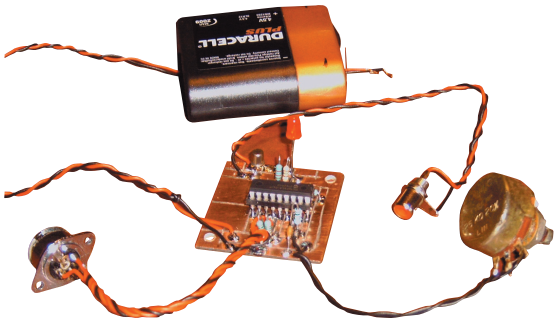
UN MANIPULATEUR ÉLECTRONIQUE SIMPLE



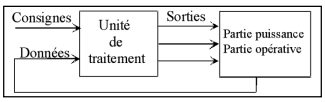
F6ALQ, Bernard LITTIERE

Avant de commencer cet article, je remercie les OM qui m'ont envoyé un courrier de remerciement ou d'encouragement pour le stabilisateur de VFO dans Radio-REF N°6 de 2002. Désolé, je n'ai pas répondu à tous. Je continue de penser qu'un bon OM n'est pas qu'un OM portefeuille.
Aussi, dans cet article, je vais essayer, par un montage simple, de faire comprendre l'utilisation d'un petit microcontrôleur, le bon vieux PIC16F84 ou PIC16F628.

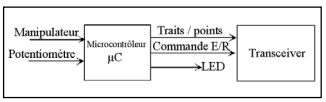
Notions de base :

Dans tout système, vous avez des entrées, une unité de traitement, puis des sorties qui vont commander la partie puissance.
Les entrées peuvent être de deux genres :

- Les consignes *(c'est le dialogue homme - machine, boutons-poussoirs, potentiomètres, roues codeuses etc.)*
- Les données informations provenant du processus *(capteurs, fin de course etc.)*.



Dans notre montage, l'unité de traitement sera le MICROCONTROLEUR (µC).



L'unité de traitement est programmée, ce qui présente l'avantage de pouvoir modifier le fonctionnement simplement en changeant le programme sans toucher au câblage.
Ici le programme sera simple, vous pourrez le modifier par la suite.

Le microcontrôleur, c'est le circuit à 18 broches.
Les broches RA0 à RA4 forment le port A ; elles peuvent servir d'entrée ou de sortie. Nous pouvons programmer individuellement chaque broche en entrée ou en sortie, c'est la configuration du port A.
Les broches RB0 à RB7 forment le port B ; elles peuvent servir d'entrée ou de sortie. Nous pouvons programmer individuellement chaque broche en entrée ou en sortie, c'est la configuration du port B.
La broche MCLR, c'est la broche " master reset " qui permet de retourner en début de programme en cas de problème.
A la mise sous tension du circuit, cette broche est actionnée automatiquement. Un petit bouton-poussoir commutant cette broche à la masse permettrait d'exécuter un " reset " en cas de problème.
Pour fonctionner, un µC ou µP (microprocesseur) a besoin d'une base de temps. Les broches OSC1/CLKIN et OSC2/CLOKOUT permettent de brancher un quartz ou un système RC pour réaliser la base de temps.
Dans notre montage, ce sera un ensemble RC formé d'un potentiomètre de 22 kΩ, d'un résistor de 10 kΩ et d'un condensateur de 100 pF qui formeront une base de temps variable : nous allons au plus simple et au moins onéreux.
RB7 et RB6 serviront d'entrée pour le manipulateur. Ne pas changer de broches ou le µC ne pourra pas se réveiller.
RB0, RB1 et RB2 seront les sorties. RB0 sera connectée à une LED qui clignotera au rythme de la graphie. RB1 sera connectée à une interface composée d'un transistor type 2N2222, 2N2218, 2N1990 ou autre, éventuellement d'un relais dont les contacts commanderont l'émetteur (dans ce cas, relier les points AA du schéma). Cette interface sera adaptée à votre émetteur. Vous pouvez extraire 20 mA sous 4,5 V de la broche RB1 sans problème. RB2 peut être utilisée pour assurer la connexion émission/ réception ; à vous de l'adapter selon vos besoins.
Le montage doit être alimenté avec une tension comprise entre 4 et 6 V, soit par des piles, soit par une alimentation stabilisée.

Schéma du montage

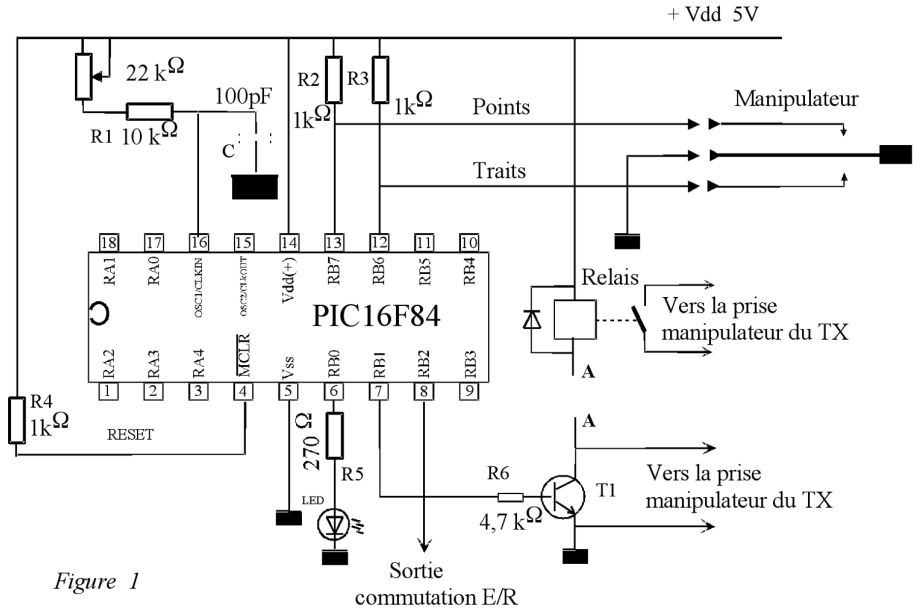


Figure 1

Programmation :

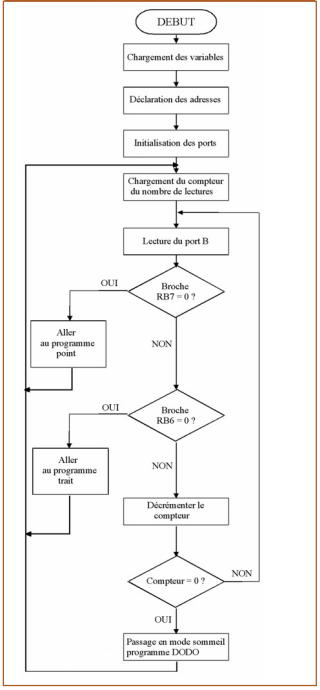
Je suis désolé, vous allez avoir besoin d'un microordinateur.
Le PIC16F84 est fabriqué par Microchip. Vous disposez uniquement de 35 codes pour le programmer.
Ces 35 codes (binaires) encore appelés instructions, déclenchent des mini-programmes à l'intérieur du µC.
Programmé en binaire, c'est très long et difficilement vérifiable. Même en traduisant le binaire en code HEXA, plus facilement mémorisable pour l'être humain, la tâche reste pénible.
Rappel du code HEXA pour ceux qui ne savent pas ou ne savent plus.

Valeur HEXA	Code binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

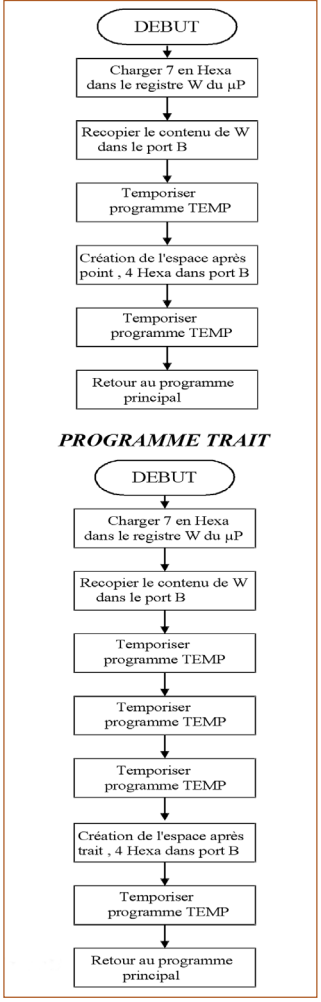
Il faut formater les codes binaires en quartés (paquet de quatre). Exemple :
0111 1110 = 7E,
2C = 0010 1100
Nous allons travailler en assembleur, langage facilement compréhensible et mémorisable. Vous trouverez les instructions, le vocabulaire, sur le site du fabricant Microchip et un logiciel gratuit MPLAB à utiliser sans modération. Pour introduire le programme dans le µC, vous aurez besoin d'un petit circuit que vous pouvez fabriquer ou acheter chez CONRAD ou autre, publicité non payée, et du petit logiciel Icprog. Nota : MPLAB permet également d'introduire le programme dans le µC.
Par la suite, je vais détailler entièrement le programme : il faut bien apprendre !
Pour comprendre ma démarche dans le programme, commençons par regarder les algorithmes nécessaires. (figure 2)
Les algorithmes me semblent suffisamment explicites pour ne pas les commenter.

Nous avons plusieurs sous-programmes que j'ai appelés INIT, PRINCIP, TEMP, COMPT, DODO, POINT, et TRAIT
INIT initialise les ports.
PRINCIP lit le port B
TEMP temporisation qui sert de base de temps.
COMPT compteur qui permet de passer en mode sommeil et réception après un certain nombre de lectures infructueuses.
DODO programme du mode sommeil.
POINT génère les points.
TRAIT génère les traits.

Programme principal



Programme point



Le programme en assembleur :

(Voir figure 3)
Le programme présenté en figure 3 est écrit en assembleur en utilisant les instructions du fabricant Microchip.
Vous devez travailler sur 3 colonnes en utilisant une tabulation pour passer d'une colonne à la suivante.
La première comportera uniquement les noms des sous-programmes ou des repères encore appelés drapeaux (flag).
La deuxième comportera les instructions.
La troisième l'opérande, c'est-à-dire sur quoi, ou avec quelle valeur s'exécute l'instruction.

Exemple :
POINT MOVLW 0X07
traduire par
Début du sous-programme POINT.
Charger le littéral suivant dans le registre W du µC. Le littéral est 07en hexa.
MOVWF PORTB
Traduire par
Recopier le contenu du registre W sur le port B.
A la fin de l'exemple ci-dessus nous aurons sur les broches du port B

RB7= 0V RB6=0V RB5=0V
RB4= 0V RB3=0V RB2=5V
RB1=5V RB0=5V

5 broches à 0 volt et 3 broches à 5 volts.

Tout ce qui est précédé d'un ; correspond à des commentaires. Il ne faut jamais hésiter à écrire des commentaires, cela permet de comprendre le programme, de le modifier plus tard sans trop de difficultés.
Les commentaires n'alourdissent pas le programme du µC, ils ne sont pas compilés et n'entrent donc pas dans le µC.
J'ai dit " compilés " ? Compiler c'est traduire le langage assembleur en langage binaire, le seul compris par le µC.
Seuls les instructions et les opérandes seront compilés. Le reste est pour la compréhension humaine.

En figure 6, vous aurez le programme compilé en HEXA. Vous pouvez à l'aide du logiciel Icprog entrer directement ce langage dans le µC.
Le fichier écrit en assembleur aura pour extension .asm,
celui compilé .hex.
Exemple :
ManipV1_0.asm et
ManipV1_0.hex

Organisation du programme :

En début du programme, titre explications, type de µC, date et version.
Les raccordements à réaliser.

Par la suite, je déclare des variables, cela permet de les modifier facilement sans faire d'analyse fatigante du programme.
Je donne des noms aux différents registres, notamment aux ports de sortie.

Pour le µC 16F84, les ports sont aux adresses 05 et 06. En effet les onze premières adresses de la mémoire RAM, qui en comporte 68, ont un rôle spécifique. La mémoire programme comporte 1024 lignes de 14 bits. Chaque ligne peut stocker une instruction et son opérande.
Même en l'absence de tension, la mémoire programme ne s'efface pas, heureusement !!
Une autre mémoire qui n'est pas utilisée ici, l'EEPROM, comporte 64 lignes de 8 bits pour stocker des données non volatiles.
Le fabricant nous affirme que nous pouvons reprogrammer environ 1000 fois avant la mort du µC : cela permet de tester les programmes sans trop de risque.
Les programmes peuvent être tapés en utilisant NOTEPADE© WORD ou autre mais il faut les sauvegarder en " .asm ".
Vous aurez besoin de MPLAB pour compiler, alors autant les taper directement avec MPLAB.

Vous venez d'installer MPLAB sur votre PC, comment l'utiliser ? Effectuez la démarche suivante pour obtenir un résultat : ▶

Figure 3

```
Programme pour réaliser un manipulateur électronique avec un 16F84
;*  Version 1.1 du 6 décembre 2004.
;*
;*  Entrée RB7 pour les points.
;*  Entrée RB6 pour les traits.
;*  Sortie RB0 pour une LED.
;*  Sortie RB1 pour utilisation.
;*  Sortie RB2 pour commuter E/R
;*****************************************************************
;Variables :
T0 =        0x06                ;Temporisation de base. Permet de faire varier la vitesse autrement
                                ;que par le potentiomètre.
T1 =        0xD8                ;Temporisation de base.(T0=0x06 et T1 = 0XD8 donne une
                                ;base de temps de +/-  47ms)
T2 =        0x4F                ;Permet de régler le nombre de lectures à vide avant la mise en sommeil
                                ;du µC et passage en réception.

;Adresses RAM utilisées : 0X10   0X11   0X12   0X13

; Adresses :
PORTA   EQU     05              ;Registre du port A que nous choisissons d'appeler PORTA
                                ;(Nous aurions pu lui donner un autre nom).
PORTB   EQU     06              ;Dito pour le port B.
INTCON  EQU     0B              ;Registre des interruptions d'adresse 0B (Nous l'appelons INTCON)


;*******************************************************************************
;Démarrage :

        org     0X000           ;Origine du programme à la ligne 0. 0X devant indique code en langage
; hexa.
        GOTO    INIT            ;Aller au sous programme INIT
;*******************************************************************************

;programme d'initialisation des ports, port A en entrée, port B BIT 0 à 5 en sortie, bit 6et 7 en entrée.

INIT    CLRF    PORTA           ;Mise à zéro des sorties par sécurité.
        CLRF    PORTB
        MOVLW   0X0F            ; 0X0F =  0000   1111 en binaire  (0X indique écriture hexa)
        TRIS    PORTA           ;PortA en entrée.
        MOVLW   0XC0            ; 0XC0 =  1100   0000  (1 = entrée  0 = sortie).
        TRIS    PORTB           ;Port B BIT 0 à 5 en sortie, bit 6et 7 en entrée.
        GOTO    PRINCIP         ;Envoi au programme principal.(PRINCIP).
;*******************************************************************************

;Programme de lecture du port b et compteur de cycles à vide avant le sommeil.
;Permet de savoir si le manipulateur est sur la position trait ou sur la position point.

PRINCIP MOVLW   T2              ;Chargement du compteur avec la variable T2
        MOVWF   0X12            ;Chargement du compteur avec la variable T2
        MOVWF   0X13            ;Chargement du compteur avec la variable T2

PRINC   BTFSC   PORTB   ,07     ;Test du bit 7, manipulateur côté point.
        GOTO    SAUT            ;Le manipulateur n'est pas du côté point, envoi au repaire.
        ;SAUT.
        CALL    POINT           ;Le manipulateur est du côté point, envoi au sous programme point.
GOTO    PRINCIP                 ; Après l'exécution du sous programme point retour en début de lecture
;(PRINCIP)

SAUT    BTFSC   PORTB   ,06     ;Test du bit 2, manipulateur côté trait.
        GOTO    COMPT           ;Le manipulateur n'est pas du côté trait, envoi au sous
                                ;programme;COMPT compteur
        CALL    TRAIT           ;Le manipulateur est du côté trait, envoi au sous programme
; TRAIT.
        GOTO    PRINCIP         ; Après l'exécution du sous programme TRAIT
;retour en début de lecture  (PRINCIP)

;*******************************************************************************

;Sous programme compteur, permet d'exécuter un certain nombre de lectures infructueuses
;avant de mettre le µC en sommeil. Le compteur est réalisé avec deux boucles imbriquées.

COMPT   DECFSZ  0X12            ;Décrémente et test l'arrivée à 0.
        GOTO    PRINC           ;Zéro du premier compteur pas atteint.
;Envoi au sous programme PRINC (nouvelle lecture)
        MOVWF   0X12            ;Zéro du premier compteur atteint
        DECFSZ  0X13
```

Lancer MPLAB
Cliquer sur Project ⇒ Project_Wizard
Device ⇒ 16F84 ⇒ suivant
Apparaît l'endroit ou MPLAB est installé et ses outils ⇒ suivant
Project Name donner un nom au projet exemple : MANIP
Project Directory choisir un répertoire ou le créer
exemple : C:\RADIO\Manipulateur
⇒ suivant Vous pouvez ajouter des fichiers déjà existants dans votre nouveau projet, sélectionner les fichiers puis Add
sinon ⇒ suivant ⇒ suivant.

Cliquer sur File ⇒ New taper votre programme dans la fenêtre.
(utiliser 3 colonnes ; voir précédemment et ne pas oublier END à la fin).
Sauvegardez dans votre répertoire, en utilisant File Save As.. et en lui donnant un nom (extension asm).

Sous la fenêtre du programme il y a une autre fenêtre.
Ex : MANIP.mcp
source file surligner source file puis cliquer droit. ⇒ ADD Files
Sélectionnez votre programme puis OUVRIR
Dans la fenêtre MANIP.mcp source file sélectionner votre programme.
Dans la barre supérieure cliquer sur Project ⇒ Build All
Vous venez de compiler votre programme en extension .hex
Si vous avez fait des erreurs, un tableau vous les signale, avec le numéro de la ligne correspondante.

Pour faire apparaître les numéros des lignes, cliquer droit sur la fenêtre du programme puis sur Properties
Dans Editor cocher Line Numbers .

Après avoir corrigé vos erreurs dans le programme en assembleur, vous devez recompiler (Project ⇒ Build All)
pour corriger le fichier .hex.
J'espère que ces quelques explications vous permettront de démarrer avec le logiciel MPLAB. Je vous laisse découvrir le reste.

Maintenant il faut introduire le programme dans le µC.
Raccordez votre programmateur au PC.
Placez le µC sur le support du programmateur.
Si vous utilisez Icprog,
Sélectionner Fichier ⇒ Ouvrir fichier
choisir votre fichier "xxxx.hex" OUVRIR ⇒
Votre buffer est rempli.
En haut à droite, faire apparaître 16F84

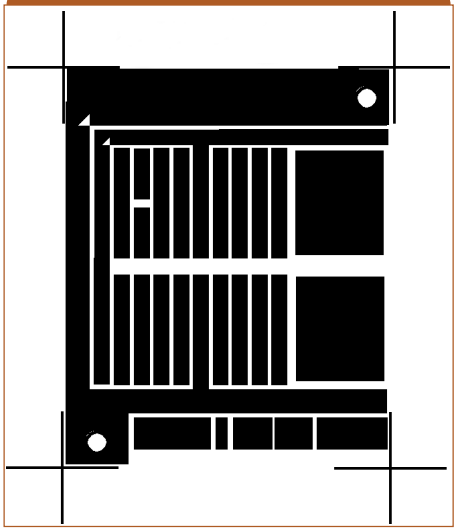
A oscillateur RC.
A fusible rien ne doit être coché.
Bien vérifier que WDT est décoché.
WDT = Watch Dog = Chien de garde.
Nous ne l'avons pas programmé.
Cliquer sur Commande ⇒ Tout Programmer

Confirmez par Yes, attendre que la programmation et la vérification s'effectuent.
Votre µC est programmé, vous pouvez l'installer sur votre montage.

Le montage :

Pour réaliser mes prototypes, j'utilise toujours une technique qui ressemble à celle des circuits imprimés pour CMS. Je grave avec un mini-disque à tronçonner. Les composants sont soudés côté cuivre. Cette technique est rapide, pas de perçage, et évite d'utiliser des produits chimiques, les grandes surfaces cuivrées permettent des soudures faciles, et des modifications éventuelles, la fixation du circuit terminé ne nécessite pas d'entretoise et le circuit résiste bien aux chocs.
Suite à mon article sur le stabilisateur de VFO, plusieurs OM m'ont demandé si j'avais réalisé un circuit imprimé : cela semblait manquer. Cette fois j'ai réalisé un circuit imprimé. Je joins le dessin du typon vu côté cuivre (Figure 4) et (Figure 5) et les photographies du prototype ; bien sûr, il est sans perçage, composants côté cuivre.

Figure 4



La consommation est d'environ 15 mA pendant l'émission, et de 0,07 mA en position réception ou sommeil du µC.
Une bonne pile alcaline de 4,5 V présente une capacité d'environ 5000 mAh, qui nous donne une autonomie en position sommeil d'environ 70 000 h soit un peu plus de huit ans ou environ 320 h de manipulation. ▶

Figure 3 (suite)

```
GOTO    PRINC                       ;Zéro du deuxième compteur pas atteint
        MOVWF   0X12                ;Zéro du deuxième compteur atteint, Recharge des compteurs
        MOVWF   0X13                ;Recharge des compteurs
        CALL    DODO                ;Envoi au sous programme DODO
        GOTO    PRINCIP             ;Envoi au début après retour du sommeil.
                ;*****************************************************************


                                    ;Programme pour mise en sommeil du microcontrôleur (permet de réduire
                                    ;la consommation).µC 16F84
                                    ;Réveil provoqué uniquement par une action sur les broches 4 à 7 du
                                    ;port B programmées en entrée, ici uniquement sur RB6 ou RB7.

DODO    CLRF    PORTB               ;Efface le contenu du portB (passage en réception).
MOVLW                               0X88
        MOVWF   INTCON              ;Pour que les interruptions soient prises en
                                    ;compte, il faut mettre 1 dans le bit 7 du
                                    ;registre des interruptions INTCON, et 1 dans
                                    ;le bit 3 pour autoriser les interruptions par
                                    ;le port B

        SLEEP                       ;Mise en sommeil du microcontrôleur pour réduire
                                    ; la consommation.

        CLRF    INTCON              ;Interdire les interruptions, après le réveil.
                                    ;Réveil provoqué par action sur RB6 ou RB7.

        RETFIE                      ; Retour au sous programme après interruption.

                ;*****************************************************************

                                    ;Temporisation à deux boucles imbriquées.

TEMP    MOVLW   T0                  ;Temporisation base de temps. Pour changer la plage de vitesse changer
        MOVWF   0X10                ;les valeurs des variables T0 et T1 en début de programme.
        MOVLW   T1
        MOVWF   0X11

BOUCLE  MOVLW   T0
        MOVWF   0X10
BOUCLE  DECFSZ  0X10                ;Décrémente, si la valeur contenue dans l'adresse 0X10 = 0 saute une
                                    ligne.
        GOTO    BOUCLE1             ;si non exécute cette ligne. Décrémente de nouveau le contenu de 0X10.
        DECFSZ  0X11
        GOTO    BOUCLE
        RETURN                      ;retourne à la ligne suivante de celle qui a envoyé à ce sous programme.
                ;*****************************************************************************


                                    ;Programme création d'un point

POINT   MOVLW   0X07                ;Charge la valeur 7 en hexa dans le registre W
                                    ;(7 = broche rb0 rb1 et rb2 à 1)
        MOVWF   PORTB               ;Recopie le contenu du registre sur le port B
        CALL    TEMP                ;Le contenu reste affiché pendant le temps de "TEMP"
        MOVLW   0X04
        MOVWF   PORTB               ;Efface le contenu du portB sauf rb2
        CALL    TEMP                ;espace derrière le point.
        RETURN                      ;Retourne au programme qui a appelé le programme "point".
                ;*****************************************************************************


                                    ;Programme création d'un trait

TRAIT   MOVLW   0X07                ;Charge la valeur 7 en hexa dans le registre W
        MOVWF   PORTB               ;Recopie le contenu du registre sur le port B
        CALL    TEMP                ;Le contenu reste affiché pendant le temps de "TEMP"
        CALL    TEMP                ;Le trait est trois fois plus long que le point !!
        CALL    TEMP                ; soit trois fois TEMP.
        MOVLW   0X04
        MOVWF   PORTB               ;Efface le contenu du portB sauf rb2
        CALL    TEMP                ;espace derrière le trait.
        RETURN                      ;Retourne au programme qui a appelé le programme "trait".
                ;*****************************************************************************
                END
```

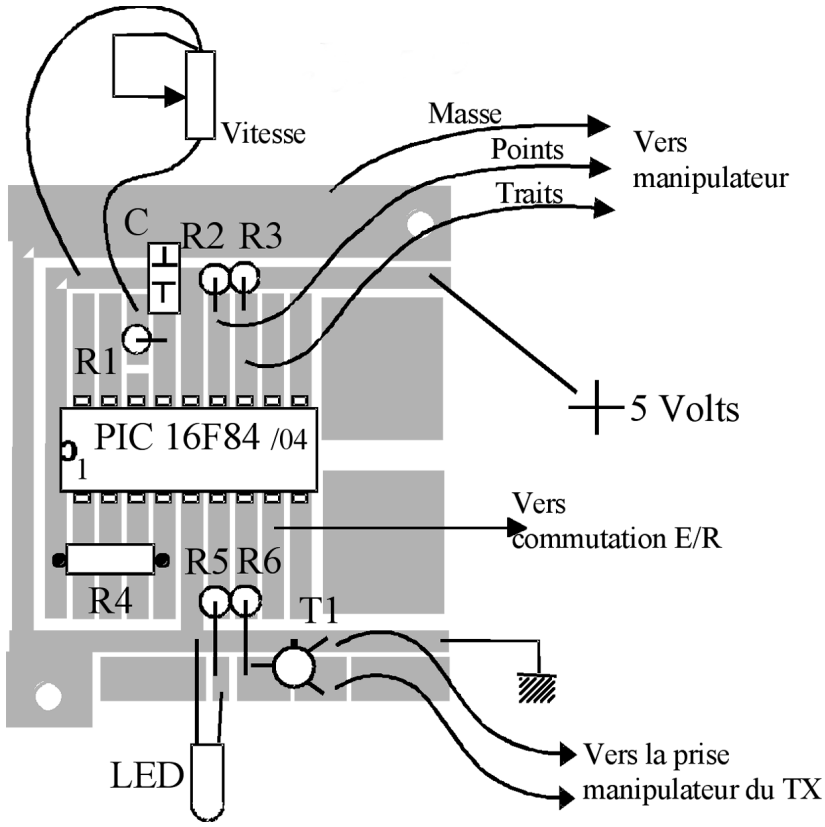
Dans ces conditions, pensez-vous rentabiliser le coût de l'interrupteur marche-arrêt ?

Un coup d'œil sur le schéma, les résistances de tirage au plus, sur les entrées du manipulateur, sont de faibles valeurs pour éviter les retours HF (1 kΩ) ; vous pouvez encore les réduire si nécessaire. Pensez à tresser les fils du manipulateur, cela limite les perturbations sur les entrées. En présence de forte puissance, il sera peut-être nécessaire d'ajouter des perles de ferrite sur les entrées et les sorties. Circuit posé sur la table, testé avec une puissance HF de 300 watts sur les bandes décamétriques, aucune perturbation n'a été remarquée. Pour améliorer le tout, il faut enfermer le circuit dans une boîte métallique bien réunie à la masse.
Pour la sortie, vous devrez peut-être apporter des modifications en fonction de votre émetteur ; sur mon TR7, la commutation s'effectue entre la masse et une tension de 9 V, donc aucun problème. Si votre tension de commutation est beaucoup plus élevée, il faudra monter un transistor qui présente un VCEO plus important. Si vous commutez une tension de grille, négative par rapport à la masse, il faudra implanter un transistor PNP.
Dans tous les cas, le montage avec un relais convient. Celui-ci sera implanté en dehors du circuit imprimé, et surtout, n'oubliez pas la diode qui limite les surtensions provoquées par la bobine du relais.
Pour modifier aisément le programme du µC, vous devez le monter sur un support.

CONCLUSION :

Par cet article, j'espère avoir motivé les OM qui hésitent à se lancer dans l'utilisation d'un microcontrôleur. Le programme proposé est très simple, il fonctionne. Avec un peu d'entraînement et d'habitude, vous pourrez l'améliorer. Vous pourrez par exemple ajouter des messages enregistrés, commander un cadencement pour commuter votre PA et vos différents relais.
Par la suite, avec d'autres montages, vous pourrez réaliser des guirlandes de Noël, des compteurs, des fréquencemètres, des filtres numériques, des commandes de moteur pas à pas, des transmissions de données etc..
Le prix du µC est compris entre 4 et 8 € suivant le modèle et le revendeur. Les PIC16F628 plus performants semblent moins chers. Attention, si vous voulez modifier vos programmes, utilisez de préférence des µC avec des mémoires Flash et EEPROM.

Figure 5



Bibliographie :

La première chose à faire est d'aller sur le site de Microchip pour récupérer MPLAB. La version que j'ai sommairement commentée est la V6 ; une nouvelle version V7 est sortie en décembre 2004.
La deuxième chose est de récupérer les caractéristiques (data sheet) du µC que vous avez choisi. Elles seront sûrement en anglais.
Sur http://www.google.fr taper PIC16F84 par exemple.
Une série d'articles est parue dans électronique pratique du N° 283 à nos jours.
Chez DUNOD, deux ouvrages écrits par Christian TAVERNIER "Les microcontrôleurs PIC, description et mise en œuvre " et "Les microcontrôleurs PIC, application" environ 40 € pièce à la boutique du REF-Union.
Nota : Je ne vends RIEN, je n'ai aucun intérêt dans aucune des publicités.

Vous trouverez les deux fichiers : MorseV1.asm et MorseV1.hex
Sur le site :
http://aeromodel.soissons.free.fr
en cliquant dans la marge sur OM.

Chers OM, à vos fers à souder, bon courage, bonne réalisation.

En figure 6, le programme compilé, que vous pouvez directement taper dans Icprog.
A gauche de 0000 à 0038 ce sont les adresses. Ne pas tenir compte des signes qui apparaissent dans la partie droite.

Figure 6

Adresses	Codes programme en hexa
0000:	2801 0185 0186 300F 0065 30C0 0066 2808
0008:	304F 0092 0093 1B86 280F 202D 2808 1B06
0010:	2813 2034 2808 0B92 280B 0092 0B93 280B
0018:	0092 0093 201C 2808 0186 3088 008B 0063
0020:	018B 0009 3006 0090 30D8 0091 3006 0090
0028:	0B90 2828 0B91 2826 0008 3007 0086 2022
0030:	3004 0086 2022 0008 3007 0086 2022 2022
0038:	2022 3004 0086 2022 0008 3FFF 3FFF 3FFF