer dans le PIC, cela va vous permettre de vérifier le fonctionnement du montage. Par la suite, le fichier HEX du PIC vous sera fourni automatiquement par le programme PICOSD, qui y aura intégré vos paramètres personnalisés. Pour l'instant, replacez le PIC programmé sur le montage (dans le bon sens !).

3°) Vous pouvez maintenant connecter le système ; mettez les ajustables à mi-course, branchez la source vidéo (caméra) et le moniteur ; à ce propos, vous aurez noté une curiosité en consultant le schéma électrique : l'entrée et la sortie vidéo sont interchangeables. Branchez alors l'alimentation, et fermez l'interrupteur de validation des messages (= inter passant). Le message sélectionné apparaît en blanc dans l'angle supérieur gauche de l'écran.

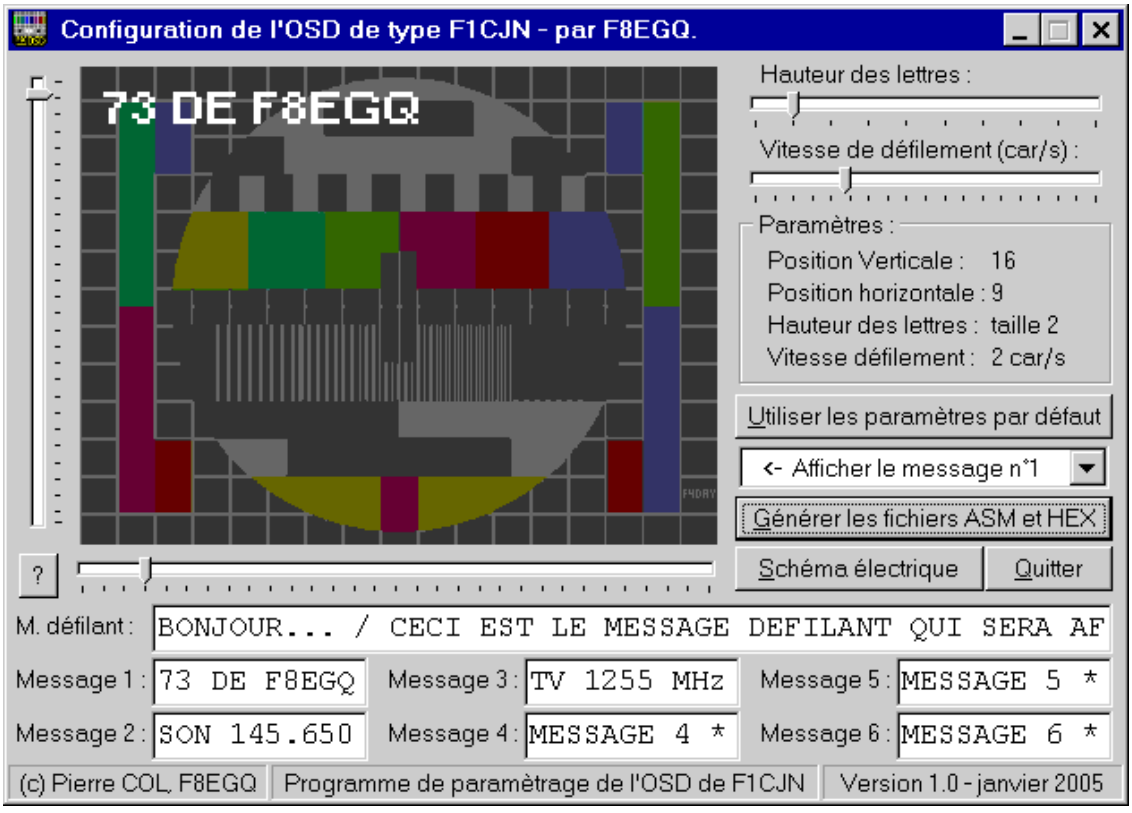
4°) Réglages :

Réglez l'ajustable de 5 kΩ (le noir, sur la photo du proto) pour modifier l'horloge du PIC, et donc la largeur des lettres. En deçà d'une certaine valeur (taille mini des lettres), l'oscillateur décroche et le message disparaît. En tournant dans l'autre sens, les lettres s'élargissent, jusqu'à ce que la largeur du message dépasse la taille de l'écran, l'affichage devient anarchique, et le signal vidéo peut être perturbé. Le meilleur compromis me semble être de se situer légèrement au-dessus de la largeur minimale, de façon à ce que le PIC ne décroche pas.
L'ajustable de 470 Ω (le blanc sur la photo du proto) permet de doser le niveau de blanc réinjecté pour constituer les lettres. Choisissez une valeur qui permette une lisibilité suffisante, sans trop saturer pour ne pas dénaturer le signal vidéo et qu'il reste dans ses limites au niveau amplitude. Cela limitera le phénomène d'écho, qui se traduit par une petite zone noire juste après une lettre trop blanche.

Vérifiez le fonctionnement correct des messages 1 à 6, ainsi que du message défilant. En principe le montage doit fonctionner dès la mise sous tension, après une éventuelle retouche des ajustables. En l'absence de message, assurez-vous qu'au moment de brider le commutateur sur les positions 1 à 7, le curseur n'est pas resté bloqué entre les positions 8 à 12. Le PIC est utilisé au-delà de sa fréquence nominale (approximativement vers 8 MHz), on ne peut donc pas exclure que dans de très rares cas, il refuse d'osciller ; essayez alors avec un autre PIC16F84 à 4 MHz, ou bien avec un PIC prévu pour fonctionner à 10 ou 20 MHz pour innocenter le reste du montage.

4°) - Le programme PICOSD

Le programme PICOSD fonctionne sous Windows 95/98/NT/2K/XP ; il permet à l'utilisateur de choisir le contenu des messages, leur position sur l'écran, la hauteur des lettres, la vitesse de défilement du message défilant. À partir des indications fournies, il va générer automatiquement le fichier "PICOSD.HEX" à programmer dans le PIC.



En décompressant l'archive PICOSD.zip, vous créez un répertoire PICOSD qui contient les quatre fichiers suivants :

- "PicOsd.exe" : c'est le fichier exécutable du programme ; depuis l'Explorateur, double-cliquez dessus pour le lancer.
- "PICOSD.HLP" : c'est le fichier d'aide qui accompagne PICOSD ; il est accessible depuis le programme en appuyant sur la touche [F1].
- "MPASM.EXE" : ce programme est l'Assembleur PIC fourni par Microchip ; en fait, PICOSD génère le fichier PICOSD.ASM (en langage "assembleur") ; puis il convertit celui-ci en fichier binaire au format hexadécimal grâce à MPASM.EXE ; on obtient alors PICOSD.HEX, à programmer dans le PIC.
- "asm.bat" : pour créer le fichier HEX, PICOSD n'appelle pas directement MPASM, mais passe par le fichier de commandes DOS "asm.bat" ; l'utilisateur averti peut alors éventuellement éditer le fichier BAT, afin de modifier les options de compilation.

Le programme en assembleur généré par PICOSD pour le PIC ("PICOSD.ASM") se rapproche évidemment beaucoup du programme initial conçu par Alain Fort F1CJN ; voici les principales modifications apportées :

- prise en charge de 6 messages au lieu de 3,
- optimisation de la longueur du message défilant,
- ajout, ou modifications esthétiques de certains caractères,
- définition de la configuration du PIC (évite le bug du WatchDogTimer),
- modifications permettant le paramétrage par des variables,
- ajout de nombreux commentaires explicatifs.

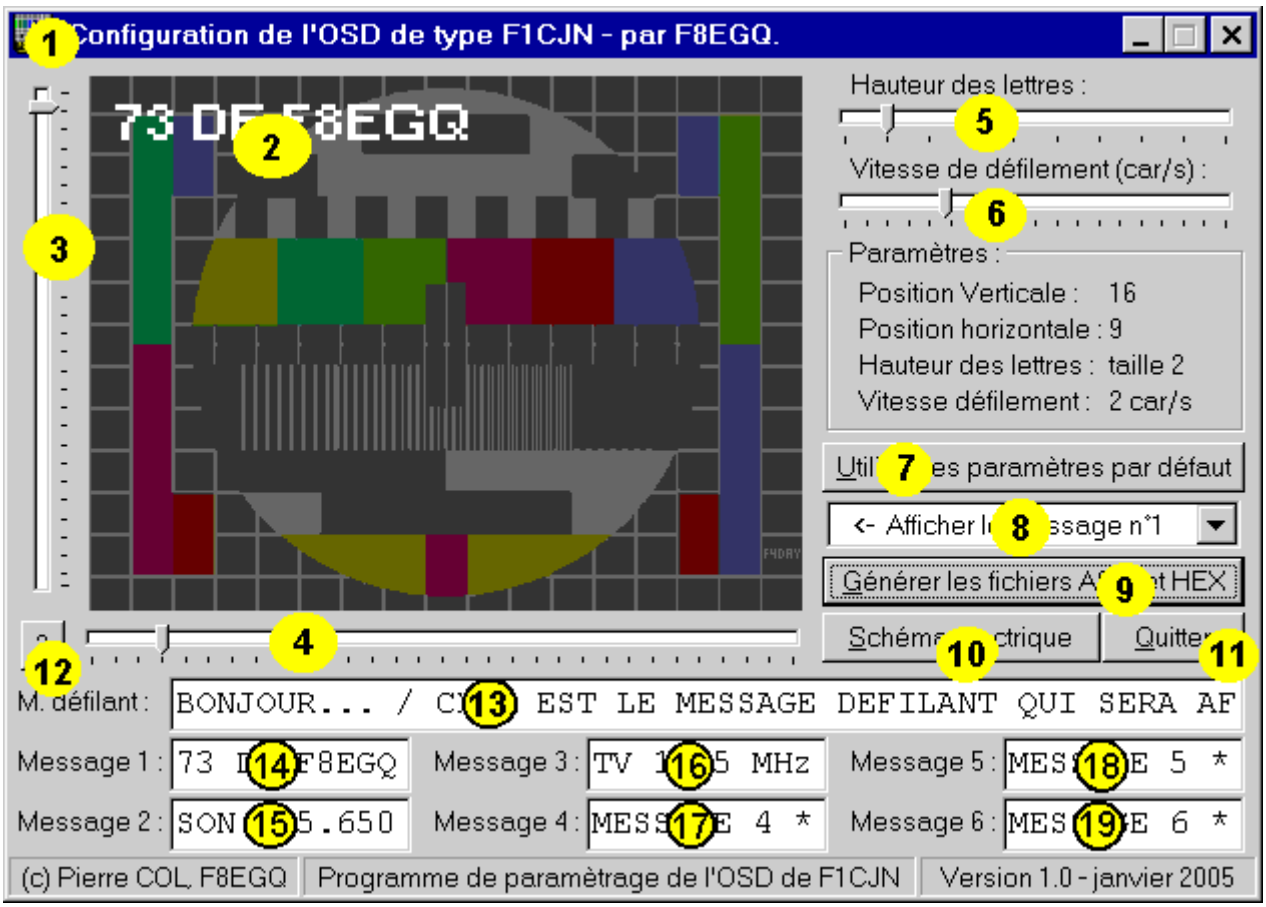
Je ne peux bien sûr pas garantir l'absence de nouveaux bugs ; en cas de soucis ou de doutes, je vous renvoie vers la version de référence, à savoir le programme d'origine (21 ko) ; vous perdrez alors les avantages de la nouvelle version ; dans ce cas, pensez bien à désactiver manuellement le WatchDogTimer dans l'utilitaire de programmation du PIC ("IC-Prog", ou autre), sinon le programme se bloquera au bout de une à deux secondes.

PICOSD

Programme de configuration pour l'incrustateur ATV conçu par F1CJN

Par Pierre COL, F8EGQ - http://col2000.free.fr/f8egq/picosd/

Regardons ensemble les différentes commandes
du programme PICOSD :



- 1 = Icône du menu système :
  Celui-ci contient notamment la commande "*Purger le registre et quitter*". Lorsque l'utilisateur quitte le programme, ce dernier sauvegarde automatiquement tous les paramètres en cours (messages, position, etc.) dans la base de registre de Windows pour la prochaine utilisation. Grâce à cette commande, pour pouvez alors supprimer ces infos. Si vous relancez le programme par la suite, cela reviendra à réinitialiser les paramètres avec les valeurs par défaut.
- 2 = Zone d'affichage :
  Elle permet de simuler (plus ou moins fidèlement) le résultat obtenu sur le moniteur pour l'affichage et le positionnement du message. Attention, la largeur réelle des caractères dépend uniquement du réglage de la fréquence d'horloge du PIC, avec l'ajustable de 5 KΩ.
- 3 = Curseur de positionnement vertical du message :
  Il permet de décaler le message vers le bas, mais attention, vous devez réglez le curseur suffisamment haut pour que les lettres soient affichées en totalité dans la zone d'affichage ; si les lettres dépassent par le bas, cela peut perturber le signal vidéo ; cette recommandation est notamment valable lorsque l'utilisateur a choisi des lettres de grande taille (de grande hauteur, en fait).

- 4 = Curseur de positionnement horizontal du message :
  Il permet de décaler le message vers la droite, mais attention, comme indiqué plus haut, la largeur réelle du texte, *comme celle du décalage*, dépend du réglage de l'ajustable 5 KΩ. Vous devez faire en sorte que la fin du message ne dépasse pas le côté droit de l'écran, car cela risquerait de perturber le signal vidéo. À vous de trouver le meilleur compromis entre le décalage horizontal et la largeur des lettres, en fonction du choix de leur hauteur.
- 5 = Curseur de réglage de la hauteur des lettres :
  Il permet de choisir la hauteur des lettres (de 1 à 10), vous pourrez ensuite adapter leur largeur grâce à l'ajustable 5 kΩ. En pratique, la valeur idéale se situe souvent entre 2 et 5, selon l'effet recherché. Dans tous les cas, le texte doit être configuré de manière à ne pas sortir de l'écran.
- 6 = Curseur de réglage de la vitesse de défilement :
  Il permet de régler la vitesse de défilement (en caractères par seconde), lorsque le message affiché est le message défilant ; la valeur indiquée est valable pour une fréquence verticale de 50 trames par seconde ; à 60 trames par secondes (normes aux USA, Canada...), la vitesse réelle sera donc plus rapide de 20% que celle annoncée. Je déconseille une vitesse trop rapide, qui est pénible à regarder (on perçoit un peu l'entrelacement des trames dans la composition des lettres).

- 7 = Bouton de réinitialisation des paramètres par défaut :
  Positionne le message en haut à gauche de l'écran, avec une hauteur des lettres de 2, et une vitesse de défilement de 2 caractères par seconde, ce qui correspond à des réglages "raisonnables" ; le contenu des messages n'est en revanche pas modifié ; pour une réinitialisation complète, voir la description de la commande 1 , ci-dessus.
- 8 = Boîte de défilement de choix du message affiché :
  Elle permet de choisir quel message sera utilisé dans la zone d'affichage pour simuler l'effet produit : message n°1 à 6, défilant, ou bien le texte "0123456789/".
- 9 = Bouton [Générer les fichiers ASM et HEX] :
  Lors de l'appui sur ce bouton le programme va créer un fichier PICOSD.ASM en langage assembleur, qui intègrera tous les paramètres définis par l'utilisateur ; il va ensuite automatiquement appeler le programme MPASM de Microchip, qui va convertir ce fichier ASM en un fichier PICOSD.HEX (en langage machine), lequel pourra alors directement être programmé dans le PIC16F84. Un fichier du listing d'assemblage est également créé : PICOSD.LST ; vous pouvez le consulter avec n'importe quel éditeur de texte pour voir le détail des opérations d'assemblage, et vérifier à la fin l'absence d'erreur.
- 10 = Bouton [Schéma électrique] :
  Il lance l'ouverture d'une fenêtre d'affichage du schéma électrique, qui peut alors éventuellement être copié dans le presse-papier pour sauvegarde ou impression ultérieure. Le schéma est aussi accessible dans le fichier d'aide.

- 11 = Bouton [Quitter] :
  Il permet de sortir du programme ; tous les paramètres (messages, position, etc.) sont alors sauvegardés dans la base de registre de Windows, et seront rechargés au prochain lancement du programme.
- 12 = Bouton [?] :
  Il permet de charger l'aide du programme, qui est également accessible par l'appui sur la touche [F1].
- 13 à 19 = Zones de saisie des messages :

  - 13 : Texte du message défilant.
  - 14 : Texte du message n°1.
  - 15 : Texte du message n°2.
  - 16 : Texte du message n°3.

- 17 : Texte du message n°4.
- 18 : Texte du message n°5.
- 19 : Texte du message n°6.

Les caractères autorisés pour les messages sont :

- les 26 lettres de l'alphabet en majuscules : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- les 10 chiffres : Ø 1 2 3 4 5 6 7 8 9
- le point (.), l'apostrophe ('), le Z minuscule (z), la barre de fraction (/),
- l'étoile (*) qui remplace le caractère personnalisé et s'affiche comme une tête ( ),
- l'espace.

Remarques diverses: Le Z minuscule permet d'écrire "MHz", ce qui est plus joli que "MHZ" ; la barre de fraction est utile pour les indicatifs en portable, en mobile ou à l'étranger, notamment. Les caractères non répertoriés ci-dessus sont affichés comme des espaces. Attention, la longueur des messages est automatiquement limitée : 11 caractères, ou 160 pour le message défilant ; or le curseur texte est en mode "insertion", et non en mode "remplacement", donc si vous avez l'impression de ne pas arriver à modifier les messages, commencez par supprimer des lettres.

Attention, j'insiste sur le fait que les réglages du texte doivent être tels que celui-ci ne doit pas "sortir de l'écran" sous peine de perturber le signal vidéo ; les premières fois, il sera sans doute nécessaire de faire plusieurs essais successifs, en retouchant éventuellement le potentiomètre de 5 kΩ (horloge du PIC). Si vos messages sont alignés à droite de l'écran et s'ils mesurent moins de 11 caractères, complétez-les avec des espaces placés en début de message (sinon les espaces sont ajoutés en fin de message, et bien qu'invisibles, ils peuvent aussi perturber la synchro).

Si vous êtes intrigué par l'aspect des différentes tailles de texte, allez jeter un coup d'oeil sur cette page : [Hauteur des lettres].

En guise de conclusion...

Ainsi s'achève cette description sans doute un peu trop longue ; j'espère que le programme PICOSD vous facilitera l'utilisation de cet incrustateur, ou que les quelques explications développées sur cette page vous convaincront de le réaliser si ce 'est déjà fait. C'est un petit dispositif peu coûteux et bien utile pour les radioamateurs pratiquant l'ATV, et puis c'est un peu notre vocation que de construire et d'expérimenter.

Bonnes bidouilles ! Pierre, F8EGQ.