Eurêka +
3 Avenue de l'Amiral Lemonnier
Maison des Associations
78160 Marly le Roi

Tél. : 01.39.58.87.92
Web : http://www.eurekaplus.org

- Compte rendu du projet -

décembre 2003

Marc Buchdahl
Antoine Retailleau

SOMMAIRE

I - Présentation du projet Vertigo

I.1 - L'association Eurêka +

Eurêka + est une association loi 1901 créée en 1985, dont le but est de vulgariser la connaissance des sciences et techniques et permettre à tous de réaliser des projets scientifiques : elle est pour cela organisée en 3 sections (Espace, Astronomie et Informatique), chacun des animateurs étant bénévole.

La section Espace compte une quinzaine d'adhérents qui ont entre 12 et 23 ans. Elle accueille ses membres tout au long de l'année scolaire, les vendredis soirs de 20H30 à 22H30. L'association permet à ces jeunes de construire des projets aérospatiaux (minifusées, fusées expérimentales et ballons stratosphériques) dont la complexité croît avec l'expérience acquise au cours des années.

Au cours des dix-huit dernières années, de très nombreux projets ont vu le jour : 44 minifusées, 16 fusées expérimentales, deux ballons stratosphériques, deux expériences en Caravelle 0G, … Tous ces projets ont été réalisés en équipe à l'initiative des jeunes.

L'association participe à de nombreuses manifestations telles que la Fête de la Science, campagnes de lancements régionales, le Festival des Clubs Espace …

Notre association est affiliée à Planète Sciences (anciennement ANSTJ) qui est chargée de faire le suivi de clubs amateurs. Depuis quarante ans, le rôle de Planète Sciences est de développer les activités spatiales pour les jeunes en France. Ainsi, depuis 1969 sont organisées des campagnes de lancements pour fusées. Planète Sciences est l'organisateur de ces campagnes et le CNES (Centre National d'Etudes Spatiales) est présent pour mettre en œuvre les propulseurs (pour des raisons de sécurité).

I.2 - Historique du projet Vertigo

La naissance du projet Vertigo remonte à septembre 2001. L'idée de base est de mettre en œuvre une mesure de vitesse par ultrasons. A partir de là, nous avons étudié la chaîne de mesure nécessaire à une telle expérience, et nous avons conclu qu'elle pourrait être embarquée dans une fusée expérimentale propulsée par un Isard. En effet, en faisant attention sur les matériaux utilisés, cela permettait d'avoir déjà une bonne plage de variation pour la vitesse.

Le projet s'est déroulé sur deux ans :

- pendant la première année, nous avons définit les besoins en terme de capteurs et d'électronique, et nous avons commencé les cartes les plus importantes, ainsi que certaines pièces maîtresses de la mécanique.
- Pendant la seconde année, nous avons terminé la réalisation physique de chaque élément de la fusée, puis assemblé les pièces et les cartes dans la structure, et enfin nous avons programmé l'électronique pour le vol.

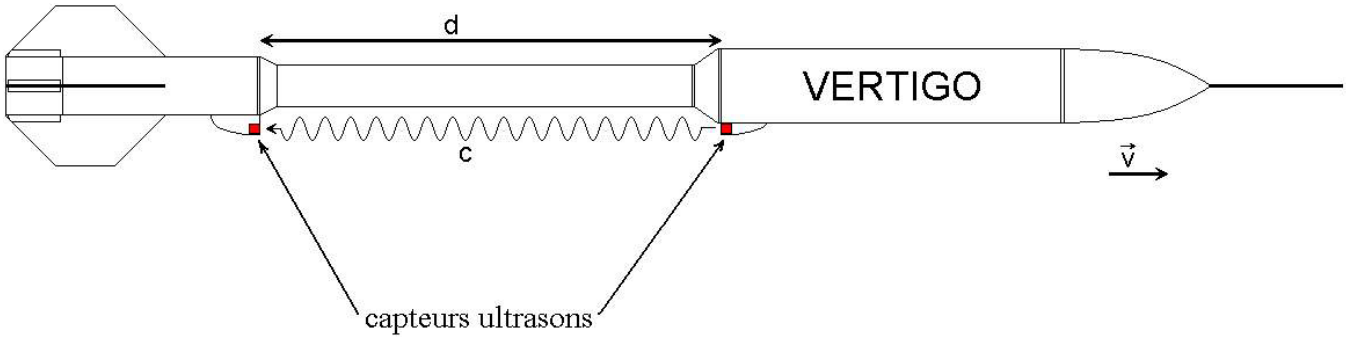
La fusée a été lancée à la campagne nationale de Sissonne, en juillet-août 2003. Elle a volé le samedi 2 août, vers 18 heures, et a effectué un vol nominal.

II - Vertigo en détails

Dans cette partie, nous détaillerons les détails techniques de la fusée, en décrivant les différentes mesures embarquées, ainsi que le système de récupération.

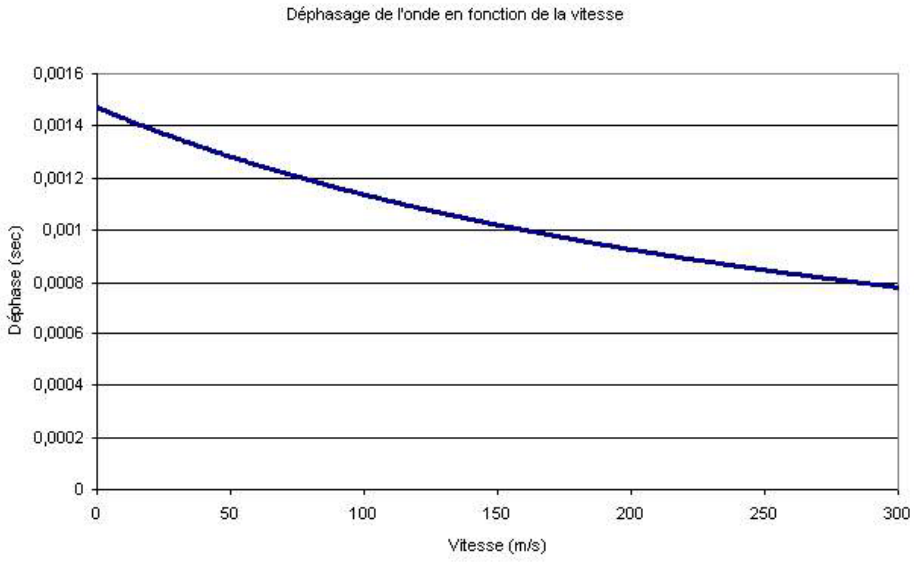
II.1 - La mesure de vitesse par ultrasons

L'idée physique de la mesure est simple : il s'agit de mesurer le temps nécessaire à une onde ultrasonore pour se propager le long de la fusée. A cet effet, la fusée est équipée de deux petits capteurs – un émetteur et un récepteur à ultrasons – placés sur le fuselage extérieur, et séparés d'une distance fixe d'environ 50 cm. 80 fois par secondes, l'émetteur envoie vers le bas de la fusée une salve d'ultrasons, et la fusée chronomètre le temps mis par ces impulsions pour arriver au bas de la fusée, au niveau du récepteur.



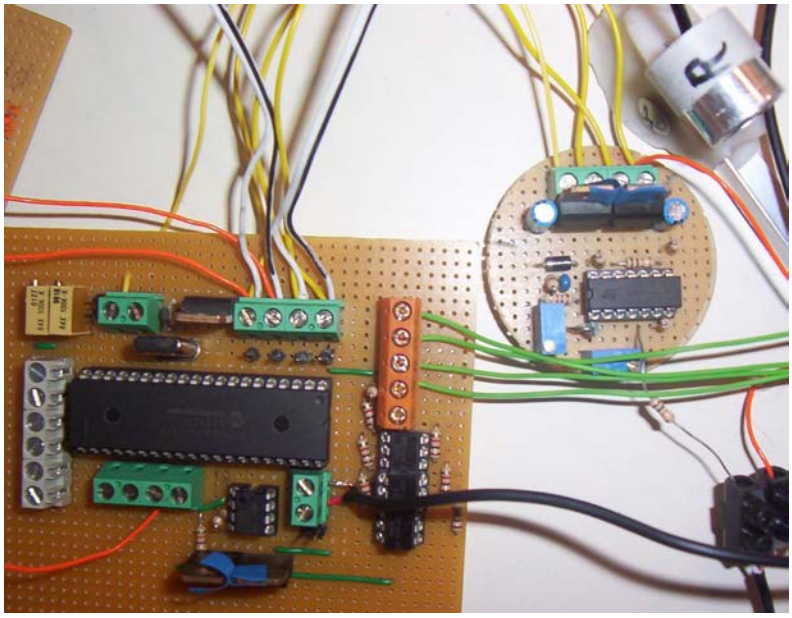
Lorsque la fusée vole vite, son vent relatif porte l'onde plus rapidement que lorsqu'elle est au repos. En notant d la distance séparant l'émetteur du récepteur, c la célerité du son, et v la vitesse de la fusée, on trouve qu'en première approximation, le temps de propagation est :

$$t=\frac{d}{c+v}$$

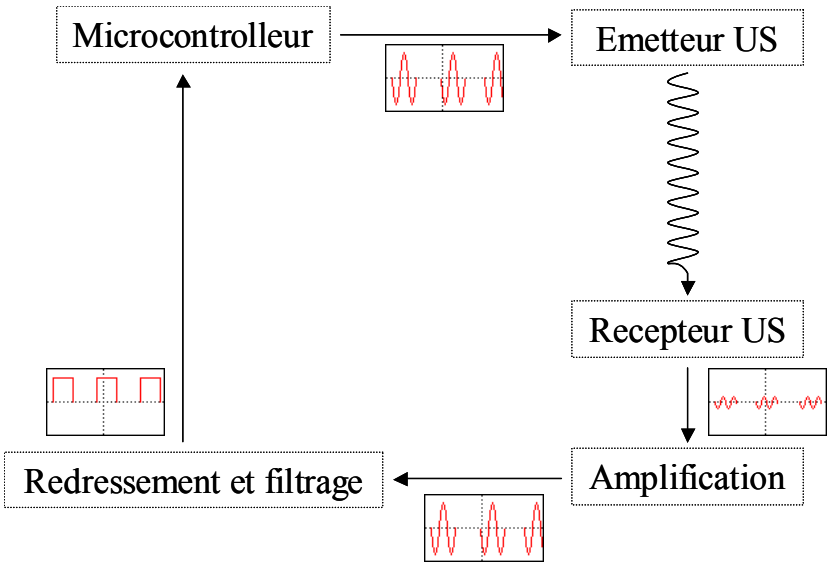


Cela revient à dire que la vitesse du flux d'air autour de la fusée s'ajoute à la célérité du son. Nous avons rajouté l'hypothèse que l'écoulement de l'air est homogène autour de la fusée, sans turbulence.

Concrètement, pour réaliser l'expérience, nous avons utilisé des ultrasons à 40kHz, afin d'éviter les perturbations par les sons extérieurs, et nous avons donné aux capteurs le plus de puissance possible, en alimentant l'émetteur avec 24V d'amplitude.



La formation des trames et le calcul de la vitesse étaient réalisés par un microcontrôleur de type PIC 16F877 cadencé à 20 MHz. Le PIC regardait également la longueur de la salve reçue et la comparait à celle émise initialement, afin de vérifier que le signale reçu n'était pas un parasite. Les données envoyées à l'émetteur contenaient donc un octet donnant la vitesse mesurée, et un octet indiquant la fiabilité de la mesure.



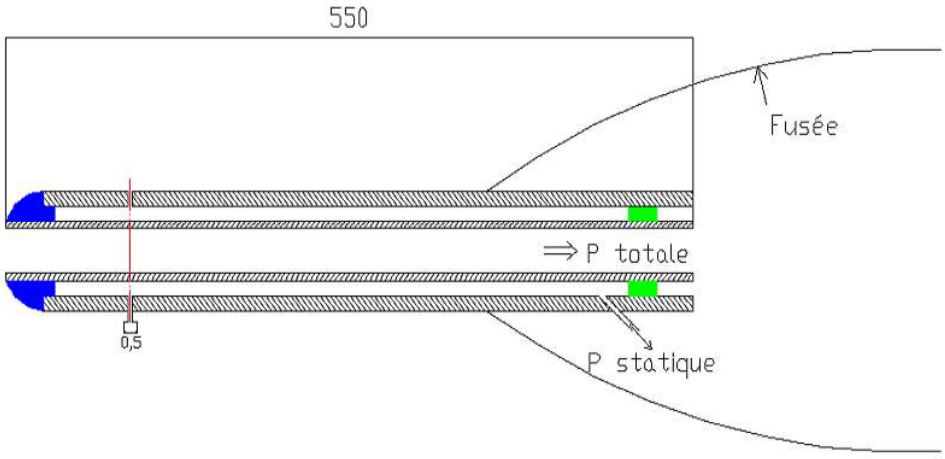
La programmation a du être assez fine (directement en assembleur, en utilisant des interruptions, cf le code source en annexe) pour permettre une bonne précision de la fréquence 40kHz, et du chronomètre. Ce microcontrôleur était aussi chargé de réaliser les mesures analogiques des autres expériences, et d'élaborer les trames pour la télémesure envoyées à l'émetteur.

II.2 - La mesure de vitesse par tube de Pitot

L'objectif de la fusée étant de valider un système de mesure original (les ultrasons !), il est nécessaire d'avoir une autre mesure qui est couramment utilisée pour sa fiabilité. Justement, le tube de Pitot équipe régulièrement les avions, et a souvent été embarqué sur des fusées expérimentales du fait de son bon rapport fiabilité / complexité.

Il s'agit d'un tube placé en haut de la fusée, sur lequel sont effectuées deux mesures de pression :

- la pression dynamique, qui est prélevée au bout du tube, et qui augmente avec la vitesse.
- la pression dynamique, qui est prélevée sur le côté du tube, et qui diminue lorsque l'altitude de la fusée augmente.



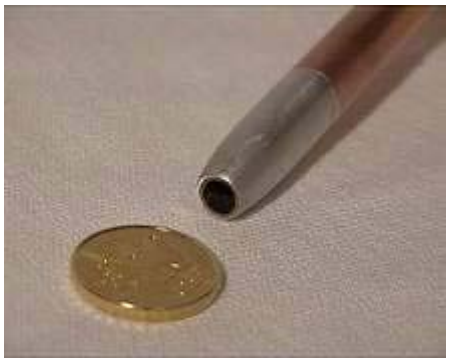
Notons ρ la masse volumique de l'air, V la vitesse de la fusée, et P_{stat} et P_{dyn} respectivement la pression statique et dynamique. Alors d'après le théorème de Bernoulli :

$$V = \sqrt{\frac{2 \times (P_{dyn} - P_{stat})}{\rho}}$$

En théorie, ρ dépend de l'altitude de la fusée, donc de P_{stat}, mais compte tenu de sa faible variation, nous la considérons comme constant. Ainsi, la vitesse de la fusée est proportionnelle à la racine carrée de la différence entre la pression dynamique et la pression statique.

Pour mesurer ces deux pressions, nous avons choisi les capteurs MPX5100, car ils ont l'avantage d'être compensés en température, et d'être amplifiés en interne, dont la sortie est directement conditionnée entre 0V et 5V.

Voici les prises de pression statique (à gauche) et dynamique (à droite) :



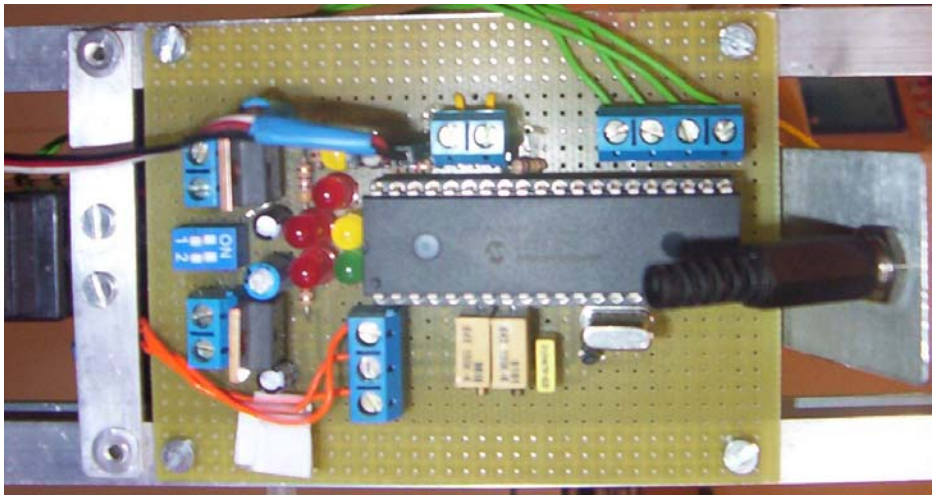
Ces deux pressions sont mesurées par un capteur à deux entrées :

Les deux tensions délivrées par les capteurs sont filtrées (par un passe bas dont la fréquence de coupure est la moitié de la fréquence d'échantillonnage, pour satisfaire le théorème de Shannon)., puis envoyées au PIC 16F877, qui effectue les conversions analogiques numériques, et génère les trames pour l'émetteur.

II.3 - La détection de culmination

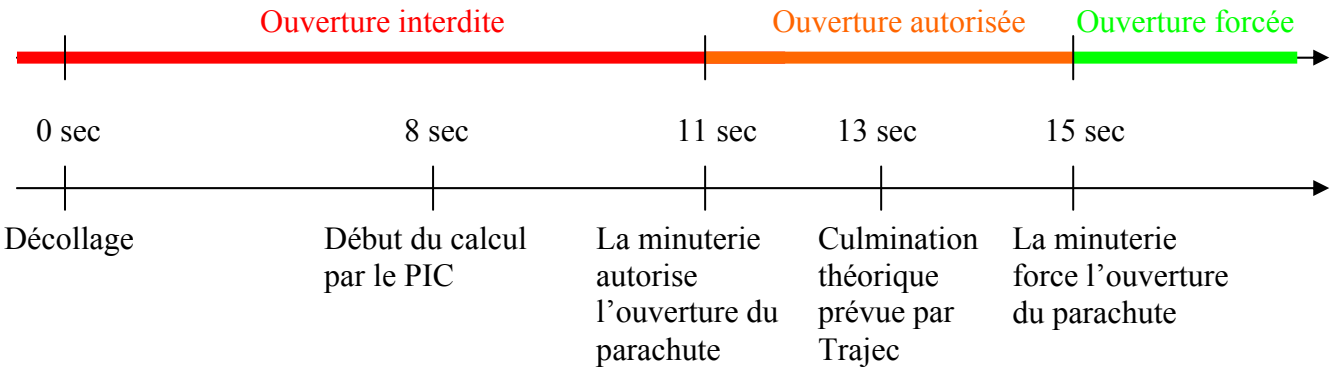
L'objectif de cette expérience est de concevoir un système qui ouvre le parachute exactement à la culmination de la fusée, c'est à dire à l'apogée de sa trajectoire. Habituellement, les fusées ouvrent leur parachute à un instant qui a été pré-programmé grâce aux calculs du logiciel Trajec. Mais ces résultats sont souvent très approximatifs, et des erreurs importantes peuvent occasionner une détérioration du parachute.

Pour faire un tel système, nous avons choisi de mesurer l'altitude de la fusée, et de dériver celle-ci par rapport au temps. Lorsque que la dérivée s'annule, c'est que la fusée est à son apogée. Concrètement, lorsque la fusée monte, l'altitude augmente, et lorsque la fusée culmine, l'altitude n'augmente plus.



L'altitude était donc mesurée par un capteur de pression (MPX5100) placé au niveau de la prise de pression statique du tube de Pitot. La sortie du capteur, une fois filtrée, était directement envoyée à la minuterie qui n'était rien d'autre qu'un PIC 16F877. Celui-ci mesurait la pression tout les 1/10ème de seconde, et la culmination était détectée lorsque la pression ne variait plus (ou très peu).

Mais ce type de système doit être sécurisé, c'est à dire que si le capteur ne détecte pas l'apogée, la minuterie doit quand même ouvrir le parachute. Inversement, si l'apogée est détectée accidentellement trop tôt, la minuterie doit ignorer le signal du capteur. Ainsi la minuterie a été programmée comme suit :



II.4 - Les phases de vol

Pour vérifier que la fusée a bien fonctionnée, il est intéressant de transmettre au sol des données sur les phases de vol. Ainsi, nous avons choisi de retransmettre :

- des informations sur l'état du système de récupération
- les états du programme de détection de culmination

Concernant le système de récupération, chaque information binaire était codée sur un bit, de la façon suivante :

1. porte ouverte/fermée.
2. parachute déployé : un jack était placé sur la sangle du parachute, et lorsque celui-ci s'ouvrait, le choc sur la sangle arrachait la prise jack.

Les informations du programme de détection de culmination étaient prélevées au niveau de la minuterie, sur les sorties du PIC :

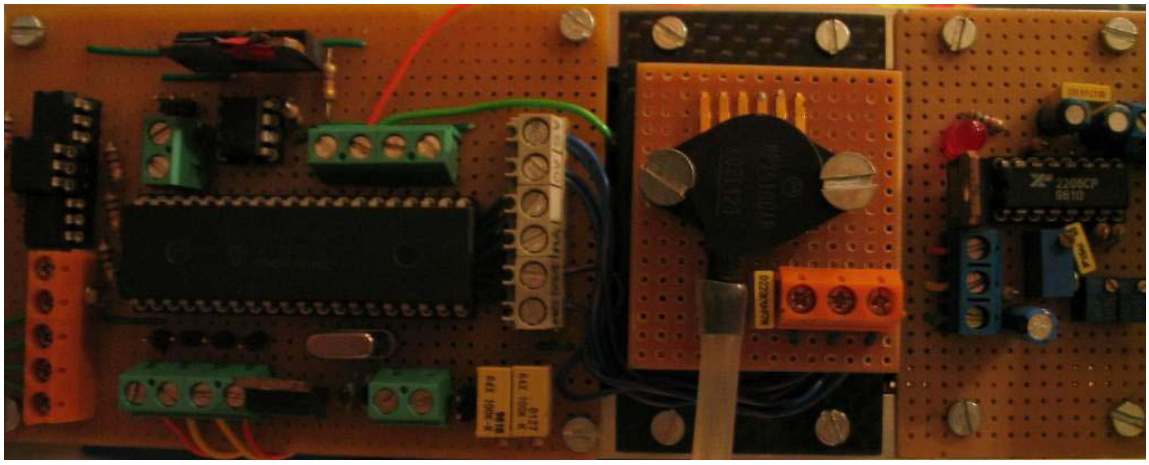
3. état du jack de décollage : indique si la fusée a décollé.
4. fenêtrage temporel : indique quand la minuterie autorise la détection de culmination à donner l'ordre d'ouvrir le parachute.
5. détection de culmination : indique le moment où la fusée a détecté son apogée. Ce bit peut passer à 1 en dehors du fenêtrage temporel.
6. ordre d'ouverture du parachute : indique que la minuterie a déclenché l'ouverture du système de récupération.

Pour récapituler, voici la composition de l'octet « phases de vol » :

Start	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	Stop
0	porte	parachute	jack	fenêtrage	détection	ordre	/	/	1

II.5 - La télémesure

La télémesure sert à transmettre les données de nos expériences au sol. Pour cela, nous avons utilisé un émetteur, le Kiwi Millenium prêté par Planète Sciences. Nous n'avions qu'à nous occuper de la partie émission (codage des trames et modulation), car toute la partie réception (démodulation et décodage des trames) est assurée par Planète Sciences. Vu le nombre de données à transmettre, nous avons choisi un standard numérique, le protocole SNR.



Le travail numérique est effectué par notre PIC 16F877 :

- La tension de chaque capteur est convertie numériquement en un octet, 80 fois par secondes.
- L'octet est transmit sous forme série, à 4800 Bauds.
- Chaque octet est précédé d'un bit de start à 0, et d'un bit de stop à 1.
- Une fois les 5 octets de mesures envoyées, nous transmettons un octet de synchronisation égal à h'FF'.
- Pour chaque capteur, nous avions donc un débit théorique de 80 valeurs par secondes.

Voici un récapitulatif de la trame complète générée par le PIC :

	Octet 1		Octet 2		Octet 3		Octet 4		Octet 5		Synchro	
0	xxxxxxxx	10	xxxxxxxx	10	xxxxxxxx	10	xxxxxxxx	10	xxxxxxxx	10	11111111	1

Le travail analogique est effectué par un XR2206. Il s'agit d'un convertisseur tension-fréquence , aussi appelé VCO, que nous utilisons en mode FSK c'est à dire que lorsque nous voulons envoyer un signal à 1, nous générons une fréquence f_1 = 15 kHz, et pour envoyer un signal à 0, nous générons une fréquence f_2 = 9 kHz.

Le signal obtenu en sortie du XR2206 est donné tel quel au Kiwi Millenium, configuré en mode Toucan (modulation externe). L'émetteur est muni d'une antenne de 52 cm.

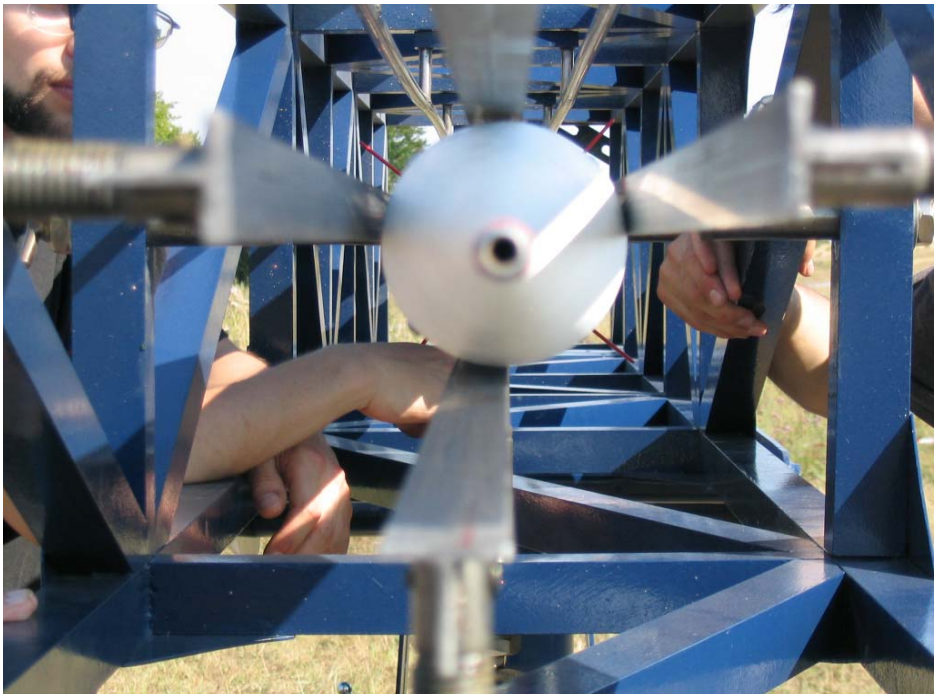
Enfin, nous avons particulièrement soigné les câblages des signaux analogiques (capteurs, modulation du VCO, …) et des signaux hautes fréquences (signaux ultrasons à 40 kHz, signal de l'émetteur, ..), en utilisant des câbles blindés. Le Kiwi était aussi séparé du reste de l'électronique par un plan de masse, afin de limiter son rayonnement sur les cartes sensibles. La structure du compartiment électronique, qui était en carbone, était reliée à la masse électrique, afin de protéger la fusée du rayonnement de l'antenne.

III - Conclusion du projet

III.1 - Le lancement

La fusée a pu être terminée à temps en travaillant d'arrache pied jusqu'en juillet. Elle a été lancée lors de la campagne de Sissonne, le samedi 2 août, en fin de journée. Son lancement été initialement prévu en début de matinée, mais lors de sa première mise en rampe, nous nous sommes aperçus que la télémesure ne fonctionnait pas. En effet, l'émetteur était alimenté en 18V, alors qu'il accepte qu'une tension de 15V. Pendant les qualifications, cette erreur n'avait pas posé de problèmes car nous n'avions utilisé que des piles usagées !

Finalement, la fusée a pu décoller vers 18 heures. Elle est partie rapidement, en larguant sans problèmes ses demi coquilles en sortie de rampe.



Il semble que son parachute se soit ouvert environ 13 secondes après le décollage, presque à culmination. Mais à près de 1200 mètres, nous avions bien du mal à la voir. Elle est retombée doucement, en se rapprochant heureusement de la zone de lancement. Le lendemain, nous l'avons retrouvé à 50 mètres de sa rampe ! Hormis un aileron plié à l'atterrissage, elle nous était revenue intacte.

III.2 - Les résultats des expériences

Bien que le camion de télémesure ait reçu le signal émis par la fusée pendant tout le vol (et même après l'atterrissage), le signal modulé a été fortement perturbé. Par conséquent, les trames sont sans cesse désynchronisées, et il nous est malheureusement impossible de décoder les données de nos capteurs. Nous ne savons donc pas si la mesure de vitesse par ultrasons, ni même si la détection de culmination ont réellement fonctionné.

De ce que nous avons vu, le parachute est sorti environ 13 secondes après le décollage, ce qui signifie que la détection semble avoir fonctionné. Mais nous ne pouvons pas en être sûr, car seules les phases de vol transmises par la télémesure auraient pu confirmer le bon fonctionnement du dispositif.

Pour la mesure de vitesse par ultrasons, nous n'avions pas eu le temps de le tester (nous avions envisagé de faire un essai en voiture), mais il semblait quand même peu sensible aux vibrations ou aux chocs, ce qui nous laisse espérer que l'expérience a fonctionné …

III.3 - Remerciements

Nous remercions toutes les personnes qui ont participé de près comme de loin au succès du projet Vertigo.

En particulier, un grand merci à :

- Les membres d'Eurêka + qui se sont mobilisés pendant l'année ou sur la campagne.
- La commune de Marly le Roi, qui offre à l'Association des locaux et des subventions.
- Patrick Rommeluere, notre suiveur, plus connu sous le nom de Zen.
- Nicolas Couronneau, et Patrick Lemaire, pour leur aide autour du PIC.
- Emmanuel Jolly, sans qui nous n'aurions pas pu débuguer la télémesure.
- Pierre Debaets, grâce à qui nous avons une magnifique photo de Vertigo au décollage
- L'association Planète Sciences, pour l'organisation de la campagne 2003.
- Le Cnes, qui permet chaque année aux jeunes de réaliser leurs rêves.
- Microchip, pour ses échantillons gratuits !

ANNEXES

Annexe 1 - Minuterie

Voici le code source de la minuterie à détection de culmination, pour le PIC 16F877 à 20MHz :

```
;*******************************************************************************
;                                                                              *
;                            V E R T I G O                                     *
;                              Eurêka +                                        *
;                                                                              *
;         Minuterie pour servomoteur à détection de culmination                *
;                                                                              *
;*******************************************************************************
;                                                                              *
;    NOM: minut877.asm                                                         *
;    Date: 16 juillet 2003                                                     *
;    Version: 3.0                                                              *
;    Circuit: minuterie 4MHz                                                   *
;    Auteur: Marc Buchdahl                                                     *
;                                                                              *
;*******************************************************************************
;                                                                              *
;    Fichier requis: P16F877.inc                                               *
;                                                                              *
;*******************************************************************************
;                                                                              *
;    Notes:                                                                    *
;                                                                              *
;    A partir du décollage, on attend sans rien faire pendant 10sec, puis      *
;    pendant 3sec, on active la mémorisation de la culmination. Ainsi,         *
;    si à la culmination est mémorisée pendant ces 3sec, on ouvrira dès        *
;    le début du fenêtrage.                                                    *
;    Après, pendant 4sec, on active le fenêtrage pour la détection de          *
;    culmination, et à la fin du fenêtrage, on force l'ouverture.              *
;                                                                              *
;*******************************************************************************
;                                                                              *
;    Ressources:                                                               *
;                                                                              *
;      RB7 : sortie PWM servo                                                  *
;      RB4 : sortie détection de la culmination                                *
;      RB3 : sortie ordre ouverture                                            *
;      RB2 : sortie fenêtrage                                                  *
;      RB1 : sortie détection jack décollage                                   *
;      RB0 : entrée jack                                                       *
;      RA0 : entrée analogique capteur de pression                             *
;      RA2 : entrée analogique Vréf-                                           *
;      RA3 : entrée analogique Vréf+                                           *
;      RD1 : sortie phases de vol (fenetrage)                                  *
;      RD2 : sortie phases de vol (culmination)                                *
;      RD3 : sortie phases de vol (jack)                                       *
;                                                                              *
;*******************************************************************************


       LIST      p=16F877            ; Définition de processeur
       #include <p16F877.inc>        ; fichier include

       __CONFIG    _CP_OFF & _DEBUG_OFF & _WRT_ENABLE_OFF & _CPD_OFF & _LVP_OFF & _BODEN_OFF & _PWRTE_ON
& _WDT_OFF & _HS_OSC

;_CP_OFF              Pas de protection
;_DEBUG_OFF           RB6 et RB7 en utilisation normale
;_WRT_ENABLE_OFF      Le programme ne peut pas écrire dans la flash
;_CPD_OFF          Mémoire EEprom déprotégée
;_LVP_OFF             RB3 en utilisation normale
; _BODEN_OFF   Reset tension hors service
;_PWRTE_ON            Démarrage temporisé
;_WDT_OFF             Watchdog hors service
;_HS_OSC              Oscillateur haute vitesse (4Mhz<F<20Mhz)

;*******************************************************************************
;                                ASSIGNATIONS SYSTEME                          *
;*******************************************************************************

; REGISTRE OPTION_REG (configuration)
```

```
; ----------------------------------
OPTIONVAL       EQU     B'10000000'
                        ; RBPU      b7 : 1= Résistance rappel +5V hors service
                        ; INTEDG    b6 : 1= Interrupt sur flanc montant de RB0
                        ;                0= Interrupt sur flanc descend. de RB0
                        ; TOCS       b5 : 1= source clock = transition sur RA4
                        ;                 0= horloge interne
                        ; TOSE       b4 : 1= Sélection flanc montant RA4(si B5=1)
                        ;                 0= Sélection flanc descendant RA4
                        ; PSA        b3 : 1= Assignation prédiviseur sur Watchdog
                        ;                 0= Assignation prédiviseur sur Tmr0
                        ; PS2/PS0    b2/b0 valeur du prédiviseur
                          ;           000 =  1/1 (watchdog) ou 1/2 (tmr0)
                        ;           001 =  1/2              1/4
                        ;           010 =  1/4          1/8
                        ;           011 =  1/8          1/16
                        ;           100 =  1/16                 1/32
                        ;           101 =  1/32                 1/64
                        ;           110 =  1/64                 1/128
                        ;           111 =  1/128        1/256


; REGISTRE INTCON (contrôle interruptions standard)
; -------------------------------------------------
INTCONVAL       EQU     B'00000000'
                        ; GIE       b7 : masque autorisation générale interrupt
                          ;                ne pas mettre ce bit à 1 ici
                          ;                sera mis en temps utile
                        ; PEIE      b6 : masque autorisation générale périphériques
                        ; T0IE      b5 : masque interruption tmr0
                        ; INTE      b4 : masque interuption RB0/Int
                        ; RBIE      b3 : masque interruption RB4/RB7
                        ; T0IF      b2 : flag tmr0
                        ; INTF      b1 : flag RB0/Int
                        ; RBIF      b0 : flag interruption RB4/RB7

; REGISTRE PIE1 (contrôle interruptions périphériques)
; ----------------------------------------------------
PIE1VAL         EQU     B'00000000'
                        ; PSPIE     b7 : Toujours 0 sur PIC 16F786
                        ; ADIE      b6 : masque interrupt convertisseur A/D
                        ; RCIE      b5 : masque interrupt réception USART
                        ; TXIE      b4 : masque interrupt transmission USART
                        ; SSPIE     b3 : masque interrupt port série synchrone
                        ; CCP1IE    b2 : masque interrupt CCP1
                        ; TMR2IE    b1 : masque interrupt TMR2 = PR2
                        ; TMR1IE    b0 : masque interrupt débordement tmr1

; REGISTRE PIE2 (contrôle interruptions particulières)
; ----------------------------------------------------
PIE2VAL         EQU     B'00000000'
                        ; UNUSED    b7 : inutilisé, laisser à 0
                        ; RESERVED  b6 : réservé, laisser à 0
                        ; UNUSED    b5 : inutilisé, laisser à 0
                        ; EEIE      b4 : masque interrupt écriture EEPROM
                        ; BCLIE     b3 : masque interrupt collision bus
                        ; UNUSED    b2 : inutilisé, laisser à 0
                        ; UNUSED    b1 : inutilisé, laisser à 0
                        ; CCP2IE    b0 : masque interrupt CCP2

; REGISTRE ADCON0 et ADCON1 (ANALOGIQUE/DIGITAL)
; ----------------------------------------------
ADCON0VAL       EQU     B'10000001'     ; AD sur ON, sélectionner RA0
ADCON1VAL       EQU     B'00001111'     ; justification à gauche, RA0 anlogique, RA2 Vréf-, RA3 Vréf+

; DIRECTION DES PORTS I/O
; -----------------------

DIRPORTA        EQU     B'00111111'     ; Direction PORTA (1=entrée)
DIRPORTB        EQU     B'01100001'     ; Direction PORTB
DIRPORTC        EQU     B'11111111'     ; Direction PORTC
DIRPORTD        EQU     B'11110001'     ; Direction PORTD
DIRPORTE        EQU     B'00000111'     ; Direction PORTE

;*****************************************************************************
;                                   DEFINE                                   *
;*****************************************************************************

#DEFINE JACK    PORTB,0         ; entrée jack de décollage
#DEFINE PWM     PORTB,7         ; sortie servomoteur
#DEFINE LEDjack         PORTB,1         ; sortie détection de décollage
#DEFINE LEDfen PORTB,2          ; sortie fenêtrage
#DEFINE LEDordre        PORTB,3         ; sortie ordre d'ouverture
#DEFINE LEDdetect       PORTB,4         ; sortie détection de la culmination
#DEFINE DETECTED        FLAGS,0         ; flag indiquant si la culmination a été détectée ou non
#DEFINE FENET   PORTD,1         ; sortie phases de vol
#DEFINE CULMI   PORTD,2         ; sortie phases de vol
#DEFINE DECOL   PORTD,3         ; sortie phases de vol

;*****************************************************************************
;                                   MACRO                                    *
;*****************************************************************************
```

```
BANK0   macro                           ; passer en banque0
                bcf     STATUS,RP0
                bcf     STATUS,RP1
        endm

BANK1   macro                           ; passer en banque1
                bsf     STATUS,RP0
                bcf     STATUS,RP1
        endm

BANK2   macro                           ; passer en banque2
                bcf     STATUS,RP0
                bsf     STATUS,RP1
        endm

BANK3   macro                           ; passer en banque3
                bsf     STATUS,RP0
                bsf     STATUS,RP1
        endm

REEPROM macro                           ; lire eeprom(adresse & résultat en w)
                clrwdt                  ; reset watchdog
                bcf     STATUS,RP0      ; passer en banque2
                bsf     STATUS,RP1
                movwf   EEADR           ; pointer sur adresse eeprom
                bsf     STATUS,RP0      ; passer en banque3
                bcf     EECON1,EEPGD    ; pointer sur eeprom
                bsf     EECON1,RD       ; ordre de lecture
                bcf     STATUS,RP0      ; passer en banque2
                movf    EEDATA,w        ; charger valeur lue
                bcf     STATUS,RP1      ; passer en banque0
        endm


                        ; Sauts inter-pages
                        ; -----------------

GOTOX   macro  ADRESSE                  ; saut inter-page
        local BIT4 = (ADRESSE & 0x1000)      ; voir bit 12
        local BIT3 = (ADRESSE & 0x0800) ; voir bit 11
        local  ICI                      ; adresse courante
ICI
        local PICI = (ICI+2 & 0x1800) ; page du saut
        IF BIT3                         ; si page 1 ou 3
        bsf    PCLATH , 3               ; b3 de PCLATH = 1
        ELSE                            ; sinon
        bcf    PCLATH , 3               ; b3 de PCLATH = 0
        ENDIF
        IF BIT4                         ; si page 2 ou 3
        bsf    PCLATH , 4               ; b4 de PCLATH = 1
        ELSE                            ; sinon
        bcf    PCLATH , 4               ; b4 de PCLATH = 0
        ENDIF
        goto (ADRESSE & 0x7FF | PICI) ; adresse simulée
        endm


PCLAX   macro  ADRESSE                  ; positionne PCLATH pour
                                        ; les sauts sans le saut
        local BIT4 = (ADRESSE & 0x1000)      ; voir bit 12
        local BIT3 = (ADRESSE & 0x0800) ; voir bit 11

        IF BIT3                         ; si page 1 ou 3
        bsf    PCLATH , 3               ; b3 de PCLATH = 1
        ELSE                            ; sinon
        bcf    PCLATH , 3               ; b3 de PCLATH = 0
        ENDIF
        IF BIT4                         ; si page 2 ou 3
        bsf    PCLATH , 4               ; b4 de PCLATH = 1
        ELSE                            ; sinon
        bcf    PCLATH , 4               ; b4 de PCLATH = 0
        ENDIF
        endm

GOTSX   macro  ADRESSE                  ; saut inter-page sans
                                        ; sélection de PCLATH
        local  ICI                      ; adresse courante
        local PICI = (ICI & 0x1800)   ; page du saut
ICI
        goto (ADRESSE & 0x7FF | PICI) ; adresse simulée
        endm

                        ; Sous-routines inter-pages
                        ; -------------------------

CALLX   macro  ADRESSE                  ; call inter-page
        local BIT4 = (ADRESSE & 0x1000)      ; voir bit 12
        local BIT3 = (ADRESSE & 0x0800) ; voir bit 11
        local  ICI                      ; adresse courante
ICI
        local PICI = (ICI+2 & 0x1800) ; page du saut
```

```
        IF BIT3                         ; si page 1 ou 3
        bsf     PCLATH , 3              ; b3 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 3              ; b3 de PCLATH = 0
        ENDIF
        IF BIT4                         ; si page 2 ou 3
        bsf     PCLATH , 4              ; b4 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 4              ; b4 de PCLATH = 0
        ENDIF
        call (ADRESSE & 0x7FF | PICI) ; adresse simulée
        local BIT4 = ((ICI+5) & 0x1000)      ; voir bit 12
        local BIT3 = ((ICI+5) & 0x0800)      ; voir bit 11
        IF BIT3                         ; si page 1 ou 3
        bsf     PCLATH , 3              ; b3 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 3              ; b3 de PCLATH = 0
        ENDIF
        IF BIT4                         ; si page 2 ou 3
        bsf     PCLATH , 4              ; b4 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 4              ; b4 de PCLATH = 0
        endm


CALLSX  macro   ADRESSE                 ; sous-routine inter-page sans
                                        ; sélection de PCLATH
        local   ICI                     ; adresse courante
        local PICI = (ICI & 0x1800)   ; page du saut
ICI
        call (ADRESSE & 0x7FF | PICI) ; adresse simulée
        endm

                        ; Sous-routines eeprom
                        ; --------------------

WEEPROM macro   addwrite        ; la donnée se trouve dans W
        LOCAL   loop
        bcf     STATUS,RP0      ; passer en banque2
        bsf     STATUS,RP1
        movwf   EEDATA          ; placer data dans registre
        movlw   addwrite        ; charger adresse d'écriture
        movwf   EEADR           ; placer dans registre
        bsf     STATUS,RP0      ; passer en banque3
        bcf     EECON1 , EEPGD ; pointer sur mémoire data
        bsf     EECON1 , WREN  ; autoriser accès écriture
        bcf     INTCON , GIE   ; interdire interruptions
        movlw   0x55            ; charger 0x55
        movwf   EECON2          ; envoyer commande
        movlw   0xAA            ; charger 0xAA
        movwf   EECON2          ; envoyer commande
        bsf     EECON1 , WR    ; lancer cycle d'écriture
        bsf     INTCON , GIE   ; réautoriser interruptions
loop
        btfsc   EECON1 , WR    ; tester si écriture terminée
        goto    loop            ; non, attendre
        bcf     EECON1 , WREN  ; verrouiller prochaine écriture
        bcf     STATUS , RP0   ; passer en banque0
        bcf     STATUS , RP1
        endm

;*****************************************************************************
;                           VARIABLES BANQUE 0                                *
;*****************************************************************************

; Zone de 80 bytes
; ----------------

        CBLOCK 0x20             ; Début de la zone (0x20 à 0x6F)
        FLAGS : 1               ; variable contenant des flags (FLAGS.0 : DETECTED)
        NEWPRESS : 1            ; valeur de la pression actuelle
        LASTPRESS : 1           ; valeur de l'ancienne pression
        MS : 1                  ; compte le nombre de millisecondes : entre 0 et 20
        CPT1 : 1                ; compte le nombre d'impulsions de 20msec : entre 0 et 5
        CPT2 : 1                ; compte le nombre de dixième de secondes : entre 0 et tempo*10
        PDel0 : 1               ; compteur pour la routine pause1ms
      ENDC                      ; Fin de la zone

var1    EQU H'006E'             ; adresse imposée

;*****************************************************************************
;                         VARIABLES ZONE COMMUNE                              *
;*****************************************************************************

; Zone de 16 bytes
; ----------------

        CBLOCK 0x70             ; Début de la zone (0x70 à 0x7F)
        w_temp : 1              ; Sauvegarde registre W
        status_temp : 1                 ; sauvegarde registre STATUS
        FSR_temp : 1            ; sauvegarde FSR (si indirect en interrupt)
        PCLATH_temp : 1                 ; sauvegarde PCLATH (si prog>2K)
```

```
        ENDC

;*****************************************************************************
;                       VARIABLES BANQUE 1                                    *
;*****************************************************************************

; Zone de 80 bytes
; ----------------

        CBLOCK  0xA0            ; Début de la zone (0xA0 à 0xEF)

        ENDC                    ; Fin de la zone

;*****************************************************************************
;                       VARIABLES BANQUE 2                                    *
;*****************************************************************************

; Zone de 96 bytes
; ----------------

        CBLOCK  0x110           ; Début de la zone (0x110 à 0x16F)

        ENDC                    ; Fin de la zone

;*****************************************************************************
;                       VARIABLES BANQUE 3                                    *
;*****************************************************************************

; Zone de 96 bytes
; ----------------

        CBLOCK  0x190           ; Début de la zone (0x190 à 0x1EF)

        ENDC                    ; Fin de la zone

;*****************************************************************************
;                      DEMARRAGE SUR RESET                                    *
;*****************************************************************************

        org 0x000               ; Adresse de départ après reset
        clrf    PCLATH            ; Effacer sélecteur de pages
        goto    init            ; Initialiser



; //////////////////////////////////////////////////////////////////////////

;                           P R O G R A M M E

; //////////////////////////////////////////////////////////////////////////

;*****************************************************************************
;                         INITIALISATIONS                                     *
;*****************************************************************************

init

                        ; initialisation PORTS (banque 0 et 1)
                        ; -----------------------------------
        BANK0                   ; sélectionner banque0
        clrf    PORTA           ; Sorties PORTA à 0
        clrf    PORTB           ; sorties PORTB à 0
        clrf    PORTC           ; sorties PORTC à 0
        clrf    PORTD           ; sorties PORTD à 0
        clrf    PORTE           ; sorties PORTE à 0
        movlw   ADCON0VAL       ; PORTA en mode digital/analogique
        movwf   ADCON0          ; écriture dans contrôle A/D
        bsf     STATUS,RP0      ; passer en banque1
        movlw   ADCON1VAL       ; PORTA en mode digital/analogique
        movwf   ADCON1          ; écriture dans contrôle A/D
        movlw   DIRPORTA        ; Direction PORTA
        movwf   TRISA           ; écriture dans registre direction
        movlw   DIRPORTB        ; Direction PORTB
        movwf   TRISB           ; écriture dans registre direction
        movlw   DIRPORTC        ; Direction PORTC
        movwf   TRISC           ; écriture dans registre direction
        movlw   DIRPORTD        ; Direction PORTD
        movwf   TRISD           ; écriture dans registre direction
        movlw   DIRPORTE        ; Direction PORTE
        movwf   TRISE           ; écriture dans registre direction

                        ; Registre d'options (banque 1)
                        ; -----------------------------
        movlw   OPTIONVAL       ; charger masque
        movwf   OPTION_REG      ; initialiser registre option

                        ; registres interruptions (banque 1)
                        ; ----------------------------------
        movlw   INTCONVAL       ; charger valeur registre interruption
```

```
        movwf   INTCON          ; initialiser interruptions
        movlw   PIE1VAL         ; Initialiser registre
        movwf   PIE1            ; interruptions périphériques 1
        movlw   PIE2VAL         ; initialiser registre
        movwf   PIE2            ; interruptions périphériques 2

                        ; Effacer RAM banque 0
                        ; ----------------------
        bcf     STATUS,RP0      ; sélectionner banque 0
        movlw   0x20            ; initialisation pointeur
        movwf   FSR             ; d'adressage indirect
init1
        clrf    INDF            ; effacer ram
        incf    FSR,f           ; pointer sur suivant
        btfss   FSR,7           ; tester si fin zone atteinte (>7F)
        goto    init1           ; non, boucler

                        ; autoriser interruptions (banque 0)
                        ; ---------------------------------
        clrf    PIR1            ; effacer flags 1
        clrf    PIR2            ; effacer flags 2
        bsf     INTCON,GIE      ; valider interruptions
        goto    start           ; programme principal


;*****************************************************************************
;                       PROGRAMME PRINCIPAL                                 *
;*****************************************************************************

start
        bsf     LEDjack         ; signalisation ...
        bcf     LEDordre
        bcf     LEDfen
        bcf     DECOL           ; envoit des phases de vol ...
        bcf     CULMI
        bcf     FENET
        btfsc   JACK            ; si JACK = 0, décollage (vol)
        goto    start           ; sinon boucler (attente)



; décollage
; attente pendant 10 sec :
; ------------------------

        movlw   20              ; initialisations ...
        movwf   MS
        movlw   5
        movwf   CPT1
        movlw   100             ; réglage de la temporisation à 10 SECONDES
        movwf   CPT2
        bcf     LEDjack         ; signalisation de détection du décollage
        bsf     DECOL

vol1                            ; attente (fait des impulsions PWM de 1ms) ...
        call    pause1ms        ; pause de 1ms
        bcf     PWM             ; met (ou maintient) PWM à 0
        decfsz  MS,1            ; décrémente MS et regarde si il vaut 0
        goto    vol1            ; si non, boucle
        bsf     PWM             ; si oui, met PWM à 1
        movlw   20              ; et recharge MS à 20
        movwf   MS
        decfsz  CPT1,1          ; et décrémente CPT1 et regarde s'il vaut 0
        goto    vol1            ; si non, boucle
        movlw   5               ; si oui, recharge CPT1 à 5
        movwf   CPT1
        btfsc   JACK            ; si JACK = 1 réinitialisation, sinon on continue
        goto    start
        decfsz  CPT2,1          ; décrémente CPT2 et regarde s'il vaut 0
        goto    vol1            ; si non, boucle
                                ; si oui, on entre dans le fenêtrage


; début de la prise en compte de la pression
; attente pendant 3 sec :
; -----------------------

        movlw   30              ; réglage de la temporisation à 3 SECONDES
        movwf   CPT2

vol2                            ; attente (fait des impulsions PWM de 1ms) où on peut prédétecter la
pression ...
        bsf     ADCON0,GO       ; lance (ou maintient, sans effet) la conversion
        call    pause1ms        ; pause de 1ms
        bcf     PWM             ; met (ou maintient) PWM à 0
        decfsz  MS,1            ; décrémente MS et regarde si il vaut 0
        goto    vol2            ; si non, boucle
        bsf     PWM             ; si oui, met PWM à 1
        movlw   20              ; et recharge MS à 20
        movwf   MS
        decfsz  CPT1,1          ; et décrémente CPT1 et regarde s'il vaut 0
        goto    vol2            ; si non, boucle
```

```
        movlw   5               ; si oui, recharge CPT1 à 5
        movwf   CPT1
        call    derivation      ; dérivation de l'altitude
        btfsc   JACK            ; si JACK = 1 réinitialisation, sinon on continue
        goto    start
        decfsz  CPT2,1          ; décrémente CPT2 et regarde s'il vaut 0
        goto    vol2


; fenêtrage de 4 sec :
; --------------------

        bsf     LEDfen          ; signalisation du fenêtrage
        bsf     FENET
        movlw   40              ; réglage de la temporisation à 4 SECONDES
        movwf   CPT2

fenetrage                       ; attente (fait des impulsions PWM de 1ms) où on active le détecteur de
culmination ...
        bsf     ADCON0,GO       ; lance (ou maintient, sans effet) la conversion
        call    pause1ms        ; pause de 1ms
        bcf     PWM             ; met (ou maintient) PWM à 0
        decfsz  MS,1            ; décrémente MS et regarde si il vaut 0
        goto    fenetrage       ; si non, boucle
        bsf     PWM             ; si oui, met PWM à 1
        movlw   20              ; et recharge MS à 20
        movwf   MS
        decfsz  CPT1,1          ; et décrémente CPT1 et regarde s'il vaut 0
        goto    fenetrage       ; si non, boucle
        movlw   5               ; si oui, recharge CPT1 à 5
        movwf   CPT1
        call    derivation      ; dérivation de l'altitude
        btfsc   DETECTED        ; si la culmination est détectée, ouverture
        goto    ouverture
        btfsc   JACK            ; si JACK = 1 réinitialisation, sinon on continue
        goto    start
        decfsz  CPT2,1          ; décrémente CPT2 et regarde s'il vaut 0
        goto    fenetrage       ; si non, boucle
                                ; si oui, on force l'ouverture

        bsf     LEDordre        ; signalisation de l'ouverture
        bcf     FENET


; ouverture du parachute :
; ------------------------

ouverture                       ; ouverture (fait des implusions PWM de 3 ms) ...
        bsf     LEDordre        ; signalisation de l'ouverture
        btfsc   JACK            ; si JACK = 1 réinitialisation, sinon on continue
        goto    start
        call    pause1ms
        call    pause1ms
        call    pause1ms
        bcf     PWM
        decfsz  MS,1
        goto    ouverture
        bsf     PWM
        movlw   10
        movwf   MS
        goto    ouverture


;*****************************************************************************
;                       PAUSE 1 MILLISECONDE                                 *
;*****************************************************************************

; Delai de 1000 cycles soit 1ms (including call+return)
;-------------------------------------------------------------
pause1ms
        movlw     .248      ; 1 set number of repetitions
        movwf     PDel0     ; 1 |
PLoop0
        nop                 ; rien
        decfsz    PDel0, 1  ; 1 + (1) is the time over?
        goto      PLoop0    ; 2 no, loop
PDelL1
        goto PDelL2         ; 2 cycles delay
PDelL2
        nop                 ; 1 cycle delay
        return              ; 2+2 Done
;-------------------------------------------------------------


;*****************************************************************************
;                     DERIVATION DE L'ALTITUDE                               *
;*****************************************************************************

; Test (effectué toutes les 0.1s) si la variation de pression est inférieure à un seuil
;-------------------------------------------------------------------------------------
derivation
```

```
        movf    NEWPRESS,W      ; met à jour LASTPRESS
        movwf   LASTPRESS

testCAN
        btfsc   ADCON0,GO       ; met à jour NEWPRESS
        goto    testCAN
        movf    ADRESH,W
        movwf   NEWPRESS

        movf    NEWPRESS,W      ; teste si |LASTPRESS - NEWPRESS| < EPSILON
        subwf   LASTPRESS,W
        sublw   3               ; on prend EPSILON à 3
        btfss   STATUS,C        ; skip if [EPSILON - |LASTPRESS - NEWPRESS|] > 0
        return                  ; si non, return
        bsf     DETECTED        ; si oui, on a détecté la culmination
        bsf     LEDdetect       ; signalisation de la détection
        bsf     CULMI
        return
;-------------------------------------------------------------------------------------


        END                     ; directive fin de programme
```

Annexe 2 - Expériences

Voici le code source de la carte qui gère les capteurs (phases de vol, pression et ultrasons) et qui génère les trames de la télémesure, pour le PIC 16F877 à 20MHz :

```
;*****************************************************************************
;                                                                            *
;                           V E R T I G O                                    *
;                              Eurêka+                                       *
;                                                                            *
;                       GESTION DES ULTRASONS                                *
;          envoi des trames et mesure de la vitesse et de la fiabilité       *
;                                                                            *
;*****************************************************************************
;                                                                            *
;    NOM : exp877.asm                                                        *
;    Fichier requis : P16F877.inc                                            *
;    Date : 15 juillet 2003                                                  *
;    Version : 2.0                                                           *
;    Circuit : expériences 16F877 à 20MHz                                    *
;    Auteur : Marc Buchdahl                                                  *
;                                                                            *
;*****************************************************************************
;                                                                            *
;    Notes :                                                                 *
;                                                                            *
;    Les trames de 40kHz sont émises sur EMETT, par salves de 3.3msec.       *
;    Après le début de la salve, on mesure l'instant du premier front        *
;    montant de RECEPT grâce à TMR0. Cela donne SPEED.                       *
;    On mesure ensuite le temps entre le premier front montant et le         *
;    premier front descendant. Cela donne la RELIAB.                         *
;    On envoie à tour de rôle la vitesse, puis la fiabilité, puis les        *
;    valeurs analogiques, puis les phases de vol, et enfin la synchro.       *
;                                                                            *
;*****************************************************************************
;                                                                            *
;    Ressources :                                                            *
;                                                                            *
;    RC2 : RECEP (entrée Récepteur)                                          *
;    RC3 : EMETT (sortie Emetteur)                                           *
;    RD1 : FREQ (sortie Fréquence 40kHz)                                     *
;    RD0 : SALVE (sortie Salve 3.2msec)                                      *
;    RC6 : TX (sortie Série)                                                 *
;    RA0 : STATIK (entrée analogique pour la pression statique)              *
;    RA1 : DYNAMIK (entrée analogique pour la pression dynamique)            *
;    RA2 : Vréf-                                                             *
;    RA3 : Vréf+                                                             *
;    RE2 : phase de vol (entrée)                                             *
;    RE1 : phase de vol (entrée)                                             *
;    RE0 : phase de vol (entrée)                                             *
;    RA5 : phase de vol (entrée)                                             *
;                                                                            *
;*****************************************************************************


       LIST      p=16F877           ; 16F877 quartz à 20 MHZ
       #include <p16F877.inc>       ; fichier include

       __CONFIG    _CP_OFF & _DEBUG_OFF & _WRT_ENABLE_OFF & _CPD_OFF & _LVP_OFF & _BODEN_OFF & _PWRTE_ON
& _WDT_OFF & _HS_OSC


;*****************************************************************************
;                                ASSIGNATIONS SYSTEME                        *
;*****************************************************************************

; REGISTRE OPTION_REG (configuration)
; -----------------------------------
OPTIONVAL      EQU     B'11000011'
                       ; b7 : disables pull-up on portB
                       ; b6 : enables interrupt sur périphériques
                       ; b3 : assignation du prédiviseur sur TMR0, avec prescaler TMR0 = 1/16

; REGISTRE INTCON (contrôle interruptions standard)
; -------------------------------------------------
INTCONVAL      EQU     B'01000000'
                       ; GIE       b7 : masque autorisation générale interrupt
                         ;                ne pas mettre ce bit à 1 ici
                         ;                sera mis en temps utile
                       ; PEIE      b6 : masque autorisation générale périphériques

; REGISTRE PIE1 (contrôle interruptions périphériques)
; ----------------------------------------------------
PIE1VAL        EQU     B'00000001'
```

```
                    ; TMR1IE    b0 : masque interrupt débordement tmr1

; REGISTRE PIE2 (contrôle interruptions particulières)
; ----------------------------------------------------
PIE2VAL         EQU     B'00000000'

; REGISTRES USART (configuration du module série)
; -----------------------------------------------
TXSTA_MASK      EQU     B'00100000'     ; masque pour TXSTA (pas de 9em bit, asynchrone)
RCSTA_MASK      EQU     B'10000000'     ; masque pour RCSTA (enable les RX et TX)
SPBRG_MASK      EQU     D'64'           ; vitesse(baud) = Fosc/(64(x+1))

; REGISTRE ADCON0 et ADCON1 (ANALOGIQUE/DIGITAL)
; ----------------------------------------------
ADCON0_MASK     EQU             B'10000001'     ; masque pour ADCON0
                                                ; diviseur Fosc/32
                                                ; sélection de RA0
                                                ; mise en service du CAN

ADCON1_MASK     EQU             B'00001100'     ; masque pour ADCON1
                                                ; justification à gauche
                                                ; RA0, RA1 et RA4 entrée analogiques
                                                ; RA2 et RA3 respectivement Vréf- et Vréf+
; REGISTRE T1CON (Timer1)
; -----------------------
T1CON_MASK      EQU     B'00000001'     ; masque pour T1CON
                                        ; presc 1/1
                                        ; mode "timer"
                                        ; TMR1 sur ON

; DIRECTION DES PORTS I/O
; -----------------------

DIRPORTA        EQU     B'00111111'     ; Direction PORTA (1=entrée et 0=sortie)
DIRPORTB        EQU     B'11111111'     ; Direction PORTB
DIRPORTC        EQU     B'10110111'     ; Direction PORTC
DIRPORTD        EQU     B'11111100'     ; Direction PORTD
DIRPORTE        EQU     B'00000111'     ; Direction PORTE


;*****************************************************************************
;                             ASSIGNATIONS PROGRAMME                          *
;*****************************************************************************

; exemple
; -------
MASQUE          EQU     H'00FF'


;*****************************************************************************
;                                   DEFINE                                    *
;*****************************************************************************

#DEFINE EMETT   PORTC,3         ; Emetteur (sortie)
#DEFINE RECEP   PORTC,2         ; Récepteur (entrée)
#DEFINE FREQ    PORTD,1         ; Test des 40kHz (sortie)
#DEFINE SALVE   PORTD,0         ; Test des 3.2ms (sortie)
#DEFINE REQST   PORTD,7         ; Request (entrée)
#DEFINE READY   PORTD,6         ; Ready (sortie)
#DEFINE CODE1   PORTD,5         ; CODE1 (entrée)
#DEFINE CODE0   PORTD,4         ; CODE0 (entrée)
#DEFINE OSCILLE FLAGS,0         ; indique si oscillations en cours (1=oui)
#DEFINE ECOUTE  FLAGS,1         ; indique si on écoute le front montant de R (1=oui)
#DEFINE FINI    FLAGS,2         ; indique si on écoute le front descendant de R (1=oui)

;*****************************************************************************
;                                   MACRO                                     *
;*****************************************************************************

BANK0   macro                           ; passer en banque0
                bcf     STATUS,RP0
                bcf     STATUS,RP1
        endm

BANK1   macro                           ; passer en banque1
                bsf     STATUS,RP0
                bcf     STATUS,RP1
        endm

BANK2   macro                           ; passer en banque2
                bcf     STATUS,RP0
                bsf     STATUS,RP1
        endm

BANK3   macro                           ; passer en banque3
                bsf     STATUS,RP0
                bsf     STATUS,RP1
        endm

REEPROM macro                           ; lire eeprom(adresse & résultat en w)
                clrwdt                  ; reset watchdog
                bcf     STATUS,RP0      ; passer en banque2
```

```
                bsf     STATUS,RP1
                movwf   EEADR           ; pointer sur adresse eeprom
                bsf     STATUS,RP0      ; passer en banque3
                bcf     EECON1,EEPGD    ; pointer sur eeprom
                bsf     EECON1,RD       ; ordre de lecture
                bcf     STATUS,RP0      ; passer en banque2
                movf    EEDATA,w        ; charger valeur lue
                bcf     STATUS,RP1      ; passer en banque0
        endm


                        ; Sauts inter-pages
                        ; -----------------

GOTOX   macro   ADRESSE                 ; saut inter-page
        local BIT4 = (ADRESSE & 0x1000)       ; voir bit 12
        local BIT3 = (ADRESSE & 0x0800) ; voir bit 11
        local   ICI                     ; adresse courante
ICI
        local PICI = (ICI+2 & 0x1800) ; page du saut
        IF BIT3                         ; si page 1 ou 3
        bsf     PCLATH , 3              ; b3 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 3              ; b3 de PCLATH = 0
        ENDIF
        IF BIT4                         ; si page 2 ou 3
        bsf     PCLATH , 4              ; b4 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 4              ; b4 de PCLATH = 0
        ENDIF
        goto (ADRESSE & 0x7FF | PICI) ; adresse simulée
        endm


PCLAX   macro   ADRESSE                 ; positionne PCLATH pour
                                        ; les sauts sans le saut
        local BIT4 = (ADRESSE & 0x1000)       ; voir bit 12
        local BIT3 = (ADRESSE & 0x0800) ; voir bit 11

        IF BIT3                         ; si page 1 ou 3
        bsf     PCLATH , 3              ; b3 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 3              ; b3 de PCLATH = 0
        ENDIF
        IF BIT4                         ; si page 2 ou 3
        bsf     PCLATH , 4              ; b4 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 4              ; b4 de PCLATH = 0
        ENDIF
        endm

GOTSX   macro   ADRESSE                 ; saut inter-page sans
                                        ; sélection de PCLATH
        local   ICI                     ; adresse courante
        local PICI = (ICI & 0x1800)   ; page du saut
ICI
        goto (ADRESSE & 0x7FF | PICI) ; adresse simulée
        endm

                        ; Sous-routines inter-pages
                        ; -------------------------

CALLX   macro   ADRESSE                 ; call inter-page
        local BIT4 = (ADRESSE & 0x1000)       ; voir bit 12
        local BIT3 = (ADRESSE & 0x0800) ; voir bit 11
        local   ICI                     ; adresse courante
ICI
        local PICI = (ICI+2 & 0x1800) ; page du saut
        IF BIT3                         ; si page 1 ou 3
        bsf     PCLATH , 3              ; b3 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 3              ; b3 de PCLATH = 0
        ENDIF
        IF BIT4                         ; si page 2 ou 3
        bsf     PCLATH , 4              ; b4 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 4              ; b4 de PCLATH = 0
        ENDIF
        call (ADRESSE & 0x7FF | PICI) ; adresse simulée
        local BIT4 = ((ICI+5) & 0x1000)       ; voir bit 12
        local BIT3 = ((ICI+5) & 0x0800)       ; voir bit 11
        IF BIT3                         ; si page 1 ou 3
        bsf     PCLATH , 3              ; b3 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 3              ; b3 de PCLATH = 0
        ENDIF
        IF BIT4                         ; si page 2 ou 3
        bsf     PCLATH , 4              ; b4 de PCLATH = 1
        ELSE                            ; sinon
        bcf     PCLATH , 4              ; b4 de PCLATH = 0
        endm
```

```
CALLSX macro  ADRESSE               ; sous-routine inter-page sans
                                    ; sélection de PCLATH
       local  ICI                   ; adresse courante
       local PICI = (ICI & 0x1800)  ; page du saut
ICI
       call (ADRESSE & 0x7FF | PICI); adresse simulée
       endm

                     ; Sous-routines eeprom
                     ; --------------------

WEEPROM macro  addwrite       ; la donnée se trouve dans W
       LOCAL  loop
       bcf    STATUS,RP0     ; passer en banque2
       bsf    STATUS,RP1
       movwf  EEDATA         ; placer data dans registre
       movlw  addwrite       ; charger adresse d'écriture
       movwf  EEADR          ; placer dans registre
       bsf    STATUS,RP0     ; passer en banque3
       bcf    EECON1 , EEPGD ; pointer sur mémoire data
       bsf    EECON1 , WREN  ; autoriser accès écriture
       bcf    INTCON , GIE   ; interdire interruptions
       movlw  0x55           ; charger 0x55
       movwf  EECON2         ; envoyer commande
       movlw  0xAA           ; charger 0xAA
       movwf  EECON2         ; envoyer commande
       bsf    EECON1 , WR    ; lancer cycle d'écriture
       bsf    INTCON , GIE   ; réautoriser interruptions
loop
       btfsc  EECON1 , WR    ; tester si écriture terminée
       goto   loop           ; non, attendre
       bcf    EECON1 , WREN  ; verrouiller prochaine écriture
       bcf    STATUS , RP0   ; passer en banque0
       bcf    STATUS , RP1
       endm

;*****************************************************************************
;                       VARIABLES BANQUE 0                                   *
;*****************************************************************************

; Zone de 80 bytes
; ----------------

       CBLOCK 0x20           ; Début de la zone (0x20 à 0x6F)
       FLAGS : 1             ; variable contenant les flags
                             ; b0 : OSCILLE
                             ; b1 : ECOUTE
                             ; b2 : FINI
       CPT : 1               ; compte le nombre d'oscillations dans une salve
       TXCPT : 2             ; compteur pour ralentir la transmission
       SPEED : 1             ; vitesse mesurée
       RELIAB : 1            ; fiabilité de la mesure
       STATIK : 1            ; valeur de la pression statique
       DYNAMIK : 1           ; valeur de la pression dynamique
       PHASES : 1            ; phases de vol
       CPTVOIE : 1           ; Numéro de la voie en cours d'émission

       ENDC                  ; Fin de la zone

var1   EQU H'006E'           ; adresse imposée

;*****************************************************************************
;                      VARIABLES ZONE COMMUNE                                *
;*****************************************************************************

; Zone de 16 bytes
; ----------------

       CBLOCK 0x70           ; Début de la zone (0x70 à 0x7F)
       w_temp : 1            ; Sauvegarde registre W
       status_temp : 1             ; sauvegarde registre STATUS
       FSR_temp : 1          ; sauvegarde FSR (si indirect en interrupt)
       PCLATH_temp : 1             ; sauvegarde PCLATH (si prog>2K)

       ENDC

;*****************************************************************************
;                      DEMARRAGE SUR RESET                                   *
;*****************************************************************************

       org 0x000             ; Adresse de départ après reset
       clrf    PCLATH          ; Effacer sélecteur de pages
       goto    init          ; Initialiser

; //////////////////////////////////////////////////////////////////////////

;                         I N T E R R U P T I O N S

; //////////////////////////////////////////////////////////////////////////

;*****************************************************************************
```

```
;                       ROUTINE INTERRUPTION                                  *
;*****************************************************************************
;-----------------------------------------------------------------------------
; Si on n'utilise pas l'adressage indirect dans les interrupts, on se passera
; de sauvegarder FSR
; Si le programme ne fait pas plus de 2K, on se passera de la gestion de
; PCLATH
;-----------------------------------------------------------------------------
                        ;sauvegarder registres
                        ;---------------------
        org 0x004               ; adresse d'interruption
        movwf   w_temp          ; sauver registre W
        swapf   STATUS,w        ; swap status avec résultat dans w
        movwf   status_temp     ; sauver status swappé
        movf    FSR , w         ; charger FSR
        movwf   FSR_temp        ; sauvegarder FSR
        movf    PCLATH , w      ; charger PCLATH
        movwf   PCLATH_temp     ; le sauver
        clrf    PCLATH          ; on est en page 0
        BANK0                   ; passer en banque0


                        ; Interruption TMR1
                        ; -----------------
; L'interruption doit être générée à chaque demi période des 40kHz,
; c'est à dire tous les 62 cyles.
; Or il y a 14 cycles entre le débordement du TMR1 et l'écriture de EMETT, donc
; on recharge TMR1 à (255 ; 255 - 62 + 14) = (255 ; 207)

        btfss   OSCILLE         ; si OSCILLE, on émet
        goto    decCPT          ; sinon, on décrémente directement CPT
        movlw   b'00001000'     ; inverse EMETT (PORTC.3)
        xorwf   PORTC,1

decCPT
        movlw   b'00000010'     ; inverse FREQ (PORTD.1)
        xorwf   PORTD,1

        decfsz  CPT,1
        goto    reload          ; si CPT <> 0, recharge directement TMR1
        movlw   255             ; sinon,
        movwf   CPT             ; recharge CPT
        movlw   b'00000001'     ; inverse OSCILLE (FLAGS.0)
        xorwf   FLAGS,1
        movlw   b'00000001'     ; inverse SALVE (PORTD.0)
        xorwf   PORTD,1

reload                          ; rechargement de TMR1L et TMR1H
        movlw   219
        movwf   TMR1L
        movlw   B'11111111'
        movwf   TMR1H


        bcf     PIR1,TMR1IF     ; effacer flag interupt


                        ;restaurer registres
                        ;-------------------
        movf    PCLATH_temp , w       ; recharger ancien PCLATH
        movwf   PCLATH          ; le restaurer
        movf    FSR_temp , w    ; charger FSR sauvé
        movwf   FSR             ; restaurer FSR
        swapf   status_temp,w   ; swap ancien status, résultat dans w
        movwf    STATUS         ; restaurer status
        swapf    w_temp,f       ; Inversion L et H de l'ancien W
                                ; sans modifier Z
        swapf    w_temp,w       ; Réinversion de L et H dans W
                                ; W restauré sans modifier status
        retfie                  ; return from interrupt




; //////////////////////////////////////////////////////////////////////////

;                           P R O G R A M M E

; //////////////////////////////////////////////////////////////////////////

;*****************************************************************************
;                           INITIALISATIONS                                   *
;*****************************************************************************

init

                        ; initialisation PORTS (banque 0 et 1)
                        ; ------------------------------------
        BANK0                   ; sélectionner banque0
        clrf    PORTA           ; Sorties PORTA à 0
```

```
        clrf    PORTB           ; sorties PORTB à 0
        clrf    PORTC           ; sorties PORTC à 0
        clrf    PORTD           ; sorties PORTD à 0
        clrf    PORTE           ; sorties PORTE à 0
        bsf     STATUS,RP0      ; passer en banque1
        movlw   DIRPORTA        ; Direction PORTA
        movwf   TRISA           ; écriture dans registre direction
        movlw   DIRPORTB        ; Direction PORTB
        movwf   TRISB           ; écriture dans registre direction
        movlw   DIRPORTC        ; Direction PORTC
        movwf   TRISC           ; écriture dans registre direction
        movlw   DIRPORTD        ; Direction PORTD
        movwf   TRISD           ; écriture dans registre direction
        movlw   DIRPORTE        ; Direction PORTE
        movwf   TRISE           ; écriture dans registre direction

                ; Registre d'options (banque 1)
                ; ----------------------------
        movlw   OPTIONVAL       ; charger masque
        movwf   OPTION_REG      ; initialiser registre option

                ; registres interruptions (banque 1)
                ; ----------------------------------
        movlw   INTCONVAL       ; charger valeur registre interruption
        movwf   INTCON          ; initialiser interruptions
        movlw   PIE1VAL         ; Initialiser registre
        movwf   PIE1            ; interruptions périphériques 1
        movlw   PIE2VAL         ; initialiser registre
        movwf   PIE2            ; interruptions périphériques 2

                ; Effacer RAM banque 0
                ; --------------------
        bcf     STATUS,RP0      ; sélectionner banque 0
        movlw   0x20            ; initialisation pointeur
        movwf   FSR             ; d'adressage indirect
init1
        clrf    INDF            ; effacer ram
        incf    FSR,f           ; pointer sur suivant
        btfss   FSR,7           ; tester si fin zone atteinte (>7F)
        goto    init1           ; non, boucler

                ; Registres de configuration de l'USART (banque ?)
                ; ------------------------------------------------

        banksel TXSTA           ; charge TXSTA
        movlw   TXSTA_MASK
        movwf   TXSTA

        banksel RCSTA           ; charge RCSTA
        movlw   RCSTA_MASK
        movwf   RCSTA

        banksel SPBRG           ; charge SPBRG
        movlw   SPBRG_MASK
        movwf   SPBRG

                ; Registres de configuration du CAN (banque ?)
                ; --------------------------------------------

        banksel ADCON0          ; charge ADCON0
        movlw   ADCON0_MASK
        movwf   ADCON0

        banksel ADCON1          ; charge ADCON1
        movlw   ADCON1_MASK
        movwf   ADCON1

                ; Registres de configuration du TMR1 (banque ?)
                ; ---------------------------------------------

        banksel T1CON           ; charge T1CON
        movlw   T1CON_MASK
        movwf   T1CON

        BANK0                   ; passe en bank0

                ; autoriser interruptions (banque 0)
                ; ----------------------------------
        clrf    PIR1            ; effacer flags 1
        clrf    PIR2            ; effacer flags 2
        bsf     INTCON,GIE      ; valider interruptions
        goto    start           ; programme principal

; On arrive dans le programme principal en étant en banque 0.
; Ceci permet de lire TMR0 et toutes nos variables.


;*******************************************************************************
;                       PROGRAMME PRINCIPAL                                   *
;*******************************************************************************

start
```

```
; calcul de SPEED et de RELIAB grâce à TMR0
; ----------------------------------------

; distance enter émetteur et recepteur US : 60cm
; célérité du son : 340m/s
; à v=0, le temps de propagation est : 0.001765sec
; à v=200, le temps de propagation est : 0.001111sec, ce qui correspond à une distance de 0.38cm pour
une vitesse nulle
; v=0 => 8825cycles => 552
; v=200 => 5556cycles => 347
; on veut obtenir TMR0=0 pour v=200, donc à 0.001111sec càd au bout de 89 oscillations, on remet TMR0 à
0 et on écoute
; on obtiendra donc TMR0=204 pour v=0

; si (OSCILLE = 1 et CPT = 255 - 89 = 167),
; alors (TMR0 := 0 et ECOUTE := 1)
        btfss   OSCILLE         ; skip si OSCILLE = 1
        goto    label1
        movf    CPT,W
        sublw   167
        btfss   STATUS,Z        ; skip si 240 - CPT = 0
        goto    label1
        clrf    TMR0
        bsf     ECOUTE

label1
; si (ECOUTE = 1 et RECEP = 1)
; alors (SPEED := TMR0 et TMR0 := 0 et ECOUTE := 0 et FINI := 0)
        btfss   ECOUTE          ; skip si ECOUTE = 1
        goto    label2
        btfss   RECEP           ; skip si RECEP = 1
        goto    label2
        movf    TMR0,W
        movwf   SPEED
        clrf    TMR0
        bcf     ECOUTE
        bcf     FINI

label2
; si (FINI = 0 et RECEP = 0)
; alors (RELIAB := TMR0 et FINI := 1)
        btfsc   FINI            ; skip si FINI = 0
        goto    label3
        btfsc   RECEP           ; skip si RECEP = 0
        goto    label3
        movf    TMR0,W
        movwf   RELIAB
        bsf     FINI

label3
        BANK1
        btfsc   TXSTA,TRMT      ; regarde si buffer vide
        goto    ecriture        ; si oui écrire dans l'USART
        BANK0                   ; si non
        goto    start           ; et boucler


; écriture des trames dans TXREG
; ------------------------------

ecriture
        BANK0
        incf    CPTVOIE,f
        movf    CPTVOIE,W       ; teste si CPTVOIE = 1
        sublw   1
        btfsc   STATUS,Z
        goto    wrSPEED
        movf    CPTVOIE,W       ; teste si CPTVOIE = 2
        sublw   2
        btfsc   STATUS,Z
        goto    wrRELIAB
        movf    CPTVOIE,W       ; teste si CPTVOIE = 3
        sublw   3
        btfsc   STATUS,Z
        goto    wrSTATIK
        movf    CPTVOIE,W       ; teste si CPTVOIE = 4
        sublw   4
        btfsc   STATUS,Z
        goto    wrDYNAMIK
        movf    CPTVOIE,W       ; teste si CPTVOIE = 5
        sublw   5
        btfsc   STATUS,Z
        goto    wrPHASES
        movlw   h'FF'           ; sinon CPTVOIE = 6  donc synchro
        movwf   TXREG
        clrf    CPTVOIE
        goto    start
wrSPEED
        incfsz  SPEED,W         ; teste si SPEED = FF
        goto    wrSPEED2        ; si non, écrire SPEED
```

```
        decf    SPEED,W         ; si oui, mettre FE dans TXREG et boucler
        movwf   TXREG
        goto    start
wrSPEED2
        movf    SPEED,W         ; met la valeur de SPEED dans W
        movwf   TXREG           ; place W dans buffer
        goto    start           ; boucler
wrRELIAB
        incfsz  RELIAB,W        ; teste si RELIAB = FF
        goto    wrRELIAB2       ; si non, écrire RELIAB
        decf    RELIAB,W        ; si oui, mettre FE dans TXREG et boucler
        movwf   TXREG
        goto    start
wrRELIAB2
        movf    RELIAB,W        ; met la valeur de RELIAB dans W
        movwf   TXREG           ; place W dans buffer
        goto    start           ; boucler
wrSTATIK
        btfsc   ADCON0,GO       ; regarde si CAN fini
        goto    wrSTATIK2       ; si non, on écrit directement l'ancienne valeur de STATIK
        movf    ADRESH,W        ; si oui, on met à jour STATIK
        movwf   STATIK
        incfsz  STATIK,W        ; teste si STATIK = FF
        goto    wrSTATIK2       ; si non, écrire wrSTATIK
        decf    STATIK,W        ; si oui, mettre FE dans TXREG et boucler
        movwf   TXREG
        goto    start
wrSTATIK2
        movf    STATIK,W        ; met la valeur de STATIK dans W
        movwf   TXREG           ; place W dans buffer
        goto    start           ; boucler
wrDYNAMIK
        btfsc   ADCON0,GO       ; regarde si CAN fini
        goto    wrDYNAMIK2      ; si non, on écrit directement l'ancienne valeur de DYNAMIK
        movf    ADRESH,W        ; si oui, on met à jour DYNAMIK
        movwf   DYNAMIK
        incfsz  DYNAMIK,W       ; teste si DYNAMIK = FF
        goto    wrDYNAMIK2      ; si non, écrire wrDYNAMIK
        decf    DYNAMIK,W       ; si oui, mettre FE dans TXREG et boucler
        movwf   TXREG
        goto    start
wrDYNAMIK2
        movf    DYNAMIK,W       ; met la valeur de DYNAMIK dans W
        movwf   TXREG           ; place W dans buffer
        goto    start           ; boucler
wrPHASES
        movf    PORTE,W         ; met les phases de vol (PORTE) dans PHASES
        movwf   PHASES
        btfss   PORTA,5         ; met RA5 dans PHASES
        goto    wrPHASES2
        bsf     PHASES,3
        movf    PHASES,W
        movwf   TXREG
        goto    start           ; boucler
wrPHASES2
        bcf     PHASES,3
        movf    PHASES,W
        movwf   TXREG
        goto    start

        END                     ; directive fin de programme
```