SUIVEUR SOLAIRE

SUPPORT A ORIENTATION AUTOMATIQUE POUR PANNEAUX PHOTOVOLTAÏQUES

Le 8 février 2018

TABLE DES MATIERES

I - INTRODUCTION :

Le projet a consisté à concevoir et construire un suiveur solaire équipé d'un ensemble de 6 panneaux photovoltaïques. La solution envisagée pour placer les panneaux perpendiculairement aux rayons du soleil a été de calculer l'azimut et la hauteur du soleil à chaque minute tout au long de la journée. Cette technique, quoique assez complexe, permet d'effectuer un suivi relativement précis. Nous pourrions nous contenter d'une précision de 2° ou 3° environ. Il s'avère que les calculs ont permis d'atteindre une précision de l'ordre de 0,2 à 0,3°. Le système a été conçu pour permettre un déplacement par pas de 1 degré. La précision globale obtenue est donc de plus ou moins 1 degré.
Le cœur du système est un microcontrôleur
Le positionnement à l'aide d'un asservissement et à l'aide de photorésistances a volontairement été écarté pour privilégier une électronique plus simple à mettre en œuvre.
Revers de la médaille : Une programmation en assembleur toute fois assez complexe.
La construction (de type amateur) a été effectuée en partie avec des matériaux et matériels de récupération.
Matériel récupéré :

Moteurs sans balai d'imprimantes laser avec train d'engrenages et électronique associée.
Capteurs de position optoélectroniques à fourche.

Le suiveur est équipé de 6 panneaux Solarword d'une puissance de 275 W crête. Chaque panneaux est connecté à un micro-onduleur ENPHASE M250. La puissance globale du système est de 1500 W crête. L'emsemble est également relié à une passerelle Enphase ; ce qui permet de gérer la production à partir d'un ordinateur connecté à INTERNET.

II - MECANIQUE :

Le suiveur est constitué de 2 ensembles mécaniques :

1°) : Un ensemble mobile pivotant sur un axe vertical. Ce bloc peux orienter les panneaux photovoltaïques en hauteur par l'intermédiaire d'un axe horizontal.
Les panneaux photovoltaïques pivotent verticalement d'environ 18° à 90°. La position « 18° » correspond à une valeur minimale d'inclinaison des panneaux solaires par rapport à la verticale. Le fait que les panneaux ne puissent pas se positionner verticalement lorsque le soleil se trouve à l'horizon n'a pratiquement aucune incidence sur les performances de production. A ce moment là, le soleil est en phase de se lever ou de se coucher et la luminosité est très fortement diminuée. Dans cette position, le soleil se retrouve d'ailleurs masqué par les immeubles et les arbres environnants. La position « 90° » correspond à la hauteur maximum du soleil; Cette position n'est atteinte uniquement que sous les tropiques. La position « 90° » est utilisée pour positionner le suiveur en position de sécurité. En cas de vent trop violent, les panneaux sont placés horizontalement. Pour conserver une légère pente et favoriser l'écoulement de l'eau de pluie, les panneaux sont positionnés en réalité avec un angle de 88° plutôt que 90° (soit une pente d'environ 3,5%).

2°) : Un bloc fixe permettant d'entraîner l'ensemble mobile d'EST en OUEST de « moins 125° environ » à « plus 125° environ » par rapport à la position plein SUD. Ce bloc constitue le socle du système. Il est principalement constitué d'une platine d'environ 60 cm de diamètre sur laquelle tourne le bloc mobile par l'intermédiaire de 6 roulements à billes. Sur cette platine est fixée une couronne dentée de 144 dents permettant d'entraîner en rotation le bloc mobile par l'intermédiaire du moteur AZIMUT.

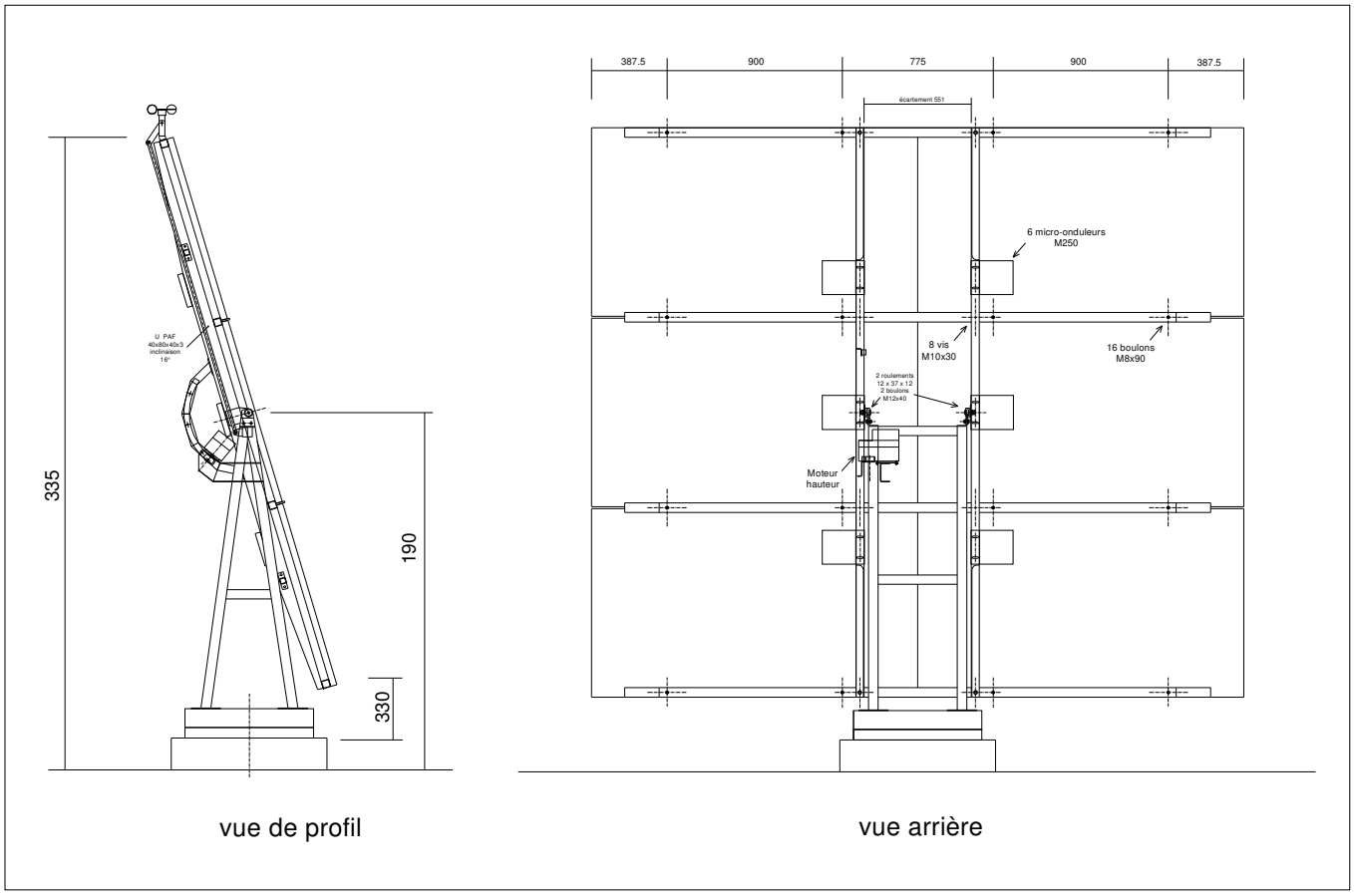
Le système est donc entraîné par 2 moteurs :
1°) - Moteur « hauteur » (sur l'ensemble mobile): Ce moteur est chargé de contrôler la position des panneaux pour suivre le soleil dans sa course en hauteur de son lever à son coucher.
2°) - Moteur « azimut » (sur l'ensemble fixe): Ce moteur est chargé de contrôler la position des panneaux pour suivre le soleil dans sa course en azimut d'EST en OUEST de son lever à son coucher.

Chacun des ces 2 ensembles mécaniques comporte:

Un moteur sans balai avec train d'engrenage de réduction de vitesse (motoréducteur).
Une cellule de positionnement (capteur à fourche) fixée sur le dernier étage de réduction de chacun des motoréducteurs.
Un contact (microswitch) permettant d'établir la position « zéro » (angle « hauteur » ou angle « azimut »)

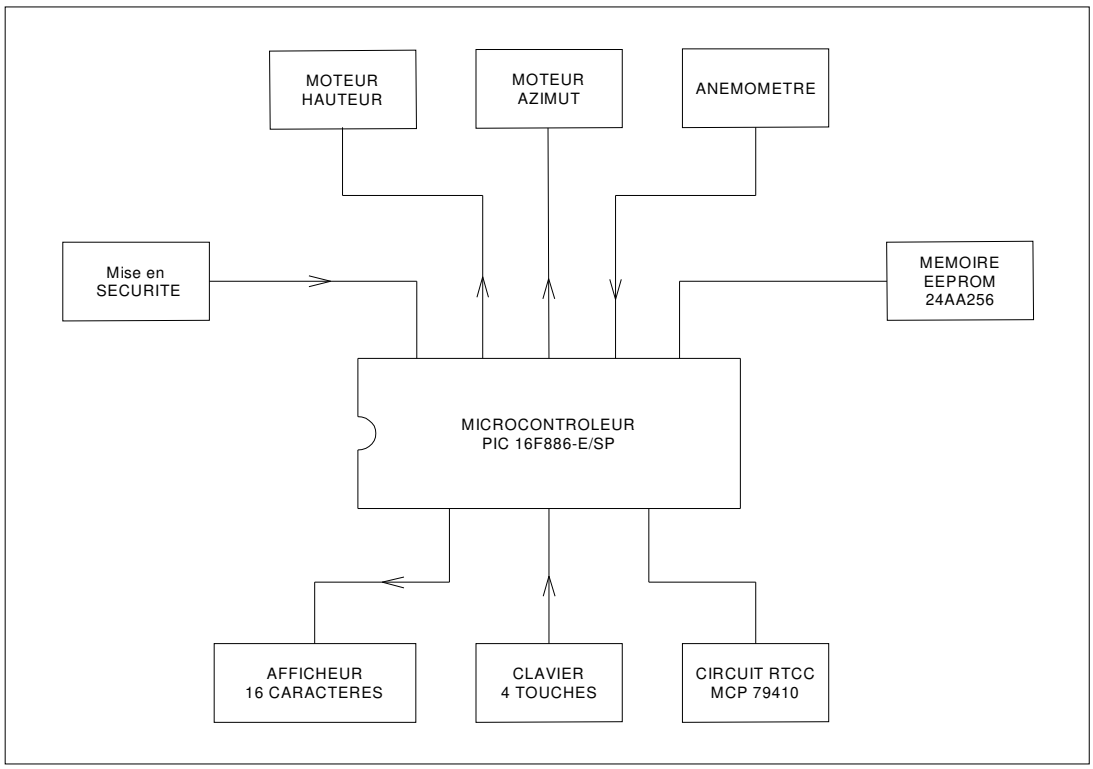
Pour réduire le prix de revient de l'ensemble, les motoréducteurs ont entièrement été fabriqués. Ayant récupéré les moteurs, il a simplement suffit de faire l'achat de quelques pignons (en nylon). La fabrication de ces motoréducteurs a également été possible grâce au tournage de quelques axes à l'aide du tour à métaux que je possède.

Figure II – 1 Vue d'ensemble



III - PRINCIPAUX COMPOSANTS :

Figure III – 1 Synoptique :



1 - Motoréducteur AZIMUT :

Le motoréducteur AZIMUT a été réalisé avec le moteur sans balai d'une imprimante laser Lexmark T520.
Ce moteur a été récupéré avec toute l'électronique de contrôle et quelques pignons associés.
Caractéristiques : Modèle DR-6236-112 24V 1800 t/mn 2,1A 0,16 Nm
Le moteur est alimenté par l'intermédiaire du connecteur CN1. Pour fonctionner, il faut fournir à ce moteur une fréquence d'horloge à appliquer sur la broche 3 (CLK) du connecteur CN1.
Cette fréquence d'horloge a été fournie par un circuit NE555 monté en multivibrateur sur une petite plaquette installée à l'intérieur du motoréducteur et alimenté en 24V (tension d'alimentation moteur) puis tension ramenée à 5V avec un régulateur 7805.
La fréquence a été réglée de telle sorte que le moteur tourne à une vitesse d'environ 1800 t/mn.
Le moteur est contrôlé par les circuits SANYO LB11822 et LB11825.

Le motoréducteur a été réalisé à l'aide de 4 trains d'engrenages.
Rapport 1er train d'engrenage : 9/62
Rapport 2ème train d'engrenage : 16/200
Rapport 3ème train d'engrenage : 12/72
Rapport 4ème train d'engrenage : 12/40
Rapport global : 1/1722,22
Soit une vitesse de rotation en sortie d'environ : 1 tour/mn

Un capteur optique à fourche a été installé à l'intérieur du motoréducteur pour détecter la position en rotation de l'axe de sortie par l'intermédiaire d'un disque comportant 25 encoches. Le déplacement d'une encoche à la suivante permet de faire tourner les panneaux d'un angle de 1°.

Figure 1 – 1 Moteur AZIMUT

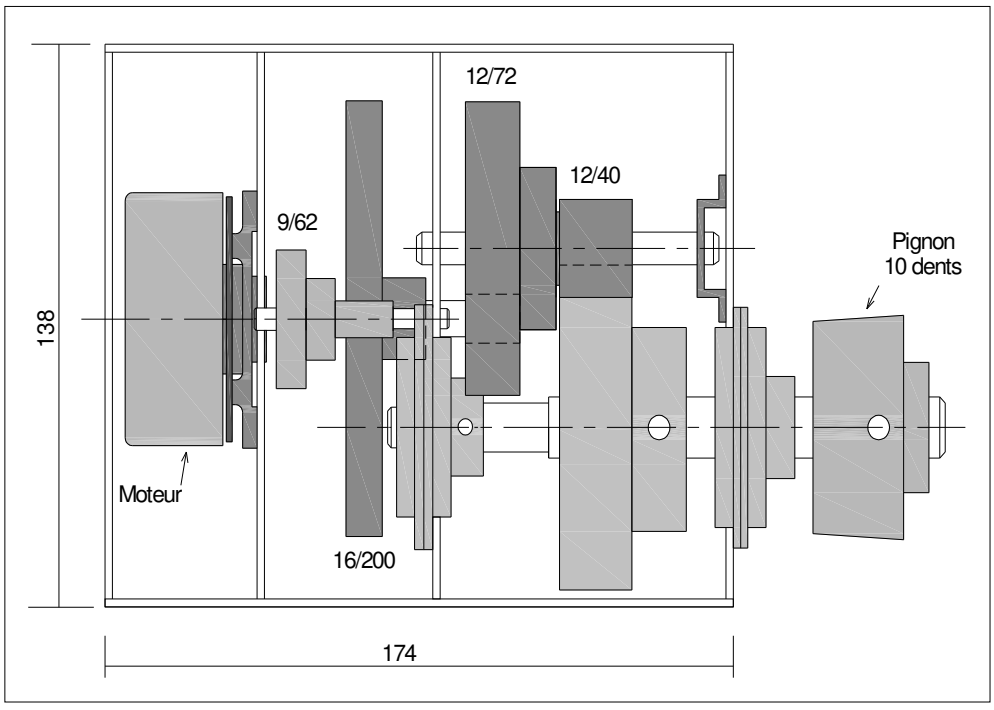
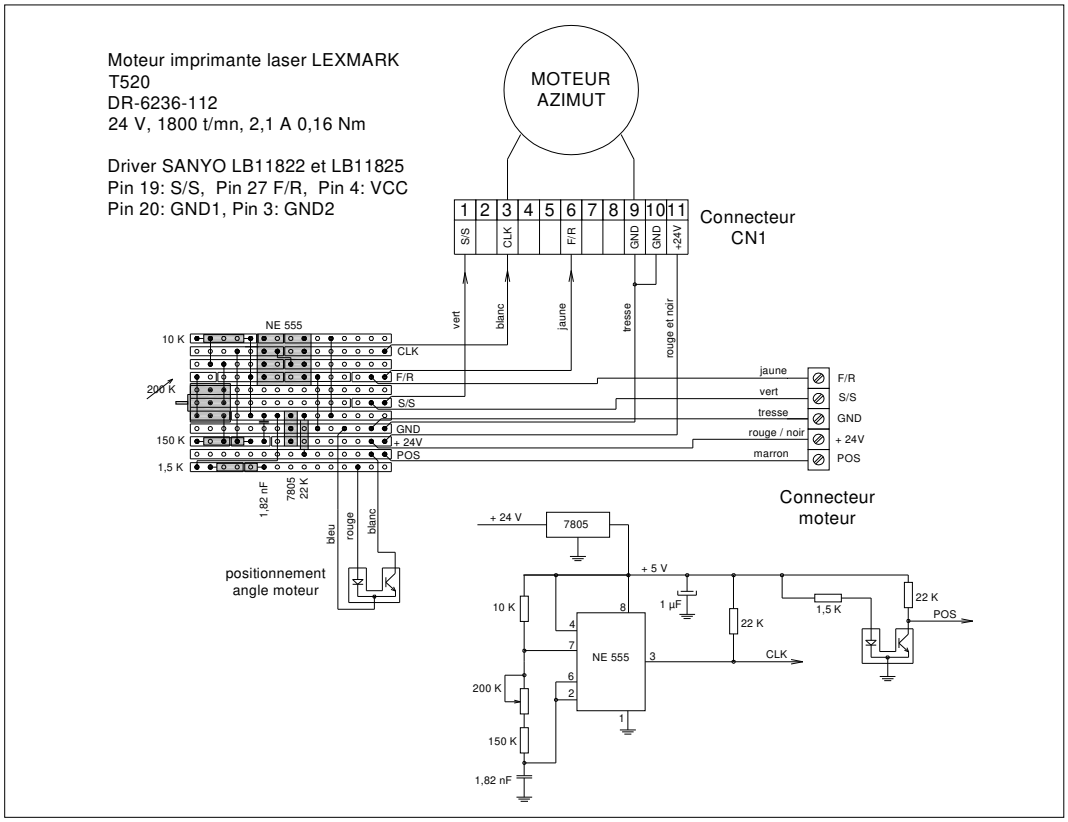


Figure 1 – 2 Câblage intérieur du motoréducteur AZIMUT



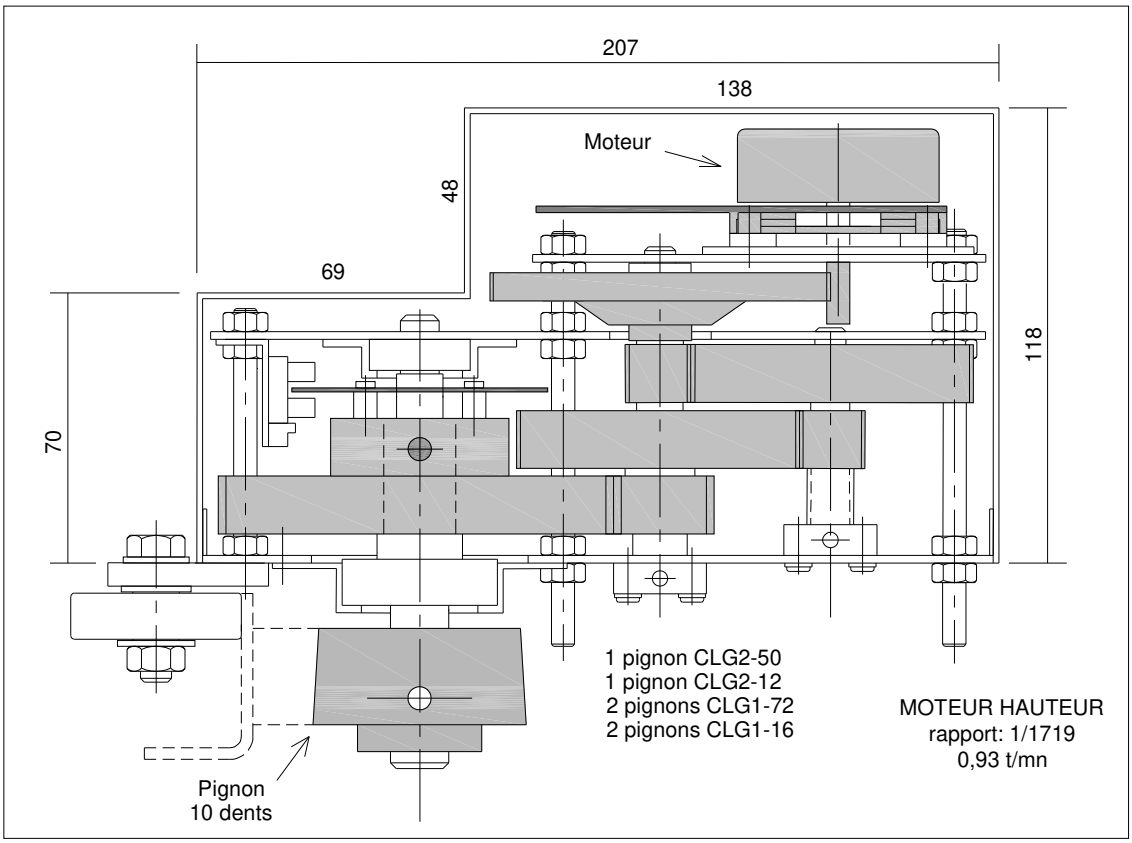
2 - Motoréducteur HAUTEUR :

Le motoréducteur HAUTEUR a été réalisé avec le moteur sans balai d'une imprimante laserjet HP4.
Ce moteur a été récupéré avec toute l'électronique de contrôle et quelques pignons associés.
Il est contrôlé par un circuit intégré spécialisé : le LB1824 qui a notamment la particularité de commander le moteur dans les 2 sens grâce à la broche 12 (F/R) forward/reverse pin (forward -> low, reverse -> high).
Le moteur a une puissance approximative de 20 W et est alimenté en courant continu de 24 V.
La mise en marche du moteur est effectuée par la broche 13 (S/S) start/stop pin (start -> low, stop -> high).
Le circuit peut être directement commandé avec la tension de 5 V fournie par la carte électronique du microcontrôleur.

Le motoréducteur a été réalisé à l'aide de 4 trains d'engrenages.
Rapport global : 1/1719
Vitesse approximative en sortie du réducteur : 0,93 tour/mn

Un capteur optique à fourche a été installé à l'intérieur du motoréducteur pour détecter la position en rotation de l'axe de sortie par l'intermédiaire d'un disque comportant 25 encoches. Le déplacement d'une encoche à la suivante permet de faire tourner les panneaux d'un angle de 1°.

Figure 2 – 1 Moteur HAUTEUR



3 – Câblage général :

Le système comporte 3 coffrets :

CC1 : Situé dans la partie basse du suiveur.
CC2 : Situé juste au dessous de l'axe de rotation horizontal.
CC3 : Situé à l'intérieur d'un local technique

Les 2 coffrets CC1 et CC2 situés sur le suiveur sont reliés par l'intermédiaire d'un tube en acier contenant 2 câbles de liaison.

Un premier câble véhicule la tention 230V du secteur (récupération de l'energie Photovoltaïque).
Le deuxième câble véhicule les différents signaux pour la commande du système.

En dehors de la liaison entre ces 2 coffrets,

CC1 reçoit les câbles provenant :

1 – du moteur « azimut »
2 – du contacteur « A0 »
3 – de l'alimentation générale 230V provenant du tableau général de répartition
4 – du câble pour la commande « marche/arrêt » situé à l'intérieur du local technique.
5 – du câble (12 conducteurs) qui véhicule les signaux entre le suiveur et le coffret de commande situé à l'intérieur de ce même local technique.
En outre, ce coffret renferme l'alimentation 24V = montée sur rail DIN

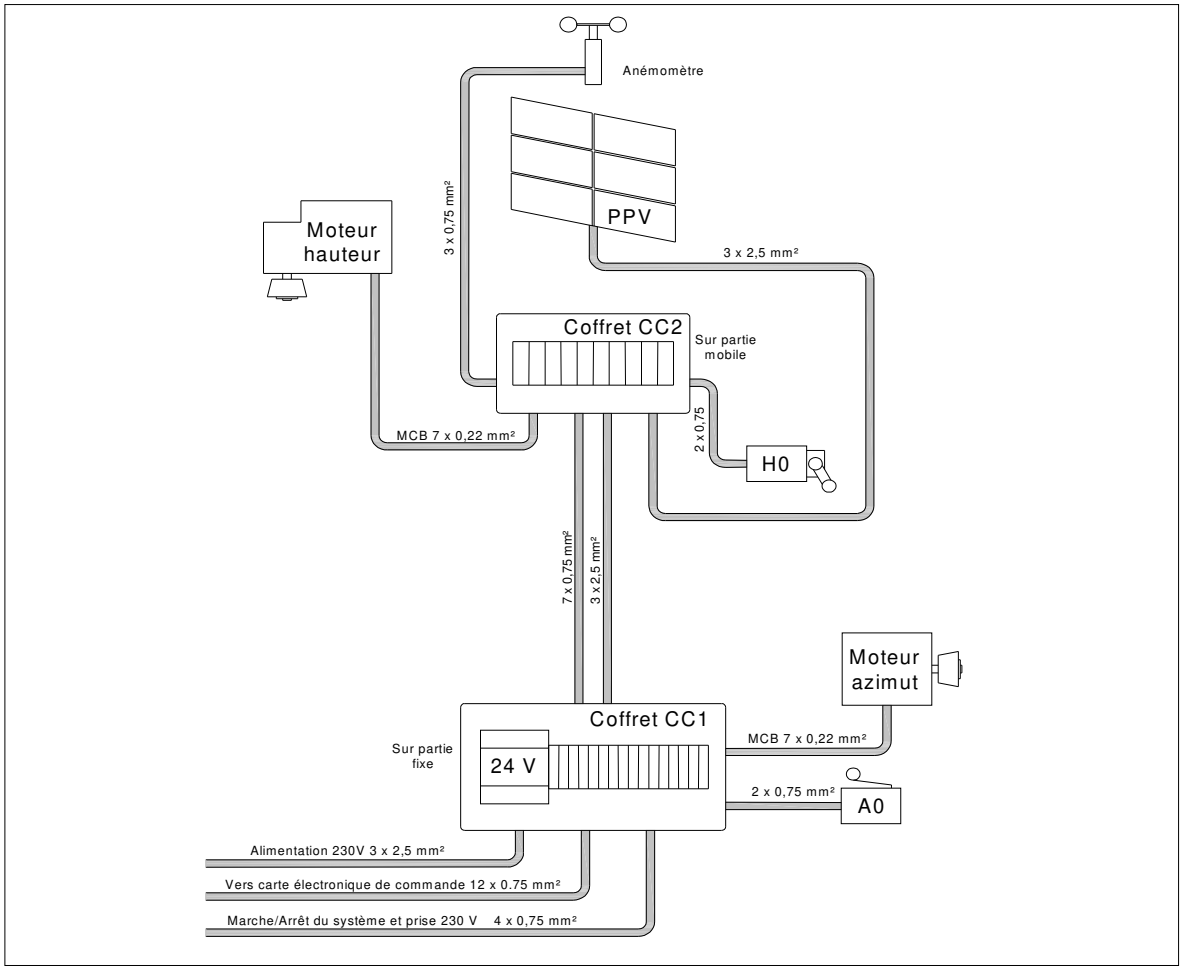
CC2 reçoit les câbles provenant :

1 – des panneaux photovoltaïques
2 – du moteur « hauteur »
3 – du contacteur « H0 »
4 – de l'anémomètre

CC3 situé à l'intérieur du local technique comporte la carte électronique de commande du système et reçoit le câble (12 conducteurs) qui véhicule les signaux de commande.

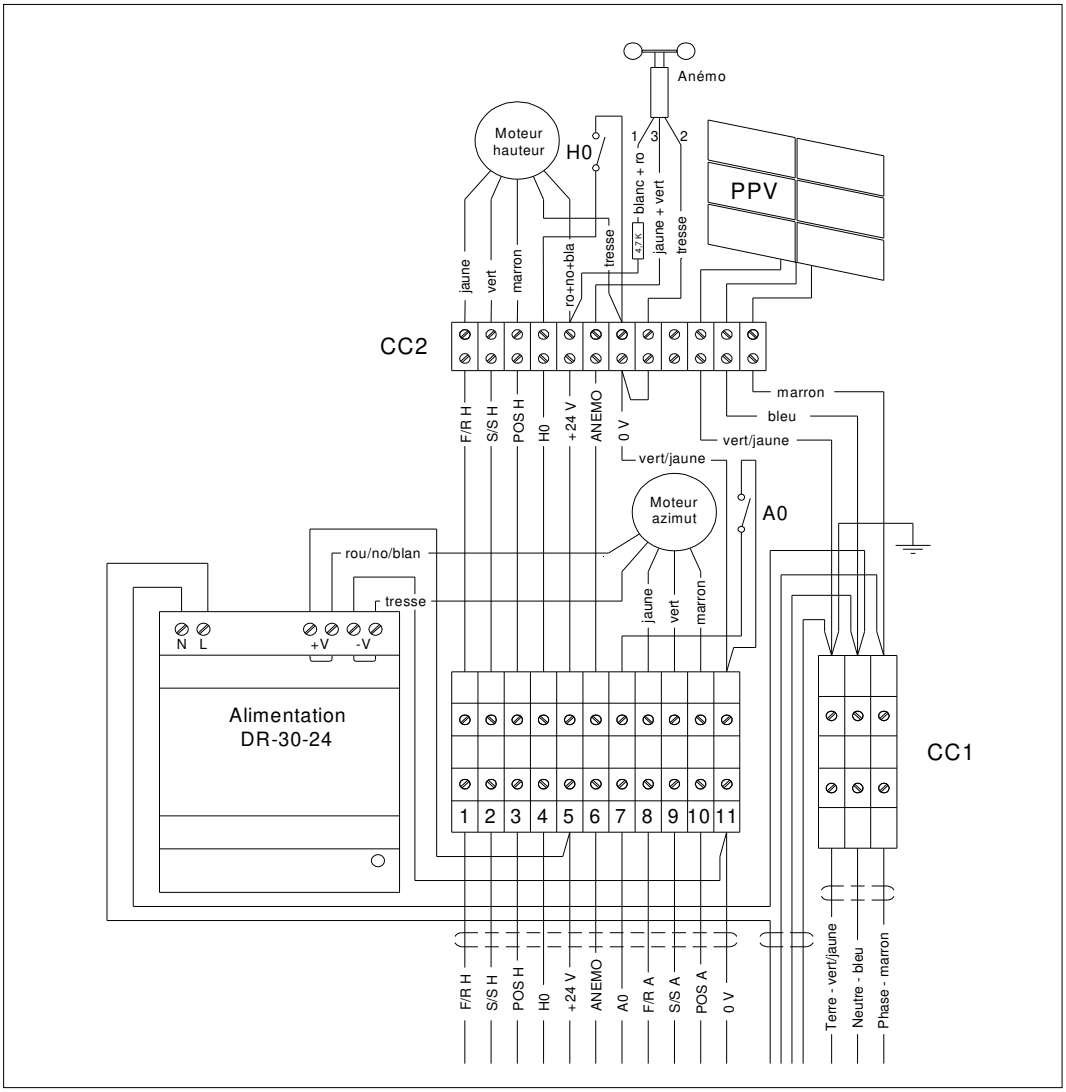
Ce coffret comporte également 6 commutateurs et boutons poussoirs pour permettre une commande manuelle du système.

Figure 3 – 1 Câblage général



4 – Câblage détaillé :

Figure 4 – 1 Câblage détaillé



5 – Tableau et coffret de la carte électronique:

La carte électronique a été installée dans le coffret CC3 lui-même installé à l'intérieur du local technique. Ce coffret aurait pu être installé à proximité du suiveur (ou sur le suiveur lui-même) donc à l'extérieur dans un coffret étanche.
Le choix d'installer ce coffret dans un local technique (chauffé en hiver) a permis de réaliser facilement la mise au point du programme. La distance entre le suiveur et ce local technique n'est que d'une quinzaine de mètres.
Le boitier contient également les commutateurs permettant une commande manuelle du suiveur.
Le boitier CC3 été installé sur un tableau qui rassemble également la passerelle ENPHASE et quelques prises (voir Figure 5-3).
Le suiveur est visible depuis ce tableau à travers une large fenêtre du local.

Figure 5 – 1 Câblage du coffret de la carte électronique

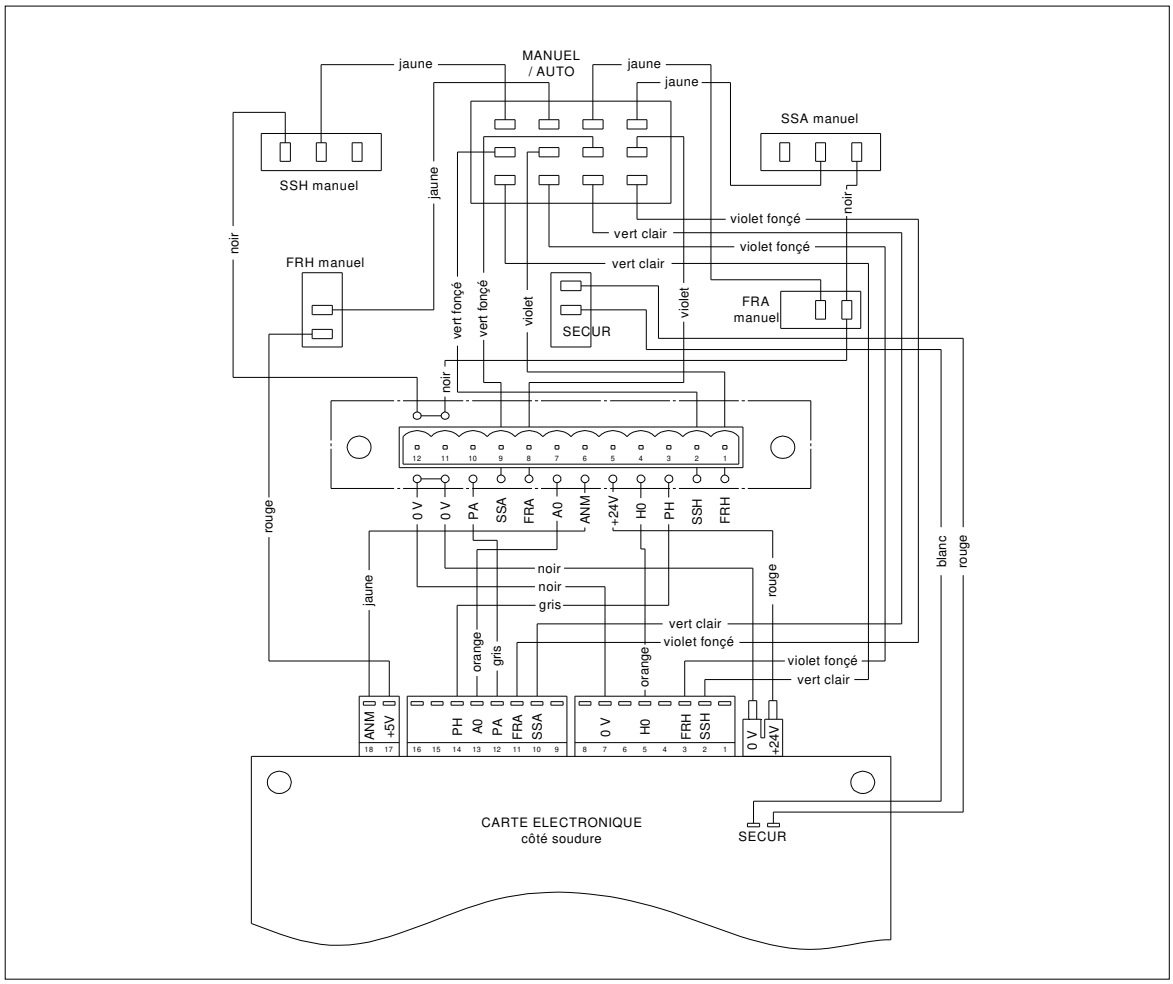


Figure 5 – 2 Coffret de la carte électronique

Figure 5 – 3 Tableau de contrôle et de commande

Sur le coffret de la carte électronique, on peut voir apparaître :
La date : 24/02/2018, l'heure : 12h28
La hauteur de soleil : 36 ,6°, l'azimut : 166,2°

Sur la passerelle ENVOY, on peut voir apparaître :
L'adresse IP de la passerelle : 192.168.1.10
+Web : Indique que la passerelle est bien connectée à INTERNET
1,45 W : Indique la puissance instantannée fournie
3,03 MWh Indique la production totale fournie depuis l'installation du suiveur.

Sur le coffret CC3 de la carte électronique on peut voir :
Le commutateur permettant de passer en mode manuel (en haut, au centre).
Le commutateur permettant d'orienter le mouvement des panneaux d'est en ouest (en bas, à gauche)
Le commutateur permettant d'orienter le mouvement des panneaux de haut en bas (en bas, à droite)
Le bouton poussoir permettant de commander le mouvement des panneaux d'est en ouest (en haut, à gauche)
Le bouton poussoir permettant de commander le mouvement des panneaux de haut en bas (en haut, à droite)
Le commutateur permettant de positionner les panneaux en mode sécurité (horizontalement) (en bas, au centre)

En bas, à gauche, un interrupteur permet la mise en marche ou l'arrêt du suiveur. Cet interrupteur n'agit que sur l'alimention de la carte électronique et la tension 24V alimentant les moteurs hauteur et azimut.
Il n'agit en aucun cas sur la tension 230V des panneaux photovoltaïques.

En bas et au milieu du tableau un switch ETHERNET permet de relier la passerelle et plusieurs ordinateurs à la BOX INTERNET.

6 – Carte électronique :

La carte électronique a été réalisée sur un circuit imprimé simple face.
L'afficheur est directement implanté sur la carte ainsi que le clavier 4 touches.

Figure 6 – 1 Carte électronique

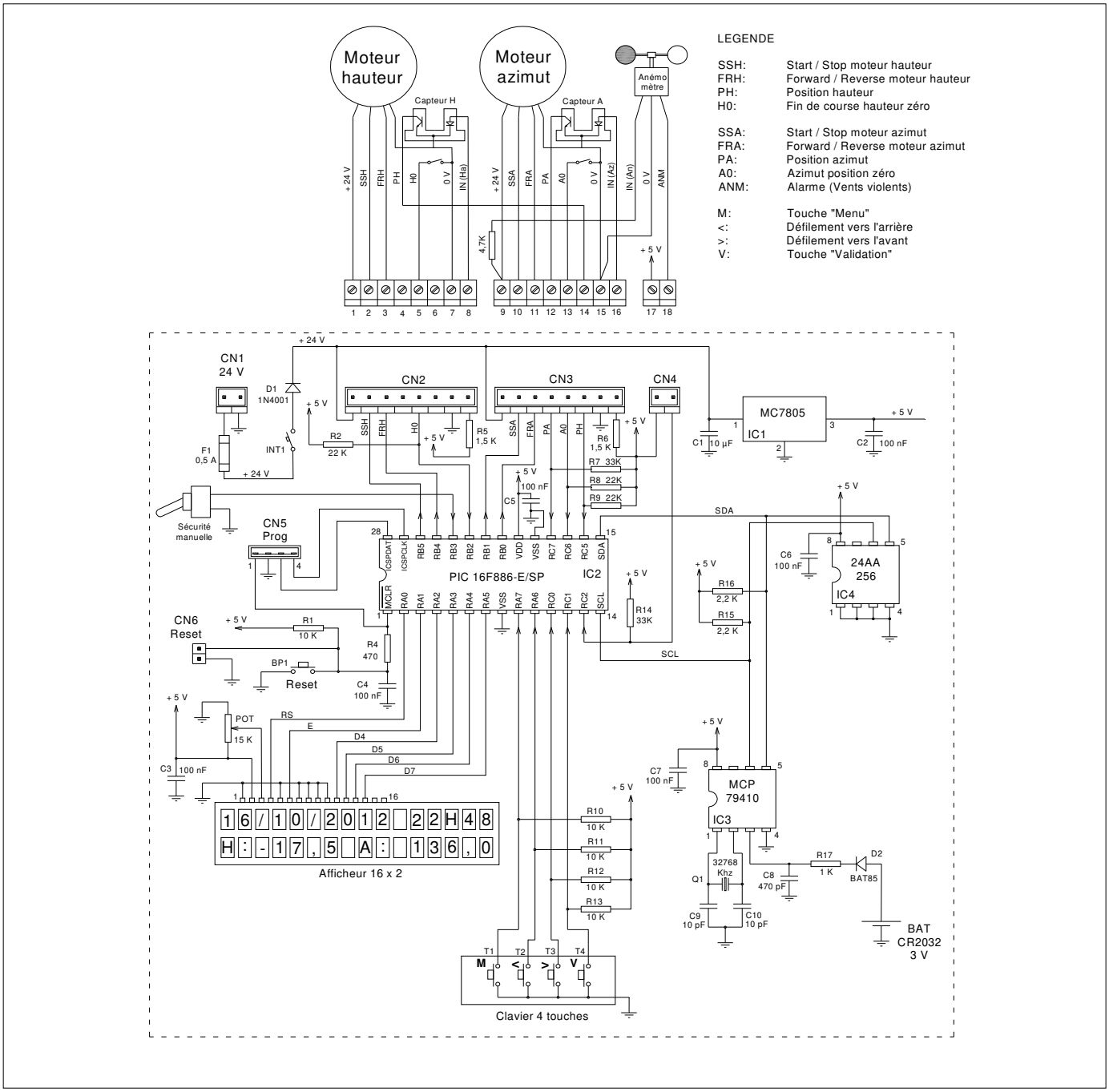


Figure 6 – 2 Typon

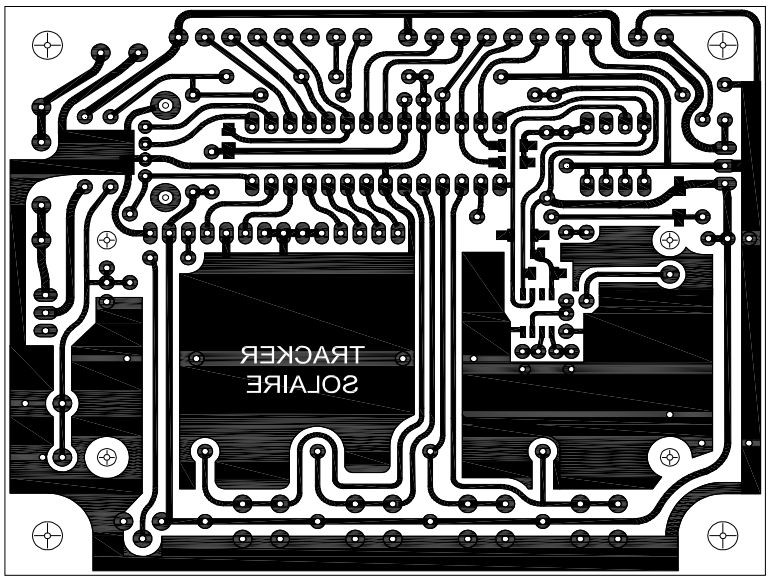
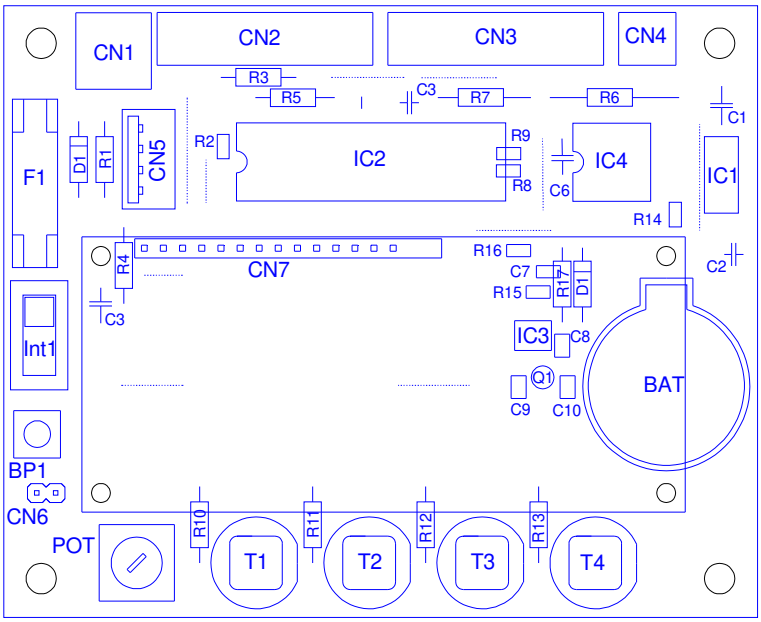


Figure 6 – 3 Implantation des composants



7 – Conposants de la carte électronique :

Figure 7 : Nomenclature des composants

Qté	Désignation	Repère	Fabriquant	Réf fabricant
	Carte Circuit imprimé			
1	Carte présensibilisée 160x100		CIF	AA16
1	Embase verticale 5 mm - 2 contacts	CN1		
1	Embase coudée 3,5 mm - 2 contacts	CN4	MULTICOMP	MC000088
2	Embasecoudée 3,5 mm - 8 contacts	CN2, CN3	MULTICOMP	MC000093
1	Prise USB	CN5		
1	Connecteur 2x1 - pas de 2,54 (reset)	CN6		
1	Connecteur 16x1 - pas de 2,54 mm			
1	Support pile bouton CR2032	BAT		
1	Pile bouton CR2032	BAT	MULTICOMP	CR2032
4	Touche CI	T1,T2,T3,T4		
1	Bouton poussoir CI (reset)	BP1		
1	Afficheur 16x2			CCM-1620
1	Régulateur de tension 5 V	IC1	ON Semicond	MC7805CTG
1	PIC 16F886-E/SP	IC2	MICROCHIP	16F886-E/SP
1	Support CI 28 broches - 7,62 mm	IC2	MULTICOMP	2227MC-28-03-05-F1
1	Real Time Clock and Calendar	IC3	MICROCHIP	MCP79410-I/SN
1	Mémoire EEPROM 256K	IC4	MICROCHIP	24AA256-I/P
1	Interrupteur CI à glissière	Int1		
1	Porte fusible 5x20	F1		
1	Fusible 5x20 - 0,5 A	F1		
1	Quartz 32768 Khz	Q1	CITIZEN	CFS206-32.768-KDFZ
1	Diode 1N4001	D1		
1	Diode Schottky BAT85	D2	VISHAY	BAT-85 T/R
1	Résistance 470 ohm - 0,25 W	R4		
1	Résistance 1 K - 0,25 W	R17		
2	Résistance 1,5 K - 0,25 W	R5, R6		
5	Résistance 10 K - 0,25 W	R1, R10, R11, R12, R13	MULTICOMP	MCF 0.25W 10K
2	Résistance 33 K - 0,25 W	R3, R7		
4	Résistance 22 K CMS 1206	R2, R8, R9, R14		
2	Résistance 2,2 K CMS 1206	R15, R16	VISHAY	CRCW12062K20FK
2	Condensateur 10 pF	C9, C10	VISHAY	K100J15C0GF53L2
1	Condensateur 100 pF	C8	VISHAY	K101J15C0GF53L2
6	Condensateur 100 nF	C2, C3 C4, C5, C6, C7	AVX	SR201C104KAR
1	Condensateur 1 µF (ou 10 µF - 6,3 V)	C1	AVX	SR215E105MAR
	Coffret et accessoires			
1	Coffret ABS 171 x121 x55		MULTICOMP	G313
1	Commutateur 4 pôles 4PDT On-On		MULTICOMP	1M41T1B1M1QE
3	Commutateur 1 pôle SPST On-Off		MULTICOMP	1MS1T1B5M1QE
2	Bouton poussoir			
1	Fiche verticale 5 mm - 2 contacts			
1	Fiche verticale 3,5 mm - 2 contacts		MULTICOMP	MC000056
2	Fiche verticale 3,5 mm - 8 contacts		MULTICOMP	MC000061

7-1 Microcontrôleur :

Le type de microcontrôleur à utiliser dépend de plusieurs facteurs et a été choisi en fonction des différentes contraintes du système.
Le choix s'est porté sur le **16F886-E/SP** de MICROCHIP :
Il possède 24 ports disponibles pour les entrées/sorties et un bus I²C permettant de s'interconnecter notamment avec un circuit RTCC et une mémoire de type EEPROM.

Entrées / Sorties

1°) – Afficheur – 6 Entrées/Sorties :

6 entrées/sorties pour une commande en 4 bits

2°) – Clavier – 4 Entrées :

- 1 entrée Touche « M » (Menu)
- 1 entrée Touche « <» (diminuer la valeur)
- 1 entrée Touche « > » (augmenter la valeur)
- 1 entrée Touche « V » (validation)

3°) – Moteur « Hauteur » - 4 Entrées/Sorties :

1 entrée pour la détection de la position zéro -**H0** (Hauteur zéro)
1 entrée phototransistor pour la position angulaire – **PH** (Position Hauteur))
1 sortie pour la commande « marche/arrêt » - **SSH** (Start/Stop Hauteur)
1 sortie pour la commande « sens de rotation » - **FRH** (Forward/Reverse Hauteur)

4°) – Moteur « Azimut » - 4 Entrées/Sorties :

1 entrée pour la détection de la position zéro – **A0** (Azimut zéro)
1 entrée phototransistor pour la position angulaire – **PA** (Position Azimut)
1 sortie pour la commande « marche/arrêt » - **SSA** (Start/Stop Azimut)
1 sortie pour la commande « sens de rotation » - **FRA** (Forward/Reverse Azimut)

5°) – Module « RTCC » et mémoire 24AA256 – 2 Entrées/Sorties:

Liaison I²C pour le circuit RTCC MCP79410 et Mémoire 24AA256.

6°) – Sécurité (Alarme) – 2 Entrées :

1 entrée – **ANM** (Anémomètre) Détection vents violents
1 entrée – **SECUR** Mise en sécurité manuelle

Affectation des broches d'entrées/sorties du microcontrôleur :

Figure 7 – 1 Affectation des broches du microcontôleur

Broches	Mnémo	Désignation et Affectation	E/S
RA0	RS	Afficheur 16x2 – Broche 4 - Register selection input	Entrée
RA1	E	Afficheur 16x2 – Broche 6 - Enable signal	Entrée
RA2	DB4	Afficheur 16x2 – Broche 11 - Data bus line	
RA3	DB5	Afficheur 16x2 – Broche 12 - Data bus line	
RA4	DB6	Afficheur 16x2 – Broche 13 - Data bus line	
RA5	DB7	Afficheur 16x2 – Broche 14 - Data bus line	
RA6	T2	Touche clavier "Reculer"	Entrée
RA7	T1	Touche clavier "Menu"	Entrée
RB0	FRA	Forward/Reverse Azimut - Sens de rotation moteur Azimut	Sortie
RB1	SSA	Start/Stop Azimut – Mise en marche et arrêt moteur Azimut	Sortie
RB2	H0	Position Zéro de la Hauteur (correspondant à 16 ou 17°)	Entrée
RB3	SECUR	Mise en sécurité manuelle	Entrée
RB4	FRH	Forward/Reverse Hauteur - Sens de rotation moteur Hauteur	Sortie
RB5	SSH	Start/Stop Hauteur – Mise en marche et arrêt moteur Hauteur	Sortie
RB6	ICSPCLK	Signal Clock pour la programmation du 16F886	
RB7	ICSPDAT	Signal des données pour la programmation du 16F886	
RC0	T3	Touche clavier "Avancer"	Entrée
RC1	T4	Touche clavier "Validation"	Entrée
RC2	ANM	Anémomètre - Alarme vent violent	Entrée
RC3	SCL	Bus I²C – Signal Clock	E/S
RC4	SDA	Bus I²C – Signal des données	E/S
RC5	PH	Position du moteur Hauteur	Entrée
RC6	A0	Position Zéro de l'Azimut	Entrée
RC7	PA	Position du moteur Azimut	Entrée

7-2 Circuit intégré RTCC MICROCHIP MCP79410 :

Ce circuit spécialisé permet de délivrer de façon fiable la date et heure en permanence.
La date et l'heure sont conservées par l'intermédiaire d'une pile de 3V (CR2032) en cas de coupure de courant.
Le circuit gère un calendrier.
Une fois initialisé et paramétré, le circuit fournit :

- L'année
- Un indicateur permettant de savoir si l'année courante est bissextile ou non bissextile.
- Le mois
- Le jour de la semaine
- La date (jour dans le mois)
- L'heure (en format 12h ou 24h)
- Les minutes et les secondes (les secondes ne seront pas utilisées dans le programme)

Le circuit est piloté par un quartz de 32,768 Khz.
Le circuit MCP79410 possède un système de calibration qui permet réduire au maximum la dérive dans le temps.
Ainsi, la dérive annuelle peut être inférieure à quelques dizaines de secondes.
La procédure de calibrage est exposée au paragraphe 29.
Le circuit est relié au microcontrôleur par son bus I²C.

7-3 Mémoire EEPROM MICROCHIP 24AA256 :

La position du soleil fait appel à la trigonométrie sphérique et nous avons eu besoin d'exécuter des calculs faisant intervenir SINUS, COSINUS, ARCSINUS. Pour se faire une table des sinus a été établie et stockée dans cette mémoire EEPROM.
La table établit une correspondance entre les angles de 0° à 90° et la valeur correspondante du SINUS.
Pour une bonne précision des calculs, les angles sont exprimés en degrés et centièmes de degrés et la valeur de leurs SINUS permet une précision de 6 chiffres après la virgule.
Pour exprimer 6 chiffres après la virgule, la partie décimale du SINUS est stockée dans la mémoire EEPROM sur 3 octets. Les angles occupent 9000 zones de mémoire pour pouvoir prendre en compte les 1/100 de degrés (90 x 100).
La zone globale de mémoire nécessaire est donc de 9000 x 3 = 27000 octets (soit 216000 bits).
Le circuit 24AA256 (256000 bits) possède donc une capacité mémoire suffisante pour stocker la table ainsi définie.
Les COSINUS sont calculés à l'aide de la formule suivante : COS (x) = SIN (90-x)
Le circuit est relié au microcontrôleur par son bus I²C.

7-4 AFFICHEUR LCD 2x16 :

Un afficheur LCD 2x16 est utilisé et affiche sur la première ligne la date et l'heure ; et sur la deuxième ligne la hauteur et l'azimut du soleil suivant l'exemple ci-dessous:

```
06/02/2018 18h33
H= -3,3 A= 251,8
```

Nota :

La hauteur du soleil est affichée avec 1 décimale avec sa valeur « positive » ou « négative ».
L'azimut sera affiché avec 1 décimale de 0° à 360° à partir du Nord.
L'afficheur indiquera donc 180° à midi (heure solaire).
L'orientation des panneaux solaires correspond aux valeurs affichées, dans la limite de visibilité du soleil ; c'est-à-dire lorsque celui-ci est au dessus de l'horizon.
La nuit, les calculs continuent à indiquer la position du soleil sur l'afficheur LCD sans que cela ne soit un inconvénient. Bien entendu, les panneaux ne sont plus orientés vers le soleil puisque celui-ci à disparu au-dessous de l'horizon.

L'afficheur est directement géré par le microcontrôleur en mode 4 bits et utilise 6 broches d'entrées/sorties du 16F688.

7-5 CLAVIER 4 touches :

Un clavier de 4 touches a été réalisé avec de simples boutons poussoirs.
Il permet de régler :

1°) la date et l'heure du système.

2°) Les coordonnées GPS de l'installation du suiveur ; Latitude et longitude terrestre.

3°) Le paramétrage du seuil de sécurité pour indiquer au système la vitesse du vent à partir de laquelle les panneaux se mettent en sécurité en position horizontale : Réglage entre 10 km/h et 50 km/h.
Ce paramètre est réglé en fonction des effets du vent sur le support.

4°) Le paramétrage du retour en fonctionnement normal après mise en sécurité : réglage entre 1mn et 59mn
Ceci est le lapse de temps correspondant au retour en position normale après que la vitesse du vent soit redescendue en dessous du seuil défini au précédent réglage. Ceci permet d'éviter au maximum un déplacement intempestif du système de la position normale à la position de sécurité et inversement au cours d'une même journée.

8 - Anémomètre :

Pour pallier au risque de dommage pouvant être causé au suiveur par des vents violents ou par une tempête, un anémomètre (fabrication maison) est installé au sommet de la structure pour mesurer la vitesse du vent.
En cas de vents trop forts, les panneaux vont se placer en « position de sécurité » à l'horizontale pour diminuer au maximum la prise au vent.
L'anémomètre a été réalisé avec du matériel de récupération (axe d'un moteur laser d'imprimante monté sur mini-roulement à billes) et de coquilles semi-sphériques en plastique pour constituer les coupelles.
Un capteur optoélectronique à fourche a été installé pour compter les impulsions générées par un disque à 8 encoches solidaire de l'axe de rotation. La fréquence du signal obtenu est directement proportionnelle à la vitesse du vent.

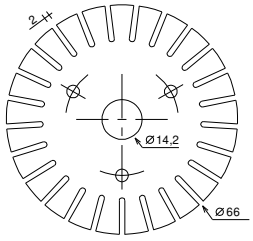
9 - Capteurs optoélectroniques :

Les capteurs de position sont des capteurs à fourche à phototransistor. Ils ont également été récupérés sur des imprimantes laser. Les capteurs récupérés sont à 3 broches :
Broche 1 : Sortie collecteur ouvert (il faut rajouter une résistance 33K reliée au + 5V)
Broche 2 : GND (0V)
Broche 3 : Entrée ; constituée par une diode émettrice polarisée en direct et alimentée de façon à introduire un courant de l'ordre de 3 mA à 5 mA (Il faut rajouter une résistance de 1,5 K en série reliée à la tension d'alimentation de 5 V).

3 capteurs sont utilisés sur le système :

- 1 capteur dans le motoréducteur HAUTEUR.
- 1 capteur dans le motoréducteur AZIMUT.
- 1 capteur pour l'anémomètre.

Figure 7-1 - Disque à 25 encoches pour motoréducteurs HAUTEUR et AZIMUT



IV - CALCUL DE LA POSITION DU SOLEIL :

1 - Généralités

L'objectif est de calculer la position du soleil en fonction de la date et de l'heure.
C'est à dire d'obtenir la valeur des 2 composantes principales qui sont : la hauteur (H) et l'azimut (A) du soleil à n'importe quel moment de l'année et à n'importe quelle heure dans la journée.
Comme dit précédemment, on utilisera une table de correspondance afin de déterminer la valeur d'un SINUS pour un angle exprimé en degrés et centièmes de degrés (précision de la valeur du sinus : 6 chiffres après la virgule).
Cette table sera stockée en mémoire EEPROM 24AA256.

Le cosinus d'un angle pourra être obtenu à partir de la relation : Cos (x) = sin (90-x).

Termes et définitions (toutes les valeurs sont en degrés sauf J et Hutc) :

J : Quantième du jour dans l'année. De 1 à 365 ou 366. 1 pour le 1er janvier
Hutc : Heure UTC
M : Anomalie moyenne
C : Equation du centre
Lon : Longitude écliptique
R : Réduction à l'équateur
Dec : Déclinaison du soleil
Lat : Latitude terrestre (coordonnée GPS)
L : Longitude terrestre (coordonnée GPS)
Ah : Angle horaire
H : Hauteur du soleil
A : Azimut du soleil

Les 2 principales formules de calcul utilisées ont été les suivantes :

Hauteur du soleil

sin H = sin(Lat) sin(Dec) + cos(Lat) cos(Dec) cos(Ah)

d'où H = arcsin (sin(H)) valeur prise dans la table des SINUS

Azimut du soleil

cos A = (cos(Dec) sin(Lat) cos(Ah) - sin(Dec) cos(Lat)) / cos(H)

d'où A = arcsin (sin(90-A)) valeur prise dans la table des SINUS

2 – Quantième du jour de l'année (J)

Il s'agit de la numérotation des jours de l'année à partir du 1er janvier.
Ce nombre compris entre 1 et 366 sera déterminé par le programme à partir de la date.
La date et l'heure sont obtenus par l'intermédiaire du circuit spécialisé RTCC MCP79410.

3 – Anomalie moyenne (M)

M = 357,53 + (360 J / 365,25) soit : **M = 357,53 + (0,9856 J)**

4 – Équation du centre (C)

C = 1,914 sin(M)

5 – Longitude écliptique (Lon)

Lon = 280,5 + (0,9856J) + C

6 – Réduction à l'équateur

R = - 2,468 sin(2 Lon) + 0,053 sin(4 Lon)

7 – Déclinaison du soleil

Dec = 22,8 sin(Lon) + 0,6 $\sin^3$ (Lon)

8 – SINUS de la déclinaison

SIN(Dec) = SIN [22,8 SIN(Lon) + 0,6 SIN^3 (Lon)]

La déclinaison varie de – 23,44° à + 23,44°
Le SINUS de la déclinaison prendre une valeur positive ou négative.

9 – COSINUS de la déclinaison

Il est obtenu à partir de la formule : COS(Dec) = SIN(90-Dec) et la table de correspondance des sinus
La valeur obtenue est toujours positive.

10 – SINUS de la latitude

La valeur de la latitude (Lat) est stockée dans un registre spécifique « latitude ». Cette valeur est paramétrée à la mise en service du suiveur.
Le sinus de la latitude est obtenu par l'intermédiaire de la table des SINUS

11 – COSINUS de la latitude

COS(Lat) = Sin(90-Lat)

12 – Angle horaire

Ah = ((Hutc – 12) x 15) – (C + R) + L

C et R définissent l'équation du centre et la réduction à l'équateur.
La somme de ces 2 valeurs définit l'équation du temps.
Ces valeurs ont été calculées auparavant.

Pour les calculs, l'heure est convertie en décimal

Exemples :

8 H 00	= 8,00	8 H 10	= 8,17
8 H 01	= 8,02	8 H 15	= 8,25
8 H 02	= 8,03	8 H 30	= 8,50

La valeur : Hutc varie de 0 H 00 à 24 H 00
L'angle horaire est exprimé en degrés
Il varie de – 180° à + 180° (sans tenir compte de l'équation du temps)
L'angle est nul à midi vrai (Heure solaire)

Par convention, l'angle est négatif vers l'EST et positif vers l'OUEST.
L est la longitude terrestre (coordonnées GPS) en degrés et centièmes de degrés

Nous avons simplement besoin de connaître : COS (Ah)

1°) – Sauvegarder la valeur du signe de (Ah) dans un registre (REGA)
Cette valeur sauvegardée servira un peu plus tard.

2°) – Tester si la valeur absolue de l'angle horaire est inférieur ou supérieur à 90

a) – Angle < 90
Calculer : 90 – Ah ; puis établir SIN (90-Ah) à partir des tables de valeurs
COS(Ah) = SIN (90-Ah)
Affecté le signe plus "+" à COS(Ah)

b) – Angle > 90
Calculer le complément: Ah – 180
Calculer : 90 moins valeur absolue de (Ah-180)
COS(Ah) = SIN(90-ABS(AH-180)) à partir des tables de valeurs
Affecter le signe moins "-" à COS(Ah)

La valeur de la longitude (L) est stockée dans un registre spécifique « longitude ». Cette valeur est paramétrée à la mise en service du suiveur.
Le sinus de la longitude est obtenu par l'intermédiaire de la table des SINUS

13 – Hauteur du soleil

sin H = sin(Lat) sin(Dec) + cos(Lat) cos(Dec) cos(Ah)

Après calcul de Sin(H), on détermine la valeur de (H) à partir de la table de valeur : Sin(H)

14 – COSINUS hauteur du soleil

COS(H) = SIN (90 – H) obtenu à partir de la table de valeur des Sinus

15 – Azimut du soleil

cos A = (cos(Dec) sin(Lat) cos(Ah) - sin(Dec) cos(Lat)) / cos(H)

La valeur obtenue par la formule peut être positive ou négative.

La valeur est négative lorsque le soleil est à l'EST et varie de -180° à 0°.
La valeur est positive lorsque le soleil est à l'OUEST et varie de 0° à +180°.

Une valeur positive indique un angle d'azimut inférieur à 90° (en valeur absolue).
On peut donc directement obtenir la valeur de l'angle d'azimut en utilisant la formule :
Cos(A) = Sin(90-A) et en affectant à la valeur de « A » trouvée le signe de l'angle horaire.

Une valeur négative indique un angle d'azimut supérieur à 90° (en valeur absolue).
On peut donc directement obtenir la valeur de l'angle d'azimut en utilisant la formule :
Cos(A) = Sin(90-A), en retranchant la valeur trouvée de 180° et en affectant à la valeur finale trouvée le signe de l'angle horaire.
Exemple : Cos(A) = - 0,5 et angle horaire positif
Sin(90-A) = 0,5 soit 90 – A = 30, d'où A = 60
180 – 60 = 120. Valeur de l'azimut : + 120°

Nota :
L'afficheur LCD affichera une valeur comptée à partir du NORD de 0° à 360° (180° au midi solaire).

(Ceci pour correspondre avec les valeurs fournies par les tables d'éphémérides : solartopo.com, SunEarthTools.com, imcce.fr).

16 – Précision des calculs

Les formules ainsi définies ici permettent d'atteindre une précision de l'ordre de 0,2°.
Cette précision a pu être évaluée par comparaison avec les valeurs fournies par les différents instituts de calculs des éphémérides (solartopo.com, SunEarthTools.com, imcce.fr).

VI - FONCTIONNEMENT:

1 - Introduction :

Le programme devra essentiellement calculer l'angle d'azimut et l'angle de la hauteur du soleil selon les formules précisées aux paragraphes précédents et devra commander les 2 moteurs de positionnement.
Les moteurs AZIMUT et HAUTEUR peuvent être commandés par intervalles de 1°
Le module RTCC (circuit intégré MCP79410) permettra de connaître en permanence la date et l'heure.
La date permettra de déterminer la valeur du paramètre « J » (Quantième du jour dans l'année)
L'heure, bien entendu, est indispensable pour effectuer les calculs de positionnement.
L'afficheur LCD 2x16 affichera sur la première ligne la date et l'heure ; et sur la deuxième ligne la hauteur et l'azimut.

2 - Déroulement des opérations :

1 ère Phase :

Lors de la mise sous tension du système, le suiveur doit s'initialiser à la position : AZIMUT : Position plein SUD et HAUTEUR : « minimale » avant que les panneaux ne se positionnent perpendiculairement par rapport aux rayons du soleil.
En effet, le système tel qu'il est construit ne possède pas de capteurs de position (de type codeur de rotation) et le positionnement est établi par comptage d'impulsions à partir d'une position d'origine.

Initialisation pour la hauteur :
Par construction, le système ne peut pas orienter les panneaux verticalement mais seulement avec un angle d'environ 18° par rapport à la verticale
Si le suiveur n'est pas déjà sur sa position minimale (soit environ 18°), une commande de mouvement du moteur « Hauteur » dans le sens de la descente sera initiée pour que le système atteigne sa position minimale.
Un contact « position minimale » sera actionné lors de l'initialisation.

Initialisation pour l'azimut :
La position 0 de l'azimut correspond au plein SUD (ou midi vrai). Un capteur permettra de détecter cette position.
Pour atteindre cette position, le système va donc se déplacer en azimut vers l'EST ou vers l'OUEST en fonction de la position qu'il occupera juste avant la commande d'initialisation.
Le capteur sera simplement constitué d'un « microswitch » actionné par une came solidaire de l'axe de rotation en azimut. Lorsque le suiveur sera positionné vers l'OUEST, le contact sera « fermé » (valeur logique 0).
Lorsque le suiveur sera orienté vers l'EST, le contact sera « ouvert » (valeur logique 1).
Lors de l'initialisation en azimut, un mouvement du moteur « AZIMUT » sera initié vers l'EST si le « microswitch » est fermé (valeur logique 0). Il sera initié vers l'OUEST si le « microswitch » est relâché (valeur logique 1).
L'arrêt moteur se fera lorsque le « microswitch » changera d'état.
Pour une position précise de la position « ZERO », cette position devra être atteinte toujours de la même manière ; c'est à dire lorsque le contact de position « ZERO » se fermera (passage de la valeur logique 1 à la valeur logique 0).
Pour l'initialisation avec déplacement d'EST en OUEST, aucun problème. La position « ZERO » sera atteinte directement car le contact passera directement de la valeur logique « 1 » à la valeur logique « 0 ».
Pour l'initialisation avec déplacement d'OUEST en EST, le moteur « azimut » s'arrêtera lorsque le contact sera relâché. A cet instant, et après quelques dixièmes de secondes, un déplacement en sens inverse devra être

établi, puis un arrêt programmé lorsque le contact sera à nouveau actionné (passage de la valeur logique 1 à la valeur logique 0).
Nota :
Lorsque le système sera déjà en position « ZERO », une nouvelle initialisation engendrera donc une mise en marche du moteur « Azimut » vers l'EST (valeur logique 0 du contact « azimut ») pendant un court instant (jusqu'à ce que le contact de position zéro soit relâché). Après une courte temporisation le moteur « Azimut » se remettra en marche en sens inverse pour s'arrêter en position « ZERO » lorsque le contact sera à nouveau « fermé ».

2ème Phase :

Le système pourra ensuite commencer les calculs. Seules les minutes seront prises en compte pour les calculs. En effet ce laps de temps sera suffisant pour suivre le soleil. L'astre se « déplace » à raison de 0,25° par minute pour sa vitesse moyenne et de 0,65° / mn à sa vitesse maximum au solstice d'été vers midi.
Les calculs seront effectués en boucle. Et la valeur trouvée pour l'azimut et la hauteur sera donc susceptible de changer à chaque minute.
Juste après l'initialisation, les moteurs AZIMUT et HAUTEUR vont être actionnés chacun à leur tour pour se positionner en fonction de la valeur calculée par le programme.
Il suffit d'envoyer à chacun des moteurs un ordre de rotation pour qu'ils puissent atteindre la valeur de consigne.
Les moteurs peuvent être commandés par pas de 1°. Un disque de 25 encoches solidaire de l'axe de sortie des motoréducteurs permet de compter les incrémentations degré par degré
Chaque moteur est commandé par un niveau bas sur la broche S/S (Start/stop) de chacun des moteurs et un niveau approprié sur la broche F/R (Forward/Reverse) pour le sens de rotation.
La position en azimut et en hauteur sera sauvegardée dans un registre et permettra le déplacement ou non du suiveur une minute plus tard.

3ème Phase :

Une minute plus tard : nouvelles valeurs calculées.
La position réelle du suiveur correspond à un nombre entier de degrés
La position calculée (en dixièmes de degrés) est convertie en un nombre entier de degrés.
Si la position calculée est identique à la valeur précédente aucun mouvement ne sera initié.
Si la position calculée diffère de la précédente, un déplacement sera initié pour l'un ou l'autre des moteurs (ou bien des 2) dans le sens approprié et correspondant à la valeur calculées.

4ème Phase :

Coucher du soleil
Lorsque les panneaux vont atteindre la valeur minimale d'inclinaison (18°) par rapport à la verticale, le mouvement en hauteur des panneaux va être interrompu. Pendant quelques temps encore, le système va continuer de se déplacer en azimut vers l'ouest tant que le soleil ne sera pas encore couché.
Lorsque le soleil va atteindre l'horizon (Hauteur du soleil : 0°), le programme à cet instant va positionner les panneaux en hauteur à 88° (mise en sécurité pour la nuit) et le mouvement en azimut va être interrompu.
Le suiveur va rester à cette position tant que la hauteur restera en dessous de 0° ; c'est-à-dire jusqu'au lendemain matin.
Le lendemain matin, lorsque la hauteur deviendra à nouveau positive, le programme va exécuté la séquence d'initialisation, puis les panneaux vont ensuite se repositionner automatiquement.

3 - Sécurité :

Au coucher du soleil, les panneaux seront placer horizontalement (ou à 88°par rapport à la verticale) et ceci toute la nuit.
Une mise en sécurité du système en cas de vents violents consistera à placer les panneaux en position horizontale (Angle de hauteur à 90°) ou légèrement en dessous (88° par exemple pour favoriser l'évacuation de l'eau de pluie. Le système sera équipé d'un anémomètre et le programme mesurera en permanence la vitesse du vent.
2 paramètres seront stockés en mémoire EEPROM
Il s'agit de :

1°) – Seuil de mise en sécurité du système :
Ce paramètre correspond à la vitesse du vent au-delà de laquelle le système risque de subir de fortes contraintes mécaniques surtout lorsque les panneaux seront placés verticalement.
La valeur de réglage minimale du paramètre sera de 10. La valeur maximale sera de 50
La valeur par défaut sera de 25 (correspondant à 25 Km/h)

2°) – Durée de temporisation de mise en sécurité :
Lorsque le système se positionnera en sécurité, il restera dans cette position une durée minimale ; ceci afin d'éviter un risque de retour trop rapide en fonctionnement normal puis à nouveau en condition de mise en sécurité selon les caprices du vent.
Cette temporisation sera réglée à environ 30 mn et pourra ensuite être modifiée suivant le cas par le programme de réglage des paramètres (valeur de réglage de ce paramètre : entre 2 et 59 mn).

4 - Contraintes mécaniques - Inertie:

Malgré la démultiplication assurée par les réducteurs et pour permettre un positionnement précis, les 2 moteurs seront freinés lors de l'arrêt pour vaincre l'inertie de la partie mécanique en mouvement. Sans freinage, le système risque de se positionner au delà de la position programmée. Pour se faire, il suffit d'introduire lors de chaque arrêt une commande en sens inverse de la rotation pendant une durée de 60 à 100 ms et après un intervalle de temps de 0,5 s. Après quelques tests, il s'avère que la durée de rotation en sens inverse de 60 ms est appropriée pour le moteur « Azimut » et 100 ms pour le moteur « Hauteur ». La durée de 0,5 s permet d'arrêter les moteurs afin que le faisceau lumineux des capteurs à fourche se situe à équidistance entre 2 encoches.

VI - LE PROGRAMME:

1 – Initialisation du 16F886 :

```
        ERRORLEVEL -302
        ERRORLEVEL -305

        LIST      p=16F886            ; Définition de processeur
        #include <p16F886.inc>        ; fichier include

        __CONFIG    _CONFIG1, _LVP_OFF & _FCMEN_ON & _IESO_OFF & _BOR_OFF & _CPD_OFF & _CP_OFF &
_MCLRE_OFF & _PWRTE_ON & _WDT_OFF & _INTOSCIO
        __CONFIG    _CONFIG2, _WRT_OFF & _BOR21V

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

; REGISTRE OPTION_REG (configuration)
; -----------------------------------
OPTIONVAL       EQU     B'10000111'

; REGISTRE INTCON (contrôle interruptions standard)
; -------------------------------------------------
INTCONVAL       EQU     B'00000000'

; REGISTRE PIE1 (contrôle interruptions périphériques)
; ----------------------------------------------------
PIE1VAL         EQU     B'00000000'

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


; REGISTRE OSCCON (contrôle de la vitesse de l'oscillateur interne)
; -----------------------------------------------------------------
OSCCONVAL       EQU     B'01100000'    ; (4 MHz)

; REGISTRE CMCON (COMPARATEURS)
; -----------------------------------
CMCONVAL        EQU     B'00000111'

; REGISTRE VRCON (voltage reference module)
; -----------------------------------
CVRCONVAL       EQU     B'00000000'

; REGISTRE ADCON1 (ANALOGIQUE/DIGITAL)
; -----------------------------------
ADCON1VAL       EQU     B'00000000' ; PORTA en mode digital

; REGISTRE ANSEL (contrôle du convertisseur A/D)
; ----------------------------------------------
ANSELVAL        EQU     B'00000000'
ANSELHVAL       EQU     B'00000000'

; DIRECTION DES PORTS I/O (0=sortie, 1=entrée)
; --------------------------

DIRPORTA        EQU     B'11000000'    ; Direction PORTA
DIRPORTB        EQU     B'00001100'    ; Direction PORTB
DIRPORTC        EQU     B'11100111'    ; Direction PORTC

; REGISTRE WPUB (Résistances de rappel du PORTB)
; ----------------------------------------------
WPUBVAL         EQU     B'00000000'    ; Résistances de rappel non validées


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

LCD_FUNCTION_SET                    EQU     B'00101000'
LCD_CURSOR_DISPLAY_SHIFT            EQU     B'00011100'
LCD_DISPLAY_CONTROL                 EQU     B'00001100'
LCD_ENTRY_MODE_SET                  EQU     B'00000110'
```

```
ADRS_RTCC                           EQU     B'11011110'    ; Microchip MCP 79410
ADRS_EEPROM                         EQU     B'10100000'    ; Microchip 24AA256


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

#define CARRY           STATUS,C
#define ZERO            STATUS,Z

#define LCD_PORT        PORTA
#define LCD_E           PORTA,1         ; LCD: Ligne de commande de contrôle de l'afficheur
#define LCD_RS          PORTA,0         ; LCD: Ligne de sélection de l'afficheur
#define LCD_D4          2
#define LCD_D5          3
#define LCD_D6          4
#define LCD_D7          5

#define SCL             PORTC,3         ; Serial clock
#define SDA             PORTC,4         ; Serial data (bidir)
#define SDA_DIR         TRISC,4         ; Serial data dir (E/S)

#define BISSEXTILE      REG_TRACK,2

#define SNEGx           REG_MATH , 7
#define SNEGy           REG_MATH , 6
#define SNEGs           REG_MATH , 5
#define SNEGas          REG_MATH,4
#define SNEGaz          REG_MATH,3
#define INTFLAG         REG_MATH,2

#define SNEG            REG_LCD,7

#define BP_M            PORTA,7         ; Bouton poussoir "Menu"
#define BP_AR           PORTA,6         ; Bouton poussoir défilement vers l'arrière
#define BP_AV           PORTC,0         ; Bouton poussoir "défilement vers l'avant
#define BP_VAL          PORTC,1         ; Bouton poussoir "Validation"

#DEFINE SSH             PORTB,5         ; Start/Stop du moteur HAUTEUR
#DEFINE FRH             PORTB,4         ; Forward/Reverse (Sens de rotation) du moteur HAUTEUR
#DEFINE PH              PORTC,5         ; Capteur de position du moteur HAUTEUR
#define H0              PORTB,2         ; Capteur HAUTEUR minimale (17°)
#DEFINE SSA             PORTB,1         ; Start/Stop du moteur AZIMUT
#DEFINE FRA             PORTB,0         ; Forward/Reverse (Sens de rotation) du moteur AZIMUT
#DEFINE PA              PORTC,7         ; Capteur de position du moteur AZIMUT
#DEFINE A0              PORTC,6         ; Capteur AZIMUT 0 (midi)
#DEFINE ANM             PORTC,2         ; Anémomètre. Alarme (Vents forts)
#DEFINE SECUR           PORTB,3         ; Mise en sécurité manuelle


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

        #include <MACRO.asm>

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

; Zone de 16 bytes  (16)
; ----------------

        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)

        x:6 , y:3 , z:3

        ENDC

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

; Zone de 80 bytes  (80 bytes déclarés et utilisés)
; -------------------------------------
```

```
        CBLOCK  0x20            ; Début de la zone (0x20 à 0x6F)

        seconde , minute , heure , jour , date , mois , annee
        REG_TRACK, REG_MATH , BitCompte , OctCompte , Position
        temp1, temp2, temp3, cmpt0, cmpt1, dchange
        REG_LCD , LCD_DATA , LCD_PORT_SAVE
        t:9
        Cadran:3 , Angle:3
        ArcA:3 ,ArcB:3 , ArcR:3
        Calcul1:3 , Calcul2:3 , Calcul3:3 , MoisAnnee
        VitessVent:3, latitude:2, longitude:3, vlimit, tlimit
        RegHaut:3  , RegAzim:3
        PreelH:3 , PreelA:3
        I2C_DONNEE, I2C_flag, I

        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
        goto    init            ; Initialiser

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

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

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

;*****************************************************************************
;                      ROUTINE INTERRUPTION                                  *
;*****************************************************************************

        org 0x004                       ; adresse d'interruption

                        ;sauvegarder registres
                        ;----------------------
;       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                  ; passer en page 0
;       BANK0                           ; passer en banque 0
```

```
                        ; switch vers différentes interrupts
                        ; inverser ordre pour modifier priorités
                        ; mais attention alors au test PEIE
                        ; effacer les inutiles
                        ;---------------------------------------

                        ; Interruption TMR0
                        ; -----------------

;       btfsc   INTCON,TMR0IE           ; tester si interrupt timer autorisée
;       btfss   INTCON,TMR0IF           ; oui, tester si interrupt timer en cours
;       goto    restorereg              ; non test suivant
;       CALL    inttmr0                 ; oui, traiter interrupt tmr0
;       bcf     INTCON,TMR0IF           ; effacer flag interrupt tmr0
;       goto    restorereg              ; et fin d'interruption
                                ; SUPPRIMER CETTE LIGNE POUR
                                ; TRAITER PLUSIEURS INTERRUPT
                                ; EN 1 SEULE FOIS

                        ;restaurer registres
                        ;-------------------
;restorereg
;       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

;*****************************************************************************
;                         INTERRUPTION TIMER 0                                *
;*****************************************************************************
;inttmr0
;       return

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

;                           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

        BANK1
        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   WPUBVAL         ; charger le masque
        movwf   WPUB            ; initialiser registre

                        ; Registre de l'oscillateur
                        ; --------------------------
        movlw   OSCCONVAL               ; charger le masque
        movwf   OSCCON                  ; initialiser registre

                        ; 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

                        ; registres A/D (banque 3)
                        ; ---------------------------------
        BANK3
        movlw   ANSELVAL
        movwf   ANSEL
        movlw   ANSELHVAL
        movwf   ANSELH

                        ; Effacer RAM banque 0
                        ; ---------------------
        BANK0
        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

;  Commande des moteurs

;       bcf             FRA                     ; Sens du déplacement vers l'OUEST
;       bsf             FRA                     ; Sens du déplacement vers l'EST
;       bcf             SSA                     ; Marche du moteur AZIMUT
;       bsf             SSA                     ; Arrêt du moteur AZIMUT

;       bcf             FRH                     ; Sens du déplacement vers le BAS
;       bsf             FRH                     ; Sens du déplacement vers le HAUT
;       bcf             SSH                     ; Marche du moteur HAUTEUR
;       bsf             SSH                     ; Arrêt du moteur HAUTEUR
```

2 – Les macros :

Le programme utlisent des macros. Celles-ci sont stockées dans le fichier : « MACRO.asm »
Ce fichier est appelé lors de l'initialisation par l'instruction : `#include <MACRO.asm>`

```
Liste des macros

                        ; Changement de banques
                        ; ----------------------

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

                        ; 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
        ENDIF
        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

                        ; opérations sur BUS I2C
                        ; ----------------------

SDA_IN  macro                           ;BUS I2C
        bsf     STATUS,RP0              ;bank1
        bsf     SDA_DIR                 ;SDA en entrée
        bcf     STATUS,RP0              ;bank0
        endm

SDA_OUT macro                           ;BUS I2C
        bsf     STATUS,RP0              ;bank1
        bcf     SDA_DIR                 ;SDA en sortie
        bcf     STATUS,RP0              ;bank0
        endm

                        ; opérations sur x, y et z (3 octets)
                        ; -----------------------------------

movlx   macro   lx2,lx1,lx0     ;Valeur littérale vers x
        movlw   lx2
        movwf   x+2
        movlw   lx1
        movwf   x+1
        movlw   lx0
        movwf   x+0
        endm

movly   macro   ly2,ly1,ly0     ; Valeur littérale vers y
        movlw   ly2
        movwf   y+2
        movlw   ly1
        movwf   y+1
        movlw   ly0
        movwf   y+0
        endm

movlz   macro   lz2,lz1,lz0     ; Valeur littérale vers z
        movlw   lz2
        movwf   z+2
        movlw   lz1
        movwf   z+1
        movlw   lz0
        movwf   z+0
        endm

movfx   macro   fx              ; Registre fx transféré dans x
        movfw   fx+2
        movwf   x+2
        movfw   fx+1
        movwf   x+1
        movfw   fx+0
        movwf   x+0
        endm

movfy   macro   fy              ; Registre fy transféré dans y
        movfw   fy+2
        movwf   y+2
        movfw   fy+1
        movwf   y+1
        movfw   fy+0
        movwf   y+0
        endm

movfz   macro   fz              ; Registre fz transféré dans z
        movfw   fz+2
        movwf   z+2
        movfw   fz+1
        movwf   z+1
        movfw   fz+0
        movwf   z+0
        endm

movxf   macro   xf              ; Registre xf transféré dans x
        movfw   x+2
        movwf   xf+2
```

```
        movfw   x+1
        movwf   xf+1
        movfw   x+0
        movwf   xf+0
        endm

movyf   macro   yf              ; Registre yf transféré dans y
        movfw   y+2
        movwf   yf+2
        movfw   y+1
        movwf   yf+1
        movfw   y+0
        movwf   yf+0
        endm

movzf   macro   zf              ; Registre zf transféré dans z
        movfw   z+2
        movwf   zf+2
        movfw   z+1
        movwf   zf+1
        movfw   z+0
        movwf   zf+0
        endm
```

3 – Le programme principal :

Le programme principal exécute séquentiellement un ensemble de sous-programmes permettant de réaliser chacune des fonctions nécessaires au bon déroulement des opérations. La liste de ces sous-programmes est donnée au paragraphe suivant.

Outre ces sous-programmes, le programme principal comporte :

1°) Une routine permettant la modification des paramètres :

a) Coordonnées GPS (Latitude et longitude) du suiveur.
b) Paramètres de sécurité (vitesse du vent et délai de positionnement en sécurité)
c) Mise à jour de la date et mise à l'heure solaire.

L'entrée dans la routine de paramétrage se fait en appuyant pendant au moins 3,5 s sur la touche « M ».

2°) Une routine permettant de surveiller la vitesse du vent pour permettre la mise en sécurité du système en cas de vents violents.

3°) Une routine permettant de contrôler la hauteur minimale de 18° pour l'inclinaison des panneaux et interdire un déplacement vertical au-delà de cette limite.

4 – Le registre REG_TRACK :

Ce registre a été défini pour facilité la programmation.
Il permet de stocker différentes valeurs importantes.
Reflet de l'état du système, il permet au programme principal (et plusieurs sous-programmes) de connaître rapidement l'information importante dont il a besoin pour fonctionner.

Bit 0 : Etat du tracker (en fonctionnement ou en veille)
Bit 0 = 0 : Tracker en fonctionnement - Hauteur du soleil supérieure à 0°
Bit 0 = 1 : Tracker en veille - Hauteur du soleil inférieure à 0°

Bit 1 : Heure d'hiver ou heure d'été
Bit 1 = 0 : Heure d'hiver
Bit 1 = 1 : Heure d'été

Bit 2 : Année bissextile
Bit 2 = 0 : L'année n'est pas bissextile
Bit 2 =1 : L'année est bissextile

Bit 3 : Détection alerte vents forts
Bit 3 =0 : Pas de vent ou vents faibles et modérés
Bit 3 =1 : Vents forts

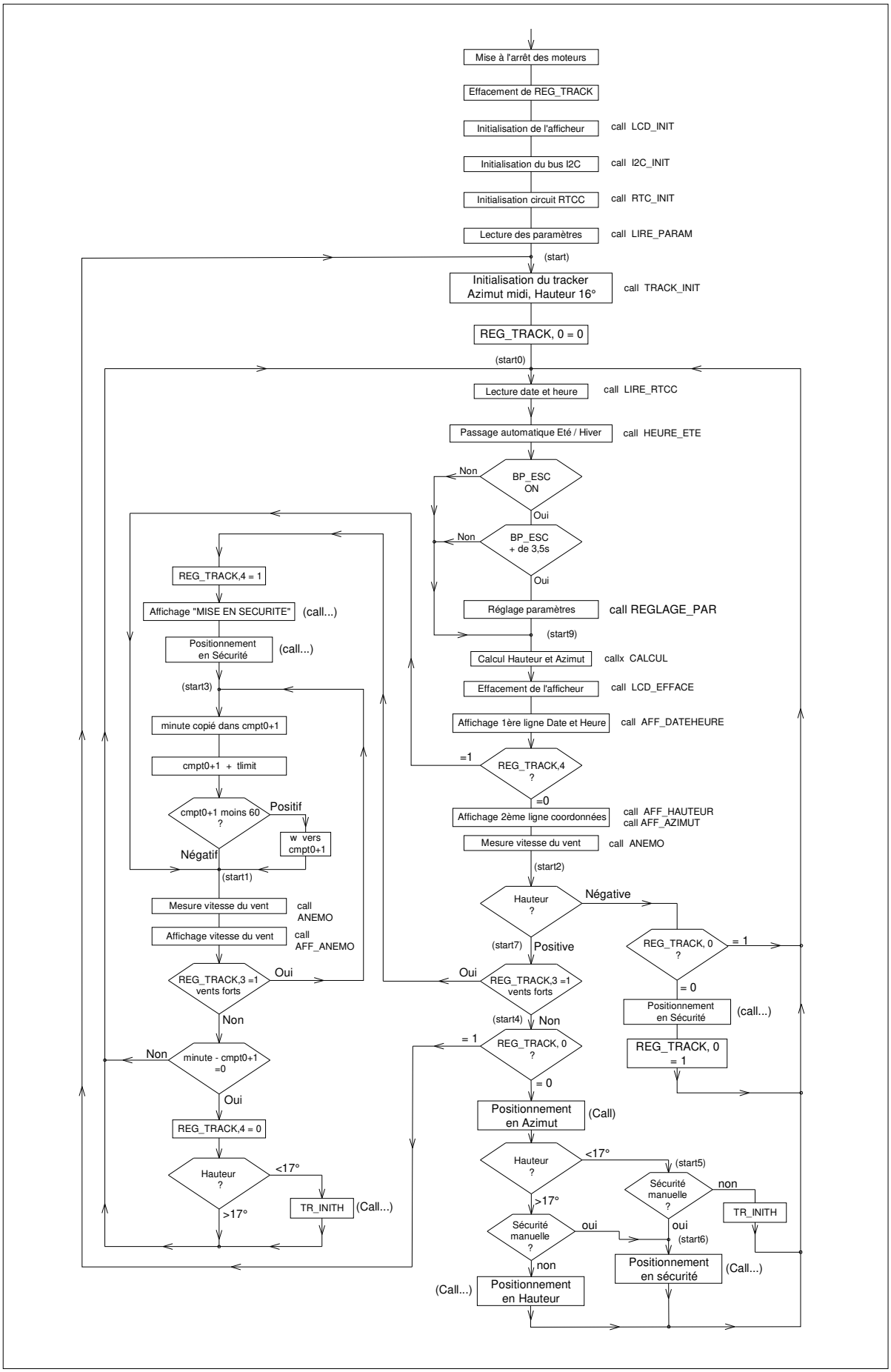
Bit 4 : Position des panneaux en sécurité
Bit 4 = 0 : Tracker en fonctionnement normal
Bit 4 = 1 : Position sécurisée. Panneaux placés horizontalement (à 88°)

Bit 5 : Non utilisé

Bit 6 : Non utilisé

Bit 7 : Non utilisé

Figure 3 – 1 Programme principal



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

        bsf     SSH             ; Moteur HAUTEUR arrêté
        bsf     SSA             ; Moteur AZIMUT arrêté
        clrf    REG_TRACK       ; Effacement du registre REG_TRACK
        call    LCD_INIT        ; Initialisation de l'afficheur LCD
        call    I2C_INIT        ; Initialisation du bus I2C
        call    RTC_INIT        ; Initialisation du circuit RTCC (Calibration)
        call    LIRE_PARAM      ; Lecture en EEPROM des coordonnées et paramètres

start
        call    TRACK_INIT              ; Initialisation physique Azimut 0, Hauteur 16°
        bcf     REG_TRACK,0             ; REG_TRACK,0 = 0

start0
        call    LIRE_RTCC               ; Lecture du circuit RTCC
        call    HEURE_ETE               ; Réglage automatique Heure été/hiver
        btfsc   BP_M                    ; Réglage paramètres du système
        goto    start9                  ;       | La routine est activée après un appui
        call    Tempo_2.5s              ;       | de 3,5 s sur la touche "M"
        call    Tempo_1s
        btfsc   BP_M                    ; Réglage paramètres du système
        goto    start9                  ;
        call    REGLAGE_PAR             ; Réglage des paramètres

start9
        CALLX   CALCUL                  ; Calculs pour la HAUTEUR et l'AZIMUT
        call    LCD_EFFACE              ; Effacement de l'afficheur
        call    AFF_DATEHEURE           ; Affichage DATE ET HEURE sur la 1ère ligne
        btfsc   REG_TRACK,4             ; Test si flag "vents forts" positionné
        goto    start1
        call    AFF_HAUTEUR             ; Affichage de la HAUTEUR sur la 2ème ligne
        call    AFF_AZIMUT              ; Affichage de l'AZIMUT sur la 2ème ligne
        call    ANEMO                   ; Mesure de la vitesse de rotation de l'anémomètre
        goto    start2                  ;
start7
        btfss   REG_TRACK,3             ;
        goto    start8                  ; Valeur 0: vers positionnement du système
        bsf     REG_TRACK,4             ; Valeur 1: Flag vents forts positionné
        call    AFF_SECURITE            ; Affichage "MISE EN SECURITE" sur la 2ème ligne
        call    Tempo_1s
        call    PositionnementS         ; Déplacement des panneaux vers position sécurisée
start3
        movf    minute,0                ; Lecture du registre 'minute'
        movwf   cmpt0+1                 ; minute est copié dans cmpt0+1
        movf    tlimit,0                ; Tempo de la position sécurisées (mn)
        addwf   cmpt0+1,1               ;
        movlw   d'60'                   ;
        subwf   cmpt0+1,0               ;
        btfsc   CARRY                   ;
        movwf   cmpt0+1                 ;
start1
        call    ANEMO                   ; Mesure de la vitesse de rotation de l'anémomètre
        call    LCD_EFFACE              ; Effacement de l'afficheur
        call    AFF_DATEHEURE           ; Affichage DATE ET HEURE sur la 1ère ligne
        call    AFF_ANEMO               ; Affiche la vitesse du vent en Km/h sur la 2ème ligne
        btfsc   REG_TRACK,3             ; Test si 'vents forts' (si oui ligne suivante)
        goto    start3                  ; Pour réinitialiser la tempo de mise en sécurité.
        movf    minute,0                ;
        subwf   cmpt0+1,0               ;
        btfss   ZERO                    ; Test fin de temposisation tlimit
        goto    start0                  ;
        movly   d'0',d'0',d'170'        ; Valeur 170 (pour 17°) dans "y"
        comf    y+2                     ; Signe négatif pour comparaison
        comf    y+1
        comf    y+0
        incf    y+0
        skpnz
        incf    y+1
        skpnz
        incf    y+2
        movfx   RegHaut                 ; la valeur de RegHaut est transférée dans "x"
        CALLX   FXA_2424S               ; Valeur de "RegHaut" - (moins) 170
        btfss   x+2,7                   ; Test si valeur positive ou négative du résultat
        goto    start4                  ; Hauteur > 17°
        call    TR_INITH                ; Hauteur < 17°
```

```
start4
        bcf     REG_TRACK,4             ;
        goto    start0                  ;

start2
        movfx   RegHaut                 ;
        btfss   x+2,7                   ; Test si "Hauteur" positive ou négative
        goto    start7                  ; La hauteur est positive;
        btfsc   REG_TRACK,0             ; La "Hauteur" est négative; test de "REG_TRACK,0"
        goto    start0                  ; REG_TRACK,0=1: vers début du programme principal
        call    PositionnementS         ; REG_TRACK,0=0: Positionnement en sécurité durant la nuit
        bsf     REG_TRACK , 0           ; Mise à la valeur 1 de REG_TRACK,0
        goto    start0                  ; Bouclage jusqu'à Hauteur à nouveau positive

start8
        btfss   REG_TRACK,0             ; Test de la valeur de "REG_TRACK,0"(Hauteur positive)
        goto    $+2                     ; REG_TRACK,0=0: Vers Positionnement du système
        goto    start                   ; REG_TRACK,0=1: Retour au début du programme
        call    PositionnementA         ; Positionnement du tracker en AZIMUT
        movly   d'0',d'0',d'170'        ; Valeur 170 (pour 17°) dans "y"
        comf    y+2                     ; Signe négatif pour comparaison
        comf    y+1
        comf    y+0
        incf    y+0
        skpnz
        incf    y+1
        skpnz
        incf    y+2
        movfx   RegHaut                 ; la valeur de RegHaut est transférée dans "x"
        CALLX   FXA_2424S               ; Valeur de "RegHaut" - (moins) 170
        btfsc   x+2,7                   ; Test si valeur positive ou négative du résultat
        goto    start5                  ; Hauteur<17°
        btfss   SECUR                   ; Hauteur>17°: Positionnement en sécurité (manuelle)?
        goto    start6                  ; Oui: vers positionnement en sécurité (avec hauteur>17°)
        call    PositionnementH         ; Non: vers Positionnement en hauteur
        goto    start0                  ;

start5
        btfss   SECUR                   ; Hauteur<17°: Positionnement en sécurité (manuelle)?
        goto    start6                  ; Oui: vers positionnement en sécurité (avec hauteur<17°)
        call    TR_INITH                ; Non: Initialisation à la hauteur: 16°
        goto    start0                  ;

start6
        call    PositionnementS         ; Positionnement en sécurité
        goto    start0                  ;
```

5 – Les sous-programmes :

	Sous-Programme	Type	Description	Sous-programmes utilisés	Registres utilisés	Bank	Nbr inst
1	FXA_2424S	Fonction mathématique	Addition de 2 nombres en 24 bits signés.		x, y	1	17
2	FXM_2424U	Fonction mathématique	Multiplication de 2 nombres 24 bits non signés		x, y, BitCompte	1	29
3	FXM_2424S	Fonction mathématique	Multiplication de 2 nombres 24 bits avec signes	FXM_2424U	x, y	1	39
4	FXD_2424U	Fonction mathématique	Division de 2 nombres 24 bits non signés		x, y, z, t, BitCompte	1	55
5	FXD_2424S	Fonction mathématique	Division de 2 nombres 24 bits avec signes	FXD_2424U	x, y	1	39
6	FXD_2424SA	Fonction mathématique	Division de 2 nombres 24 bits avec signes et avec arrondi	FXD_2424U FX_ARRONDI	x, y, t	1	41
7	FX_ARRONDI	Fonction mathématique	Fonction arrondi (utilisé après une division)	FXM_2424U FXD_2424U FXA_2424S	x, t, z	1	16
8	Tempo_2,5s	Temporisation	Temporisations : 350 ms, 500 ms, 1 s et 2,5 s	Tempo_10ms	temp3	0	16
9	Tempo_200ms	Temporisation	Temporisations : 10 ms, 30 ms, 100 ms, 150 ms et 200 ms	Tempo_1ms	temp2	0	18
10	Tempo_1530us	Temporisation	Temporisation 1530 µs	Tempo_1ms Tempo_530ms	temp1	0	3
11	Tempo_1ms	Temporisation	Temporisations : 250 µs, 530 µs, 1 ms		temp1	0	13
12	Tempo_43us	Temporisation	Temporisations : 11 µs, 39 µs, 43 µs		temp1	0	15
13	LCD_INIT	Affichage LCD	Initialisation de l'afficheur LCD 16x2	Tempo_30ms LCD_CONTROLE LCD_EFFACE		0	14
14	LCD_CONTROLE	Affichage LCD	Envoi d'une commande vers l'afficheur	LCD_4B	LCD_DATA	0	6
15	LCD_DONNEE	Affichage LCD	Envoi d'une donnée vers l'afficheur	LCD_4B	LCD_DATA	0	6
16	LCD_LOCATE	Affichage LCD	Positionnement de la donnée à écrire sur l'afficheur	LCD_CONTROLE		0	3
17	LCD_EFFACE	Affichage LCD	Effacement de l'afficheur	LCD_CONTROLE Tempo_1530us		0	4
18	LCD_4B	Affichage LCD	Transfert d'un ½ octet vers l'afficheur (mode 4 bits).	Tempo_43us	LCD_E, LCD_PORT, LCD_PORT_SAVE	0	20
19	I2C_INIT	BUS I2C	Initialisation du bus I2C			0	6
20	I2C_START_	BUS I2C	Envoi d'un start condition			0	4
21	IC2_STOP_	BUS I2C	Envoi d'un stop condition			0	4
22	I2C_SLAVE_ACK	BUS I2C		Tempo_11us NO_ACK		0	15
23	I2C_MASTER_ACK	BUS I2C		Tempo_11us		0	11
24	I2C_MASTER_NACK	BUS I2C		Tempo_11us		0	11
25	IC2_SEND_BYTE	BUS I2C	Envoi d'un octet	Tempo_11us I2C_SLAVE_ACK	I2C_DONNEE, I	0	20
26	IC2_RECEIVE_BYTE	BUS I2C	Réception d'un octet	Tempo_11us I2C_MASTER_ACK	I2C_DONNEE, I	0	25
27	IC2_RECEIVE_BYTE_LAST	BUS I2C	Réception dernier octet	Tempo_11us I2C_MASTER_NACK	I2C_DONNEE, I	0	23
28	NO_ACK	BUS I2C			IC2_flag	0	2
29	RTC_INIT	Horloge et calendrier	Envoi du paramètre de calibration du circuit RTCC. Valeur d'83' à l'adresse : h'08'	I2C_START_ I2C_SEND_BYTE I2C_STOP_		0	9
30	LIRE_PARAM		Lecture des paramètres à partir de la mémoire 24AA256			0	
31	LIRE_RTCC	Horloge et calendrier	Transfert les données du circuit RTCC dans différents registres de travail : minute, heure, date, mois, année	I2C_START_ I2C_SEND_BYTE I2C_RECEIVE_BYTE I2C_Receive_byte_last I2C_STOP_ Conv_HD_DateHeure	seconde, minute, heure, date, mois, annee, ADRS_RTCC	0	29

32	Conv_HD	Conversion de code	Conversion d'un codage binaire en codage BCD	FXM_2424S	x, t	1	21
33	Conv_DH	Conversion de code	Conversion d'un codage BCD en codage binaire	FXD_2424S	x, z	1	13
34	Conv_HD_DateHeure	Conversion de code	Conversion des registres seconde, minute, heure, date, mois, annee codés en binaire classique vers un codage en BCD	Conv_HD	Seconde, minute, heure, date, mois, annee	1	19
35	Conv_DH_DateHeure	Conversion de code	Conversion des registres seconde, minute, heure, date, mois, annee codés en BCD vers un codage en binaire classique.	Conv_DH	Seconde, minute, heure, date, mois, annee	1	19
36	LIRE_SINUS	Fonctions trigonométriques	Lecture d'un sinus dans la table des sinus (mémoire EEPROM MCP79410)	FXM_2424S I2C_START_ I2C_SEND_BYTE I2C_STOP_	x, y	0	24
37	Sinus	Fonction mathématique	Établi la valeur d'un sinus à partir d'un angle inférieur ou supérieur à 90°, positif ou négatif	FXA_2424S LIRE_SINUS	x, y, Cadran, Angle	1	167
38	Arcsinus	Fonction mathématique	Établi la valeur d'un angle à partir de la valeur de son sinus.	FXA_2424S FXD_2424SA Sinus	Registre utilisés à l'origine à redéfinir.	1	125
39	Cosinus	Fonction mathématique	Établi la valeur d'un cosinus à partir de la valeur d'un sinus en effectuant l'opération : cos(x) = Sin(90-x)	FXA_2424S Sinus		1	17
40	TRACK_INIT	Programme général	Etabli l'initialisation du tracker. Déplacement en position Azimut=midi et Hauteur=0.	LCD_EFFACE LCD_LOCATE LCD_DONNEE Tempo_500ms, 1s	CapteurA, CapteurH	0	72
41	ANEMO	Programme général	Correspondance entre la vitesse de rotation de l'anémomètre et la vitesse du vent.	FXA_2424S FXD_2424U	x, y, ANM, VitessVent, Cmpt1, vlimit, REG_TRACK,3	1	65
42	AFF_ANEMO	Programme général	Affiche la vitesse du vent (en km/h) lorsque le suiveur place des panneaux en sécurité.	LCD_LOCATE LCD_DONNEE Tempo_1s	VitessVent	0	30
43	PositionnementA	Programme général	Positionnement du traker en azimut	FX_ARRONDI FXA_2424S FDX_2424S Tempo_500ms, 30ms	x, RegAzim, CapteurA	0	80
44	PositionnementS	Programme général	Positionnement du traker en sécurité	PositionnementH	x, RegHaut	0	81
45	PositionnementH	Programme général	Positionnement du traker en hauteur	FX_ARRONDI FXA_2424S FDX_2424S Tempo_500ms Tempo_30ms, 10ms	x, RegHaut, CapteurH	0	81
46	HEURE_ETE	Programme général	Exécute le passage automatique en heure d'été et heure d'hiver.		mois, jour, date, heure, dchange, REG_TRACK,1	1	69
47	REGL_PAR	Programme général	Routine « Paramétrages »			0	
48	REGL_LATD	Programme général	Paramétrage de la latitude (degrés)			0	
49	REGL_LATC	Programme général	Paramétrage de la latitude (centièmes)			0	
50	REGL_LOND	Programme général	Paramétrage de la longitude (degrés)			0	
51	REGL_LONC	Programme général	Paramétrage de la longitude (centièmes)			0	
52	REGL_LONS	Programme général	Paramétrage de la longitude (signe))			0	
53	REGL_VLIM	Programme général	Paramétrage du seuil limite de la vitesse du vent pour la mise en sécurité.			0	
54	REGL_TLIM	Programme général	Paramétrage du temps minimal pendant lequel le système reste en sécurité.			0	
55	REGL_ANN	Programme général	Paramétrage de l'année	LCD_EFFACE LCD_LOCATE LCD_DONNEE AFFICHE_24BS IC2_START_ IC2_SEND_BYTE IC2_STOP Conv_DH Tempo_500ms, 200ms	Cmpt0, annee	0	56

56	REGL_MOI	Programme général	Paramétrage du mois	Idem : REGLAGE_ANN	Cmpt0, annee	0	54
57	REGL_DAT	Programme général	Paramétrage du jour dans le mois	Idem : REGLAGE_ANN	Cmpt0, mois	0	54
58	REGL_JOU	Programme général	Paramétrage du jour de la semaine			0	
59	REGL_HEU	Programme général	Paramétrage de l'heure	Idem : REGLAGE_ANN	Cmpt0, heure	0	56
60	REGL_MIN	Programme général	Paramétrage des minutes	Idem : REGLAGE_ANN	Cmpt0, minute	0	60
61	AFFICHE_8BS	Affichage LCD	Affichage d'une valeur codée sur 1 octet sur l'afficheur LCD	AFFICHE_24BS	x	0	7
62	AFFICHE_24BS	Affichage LCD	Affichage d'une valeur codée sur 24 bits sur l'afficheur LCD	FXD_2424U LCD_LOCATE LCD_DONNEE	x, y, Position, OctCompte	0	49
63	AFF_DATEHEURE	Affichage	Affiche la date et l'heure sur la 1ère ligne de l'afficheur LCD.	LCD_LOCATE LCD_DONNEE AFFICHE_8BS	x, date, mois, annee, heure, minute	0	57
64	AFF_HAUTEUR	Affichage	Affichage de la hauteur sur la 2ème ligne de l'afficheur LCD	FDX_2424S LCD_LOCATE LCD_DONNEE AFFICHE_8BS		0	
65	AFF_AZIMUT	Affichage	Affiche l'Azimut sur la 2eme ligne de l'afficheur LCD.	FDX_2424S LCD_LOCATE LCD_DONNEE AFFICHE_8BS	x, y, z, RegHaut, RegAzim, cmpt1, Calcul2	0	71
66	ARRONDI	Conversion numérique	Fonction arrondi au 10 ème	FXM_2424S FXD_2424S FXA_2424S	x, y, z	1	26
67	LATITUDE	Programme général	Latitude exprimée en centièmes de degrés	FXM_2424S FXA_2424S	x, y, latitude	1	
68	LONGITUDE	Programme général	Longitude exprimée en centièmes de degrés	FXM_2424S FXA_2424S	x, y, longitude	1	
69	CALCUL	Calculs	Calculs 10 séquences distinctes détaillées ci-dessous.			1	
	JourAnnee	Calculs	Calcul du quantième du jour de l'année.	FXA_2424S	x, MoisAnnee	1	118
	AnoMoy	Calculs	Calcul de l'anomalie moyenne.	FXA_2424S FXM_2424S FXD_2424SA	x, Calcul1	1	27
	EquCen	Calculs	Calcul de l'équation du centre.	Sinus FXM_2424S FXD_2424SA	x, y, RegHaut	1	22
	LonEcl	Calculs	Calcul de la longitude de l'écliptique.	FXM_2424S FXA_2424S FXD_2424SA	x, y, Calcul1, RegHaut, JourAnnee	1	45
	RedEqu	Calculs	Calcul de la réduction à l'Equateur	FXD_2424SA FXM_2424S Sinus FXA_2424S	x, y, Calcul1, Calcul2,	1	97
	Declin	Calculs	Calcul de la déclinaison solaire.	FXD_2424SA Sinus FXM_2424S FXA_2424S	x, y, Calcul1, Calcul3	1	94
	HeureJour	Calculs	Conversion, de l'heure en terme décimal.	FXM_2424S FXD_2424SA FXA_2424S	x, y, Calcul1, heure, minute	1	42
	AngHor	Calculs	Calcul de l'angle horaire.	FXA_2424S FXM_2424S FXD_2424SA	x, y, Calcul1, Calcul2, RegHaut		71
	Hauteur	Calculs	Calcul de la Hauteur.	FXA_2424S FXM_2424S FXD_2424SA Sinus Arcsinus	x, y, Calcul1, Calcul2, Calcul3, RegHaut	1	140
	Azimut	Calculs	Calcul de l'Azimut	Sinus, Cosinus FXA_2424S FXM_2424S FXD_2424SA Arcsinus	x, y, z, Calcul1, Calcul2, Calcul3, RegHaut, RegAzim	1	202

Le programme complet se compose d'un programme principal et et d'environ 70 sous-programmes. Ces sous-programmes listés dans le tableau ci-dessus sont détaillés dans les paragraphes qui suivent.
Le programme dans sa totalité utilise quasiment les 2 premieres BANK de la mémoire programme (soit 4096 lignes de programme).
Dans l'avant dernière colonne du tableau est indiqué dans quelle BANK se trouve le sous-programme concerné.
La deuxième BANK (BANK1) est utilisée principalement pour les calculs et toutes les fonctions qui ne font pas appel à une fonction d'affichage, une fonction du bus I²C ou une temporisation.
En effet, le passage d'une BANK à l'autre nécessite quelques précautions en veillant à utiliser la macro « CALLX » pour indiquer que l'on va chercher un sous-programme dans l'autre BANK. Le passage d'une BANK à l'autre n'est pas automatique.

1 - FXA_2424S

La routine des 7 programmes de fonctions mathématiques qui suivent a été récupérée dans la bibliothèque en accès libre sur INTERNET.

Addition de 2 nombres sur 3 octets (avec signe)

```
    movfw       x+0
    addwf       y+0,F
    movfw       x+1
    skpnc
    incfsz      x+1,W
    addwf       y+1,F
    movfw       x+2
    skpnc
    incfsz      x+2,W
    addwf       y+2,F
    movfw       y+2
    movwf       x+2
    movfw       y+1
    movwf       x+1
    movfw       y+0
    movwf       x+0
    return

FXM_2424U:
   CLRF         x+5
   CLRF         x+4
   CLRF         x+3
   MOVLW        D'24'
   MOVWF        BitCompte
   RRF          x+2,F
   RRF          x+1,F
   RRF          x+0,F
fx2424u1
   BTFSS        CARRY
    GOTO        fx2424u0
  MOVFW         y+0
  ADDWF         x+3,F
  MOVFW         y+1
  BTFSC         CARRY
  INCFSZ        y+1,W
  ADDWF         x+4,F
  MOVFW         y+2
  BTFSC         CARRY
  INCFSZ        y+2,W
  ADDWF         x+5,F
fx2424u0
  RRF           x+5,F
  RRF           x+4,F
  RRF           x+3,F
  RRF           x+2,F
  RRF           x+1,F
  RRF           x+0,F
  DECFSZ        BitCompte,F
  GOTO          fx2424u1
  RETURN
```

2 - FXM_2424U

Multiplication de 2 nombres sur 3 octets (sans signe)

```
    bcf         SNEGx
    bcf         SNEGy
    btfss       x+2,7
    goto        fxm2424s0
    bsf         SNEGx
```

```
        bsf             SNEGy
        comf            x+2
        comf            x+1
        comf            x+0
        incf            x+0
        skpnz
        incf            x+1
        skpnz
        incf            x+2

fxm2424s0
        btfss           y+2,7
        goto            fxm2424s1
        bsf             SNEGx
        btfsc           SNEGy
        bcf             SNEGx
        comf            y+2
        comf            y+1
        comf            y+0
        incf            y+0
        skpnz
        incf            y+1
        skpnz
        incf            y+2

fxm2424s1
        call            FXM_2424U
        btfss           SNEGx
        return
        comf            x+2
        comf            x+1
        comf            x+0
        incf            x+0
        skpnz
        incf            x+1
        skpnz
        incf            x+2
        return
```

3 - FXM_2424S

Multiplication de 2 nombres sur 3 octets (avec signe)

```
        bcf             SNEGx
        bcf             SNEGy
        btfss           x+2,7
        goto            fxm2424s0
        bsf             SNEGx
        bsf             SNEGy
        comf            x+2
        comf            x+1
        comf            x+0
        incf            x+0
        skpnz
        incf            x+1
        skpnz
        incf            x+2

fxm2424s0
        btfss           y+2,7
        goto            fxm2424s1
        bsf             SNEGx
        btfsc           SNEGy
        bcf             SNEGx
        comf            y+2
        comf            y+1
        comf            y+0
        incf            y+0
        skpnz
        incf            y+1
        skpnz
        incf            y+2

fxm2424s1
        call            FXM_2424U
        btfss           SNEGx
        return
        comf            x+2
        comf            x+1
```

```
          comf           x+0
          incf           x+0
          skpnz
          incf           x+1
          skpnz
          incf           x+2
          return
```

4 - FXD_2424U

Division de 2 nombres sur 3 octets (sans signe)

```
     Movlw             .24
     Movwf             BitCompte
     Movf              x+2,
     Movwf             t+2
     Movf              x+1,w
     Movwf             t+1
     Movf              x+0,w
     Movwf             t+0
     Clrf              x+2
     Clrf              x+1
     Clrf              x+0
     Clrf              z+2
     Clrf              z+1
     Clrf              z+0

fxd2424u3
     bcf        CARRY
     rlf        t+0,f
     rlf        t+1,f
     rlf        t+2,f
     rlf        z+0,f
     rlf        z+1,f
     rlf        z+2,f
     movf              y+2,w
     subwf             z+2,w
     Btfss             ZERO
     Goto              fxd2424u0
     Movf              y+1,w
     Subwf             z+1,w
     Btfss             ZERO
     Goto              fxd2424u0
     Movf              y+0,w
     Subwf             z+0,w

fxd2424u0
     btfss             CARRY
     goto              fxd2424u1
     movf              y+0,w
     subwf             z+0,f
     btfsc             CARRY
     goto              fxd2424u2
     Decf              z+1,f
     Movf              z+1,w
     Xorlw             0xff
     Btfsc             ZERO
     Decf              z+2,f

fxd2424u2
     movf              y+1,w
     subwf             z+1,f
     btfss             CARRY
     decf              z+2,f
     movf              y+2,w
     subwf             z+2,f
     bsf               CARRY

fxd2424u1
      rlf              x+0,f
      rlf              x+1,f
      rlf              x+2,f
      decfsz           BitCompte,f
      goto             fxd2424u3
      return
```

5 - FXD_2424S

Division de 2 nombres sur 3 octets (avec signe)

```
        bcf             SNEGx
        bcf             SNEGy
        btfss           x+2,7
        goto            fxd2424s0
        bsf             SNEGx
        bsf             SNEGy
        comf            x+2
        comf            x+1
        comf            x+0
        incf            x+0
        skpnz
        incf            x+1
        skpnz
        incf            x+2

fxd2424s0
        btfss           y+2,7
        goto            fxd2424s1
        bsf             SNEGx
        btfsc           SNEGy
        bcf             SNEGx
        comf            y+2
        comf            y+1
        comf            y+0
        incf            y+0
        skpnz
        incf            y+1
        skpnz
        incf    y+2

fxd2424s1
        call            FXD_2424U
        btfss           SNEGx
        return
        comf            x+2
        comf            x+1
        comf            x+0
        incf            x+0
        skpnz
        incf            x+1
        skpnz
        incf            x+2
        return
```

6 - FX_ARRONDI

Fonction arrondi (utilisé après une division)

```
        Movxf           t+3                     ; x transféré dans le registre "x"
        movzf           x                       ; registre "z" transféré dans registre "x"
        movly           h'00',h'00',h'0a'       ; valeur "10" dans "y"
        call            FXM_2424U               ; "x" est multiplié par 10
        movfy           t+6                     ; le registre "t+6" est transféré dans "y"
        call            FXD_2424U               ; "x" est divisé par "y" (résultat dans "x")
        movlw           d'5'                    ; valeur "5" dans le registre de travail
        subwf           x+0                     ; la valeur "5" est soustraite de "x+0"
        btfsc           CARRY                   ; saut si la valeur trouvée est négative
        goto            fxarrondi0
        movfx           t+3                     ; le registre "t+3" est transféré dans "x"
        return

fxarrondi0
        movfx           t+3             ; le registre "t+3" est transféré dans "x"
        movly           d'0',d'0',d'1'  ; la valeur "1" est transféré dans "y"
        call            FXA_2424S       ; le registre "x" est incrémenté
        return
```

7 - FXD_2424SA

Division de 2 nombres sur 3 octets (avec signe et arrondi)

```
        bcf             SNEGx
```

```
                bcf             SNEGy
                btfss           x+2,7
                goto            fxd2424sa0
                bsf             SNEGx
                bsf             SNEGy
                comf            x+2
                comf            x+1
                comf            x+0
                incf            x+0
                skpnz
                incf    x+1
                skpnz
                incf    x+2

        fxd2424sa0
                btfss           y+2,7
                goto            fxd2424sa1
                bsf             SNEGx
                btfsc           SNEGy
                bcf             SNEGx
                comf            y+2
                comf            y+1
                comf            y+0
                incf            y+0
                skpnz
                incf            y+1
                skpnz
                incf            y+2

        fxd2424sa1
                movyf           t+6
                call            FXD_2424U
                call            FX_ARRONDI
                btfss           SNEGx
                return
                comf            x+2
                comf            x+1
                comf            x+0
                incf            x+0
                skpnz
                incf            x+1
                skpnz
                incf            x+2
                return
```

8 - Tempo_2.5s :

Temporisation 350 ms, 500 ms, 1 s, 2,5 s
(toutes les temporisations nécessitent l'utilisation des registres temp1, temp2, temp3)

```
        movlw   d'250'          ;1 cycle
        movwf   temp3           ;1 cycle
        goto    tempo2          ;2 cycles

Tempo_1s:
        movlw   d'100'          ;1 cycle
        movwf   temp3           ;1 cycle
        goto    tempo2          ;2 cycles

Tempo_500ms:
        movlw   d'50'           ;1 cycle
        movwf   temp3           ;1 cycle
        goto    tempo2          ;2 cycles

Tempo_350ms:
        movlw   d'35'           ;1 cycle
        movwf   temp3           ;1 cycle
        goto    tempo2          ;2 cycles

tempo2:
        call    Tempo_10ms      ;
        decfsz  temp3,1         ;2 cycles si temp1=0 sinon 1 cycle
        goto    tempo2          ;2 cycles
        return
```

9 - Tempo_200ms :

Temporisation 10 ms, 30 ms, 100m s, 150 ms, 200 ms

```
        movlw   d'200'          ;1 cycle
        movwf   temp2           ;1 cycle
        goto    tempo1          ;2 cycles

Tempo_150ms:
        movlw   d'150'          ;1 cycle
        movwf   temp2           ;1 cycle
        goto    tempo1          ;2 cycles

Tempo_100ms:
        movlw   d'100'          ;1 cycle
        movwf   temp2           ;1 cycle
        goto    tempo1          ;2 cycles

Tempo_30ms:
        movlw   d'30'           ;1 cycle
        movwf   temp2           ;1 cycle
        goto    tempo1          ;2 cycles

Tempo_10ms:
        movlw   d'10'           ;1 cycle
        movwf   temp2           ;1 cycle

tempo1:
        call    Tempo_1ms       ;
        decfsz  temp2,1         ;2 cycles si temp1=0 sinon 1 cycle
        goto    tempo1          ;2 cycles
        return
```

10 - Tempo_1530us :

Temporisation 1530 µs

```
        call    Tempo_1ms
        call    Tempo_530us
        return
```

11 - Tempo_1ms :

Temporisation 1 ms, 530 ms, 250 ms

```
        movlw   d'249'          ;1 cycle
        movwf   temp1           ;1 cycle
        goto    tempo           ;2 cycles

Tempo_530us:                    ;532us soit 532 cycles (temp1*4)+4
        movlw   d'132'          ;1 cycle
        movwf   temp1           ;1 cycle
        goto    tempo           ;2 cycles

Tempo_250us:                    ;532us soit 532 cycles (temp1*4)+4
        movlw   d'62'           ;1 cycle
        movwf   temp1           ;1 cycle
        goto    tempo           ;2 cycles

tempo:
        nop                     ;1 cycle
        decfsz  temp1,1         ;2 cycles si temp1=0 sinon 1 cycle
        goto    tempo           ;2 cycles
        return
```

12 - Tempo_43µs :

Temporisation 11 µs, 39 µs, 43 µs

```
        movlw   d'14'           ;1 cycle
        movwf   temp1           ;1 cycle
Tempo_43:
        decfsz  temp1,1         ;2 cycles si temp1=0 sinon 1 cycle
        goto    Tempo_43        ;2 cycles
```

```
        return

Tempo_39us:                     ;39us soit 39 cycles   (temp1*3)+2
        movlw   d'12'           ;1 cycle
        movwf   temp1           ;1 cycle

Tempo_39:
        decfsz  temp1,1         ;2 cycles si temp1=0 sinon 1 cycle
        goto    Tempo_39        ;2 cycles
        return

Tempo_12us:                     ;11us soit 11 cycles   (temp1*3)+3
        movlw   d'3'            ;1 cycle
        movwf   temp1           ;1 cycle
        nop

Tempo_12:
        decfsz  temp1,1         ;2 cycles si temp1=0 sinon 1 cycle
        goto    Tempo_11        ;2 cycles
        return

Tempo_11us:                     ;11us soit 11 cycles   (temp1*3)+2
        movlw   d'3'            ;1 cycle
        movwf   temp1           ;1 cycle

Tempo_11:
        decfsz  temp1,1         ;2 cycles si temp1=0 sinon 1 cycle
        goto    Tempo_11        ;2 cycles
        return
```

13 - LCD_INIT :

Liste des commandes pour l'afficheur LCD 2 lignes de 16 caractères:

LCD_INIT	: Initialisation de l'afficheur
LCD_WRITE_RAM_W	: Envoi un caractère (contenu dans w) sur l'afficheur
LCD_WRITE_CONTROL_W	: Envoi une commande (contenue dans w) à l'afficheur
LCD_CLEAR_DISPLAY	: Efface l'afficheur
LCD_RETURN_HOME	: Renvoi le curseur en haut a gauche
LCD_LIGNE1	: Renvoi le curseur au début de la ligne 1
LCD_LIGNE2	: Renvoi le curseur au début de la ligne 2
LCD_ON/LCD_OFF	: Allume ou éteint l'affichage
LCD_CURSOR_ON	: Affichage du curseur
LCD_CURSOR_OFF	: Pas d'affichage du curseur
LCD_BLINK_ON/LCD_BLINK_OFF	: Mode BLINK ON ou OFF
LCD_DIR_LEFT/LCD_DIR_RIGHT	: Direction de l'écriture
LCD_SH_ON/LCD_SH_OFF	:

Constantes à définir dans le programme principal:

```
LCD_DATA        ; LCD: Port ou est branché le bus de donnée ( bits 4 à 7 ) MSB IMPERATIF
LCD_E           ; LCD: Ligne de commande de contrôle de l'afficheur
LCD_RW          ; LCD: Ligne de Lecture/Ecriture de l'afficheur
LCD_RS          ; LCD: Ligne de sélection de l'afficheur
```

Variables à définir dans le programme principal:

```
LCD_mode_set,   LCD_disp_cont,   DATA_TMP,   J
LCD_curs_disp_shift,   LCD_func_set,DATA_SWAP,   PORT_SAVE
```

Initialisation de l'afficheur:

```
        call    Tempo_30ms
        movlw   h'33'                   ;Function set
        call    LCD_CONTROLE            ;Ecriture
        movlw   h'32'                   ;Function set
        call    LCD_CONTROLE            ;Ecriture
        movlw   LCD_FUNCTION_SET        ;Function set
;       movwf   LCD_func_set            ;Sauvegarde des paramètres
        call    LCD_CONTROLE            ;Ecriture
        movlw   LCD_DISPLAY_CONTROL     ;Display ON/OFF
;       movwf   LCD_disp_cont           ;Sauvegarde des paramètres
        call    LCD_CONTROLE            ;Ecriture
        call    LCD_EFFACE
        movlw   LCD_ENTRY_MODE_SET      ;Entry mode set
;       movwf   LCD_mode_set            ;Sauvegarde des paramètres
        call    LCD_CONTROLE            ;Ecriture
        movlw   LCD_CURSOR_DISPLAY_SHIFT
```

```
;       movwf   LCD_curs_disp_shift           ;Sauvegarde des paramètres
        return                                ;Init end
```

14 - LCD_CONTROLE :

Ce programme ainsi que les 4 suivants sont utilisés pour le contrôle de l'afficheur 16 caractères

```
bcf     LCD_RS
movwf   LCD_DATA
call    LCD_4B
swapf   LCD_DATA
call    LCD_4B
return
```

15 - LCD_DONNEE :

```
bsf     LCD_RS
movwf   LCD_DATA
call    LCD_4B
swapf   LCD_DATA
call    LCD_4B
return
```

16 - LCD_LOCATE :

```
addlw   d'128'
CALL    LCD_CONTROLE
return
```

17 - LCD_EFFACE :

```
movlw   h'01'
CALL    LCD_CONTROLE
CALL    Tempo_1530us
return
```

18 - LCD_4B :

```
movfw   LCD_PORT
movwf   LCD_PORT_SAVE
bcf     LCD_PORT_SAVE , LCD_D4
bcf     LCD_PORT_SAVE , LCD_D5
bcf     LCD_PORT_SAVE , LCD_D6
bcf     LCD_PORT_SAVE , LCD_D7
btfsc   LCD_DATA , 4
bsf     LCD_PORT_SAVE , LCD_D4
btfsc   LCD_DATA , 5
bsf     LCD_PORT_SAVE , LCD_D5
btfsc   LCD_DATA , 6
bsf     LCD_PORT_SAVE , LCD_D6
btfsc   LCD_DATA , 7
bsf     LCD_PORT_SAVE , LCD_D7
movfw   LCD_PORT_SAVE
movwf   LCD_PORT
bsf     LCD_E
bcf     LCD_E
call    Tempo_11us
return
```

19 - I2C_INIT:

Routines de gestion du bus I²C:

Ces routines nécessitent :

- Les deux macros : SDA_IN et SDA_OUT
- Les variables : I2C_flag, I2C_DATA, I
- Le fichier : Q4mhz.asm

```
       SDA_OUT              ;SDA en sortie
       bsf     SDA          ;SDA à 1
       bsf     SCL          ;SCL à 1
       return
```

20 - I2C_START_:

```
                            ; SDA_OUT, SDA=1, SCL=1
       bsf     SCL          ;SCL à 1
       bsf     SDA          ;SDA à 1
       bcf     SDA          ;SDA à 0
       return
```

21 - I2C_STOP_:

```
                            ;SDA_OUT
                            ;SDA=X SCL=0
       bcf     SDA          ;SDA à 0
       bsf     SCL          ;SCL à 1
       bsf     SDA          ;SDA à 1
       return
```

22 - I2C_SLAVE_ACK:

```
                            ;SDA_OUT
                            ;SDA=X SCL=0
       bcf     SCL          ;SCL à 0
       SDA_IN               ;SDA en entrée
       CALL    Tempo_11us
       bsf     SCL          ;SCL à 1
       CALL    Tempo_11us
       btfsc   SDA          ;Si 1 écriture d'un flag d'erreur
       call    NO_ACK
       bcf     SCL          ;SCL à 0
       CALL    Tempo_11us
       SDA_OUT              ;SDA en sortie
       return
```

23 - I2C_MASTER_ACK:

```
                            ;SDA_IN
                            ;SDA=X SCL=0
       bcf     SCL          ;SCL à 0
       SDA_OUT              ;SDA en sortie
       bcf     SDA          ;SDA à 0
       bsf     SCL          ;SCL à 1
       CALL    Tempo_11us
       bcf     SCL          ;SCL à 0
       CALL    Tempo_11us
       bsf     SDA          ;SDA à 1
       return
```

24 - I2C_MASTER_NACK:

```
                            ;SDA_IN
                            ;SDA=X SCL=0
       bcf     SCL          ;SCL à 0
       SDA_OUT              ;SDA en sortie
       bsf     SDA          ;SDA à 1
       bsf     SCL          ;SCL à 1
       CALL    Tempo_11us
       bcf     SCL          ;SCL à 0
       CALL    Tempo_11us
       bcf     SDA          ;SDA à 0
       return
```

25 - I2C_SEND_BYTE:

```
       movwf   I2C_DONNEE   ;w contient les data à envoyer
       movlw   d'8'
       movwf   I
  Next_data_s:
       decf    I,f
```

```
      bcf     SCL             ;SCL a 0
      CALL    Tempo_11us
      rlf     I2C_DONNEE,f    ;Rotation a gauche bit(I) dans la carry
      btfsc   STATUS,C        ;Si I=1
      bsf     SDA             ;SDA a 1
      btfss   STATUS,C        ;Si I=0
      bcf     SDA             ;SDA a 0
      bsf     SCL             ;SCL a 1
      CALL    Tempo_11us
      incf    I,f
      decfsz  I,f             ;Si I!=0
      goto    Next_data_s
      bcf     SCL             ;SCL a 0
      CALL    Tempo_11us
      call    I2C_SLAVE_ACK
      return
```

26 - I2C_RECEIVE_BYTE:

```
      movlw   d'8'
      movwf   I
      clrf    I2C_DONNEE
      SDA_IN                  ;SDA en entrée
 Next_data_r:
      decf    I,f
      bcf     SCL             ;SCL a 0
      CALL    Tempo_11us
      bsf     SCL             ;SCL a 1
      CALL    Tempo_11us
      btfsc   SDA             ;Si SDA=1
      bsf     STATUS,C        ;
      btfss   SDA             ;Si SDA=0
      bcf     STATUS,C        ;
      rlf     I2C_DONNEE,f    ;décallage à gauche et met la carry dans le LSB
      incf    I,f
      decfsz  I,f             ;Si I!=0
      goto    Next_data_r
      bcf     SCL             ;SCL a 0
      CALL    Tempo_11us
      call    I2C_MASTER_ACK
      movfw   I2C_DONNEE
      return
```

27 - I2C_RECEIVE_BYTE_LAST:

```
      movlw   d'8'
      movwf   I
      SDA_IN                  ;SDA en entrée
Next_data_rl:
      decf    I,f
      bcf     SCL             ;SCL a 0
      CALL    Tempo_11us
      bsf     SCL             ;SCL a 1
      CALL    Tempo_11us
      btfsc   SDA             ;Si SDA=1
      bsf     STATUS,C        ;
      btfss   SDA             ;Si SDA=0
      bcf     STATUS,C
      rlf     I2C_DONNEE,f    ;décallage à gauche et met la carry dans le LSB
      incf    I,f
      decfsz  I,f             ;Si I!=0
      goto    Next_data_rl
      bcf     SCL             ;SCL a 0
      CALL    Tempo_11us
      call    I2C_MASTER_NACK
      movfw   I2C_DONNEE
      return
```

28 - NO_ACK:

```
      bsf     I2C_flag,0      ;Pas d'ack de l'esclave -> problème de comm
      return
```

29 - RTC_INIT:

Initialisation du circuit RTCC (calibration)
Cette routine a pour but de transférer la valeur « 83 » à l'adresse H'08' du circuit RTCC
Cette valeur a été déterminée par la procédure décrite ci-dessous

Calibration du circuit RTCC :

Le circuit RTCC microchip MCP79410 est utilisé avec un quartz CITIZEN CFS206-32768K-DFZ
Tolérance de fréquence : +/-20 ppm
Vieillissement : +/-3 ppm (la 1ère année)
Coefficient de température : -0,034(+/-0,006) ppm/°C²

La fréquence du quartz n'est pas exactement de 32,768 KHz mais une valeur très approchée.
L'oscillateur constitué du quartz et de 2 condensateurs ne délivre pas la fréquence exacte de 32,768 KHz.
Cette fréquence varie en fonction de principaux facteurs qui sont :
Les vibrations subites par le quartz.
L'imprécision sur la valeur théorique du quartz (+/- 20 ppm ou plus)
Les variations de température
Le vieillissement des composants
Pour ces raisons, on constate une dérive de l'heure délivrée par le circuit pouvant atteindre 1 heure entière sur 1 année. Cette dérive, peut être en plus ou en moins suivant que l'heure avance ou retarde sur l'heure exacte.
Un système de calibration du circuit permet de corriger cette dérive.
La calibration consiste à rajouter ou à enlever un certain nombre d'impulsions au compteur diviseur du circuit toutes les minutes afin d'effectuer une correction.
Cette correction est introduite à l'aide du registre « CALIBRATION » situé à l'adresse 08h

Relevé des écarts de temps (avant calibration) en juin 2013:

(Lors des relevés, l'heure du tracker et l'heure exacte avaient déjà fortement divergées.

Date	Ecarts	Heure du tracker	Heure réelle	Ecart en s	Ecarts en secondes/jour
15/06/2013		08h 00mn 00s	08h 40mn 30s		
19/06/2013	4	08h 00mn 00s	08h 40mn 2s	28	+ 7 s/jour
20/06/2013	5	08h 00mn 00s	08h 39mn 54s	34	+ 7,2 s/jour
21/06/2013	6	08h 00mn 00s	08h 39mn 47s	43	+ 7,17 s/jour
24/06/2013	9	08h 00mn 00s	08h 39mn 25s	65	+ 7,22 s/jour
25/06/2013	10	08h 00mn 00s	08h 39mn 17s	73	+ 7,3 s/jour

Calcul de la calibration :
7,3 secondes / jour représentent : 7,3 secondes pour 86400 secondes (24 h x 60 mn x 60 s)
soit : 84.49 s pour 1 000 000 secondes (84,49 ppm)

La calibration intervient à chaque minute ; soit toutes les 32768 x 60 impulsions du compteur
Soit toutes les : 1 966 080 impulsions

Un bit du registre de calibration correspond à une augmentation ou une diminution de 2 impulsions du compteur diviseur (l'augmentation ou la diminution est définie par le bit 7 de registre « CALIBRATION »).
A chaque minute, le compteur diviseur est augmenté (ou décrémenté) d'une valeur de 2 fois la valeur indiquée dans le registre « CALIBRATION »
Pour une avance sur l'heure réelle, la calibration doit être positive (bit 7 du registre calibration à 0)
Pour un retard sur l'heure réelle, la calibration doit être négative (bit 7 du registre calibration à 1)

Donc, 1 bit du registre correspond à: 2 x 1 000 000 / 1 966 080 = 1,017 ppm (soit 32 s / an)
Pour un écart de 84,49 ppm, le registre de calibration doit donc être positionné à :
84,49 ppm / 1,017 = 83,08 (arrondi à 83)

En effet, lorsque l'heure du circuit RTCC avance, cela correspond à une fréquence réelle du quartz supérieure à 32768 Hertz. Le fait d'avancer le compteur subitement d'une valeur définie de plusieurs impulsions à chaque minute permet de diminuer globalement le nombre moyen d'impulsions par seconde pour ramener ce nombre d'impulsions le plus proche possible de la valeur idéale et précise de 32768.

Formule générale de calcul : ***CALIBRATION = écart (en secondes/jour) x 11,38***

Valeur du registre « CALIBRATION » pour une avance de l'heure d'environ 7,3 s/jour: **83** (avec bit 7 à 0)

Valeur binaire du registre : **0 1 0 1 0 0 1 1**

```
RTC_INIT

        Call            I2C_START_              ; Start condition
        Movlw           ADRS_RTCC+0             ; Adresse du circuit RTCC pour l'écriture
        Call            I2C_SEND_BYTE           ; Envoi de l'adresse du circuit
        Movlw           h'08'                   ; Adresse du registre "CALIBRATION"
        Call            I2C_SEND_BYTE           ; Pointage sur l'adresse mémoire h'08'
        Movlw           d'83'                   ; 83: valeur de calibration du circuit RTCC
        Call            I2C_SEND_BYTE           ; Envoi de la valeur d'83' à l'adresse h'08'
        Call            I2C_STOP_               ; Stop condition
        Return
```

30 - LIRE_PARAM:

Ce programme est exécuté à l'initialisation du suiveur et transfère les paramètres vers la mémoire RAM pour la prise en compte des paramètres de latitude, longitude, seuil pour la vitesse du vent et délai de maintien en position sécurité.

```
LIRE_PARAM                  ; Lecture de la valeur des paramètres en EEPROM 24AA256

        call    I2C_START_                  ; Bit start condition
        movlw   b'10100000'                 ; Adresse du circuit EEPROM (écriture)
        call    I2C_SEND_BYTE               ; Envoi de l'adresse du circuit EEPROM
        movlw   h'6B'                       ; 1er octet d'adresse de lecture du circuit EEPROM
        call    I2C_SEND_BYTE               ; Envoi de l'adresse du 1er octet pour la lecture
        movlw   h'D0'                       ; 2ème octet d'adresse de lecture du circuit EEPROM
        call    I2C_SEND_BYTE               ; Envoi de l'adresse du 2ème octet
        call    I2C_START_                  ; Bit start condition
        movlw   b'10100001'                 ; Adresse du circuit EEPROM (lecture)
        call    I2C_SEND_BYTE               ; Envoi de l'adresse du 1er octet pour la lecture
        call    I2C_RECEIVE_BYTE            ; Lecture de l'octet à l'adresse h'6B',h'D0'
        movwf   latitude+0
        call    I2C_RECEIVE_BYTE            ; Lecture de l'octet à l'adresse h'6B',h'D1'
        movwf   latitude+1
        call    I2C_RECEIVE_BYTE            ; Lecture de l'octet à l'adresse h'6B',h'D2'
        movwf   longitude+0
        call    I2C_RECEIVE_BYTE            ; Lecture de l'octet à l'adresse h'6B',h'D3'
        movwf   longitude+1
        call    I2C_RECEIVE_BYTE            ; Lecture de l'octet à l'adresse h'6B',h'D4'
        movwf   longitude+2
        call    I2C_RECEIVE_BYTE            ; Lecture de l'octet à l'adresse h'6B',h'D5'
        movwf   vlimit
        call    I2C_RECEIVE_BYTE_LAST       ; Lecture de l'octet à l'adresse h'6B',h'D6'
        movwf   tlimit
        call    I2C_STOP_
        return
```

31 - LIRE_RTCC:

Ce programme a pour but de lire les informations stockées dans le circuit RTCC pour les transférer dans des registres spécifiques de la mémoire RAM.
Il s'agit des registres : seconde, minute, heure, jour, date, mois et année
Si l'année est bissextile, le bit REG_TRACK,2 est mis à 1
Si l'année n'est pas bissextile le bit 2 du registre REG_TRACK reste à 0
Pour terminer, le programme convertit la date et l'heure du format BCD en format binaire

```
        Call            I2C_START_              ; Bit start condition
        Movlw           ADRS_RTCC+0             ; Adresse du circuit RTCC (écriture)
        Call            I2C_SEND_BYTE           ; Envoi de l'adresse du circuit RTCC
        Movlw           b'00000000'             ; Adresse de lecture du circuit RTCC
        Call            I2C_SEND_BYTE           ; Envoi de l'adresse du 1er octet pour la lecture
        Call            I2C_START_              ; Bit start condition
        Movlw           ADRS_RTCC+1             ; Adresse du circuit RTCC (lecture)
        Call            I2C_SEND_BYTE           ; Envoi de l'adresse du 1er octet pour la lecture
        Call            I2C_RECEIVE_BYTE        ; Lecture du 1er octet à l'adresse h00
        Movwf           seconde                 ; Transfère de la valeur lue dans "seconde"
        bcf             seconde,7               ; Effacement du bit 7 du registre "seconde"
        call            I2C_RECEIVE_BYTE        ; Lecture de l'octet suivant
        movwf           minute                  ; Transfère de la valeur lue dans "minute"
        call            I2C_RECEIVE_BYTE        ; Lecture de l'octet suivant
        movwf           heure                   ; Transfère de la valeur lue dans "heure"
        call            I2C_RECEIVE_BYTE        ; Lecture de l'octet suivant (jour de la semaine)
        movwf           jour                    ; Transfère de la valeur lue dans "jour"
        bcf             jour,3                  ; Effacement du bit 3 du registre "jour"
        bcf             jour,4                  ; Effacement du bit 4 du registre "jour"
        bcf             jour,5                  ; Effacement du bit 5 du registre "jour"
        call            I2C_RECEIVE_BYTE        ; Lecture de l'octet suivant
        movwf           date                    ; Transfère de la valeur lue dans le registre "date"
        call            I2C_RECEIVE_BYTE        ; Lecture de l'octet suivant
        movwf           mois                    ; Transfère de la valeur lue dans le registre "mois"
        bcf             BISSEXTILE              ; Effacement du flag "BISSEXTILE"
        btfsc           mois,5                  ; Test du bit 5 du registre "mois"(année bissextile)
        bsf             BISSEXTILE              ; Si bit 5 est à 1, "BISSEXTILE" est mis à b'1'
        bcf             mois,5                  ; Le bit 5 du registre "mois" est mis à 0
        call            I2C_RECEIVE_BYTE_LAST   ; Lecture du dernier octet
        movwf           annee                   ; Transfère de la valeur lue dans "annee"
        call            I2C_STOP_               ; Stop condition
        call            conv_HD_DateHeure       ; Conversion BCD vers système binaire date et heure
        return
```

32 - conv_HD:

Ce programme convertit une valeur quelconque du codage binaire (ou hexa) en codage BCD

```
movwf          t
movlx          h'00',h'00',h'00'
swapf          t,w
andlw          h'0f'
movwf          x+0
movly          h'00',h'00',h'0a'
CALLX          FXM_2424S
Movfw          t
Andlw          h'0f'
Addwf          x+0,w
return
```

33 - conv_DH:

Ce programme convertit une valeur quelconque du codage BCD en codage binaire (ou hexa)

```
clrf           x+2
clrf           x+1
movwf          x+0
movly          h'00',h'00',h'0a'
CALLX          FXD_2424S
Swapf          x+0,w
addwf          z+0,w
return
```

34 - conv_HD_DateHeure:

Ce programme convertit les valeurs du codage binaire (ou hexa) en codage BCD pour la date et l'heure

```
Movfw          seconde
Call           conv_HD
Movwf          seconde

Movfw          minute
Call           conv_HD
Movwf          minute

movfw          heure
call           conv_HD
movwf          heure

movfw          jour
call           conv_HD
movwf          jour

movfw          date
call           conv_HD
movwf          date

movfw          mois
call           conv_HD
movwf          mois

movfw          annee
call           conv_HD
movwf          annee

return
```

35 - conv_DH_DateHeure:

Ce programme convertit les valeurs du codage BCD en codage binaire (ou hexa) pour la date et l'heure

```
Movfw           seconde
Call            conv_DH
Movwf           seconde

Movfw           minute
Call            conv_DH
Movwf           minute

Movfw           heure
Call            conv_DH
Movwf           heure

Movfw           date
Call            conv_DH
Movwf           date

Movfw           mois
Call            conv_DH
Movwf           mois

Movfw           annee
Call            conv_DH
Movwf           annee

return
```

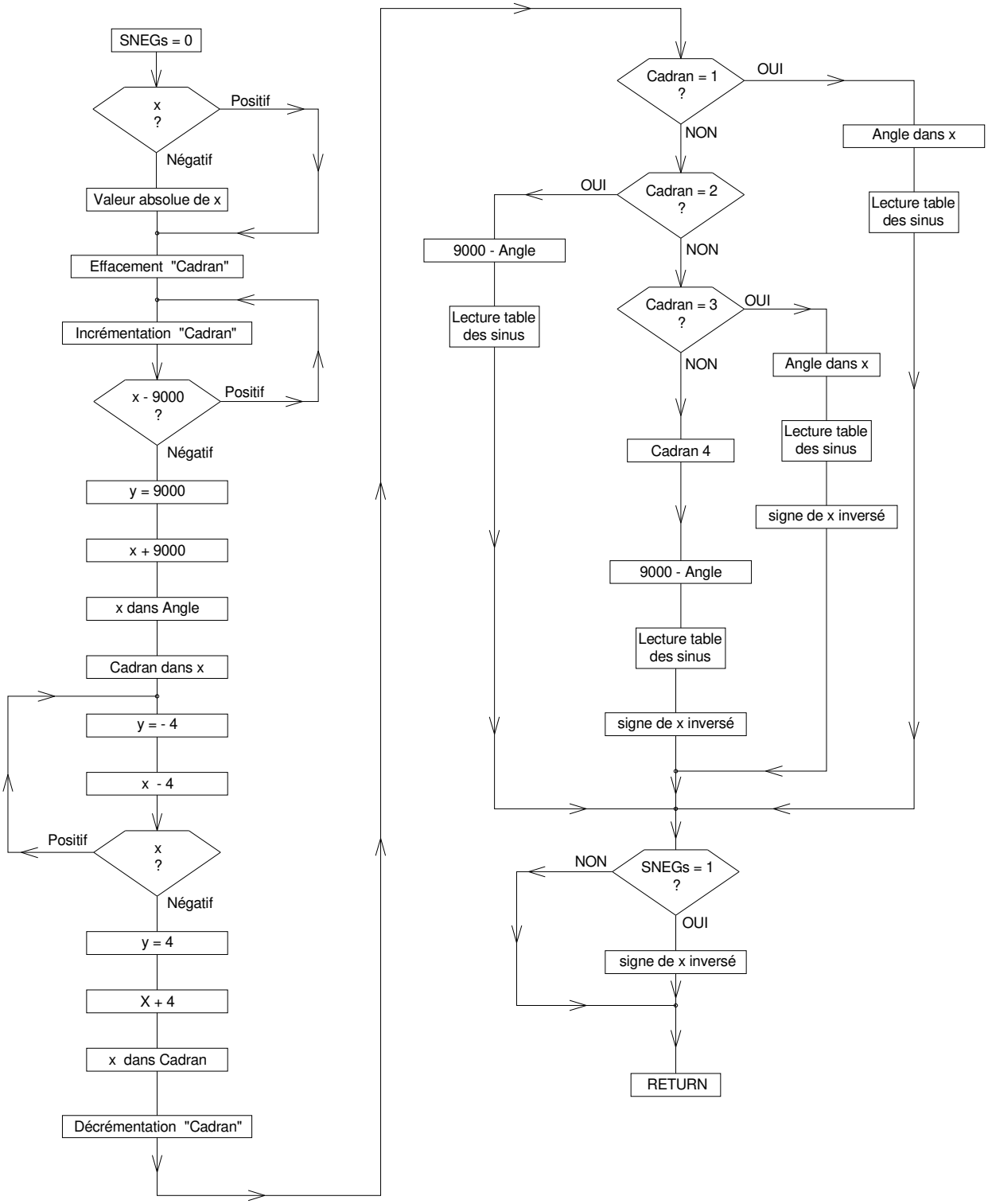
36 - LIRE_SINUS:

Ce programme vient lire la valeur d'un sinus dans la table des sinus et transfert la valeur dans le registre x
La valeur d'un angle est exprimé en centièmes de degrés (soit sur 4 chiffres).
Ce nombre peut donc varier de 0 à 8999 et il est codé sur 2 octets.
Il est transféré ensuite dans les registres x+0 et x+1
Comme les SINUS sont stockés sur 3 octets, il suffit de multiplier par 3 la valeur contenue dans le registre « x » pour pointer sur l'adresse du SINUS à obtenir. On récupère la valeur du SINUS en lisant les 3 octets à partir de cette adresse et en les transférant dans les registres x+2, x+1 et x+0.
Le nombre obtenu contient au maximum 6 chiffres.

```
movly           d'0',d'0',d'3'          ; Multiplié par 3 pour valeur sinus sur 3 octets
CALLX           FXM_2424S
call            I2C_START_
movlw           b'10100000'
call            I2C_SEND_BYTE
movfw           x+1
call            I2C_SEND_BYTE
movfw           x+0
call            I2C_SEND_BYTE
call            I2C_START_
movlw           b'10100001'
call            I2C_SEND_BYTE
call            I2C_RECEIVE_BYTE
movwf           x+2
call            I2C_RECEIVE_BYTE
movwf           x+1
call            I2C_RECEIVE_BYTE_LAST
movwf           x+0
call            I2C_STOP_
return
```

37 - SINUS:

Ce programme établit la valeur d'un SINUS à partir d'un angle inférieur ou supérieur à 90°, positif ou négatif.
La valeur du sinus est comprise entre -1 et +1
A partir de la valeur de l'angle, ont détermine d'abord dans quel cadran on se situe.
En fonction du cadran dans lequel on se trouve, on réalise ou pas le complément à 90°.
On effectue ensuite la lecture de correspondance dans la table des SINUS et on affecte à cette valeur le signe qui lui convient en fonction du cadran dans lequel on se trouvait.

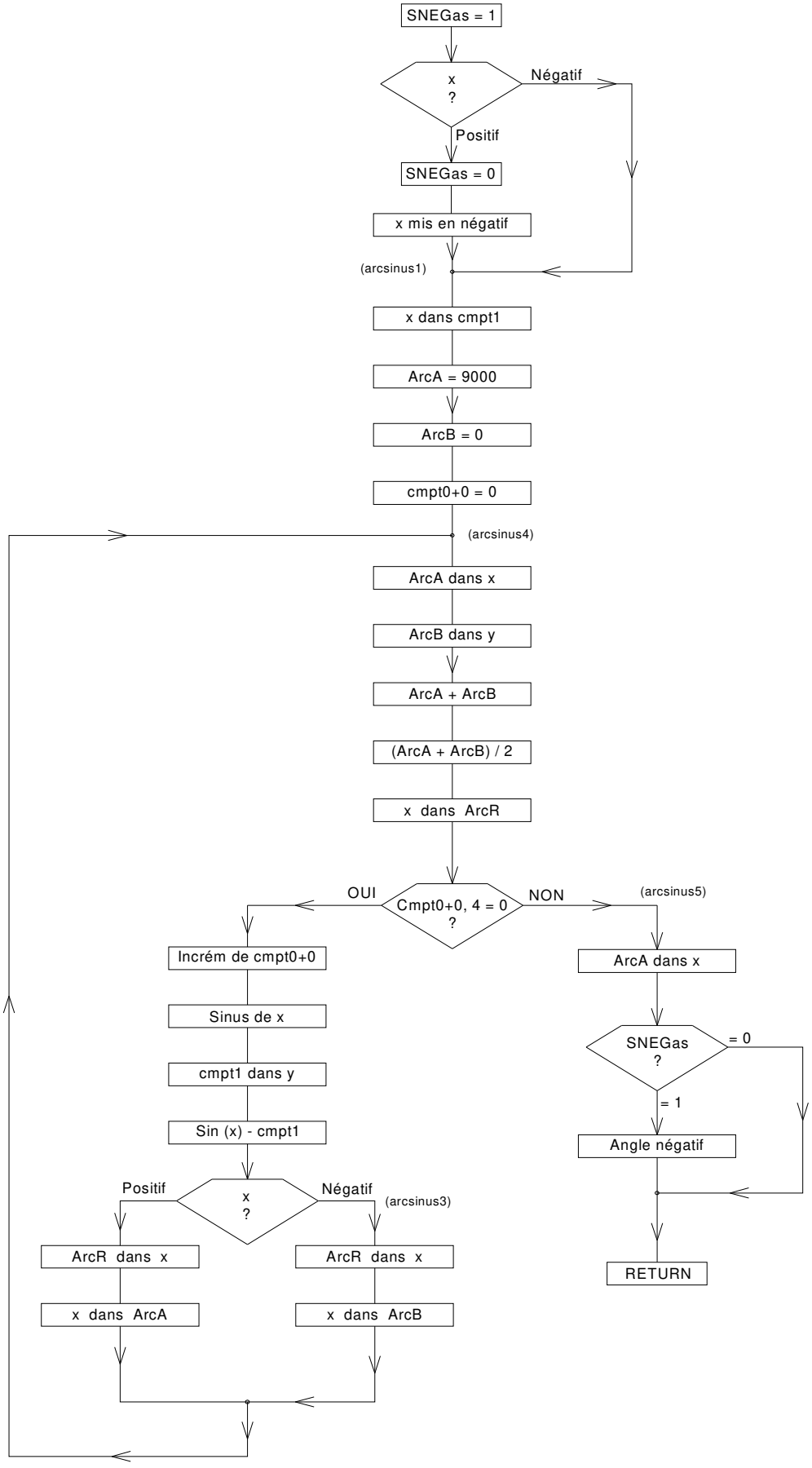


```
Sinus :

            bcf     SNEGs               ; Effacement du bit "SNEGs"
            btfss   x+2,7               ; Test si x positif ou négatif
            goto    sinus0              ; x positif, saut vers "sinus0"
            bsf     SNEGs               ; mise à 1 de "SNEGs" (x négatif)
            comf    x+2                 ; x mis en valeur positive (valeur absolue)
            comf    x+1
            comf    x+0
            incf    x+0
            skpnz
            incf    x+1
            skpnz
            incf    x+2

      sinus0
            clrf    Cadran+2            ; effacement du registre "Cadran"
            clrf    Cadran+1
            clrf    Cadran+0

      sinus1
            incf    Cadran              ; Incrémentation de "Cadran"
            movly   h'ff',h'dc',h'd8'   ; valeur (- 9000)
            call    FXA_2424S           ; x - 9000
            btfss   x+2,7               ; Test si x positif ou négatif
             goto   sinus1              ; x positif (bouclage)
            movly   h'00',h'23',h'28'   ; Valeur 9000
            call    FXA_2424S           ; x + y
            movxf   Angle               ; x transféré dans "Angle"
            movfx   Cadran              ; "Cadran" est transféré dans x

      sinus2
            movly   h'ff',h'ff',h'fc'   ; (-4) dans y
            call    FXA_2424S           ; x + y  (x - 4)
            btfss   x+2 , 7             ; Test si x positif ou négatif
            goto    sinus2              ; x positif (bouclage)
            movly   h'00',h'00',h'04'   ; Valeur 4 dans y
            call    FXA_2424S           ; x + y
            movxf   Cadran              ; x transféré dans cadran
            decf    Cadran              ; Décrémentation de "Cadran"
            btfsc   ZERO                ; Test si cadran égal à 1
            goto    cadran1             ; Cadran = 1, saut vers routine "cadran1"
            decf    Cadran              ; Décrémentation de "Cadran"
            btfsc   ZERO                ; Test si cadran égal à 2
            goto    cadran2             ; Cadran = 2, saut vers routine "cadran2"
            decf    Cadran              ; Décrémentation de "Cadran"
            btfsc   ZERO                ; Test si cadran égal à 1
            goto    cadran3             ; Cadran = 3, saut vers routine "cadran3"
            goto    cadran4             ; Cadran = 4, saut vers routine "cadran4"

      cadran1
            movfx   Angle               ; Angle transféré dans x
            CALLX   LIRE_SINUS          ; Lecture de la table des sinus
            goto    finsinus            ; Vers fin du sous-programme
      cadran2
            movlx   h'00',h'23',h'28'   ; Valeur 9000 dans x
            movfy   Angle               ; "Angle" transféré dans y
            comf    y+2                 ; Sinus en valeur négative
            comf    y+1
            comf    y+0
            incf    y+0
            skpnz
            incf    y+1
            skpnz
            incf    y+2
            call    FXA_2424S           ; 9000 - Angle
            CALLX   LIRE_SINUS          ; Lecture de la table des sinus
            goto    finsinus            ; Vers fin du sous-programme

      cadran3
            movfx   Angle               ; Angle transféré dans x
            CALLX   LIRE_SINUS          ; Lecture de la table des sinus
            comf    x+2                 ; Sinus en valeur négative
            comf    x+1
            comf    x+0
            incf    x+0
            skpnz
            incf    x+1
            skpnz
            incf    x+2
            goto    finsinus            ; Vers fin du sous-programme
```

```
        cadran4
               movlx   h'00',h'23',h'28'      ; Valeur 9000 dans x
               movfy   Angle                  ; "Angle" transféré dans y
               comf    y+2                    ; "Angle" en valeur négative
               comf    y+1
               comf    y+0
               incf    y+0
               skpnz
               incf    y+1
               skpnz
               incf    y+2
               call    FXA_2424S              ; 9000 - Angle
               CALLX   LIRE_SINUS             ; Lecture de la table des sinus
               comf    x+2                    ; Sinus en valeur négative
               comf    x+1
               comf    x+0
               incf    x+0
               skpnz
               incf    x+1
               skpnz
               incf    x+2

finsinus
               btfss   SNEGs                  ; Test de "SNEGs"
                return                        ; Fin du sous-prog si "SNEGs" = 0
               comf    x+2                    ; "SNEGs" = 1,  le signe de x est inversé
               comf    x+1
               comf    x+0
               incf    x+0
               skpnz
               incf    x+1
               skpnz
               incf    x+2
               return
```

38 - ArcSinus:

Ce programme calcule la valeur d'un angle à partir de la valeur de son SINUS.



```
ArcSinus:

               bsf     SNEGas                          ; Bit SNEGs mis à 1
               btfsc   x+2,7                           ; Test si x positif ou négatif
               goto    arcsinus1                       ; x négatif, saut vers "ArcSinus1"
               bcf     SNEGas                          ; x positif, mise à 0 de SNEGas
               comf    x+0                             ; x mis en valeur négative
               comf    x+1
               comf    x+2
               incf    x+0
               skpnz
               incf    x+1
               skpnz
               incf    x+2
       arcsinus1
               movxf   cmpt1                           ; x transféré dans "cmpt1"
               movlx   h'00',h'23',h'28'               ; Valeur 9000 dans x
               movxf   ArcA                            ; Valeur 9000 dans "ArcA"
               movlx   h'00',h'00',h'00'               ; x effacé
               movxf   ArcB                            ; Valeur 0 dans "ArcB"
               clrf    cmpt0                           ; Effacement du registre "cmpt0"
       arcsinus4
               movfx   ArcA                            ; "ArcA" dans x
               movfy   ArcB                            ; "ArcB" dans y
               call    FXA_2424S                       ;
               movly   h'00',h'00',h'02'               ; Valeur 2 dans y
               call    FXD_2424SA                      ; Division de x par 2
               movxf   ArcR                            ; x transféré dans "ArcR"
               btfsc   cmpt0,4                         ; Test si bit 4 de "cmpt0" est égal à 0
               goto    arcsinus5
               incf    cmpt0                           ; Incrémentation de "cmpt0"
               call    Sinus
               movfy   cmpt1
               call    FXA_2424S
               btfsc   x+2,7
               goto    arcsinus3
               movfx   ArcR
               movxf   ArcA
               goto    arcsinus4
       arcsinus3
               movfx   ArcR
               movxf   ArcB
               goto    arcsinus4
       arcsinus5
               movfx   ArcA
               btfss   SNEGas
               return
               comf    x+0
               comf    x+1
               comf    x+2
               incf    x+0
               skpnz
               incf    x+1
               skpnz
               incf    x+2
               return
```

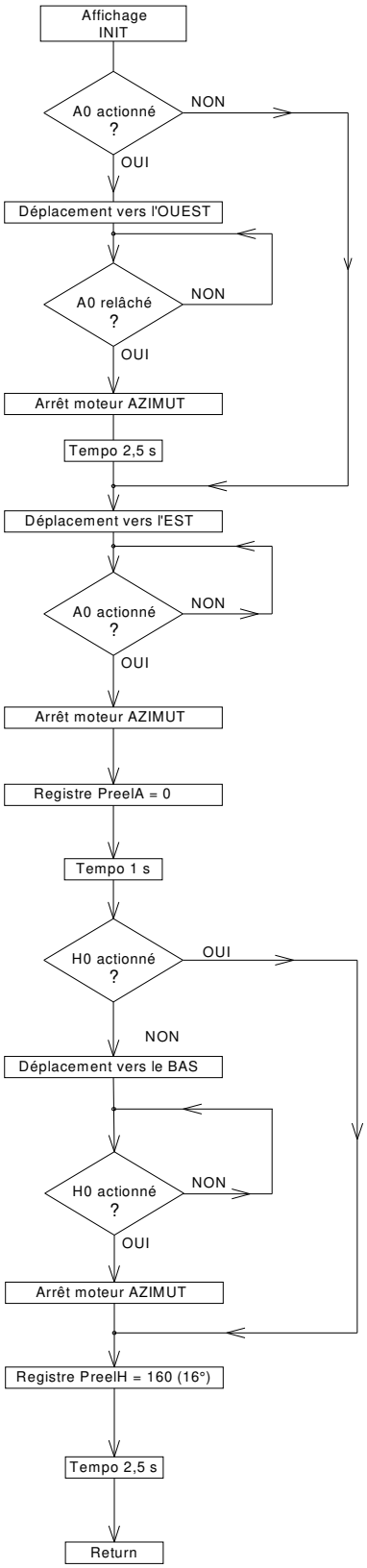
39 - Cosinus:

Ce programme calcule la valeur du COSINUS d'un angle à partir de la valeur de son SINUS.

```
       Cosinus:                    ; cos(x) = sin(90-x)

               comf    x+0
               comf    x+1
               comf    x+2
               incf    x+0
               skpnz
               incf    x+1
               skpnz
               incf    x+2
               movly   h'00',h'23',h'28'     ; Valeur 9000  soit 100 fois 90°
               call    FXA_2424S
               call    Sinus
               return
```

40 - TRACK_INIT:

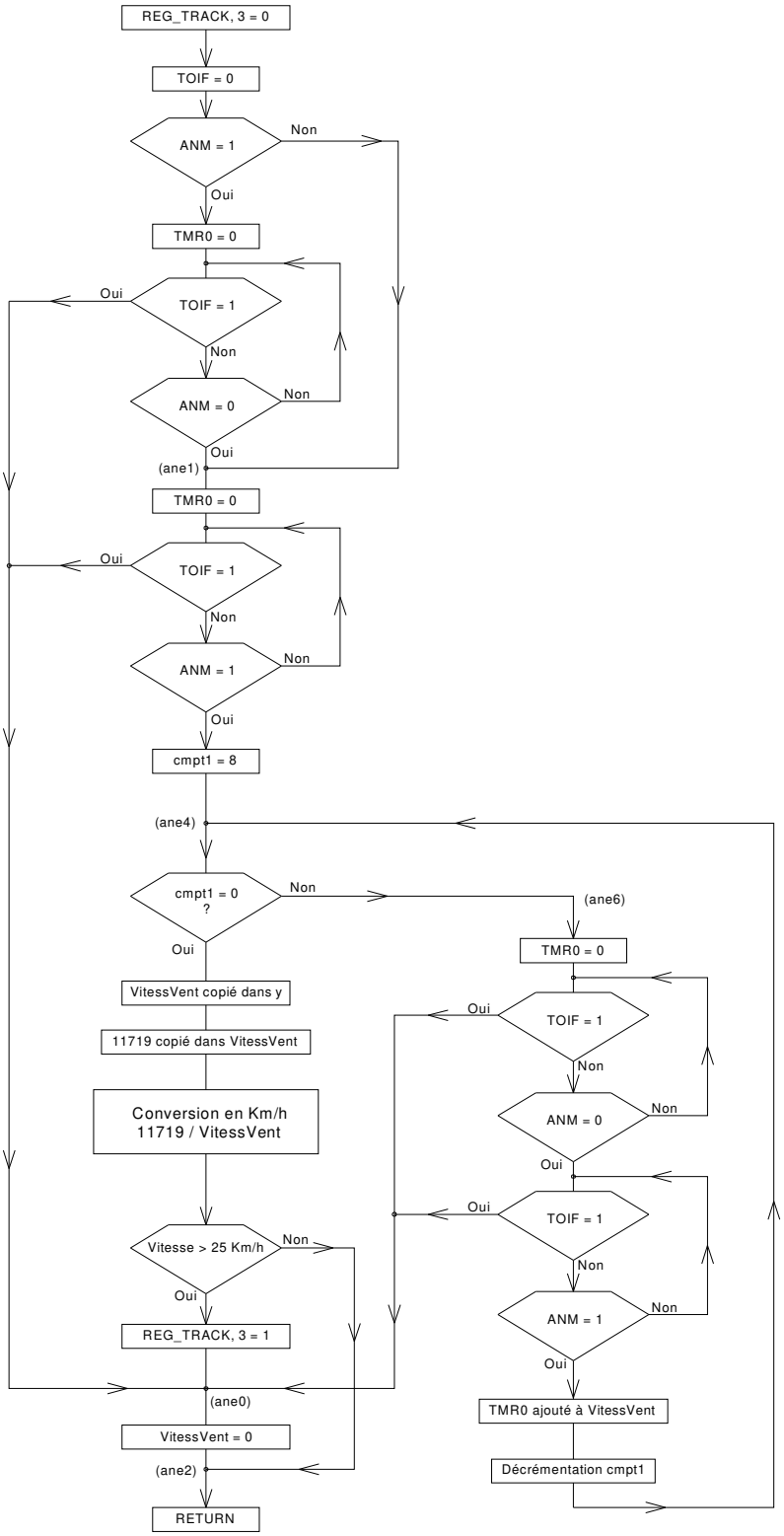
Ce programme réalise l'initialisation physique du suiveur à la position « plein SUD » et hauteur « minimale ».
Cette routine est exécutée à chaque mise sous tension du système et lorsque la hauteur du soleil repasse le matin au dessus de l'horizon.
Le repérage de chaque position d'origine est réalisé par un simple contact (A0 pour l'azimut et H0 pour la hauteur).



```
TRACK_INIT:           ; Initialisation positions... Azimut: midi, Hauteur: minimale

              CALL    LCD_EFFACE
              MOVLW   d'0'
              CALL    LCD_LOCATE
              MOVLW   "I"
              CALL    LCD_DONNEE
              MOVLW   "N"
              CALL    LCD_DONNEE
              MOVLW   "I"
              CALL    LCD_DONNEE
              MOVLW   "T"
              CALL    LCD_DONNEE

       TR_INITA:                          ; initialisation de l'azimut
              BTFSC   A0                  ; Test de la position en azimut
              GOTO    init_est            ; vers "déplacement vers l'EST
              BCF     FRA                 ; Positionnement pour déplacement vers l'ouest
              BCF     SSA                 ; Mise en marche moteur azimut vers l'OUEST
              BTFSS   A0                  ; Le capteur A0 passe-t-il à 1
              GOTO    $-1                 ; Attente pour A0 à 1 (capteur relâché)
              CALL    Tempo_2.5s          ;
              BSF     SSA                 ; Arrêt moteur AZIMUT

       init_est
              BSF     FRA                 ; Positionnement pour déplacement vers l'EST
              CALL    Tempo_2.5s          ;
              BCF     SSA                 ; Mise en marche moteur azimut vers l'EST
              BTFSC   A0                  ; Le capteur A0 passe-t-il à 0 ?
              GOTO    $-1                 ; Attente pour A0 à 0 (capteur contact fermé)
              BSF     SSA                 ; Arrêt moteur AZIMUT
              BCF     FRA                 ; Positionnement pour freinage (sens inverse)
              CALL    Tempo_10ms          ;
              BCF     SSA                 ; Marche en sens inverse 100 ms (freinage)
              CALL    Tempo_100ms         ; Durée du freinage: 100 ms
              BSF     SSA                 ; Arrêt moteur AZIMUT
              CLRF    PreelA+0            ; Initialisation du registres PreelA
              CLRF    PreelA+1            ;
              CLRF    PreelA+2            ;

       TR_INITH:                          ; Initialisation de la HAUTEUR
              CALL    Tempo_1s
              BCF     FRH                 ; Mise en position pour descente
              BTFSS   H0                  ; Test du capteur HAUTEUR (contact fermé ?)
              GOTO    fin_init
              BCF     SSH                 ; Mise en marche moteur HAUTEUR
              BTFSC   H0                  ; Test du capteur HAUTEUR (contact fermé ?)
              GOTO    $-1
              BSF     SSH                 ; Arrêt moteur HAUTEUR
              BSF     FRH                 ; Positionnement pour freinage (sens inverse)
              CALL    Tempo_10ms          ;
              BCF     SSH                 ; Marche en sens inverse 60 ms (freinage)
              CALL    Tempo_30ms          ; Durée du freinnage: 60 ms
              CALL    Tempo_30ms
              BSF     SSH                 ; Arrêt moteur HAUTEUR

       fin_init

              movlx   d'0',d'0',d'160'    ; Angle d'initialisation HAUTEUR: 16°
              movxf   PreelH              ; Transfère de la valeur 160 dans PreelH
              CALL    Tempo_1s            ; Tempo 1s
              RETURN
```

41 - ANEMO:

Ce programme effectue la correspondance entre la vitesse de rotation de l'anémomètre et la vitesse du vent.
Il utilise le signal « ANM » fourni par le capteur à fourche situé à l'intérieur de l'anémomètre.
Un étalonnage a été réalisé pour effectuer cette correspondance.
Lorsque qu'un vents fort est détecté, le suiveur se place en sécurité.
La mise en sécurité positionne les panneaux horizontalement avec une pente d'environ 3,5% ; puis arrête le fonctionnement du suiveur pour une durée de 30 mn (valeur par défaut). Cette valeur est modifiable par paramétrage de 2 à 59 mn.
A la fin de la période de vents forts, le suiveur se repositionne automatiquement en fonctionnement normal.



```
ANEMO:              ; Mesure de la vitesse du vent

                bcf     REG_TRACK,3         ; bit 3 de REG_TRACT mis à 0
                bcf     INTCON,2            ; bit TOIF mis à 0
                btfss   ANM                 ; Test si ANM = 1
                goto    ane1
                clrf    TMR0                ; ANM = 1 (attente ANM = 0)
                btfsc   INTCON,2            ; Test du bit TOIF ( Débordement TIMER0)
                goto    ane0
                btfsc   ANM
                goto    $-3

        ane1
                clrf    TMR0                ; ANM = 0 (attente ANM = 1)
                btfsc   INTCON,2
                goto    ane0
                btfss   ANM
                goto    $-3
                clrf    VitessVent+0        ; ANM = 1 (détection 1er front montant de l'impulsion)
                clrf    VitessVent+1
                clrf    VitessVent+2
                movlw   d'8'                ; Pour comptage des 8 encoches du disque à encoches
                movwf   cmpt1
                goto    ane5

        ane4
                decfsz  cmpt1,1
                goto    ane5
                goto    ane6

        ane5
                clrf    TMR0                ; Top départ de la mesure entre 2 fronts montants
                btfsc   INTCON,2            ; Test TOIF
                goto    ane0
                btfsc   ANM                 ; Attente front descendant
                goto    $-3
                btfsc   INTCON,2            ; Test TOIF
                goto    ane0
                btfss   ANM                 ; Attente front montant
                goto    $-3
                movf    TMR0,0              ; Fin de la mesure. Sauvegarde de la valeur mesurée
                movwf   y+0                 ; Sauvegarde dans y
                clrf    y+1
                clrf    y+2
                movfx   VitessVent
                CALLX   FXA_2424S           ; Mesure TMR0 cumulée dans "VitessVent"
                movxf   VitessVent          ; Sauvegarde du total dans "VitessVent"
                goto    ane4

        ane6
                movfy   VitessVent          ; Somme des 8 valeurs TMR0 totalisées copiée dans y
                movlx   0x00,0x2D,0xC7      ; Valeur 11719 dans x (pour conversion Km/h)
                movxf   VitessVent          ; "VitessVent" dans x
                CALLX   FXD_2424U           ; 11719 divisé par VitessVent
                movxf   VitessVent          ; Résultat sauvegardé dans x
                movfy   vlimit              ; Prise en compte seuil critique de mise en sécurité
                clrf    y+2
                clrf    y+1
                comf    y+2                 ; signe négatif pour comparaison
                comf    y+1
                comf    y+0
                incf    y+0
                skpnz
                incf    y+1
                skpnz
                incf    y+2
                CALLX   FXA_2424S
                btfsc   x+2,7               ; Test si "x" est négatif
                goto    ane2
                bsf     REG_TRACK,3         ; Vents forts détectés (Vitesse supérieure à vlimit)
                goto    ane2

        ane0
                clrf    VitessVent+0
                clrf    VitessVent+1
                clrf    VitessVent+2

        ane2
                return
```

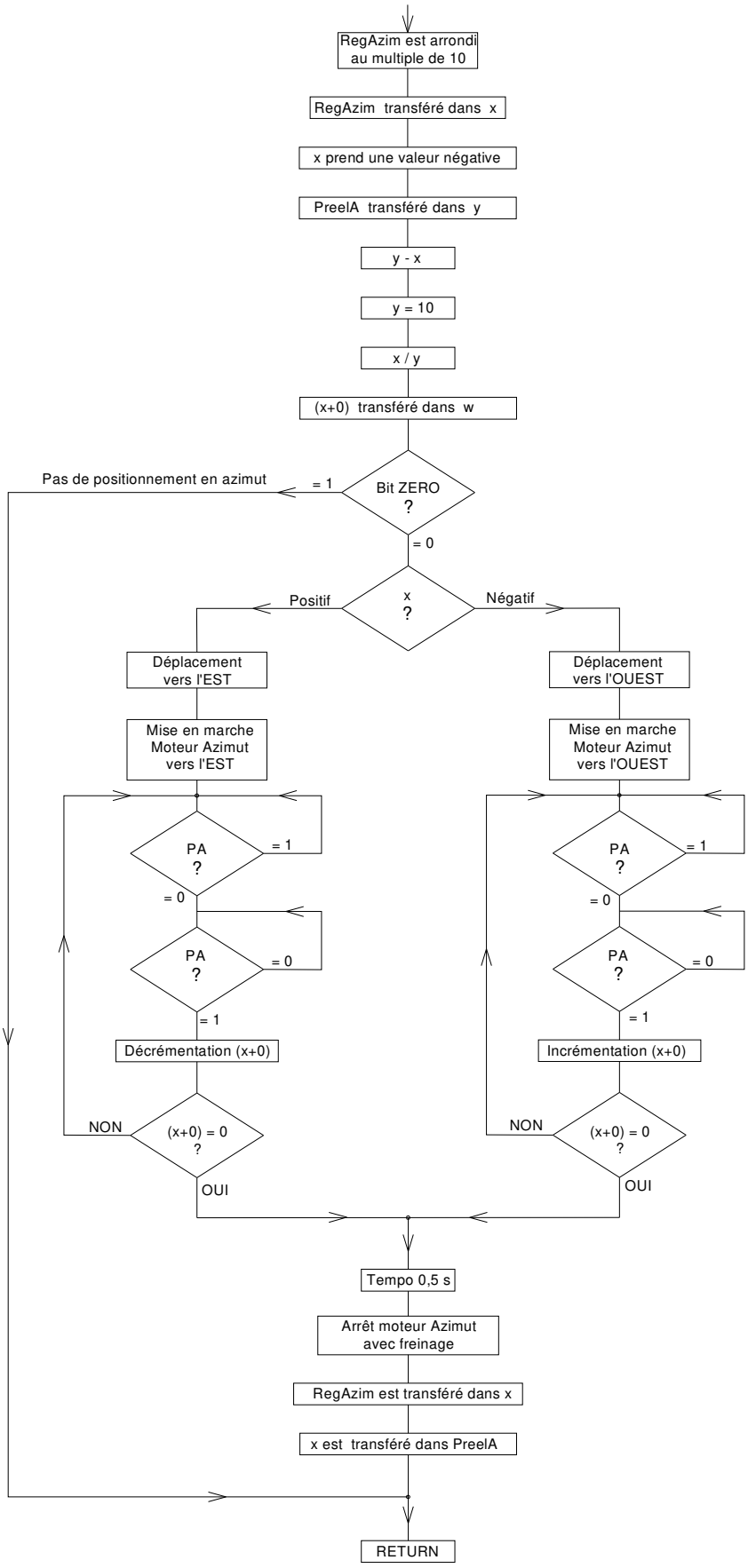
42 - AFF_ANEMO:

Ce programme réalise l'affichage de la vitesse du vent (en km/h) lorsque le suiveur place des panneaux en sécurité.

```
AFF_ANEMO:                  ; Affichage de la vitesse du vent

               movfx   VitessVent            ;
               movlw   d'64'                 ; Affichage: "VENT:" sur la 2ème ligne
               call    LCD_LOCATE
               movlw   "V"
               call    LCD_DONNEE
               movlw   "E"
               call    LCD_DONNEE
               movlw   "N"
               call    LCD_DONNEE
               movlw   "T"
               call    LCD_DONNEE
               movlw   ":"
               call    LCD_DONNEE
               movlw   " "
               call    LCD_DONNEE
               movlw   d'71'
               CALLX   AFFICHE_24BS   ; Affichage de la vitesse du vent
               movlw   " "
               call    LCD_DONNEE
               movlw   "K"
               call    LCD_DONNEE
               movlw   "m"
               call    LCD_DONNEE
               movlw   "/"
               call    LCD_DONNEE
               movlw   "h"
               call    LCD_DONNEE
               call    Tempo_1s
               call    Tempo_1s
               return
```

43 - PositionnementA:

Ce programme réalise le positionnement en azimut du suiveur. Il a pour mission de commander le moteur AZIMUT.



```
PositionnementA:              ; Positionnement en Azimut

        movfx   RegAzim               ; RegAzim est transféré dans "x"
        CALLX   ARRONDI               ; "x" est arrondi au multiple de 10
        comf    x+2                   ; Changement du signe de "x"
        comf    x+1                   ;
        comf    x+0                   ;
        incf    x+0                   ;
        skpnz                         ;
        incf    x+1                   ;
        skpnz                         ;
        incf    x+2                   ;
        movfy   PreelA                ; PreelA est transféré dans "y"
        CALLX   FXA_2424S             ; calcul de "y" moins "x" (résultat dans "x")
        movly   d'0',d'0',d'10'       ; valeur "10" dans "y"
        CALLX   FXD_2424S             ; division de "x" par 10
        movfw   x+0                   ; valeur de "x+0" dans w
        btfsc   ZERO                  ; test si la valeur est nulle (pas de saut si valeur nulle)
        return                        ; Pas de positionnement
        btfss   x+1,7                 ; test si le bit 7 de "x+1" est à "1" (saut si oui)
        goto    posiA1                ; Résultat positif: Sens de déplacement vers l'EST
        bcf     FRA                   ; Résultat négatif: Sens de déplacement vers l'OUEST
        bcf     SSA                   ; Mise en route moteur Azimut vers l'OUEST
        btfsc   PA                    ; Attente du capteur azimut à 0
        goto    $-1                   ;
        btfss   PA                    ; lorsque capteurA à 1, attente repassage à 1
        goto    $-1                   ;
        incfsz  x+0                   ; lorsque passage à 1, incrémentation de "x+0"
        goto    $-5                   ;
        goto    posiA2                ; saut vers posiA2 lorsque "x+0" = 0
posiA1
        bsf     FRA                   ; Sens de déplacement vers l'EST
        bcf     SSA                   ; Mise en route moteur Azimut vers l'EST
        btfsc   PA                    ; Attente du capteur azimut à 0
        goto    $-1                   ;
        btfss   PA                    ; lorsque capteurA à 1, attente repassage à 1
        goto    $-1                   ; et bouclage jusqu'à "x+0" égal à 0
        decfsz  x+0                   ; lorsque passage à 1, décrémentation de "x+0"
        goto    $-5                   ;
posiA2
        bsf     SSA                   ; Arrêt du moteur AZIMUT avec freinage
        btfss   FRA                   ; Détection du sens de rotation moteur AZIMUT
        goto    $+3                   ; "
        bcf     FRA                   ; "
        goto    $+2                   ; "
        bsf     FRA                   ; "
        call    Tempo_500ms           ; Pour arrêt entre 2 encoches
        bcf     SSA                   ; Freinage par rotation en sens inverse
        call    Tempo_100ms           ; Durée de la rotation en sens inverse
        bsf     SSA                   ; Arrêt moteur AZIMUT
        movfx   RegAzim               ; RegAzim est transféré dans "x"
        movxf   PreelA                ; RegAzim identique à PreelA
        return
```

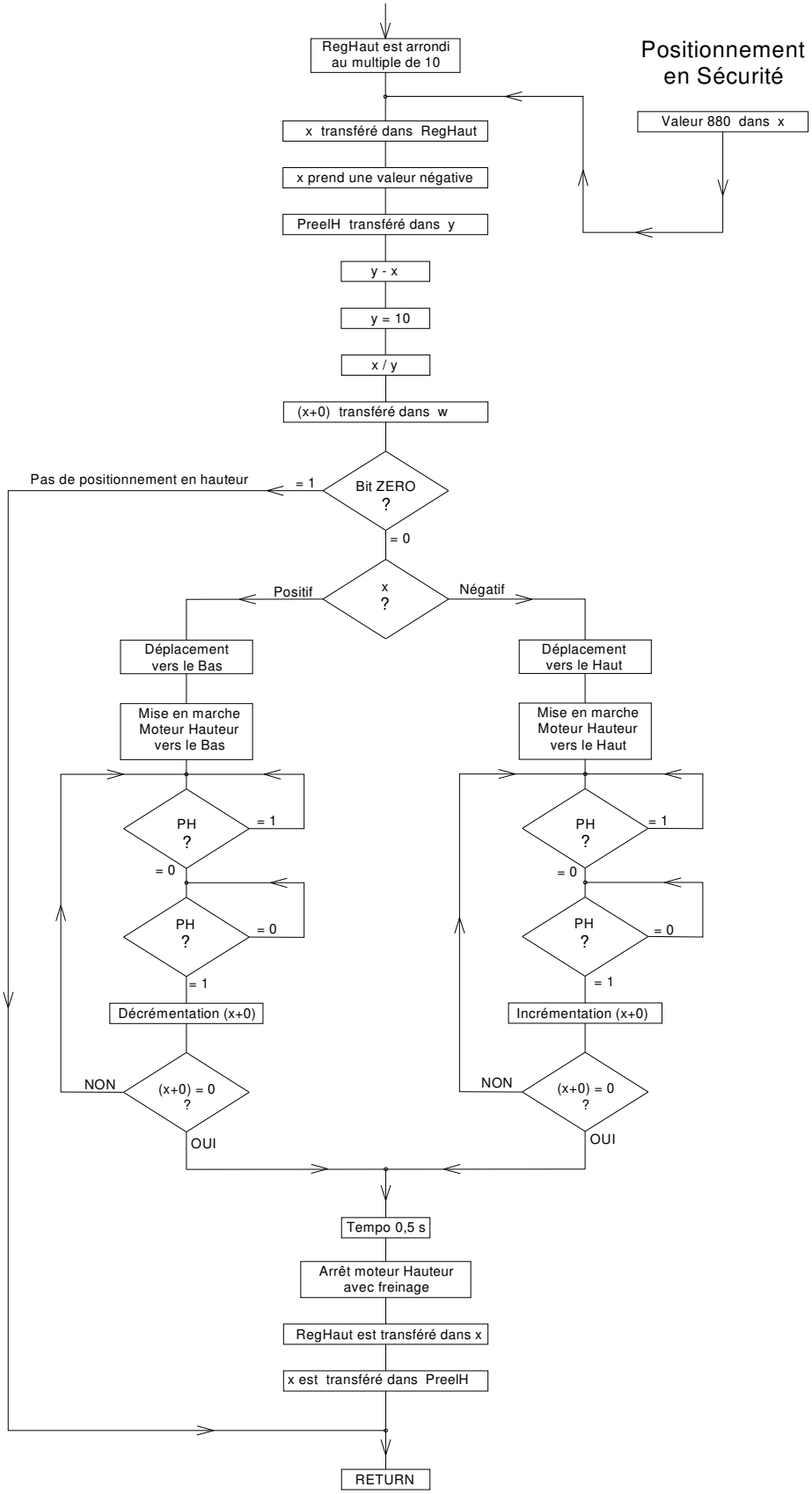
44 - PositionnementS:

Ce programme réalise le positionnement en sécurité du suiveur.
Le programme « PositionnementH » s'exécute immédiatement après.

```
        PositionnementS:                    ; Positionnement en sécurité  H = 88°

                movlx   h'00',h'03',h'70'           ; 880 pour 88° (0x03,0x70) Pente 3,5%
                movxf   RegHaut
```

45 - PositionnementH:

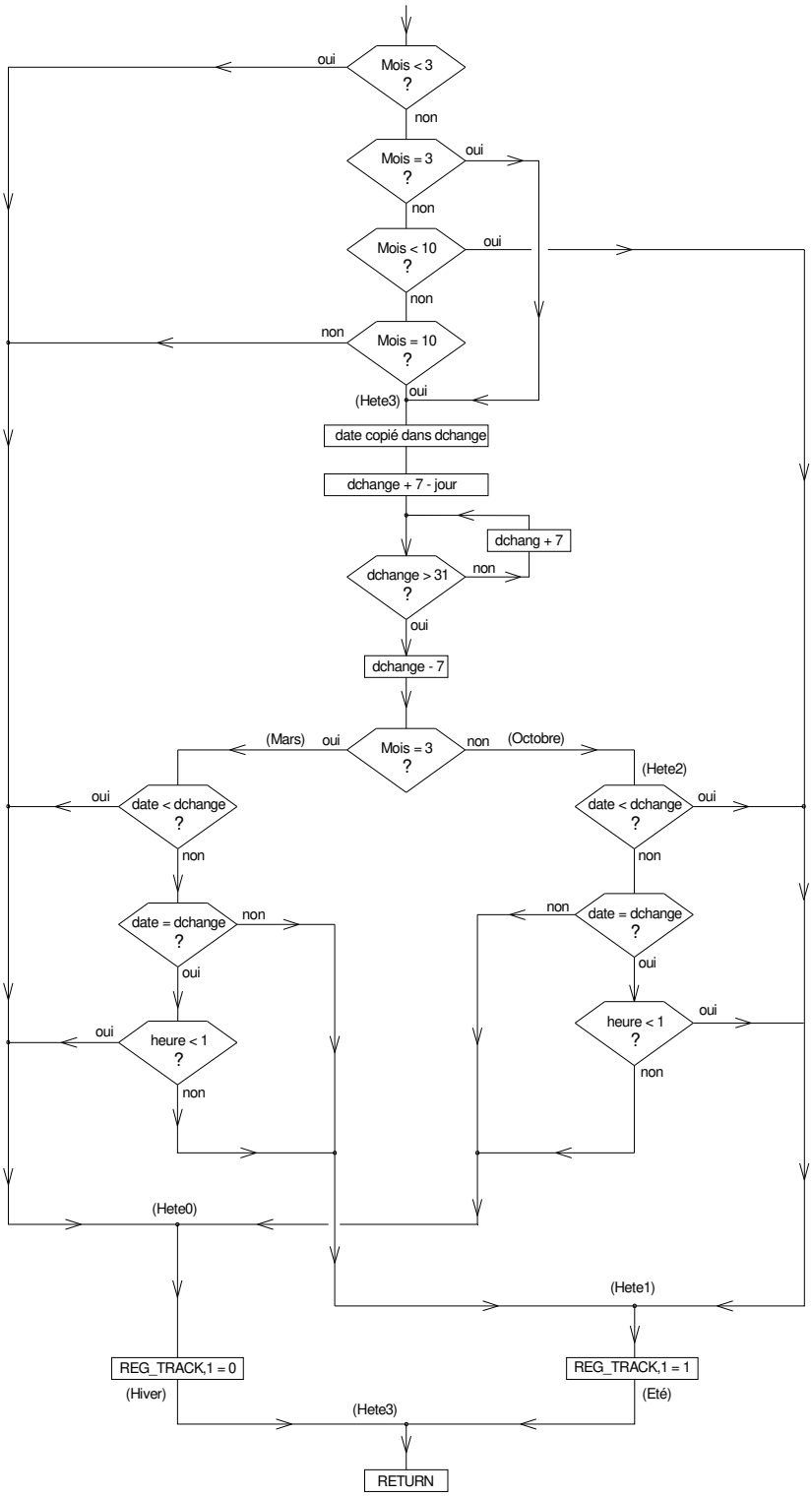
Ce programme réalise le positionnement en hauteur du suiveur en commandant le moteur « Hauteur ».



```
PositionnementH:                    ; Positionnement en Hauteur

              movfx   RegHaut               ; RegHaut est copié dans x
              CALLX   ARRONDI               ; "x" est arrondi au multiple de 10
              comf    x+2                   ; Changement du signe de x
              comf    x+1                   ;
              comf    x+0                   ;
              incf    x+0                   ;
              skpnz                         ;
              incf    x+1                   ;
              skpnz                         ;
              incf    x+2                   ;
              movfy   PreelH                ; La valeur de PreelH est copiée dans y
              CALLX   FXA_2424S             ; Soustraction y - x
              movly   d'0',d'0',d'10'       ; Valeur litérale 10 dans y
              CALLX   FXD_2424S             ; (y - x)/10
              movfw   x+0                   ; Valeur x+0 dans w
              btfsc   ZERO                  ; Test si valeur égale à 0
              return                        ; Si valeur égale à 0 pas de positionnement
              btfss   x+1,7                 ; Test si valeur négative ou positive
              goto    posiH1                ; Valeur positive: saut vers posiH1
              bsf     FRH                   ; Valeur négative: Déplacement vers le haut
              bcf     SSH
              btfsc   PH
              goto    $-1
              btfss   PH
              goto    $-1
              incfsz  x+0                   ; Incrémentation jusqu'à valeur nulle puis saut
              goto    $-5
              goto    posiH2                ; Saut vers fin du déplacement
       posiH1
              bcf     FRH
              bcf     SSH
              btfsc   PH
              goto    $-1
              btfss   PH
              goto    $-1
              decfsz  x+0                   ; Décrémentation jusqu'à valeur nulle puis saut
              goto    $-5
       posiH2
              bsf     SSH                   ; Arrêt du moteur HAUTEUR avec freinage
              btfss   FRH                   ; Détection du sens de rotation moteur HAUTEUR
              goto    $+3                   ; "
              bsf     FRH                   ; "
              goto    $+2                   ; "
              bcf     FRH                   ; "
              call    Tempo_500ms           ; Pour arrêt entre 2 encoches ( 0,5s )
              bcf     SSH                   ; Freinage par rotation en sens inverse
              call    Tempo_30ms            ; Durée de la rotation en sens inverse 60 ms
              call    Tempo_30ms            ;
              bsf     SSH                   ; Arrêt moteur HAUTEUR
              movfx   RegHaut               ; RegHaut est copié dans x
              movxf   PreelH                ; regHaut identique à PreelH
              return
```

47 - HEURE_ETE:

Exécute le passage automatique de l'heure d'été à l'heure d'hiver et inversement.
Le programme s'appui sur les valeurs fournies par le circuit RTCC MCP79410 pour connaître en particulier le dernier dimanche de mars et le dernier dimanche d'octobre.
En mars, le changement se fait à 1h00 (heure solaire) ; ce qui correspond à 2H00 (heure légale pour la France).
1h00 est ajoutée à l'heure légale et on passe subitement de 2h00 (heure légale) à 3h00 (heure légale).
En octobre, le changement se fait à 1H00 (heure solaire) ; ce qui correspond à 3H00 (heure légale pour la France).
1h00 est retranchée à l'heure légale. et on passe subitement de 3h00 (heure légale) à 2h00 (heure légale).
Le programme positionne le bit 1 du registre REG_TRACK.
Si le bit 1 du registre est à zéro, alors nous sommes en heure d'hiver.
Si le bit 1 du registre est à 1, alors nous sommes en heure d'été.
Ce programme ne sert en réalité que pour afficher l'heure légale sur l'afficheur LCD.

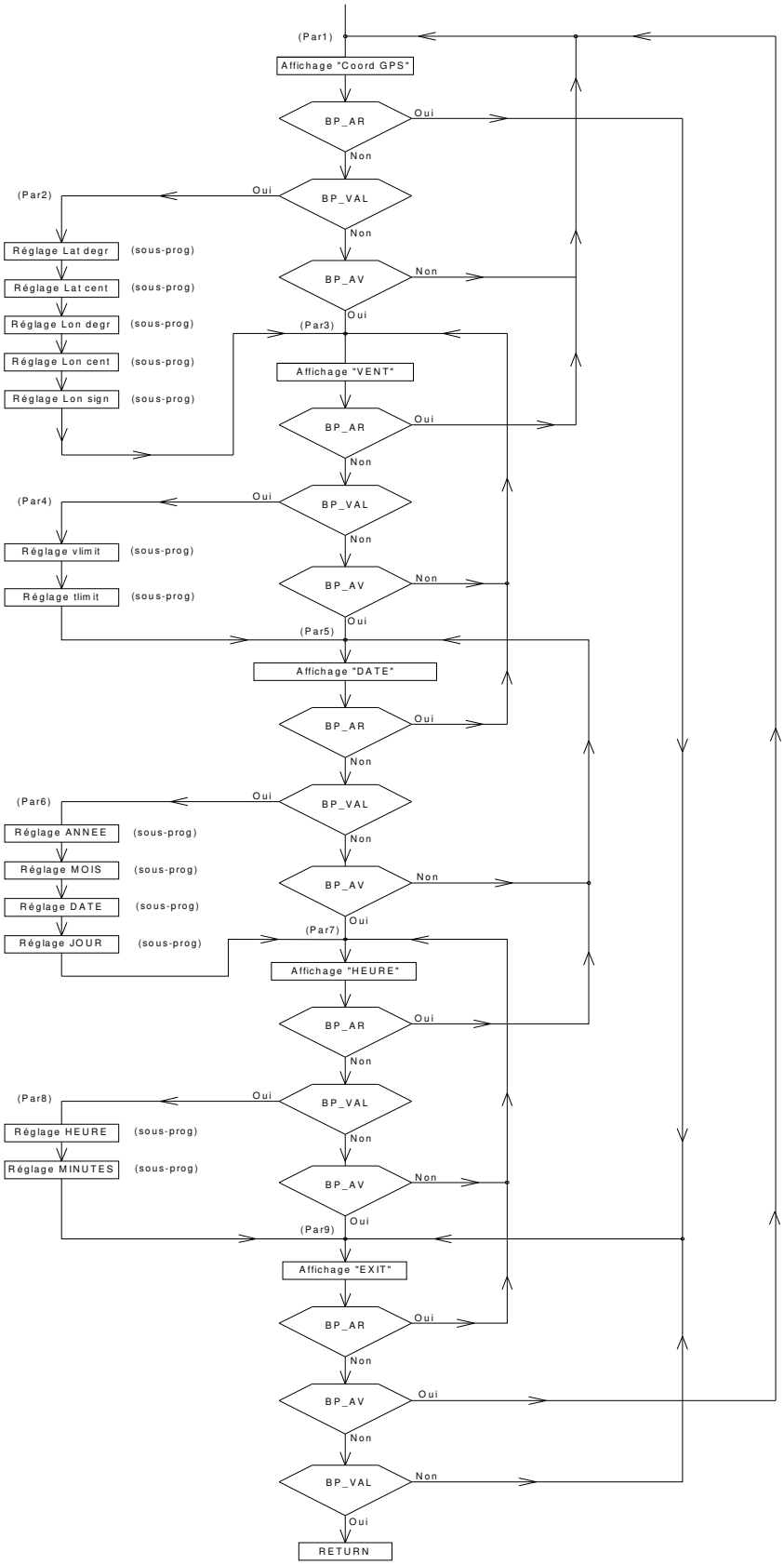


```
HEURE_ETE:                  ; Passage automatique en heure été/hiver à 01H00 (heure solaire)
                                    ; le dernier dimanche de Mars et dernier dimanche d'octobre

        movlw       d'3'
        subwf       mois,0
        btfss       CARRY           ; Test si mois est inférieur à mars
        goto        Hete0           ; Oui: Vers heure d'hiver
        btfsc       ZERO            ; Test si mois est égal à mars
        goto        Hete3           ; Oui: Saut vers routine mars et octobre
        movlw       d'10'           ; Non: Test si mois d'octobre
        subwf       mois,0
        btfss       CARRY           ; Test si mois est inférieur à octobre
        goto        Hete1           ; Oui: Vers heure d'été
        btfss       ZERO            ; Test si mois est égal à octobre
        goto        Hete0           ; Non: vers heure d'hiver

Hete3
        movf        date,0          ; date  =>  w
        movwf       dchange         ; w  =>  dchange   (dchange: date de changement d'heure)
        addlw       d'7'            ; w + 7 (résultat dans w)
        movwf       dchange         ; w  =>  dchange   (dchange a été incrémenté de 7)
        movf        jour,0          ; jour  =>  w
        subwf       dchange,0       ; dchange - w   ( dchange - jour)
        movwf       dchange         ; w  =>  dchange  ( w inchangé )
        sublw       d'31'           ; 31 - w  (résultat dans w) positif ou nul: C=1, négatif: C=0
        btfss       CARRY           ; Recherche dernier dimanche du mois (dchange)
        goto        $+4                     ;
        movf        dchange,0       ;
        addlw       d'7'            ;
        goto        $-6                     ;
        movlw       d'7'            ;
        subwf       dchange,1       ;
        movf        mois,0          ; mois  =>  w
        sublw       d'3'            ; 3 - w  (3 moins mois)
        btfss       ZERO            ; test si mois de mars
        goto        Hete2           ; Non: Saut vers mois d'octobre (Hete2)
        movf        dchange,0       ; Oui: mois de mars. dchange  =>  w
        subwf       date,0          ; date - w  (date - dchange), résultat dans w
                                    ; résultat positif ou nul: C=1, négatif : C=0
        btfss       CARRY           ; test si la date est inférieure à la date de changement d'heure
        goto        Hete0           ; Non: Saut vers heure d'hiver (Hete0)
        btfss       ZERO            ; test si la date est égale à la date de changement d'heure
        goto        Hete1           ; Non: Saut vers heure d'été (Hete1)
        movlw       d'1'
        subwf       heure,0
        btfss       CARRY           ; test si l'heure est inférieure à 2
        goto        Hete0           ; Non: Saut vers heure d'hiver (Hete0)
        btfss       ZERO            ; test si l'heure est égale à 2
        goto        Hete1           ; Non: Saut vers heure d'été (Hete1)

Hete2
        movf        dchange,0       ; dchange  =>  w
        Subwf       date,0          ; mois d'octobre,  Date - w (date - dchange), résultat dans w
                                    ; résultat positif ou nul: C=1, négatif: C=0
        btfss       CARRY           ; test si date est inférieure à la date de changement d'heure
        goto        Hete1           ; Non: Saut vers heure d'été (Hete1)
        btfss       ZERO            ; test si la date est égale à la date de changement d'heure
        goto        Hete0           ; Non: Saut vers heure d'hiver (Hete0)
        movlw       d'1'            ;
        subwf       heure,0         ;
        btfss       CARRY           ; test si l'heure est inférieure à 2
        goto        Hete1           ; Non: Saut vers heure d'été
        btfss       ZERO            ; test si l'heure est égale à 2
        goto        Hete0           ; Saut vers heure d'hiver

Hete0
        Bcf                 REG_TRACK,1     ; Hiver  (REG_TRACK,1 = 0)
        goto        $+2

Hete1
        Bsf                 REG_TRACK,1     ; Eté  (REG_TRACK,1 = 1)
        return
```

48 - REGL_PAR:

Permet le réglage des paramètres
Cette routine exécute séquentiellement :

le paramétrage des coordonnées GPS du suiveur : latitude et longitude (précision au centième de degrés).
le paramétrage du seuil de la vitesse du vent et du temps de maintien en position sécurité.
le paramétrage de la date (Année, Mois, Jour du mois et Jour de la semaine).
le paramétrage de l'Heure (Heures, Minutes et zéro seconde).



```
REGL_PAR :                  ; Activation du MENU pour le réglage des paramètres et mise à l'heure

        Par1
                call    LCD_EFFACE
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "G"                             ; Affichage "GPS"
                call    LCD_DONNEE
                movlw   "P"
                call    LCD_DONNEE
                movlw   "S"
                call    LCD_DONNEE
                call    Tempo_200ms
                btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
                goto    $+2
                goto    Par9
                btfsc   BP_VAL                  ; Validation
                goto    $+2
                goto    Par2
                btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
                goto    Par1
                goto    Par3
        Par2
                call    REGL_LATD               ; Parammétrage des degrés de la latitude
                call    REGL_LATC               ; Parammétrage des centièmes de degrés de la latitude
                call    REGL_LOND               ; Parammétrage des degrés de la longitude
                call    REGL_LONC               ; Parammétrage des centièmes de degrés de la longitude
                call    REGL_LONS               ; Paramétrage signe de la longitude

; Réglage du seuil vents forts
        Par3
                call    LCD_EFFACE
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "V"                     ; Affichage "VENT"
                call    LCD_DONNEE
                movlw   "E"
                call    LCD_DONNEE
                movlw   "N"
                call    LCD_DONNEE
                movlw   "T"
                call    LCD_DONNEE
                call    Tempo_200ms
                btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
                goto    $+2
                goto    Par1
                btfsc   BP_VAL                  ; Validation
                goto    $+2
                goto    Par4
                btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
                goto    Par3
                goto    Par5
        Par4
                call    REGL_VLIM
                call    REGL_TLIM
        Par5
                call    LCD_EFFACE
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "D"                     ; Affichage "DATE"
                call    LCD_DONNEE              ;
                movlw   "A"
                call    LCD_DONNEE
                movlw   "T"
                call    LCD_DONNEE
                movlw   "E"
                call    LCD_DONNEE
                call    Tempo_200ms
                btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
                goto    $+2
                goto    Par3
                btfsc   BP_VAL                  ; Validation
                goto    $+2
                goto    Par6
                btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
                goto    Par5
                goto    Par7
        Par6
                call    REGL_ANN                ; Réglage de l'ANNEE   (2010 à 2099)
                call    REGL_MOI                ; Réglage du MOIS      (1 à 12)
                call    REGL_DAT                ; Réglage de la DATE   (1 à 31)
```

```
        call    REGL_JOU            ; Réglage du JOUR de la semaine  (1 à 7)
Par7
        call    LCD_EFFACE
        movlw   d'0'
        call    LCD_LOCATE
        movlw   "H"                 ; Affichage: "HEURE"
        call    LCD_DONNEE          ;
        movlw   "E"
        call    LCD_DONNEE
        movlw   "U"
        call    LCD_DONNEE
        movlw   "R"
        call    LCD_DONNEE
        movlw   "E"
        call    LCD_DONNEE
        call    Tempo_200ms
        btfsc   BP_AR               ; Défilement de la valeur en décrémentation
        goto    $+2
        goto    Par5
        btfsc   BP_VAL              ; Validation
        goto    $+2
        goto    Par8
        btfsc   BP_AV               ; Défilement de la valeur en incrémentation
        goto    Par7
        goto    Par9
Par8
        call    REGL_HEU            ; Réglage de l'HEURE   (0 à 23)
        call    REGL_MIN            ; Réglage des MINUTES   (0 à 59)
Par9
        call    LCD_EFFACE
        movlw   d'0'
        call    LCD_LOCATE
        movlw   "E"
        call    LCD_DONNEE
        movlw   "X"
        call    LCD_DONNEE
        movlw   "I"
        call    LCD_DONNEE
        movlw   "T"
        call    LCD_DONNEE
        call    Tempo_200ms
        btfsc   BP_AR               ; Défilement de la valeur en décrémentation
        goto    $+2
        goto    Par7
        btfsc   BP_AV               ; Défilement de la valeur en incrémentation
        goto    $+2
        goto    Par1
        btfsc   BP_VAL              ; Validation
        goto    Par9
        return
```

49 - REGL_LATD:

Exécute le programme de réglage de la latitude pour les degrés.

```
REGL_LATD:          ; Paramétrage de la latitude (degrés)

        movf    latitude+0,0
        movwf   cmpt0
Latd1
        movwf   x+0                 ; Affichage: "LatD:" sur la 1ère ligne
        call    LCD_EFFACE
        movlw   d'0'
        call    LCD_LOCATE
        movlw   "L"
        call    LCD_DONNEE
        movlw   "a"
        call    LCD_DONNEE
        movlw   "t"
        call    LCD_DONNEE
        movlw   "D"
        call    LCD_DONNEE
        movlw   ":"
        call    LCD_DONNEE
        movlw   d'10'
        CALLX   AFFICHE_24BS        ; Affichage du paramètre
        call    Tempo_350ms
Latd2
        btfsc   BP_AV               ; Défilement de la valeur en incrémentation
```

```
                goto    Latd3
                goto    Latd5
        Latd3
                btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
                goto    Latd4
                goto    Latd7
        Latd4
                btfsc   BP_VAL                  ; Validation pour écriture dans la mémoire EEPROM
                goto    Latd2
                goto    Latd9
        Latd5
                movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
                sublw   d'88'                   ; Pour limitation à 89 degrés
                btfss   CARRY
                goto    Latd6
                incf    cmpt0
                movf    cmpt0,0
                goto    Latd1
        Latd6
                movlw   d'0'                    ; Pour limitation à 0 degré
                movwf   cmpt0
                goto    Latd1
        Latd7
                movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
                sublw   d'0'                    ; Pour limitation à 0 degré
                btfsc   CARRY
                goto    Latd8
                decf    cmpt0
                movf    cmpt0,0
                goto    Latd1
        Latd8
                movlw   d'89'                   ; Pour limitation à 89 degrés
                movwf   cmpt0
                goto    Latd1
        Latd9
                call    I2C_START_              ; BP_VAL actionné (écriture EEPROM)
                movlw   b'10100000'             ; Adresse EEPROM en lecture
                call    I2C_SEND_BYTE
                movlw   0x6B                    ; 1er octet de l'adresse 0x6B,0xD0
                call    I2C_SEND_BYTE
                movlw   0xD0                    ; 2ème octet de l'adresse 0x6B,0xD0
                call    I2C_SEND_BYTE
                movf    cmpt0,0
                movwf   latitude+0              ; valeur du registre cmpt0 vers w
                call    I2C_SEND_BYTE           ; envoi de l'octet en mémoire EEPROM
                call    I2C_STOP_
                call    Tempo_500ms
                return
```

50 - REGL_LATC:

Exécute le programme de réglage de la latitude pour les centièmes degrés.

```
REGL_LATC:              ; Paramétrage de la latitude (centièmes)

                movf    latitude+1,0
                movwf   cmpt0
        Latc1
                movwf   x+0                     ; Affichage: "LatC:" sur la 1ère ligne
                call    LCD_EFFACE
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "L"
                call    LCD_DONNEE
                movlw   "a"
                call    LCD_DONNEE
                movlw   "t"
                call    LCD_DONNEE
                movlw   "C"
                call    LCD_DONNEE
                movlw   ":"
                call    LCD_DONNEE
                movlw   d'10'
                CALLX   AFFICHE_24BS            ; Affichage du paramètre
                call    Tempo_350ms
        Latc2
                btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
                goto    Latc3
```

```
                goto    Latc5
        Latc3
                btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
                goto    Latc4
                goto    Latc7
        Latc4
                btfsc   BP_VAL                  ; Validation pour écriture dans la mémoire EEPROM
                goto    Latc2
                goto    Latc9
        Latc5
                movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
                sublw   d'98'                   ; Pour limitation à 99 centièmes
                btfss   CARRY
                goto    Latc6
                incf    cmpt0
                movf    cmpt0,0
                goto    Latc1
        Latc6
                movlw   d'0'                    ; Pour limitation à 0 centième
                movwf   cmpt0
                goto    Latc1
        Latc7
                movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
                sublw   d'0'                    ; Pour limitation à 0 centième
                btfsc   CARRY
                goto    Latc8
                decf    cmpt0
                movf    cmpt0,0
                goto    Latc1
        Latc8
                movlw   d'99'                   ; Pour limitation à 99 centièmes
                movwf   cmpt0
                goto    Latc1
        Latc9
                call    I2C_START_              ; BP_VAL actionné (écriture EEPROM)
                movlw   b'10100000'             ; Adresse EEPROM en lecture
                call    I2C_SEND_BYTE
                movlw   0x6B                    ; 1er octet de l'adresse 0x6B,0xD1
                call    I2C_SEND_BYTE
                movlw   0xD1                    ; 2ème octet de l'adresse 0x6B,0xD1
                call    I2C_SEND_BYTE
                movf    cmpt0,0
                movwf   latitude+1              ; valeur du registre cmpt0 vers w
                call    I2C_SEND_BYTE           ; envoi de l'octet en mémoire EEPROM
                call    I2C_STOP_
                call    Tempo_500ms
                return
```

51 - REGL_LOND:

Exécute le programme de réglage de la longitude pour les degrés.

```
REGL_LOND:              ; Paramétrage de la longitude (degrés)

                movf    longitude+0,0
                movwf   cmpt0
        Lond1
                movwf   x+0                     ; Affichage: "LonD:" sur la 1ère ligne
                call    LCD_EFFACE
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "L"
                call    LCD_DONNEE
                movlw   "o"
                call    LCD_DONNEE
                movlw   "n"
                call    LCD_DONNEE
                movlw   "D"
                call    LCD_DONNEE
                movlw   ":"
                call    LCD_DONNEE
                movlw   d'10'
                CALLX   AFFICHE_24BS            ; Affichage du paramètre
                call    Tempo_350ms
        Lond2
                btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
                goto    Lond3
                goto    Lond5
        Lond3
```

```
                btfsc   BP_AR                  ; Défilement de la valeur en décrémentation
                goto    Lond4
                goto    Lond7
        Lond4
                btfsc   BP_VAL                 ; Validation pour écriture dans la mémoire EEPROM
                goto    Lond2
                goto    Lond9
        Lond5
                movf    cmpt0,0                ; BP_AV actionné (incrémentation)
                sublw   d'88'                  ; Pour limitation à 89 degrés
                btfss   CARRY
                goto    Latd6
                incf    cmpt0
                movf    cmpt0,0
                goto    Lond1
        Lond6
                movlw   d'0'                   ; Pour limitation à 0 degré
                movwf   cmpt0
                goto    Lond1
        Lond7
                movf    cmpt0,0                ; BP_AR actionné (décrémentation)
                sublw   d'0'                   ; Pour limitation à 0 degré
                btfsc   CARRY
                goto    Lond8
                decf    cmpt0
                movf    cmpt0,0
                goto    Lond1
        Lond8
                movlw   d'89'                  ; Pour limitation à 89 degrés
                movwf   cmpt0
                goto    Lond1
        Lond9
                call    I2C_START_             ; BP_VAL actionné (écriture EEPROM)
                movlw   b'10100000'            ; Adresse EEPROM en lecture
                call    I2C_SEND_BYTE
                movlw   0x6B                   ; 1er octet de l'adresse 0x6B,0xD2
                call    I2C_SEND_BYTE
                movlw   0xD2                   ; 2ème octet de l'adresse 0x6B,0xD2
                call    I2C_SEND_BYTE
                movf    cmpt0,0
                movwf   longitude+0            ; valeur du registre cmpt0 vers w
                call    I2C_SEND_BYTE          ; envoi de l'octet en mémoire EEPROM
                call    I2C_STOP_
                call    Tempo_500ms
                return
```

52 - REGL_LONC:

Exécute le programme de réglage de la longitude pour les centièmes de degrés.

```
REGL_LONC:              ; Paramétrage de la longitude (centièmes)

                movf    longitude+1,0
                movwf   cmpt0
        Lonc1
                movwf   x+0                    ; Affichage: "LonC:" sur la 1ère ligne
                call    LCD_EFFACE
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "L"
                call    LCD_DONNEE
                movlw   "o"
                call    LCD_DONNEE
                movlw   "n"
                call    LCD_DONNEE
                movlw   "C"
                call    LCD_DONNEE
                movlw   ":"
                call    LCD_DONNEE
                movlw   d'10'
                CALLX   AFFICHE_24BS           ; Affichage du paramètre
                call    Tempo_350ms
        Lonc2
                btfsc   BP_AV                  ; Défilement de la valeur en incrémentation
                goto    Lonc3
                goto    Lonc5
        Lonc3
                btfsc   BP_AR                  ; Défilement de la valeur en décrémentation
                goto    Lonc4
```

```
        goto    Lonc7
Lonc4
        btfsc   BP_VAL              ; Validation pour écriture dans la mémoire EEPROM
        goto    Lonc2
        goto    Lonc9
Lonc5
        movf    cmpt0,0             ; BP_AV actionné (incrémentation)
        sublw   d'98'               ; Pour limitation à 99 centièmes
        btfss   CARRY
        goto    Lonc6
        incf    cmpt0
        movf    cmpt0,0
        goto    Lonc1
Lonc6
        movlw   d'0'                ; Pour limitation à 0 centième
        movwf   cmpt0
        goto    Lonc1
Lonc7
        movf    cmpt0,0             ; BP_AR actionné (décrémentation)
        sublw   d'0'                ; Pour limitation à 0 centième
        btfsc   CARRY
        goto    Lonc8
        decf    cmpt0
        movf    cmpt0,0
        goto    Lonc1
Lonc8
        movlw   d'99'               ; Pour limitation à 99 centièmes
        movwf   cmpt0
        goto    Lonc1
Lonc9
        call    I2C_START_          ; BP_VAL actionné (écriture EEPROM)
        movlw   b'10100000'         ; Adresse EEPROM en lecture
        call    I2C_SEND_BYTE
        movlw   0x6B                ; 1er octet de l'adresse 0x6B,0xD3
        call    I2C_SEND_BYTE
        movlw   0xD3                ; 2ème octet de l'adresse 0x6B,0xD3
        call    I2C_SEND_BYTE
        movf    cmpt0,0
        movwf   longitude+1         ; valeur du registre cmpt0 vers w
        call    I2C_SEND_BYTE       ; envoi de l'octet en mémoire EEPROM
        call    I2C_STOP_
        call    Tempo_500ms
        return
```

53 - REGL_LONS:

Exécute le programme de réglage de la longitude pour l'OUEST ou l'EST
Si longitude : OUEST « longitude+2 » est mis à 0. Si longitude EST « longitude+2 » est mis à 1

```
REGL_LONS:                  ;Signe de la longitude: OUEST=0, EST=1

        movf    longitude+2,0
        movwf   cmpt0
Lons1
        movwf   x+0                 ; Affichage: "LSign" sur la 1ère ligne
        call    LCD_EFFACE
        movlw   d'0'
        call    LCD_LOCATE
        movlw   "L"
        call    LCD_DONNEE
        movlw   "S"
        call    LCD_DONNEE
        movlw   "i"
        call    LCD_DONNEE
        movlw   "g"
        call    LCD_DONNEE
        movlw   "n"
        call    LCD_DONNEE
        movlw   d'10'
        CALLX   AFFICHE_24BS        ; Affichage du paramètre
        call    Tempo_350ms
Lons2
        btfsc   BP_AV               ; Défilement de la valeur en incrémentation
        goto    Lons3
        goto    Lons5
Lons3
        btfsc   BP_AR               ; Défilement de la valeur en décrémentation
        goto    Lons4
        goto    Lons7
```

```
        Lons4
                btfsc   BP_VAL                  ; Validation pour écriture dans la mémoire EEPROM
                goto    Lons2
                goto    Lons9
        Lons5
                movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
                sublw   d'0'                    ; Pour limitation à 1
                btfss   CARRY
                goto    Lons6
                incf    cmpt0
                movf    cmpt0,0
                goto    Lons1
        Lons6
                movlw   d'0'                    ; Pour limitation à 0
                movwf   cmpt0
                goto    Lons1
        Lons7
                movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
                sublw   d'0'                    ; Pour limitation à 0
                btfsc   CARRY
                goto    Lons8
                decf    cmpt0
                movf    cmpt0,0
                goto    Lons1
        Lons8
                movlw   d'1'                    ; Pour limitation à 1
                movwf   cmpt0
                goto    Lons1
        Lons9
                call    I2C_START_              ; BP_VAL actionné (écriture EEPROM)
                movlw   b'10100000'             ; Adresse EEPROM en lecture
                call    I2C_SEND_BYTE
                movlw   0x6B                    ; 1er octet de l'adresse 0x6B,0xD4
                call    I2C_SEND_BYTE
                movlw   0xD4                    ; 2ème octet de l'adresse 0x6B,0xD4
                call    I2C_SEND_BYTE
                movf    cmpt0,0                 ; Transfert de cmpt0 dans w
                movwf   longitude+2             ; valeur de w vers le registre cmpt3
                call    I2C_SEND_BYTE           ; envoi de l'octet en mémoire EEPROM
                call    I2C_STOP_
                call    Tempo_500ms
                return
```

51 - REGL_VLIM:

Permet le réglage du seuil à partir duquel le suiveur va se placer en position de sécurité.
(Vitesse du vent exprimée en km/h)

```
REGLAGE_VLIM:                   ; Paramétrage seuil de sécurité vents forts

                movf    vlimit,0
                movwf   cmpt0
        Vlim1
                movwf   x+0             ; Affichage: "VLIMIT:" sur la 1ère ligne
                call    LCD_EFFACE
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "V"
                call    LCD_DONNEE
                movlw   "L"
                call    LCD_DONNEE
                movlw   "I"
                call    LCD_DONNEE
                movlw   "M"
                call    LCD_DONNEE
                movlw   "I"
                call    LCD_DONNEE
                movlw   "T"
                movlw   ":"
                call    LCD_DONNEE
                movlw   d'10'
                CALLX   AFFICHE_24BS            ; Affichage du paramètre
                call    Tempo_200ms
        Vlim2
                btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
                goto    Vlim3
                goto    Vlim5
        Vlim3
                btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
```

```
                goto    Vlim4
                goto    Vlim7
        Vlim4
                btfsc   BP_VAL                  ; Validation de la valeur pour écriture dans le RTCC
                goto    Vlim2
                goto    Vlim9
        Vlim5
                movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
                sublw   d'49'                   ; Pour limitation à 50 Km/h (limite maxi 50)
                btfss   CARRY
                goto    Vlim6
                incf    cmpt0
                movf    cmpt0,0
                goto    Vlim1
        Vlim6
                movlw   d'10'
                movwf   cmpt0
                goto    Vlim1
        Vlim7
                movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
                sublw   d'10'                   ; Pour limitation à 10 Km/h (limite mini 10)
                btfsc   CARRY
                goto    Vlim8
                decf    cmpt0
                movf    cmpt0,0
                goto    Vlim1
        Vlim8
                movlw   d'50'
                movwf   cmpt0
                goto    Vlim1
        Vlim9
                call    I2C_START_              ; BP_VAL actionné (écriture EEPROM)
                movlw   b'10100000'             ; Adresse EEPROM en lecture
                call    I2C_SEND_BYTE
                movlw   0x6B                    ; 1er octet de l'adresse 0x6B,0xD3
                call    I2C_SEND_BYTE
                movlw   0xD3                    ; 2ème octet de l'adresse 0x6B,0xD3
                call    I2C_SEND_BYTE
                movf    cmpt0,0                 ; valeur du registre cmpt0 vers w
                movwf   vlimit
                call    I2C_SEND_BYTE           ; envoi de l'octet en mémoire EEPROM
                call    I2C_STOP_
                call    Tempo_500ms
                return
```

52 - REGL_TLIM:

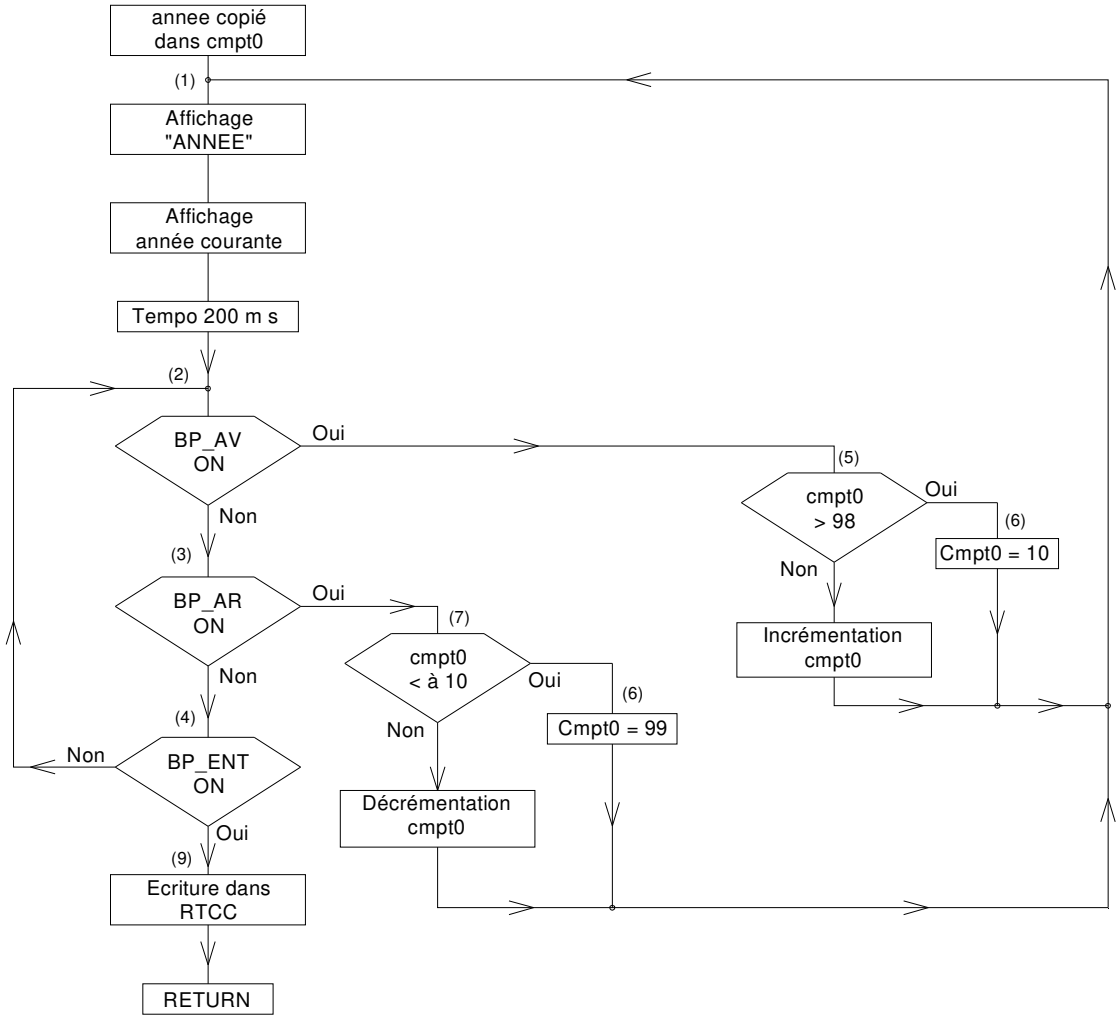
Permet le réglage du temps pendant lequel le suiveur va rester en position de sécurité lorsque la vitesse du vent va redescendre en dessous du seuil de mise en sécurité.

```
REGL_TLIM:                      ; Paramétrage tempo de sécurité vents forts

                movf    tlimit,0
                movwf   cmpt0
        Tlim1
                movwf   x+0                     ; Affichage: "TLIMIT:" sur la 1ère ligne
                call    LCD_EFFACE
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "T"
                call    LCD_DONNEE
                movlw   "L"
                call    LCD_DONNEE
                movlw   "I"
                call    LCD_DONNEE
                movlw   "M"
                call    LCD_DONNEE
                movlw   "I"
                call    LCD_DONNEE
                movlw   "T"
                movlw   ":"
                call    LCD_DONNEE
                movlw   d'10'
                CALLX   AFFICHE_24BS            ; Affichage du paramètre
                call    Tempo_200ms
        Tlim2
                btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
                goto    Tlim3
                goto    Tlim5
```

```
Tlim3
        btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
        goto    Tlim4
        goto    Tlim7
Tlim4
        btfsc   BP_VAL                  ; Validation de la valeur pour écriture dans le RTCC
        goto    Tlim2
        goto    Tlim9
Tlim5
        movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
        sublw   d'58'                   ; Pour limitation à 59 mn (limite maxi 59 mn)
        btfss   CARRY
        goto    Tlim6
        incf    cmpt0
        movf    cmpt0,0
        goto    Tlim1
Tlim6
        movlw   d'2'
        movwf   cmpt0
        goto    Tlim1

Tlim7
        movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
        sublw   d'2'                    ; Pour limitation à 2 mn (limite mini 2 mn)
        btfsc   CARRY
        goto    Tlim8
        decf    cmpt0
        movf    cmpt0,0
        goto    Tlim1
Tlim8
        movlw   d'59'
        movwf   cmpt0
        goto    Tlim1
Tlim9
        call    I2C_START_              ; BP_VAL actionné (écriture EEPROM)
        movlw   b'10100000'             ; Adresse EEPROM en lecture
        call    I2C_SEND_BYTE
        movlw   0x6B                    ; 1er octet de l'adresse 0x6B,0xD6
        call    I2C_SEND_BYTE
        movlw   0xD6                    ; 2ème octet de l'adresse 0x6B,0xD6
        call    I2C_SEND_BYTE
        movf    cmpt0,0                 ; valeur du registre cmpt0 vers w
        movwf   tlimit
        call    I2C_SEND_BYTE           ; envoi de l'octet en mémoire EEPROM
        call    I2C_STOP_
        call    Tempo_500ms
        return
```

53 - REGL_ANN:

Exécute le paramétrage de la date au niveau de l'année.
Les valeurs de paramétrage vont de 10 à 99; Ce qui signifie que l'année ne peut être réglée que de 2010 à 2099.

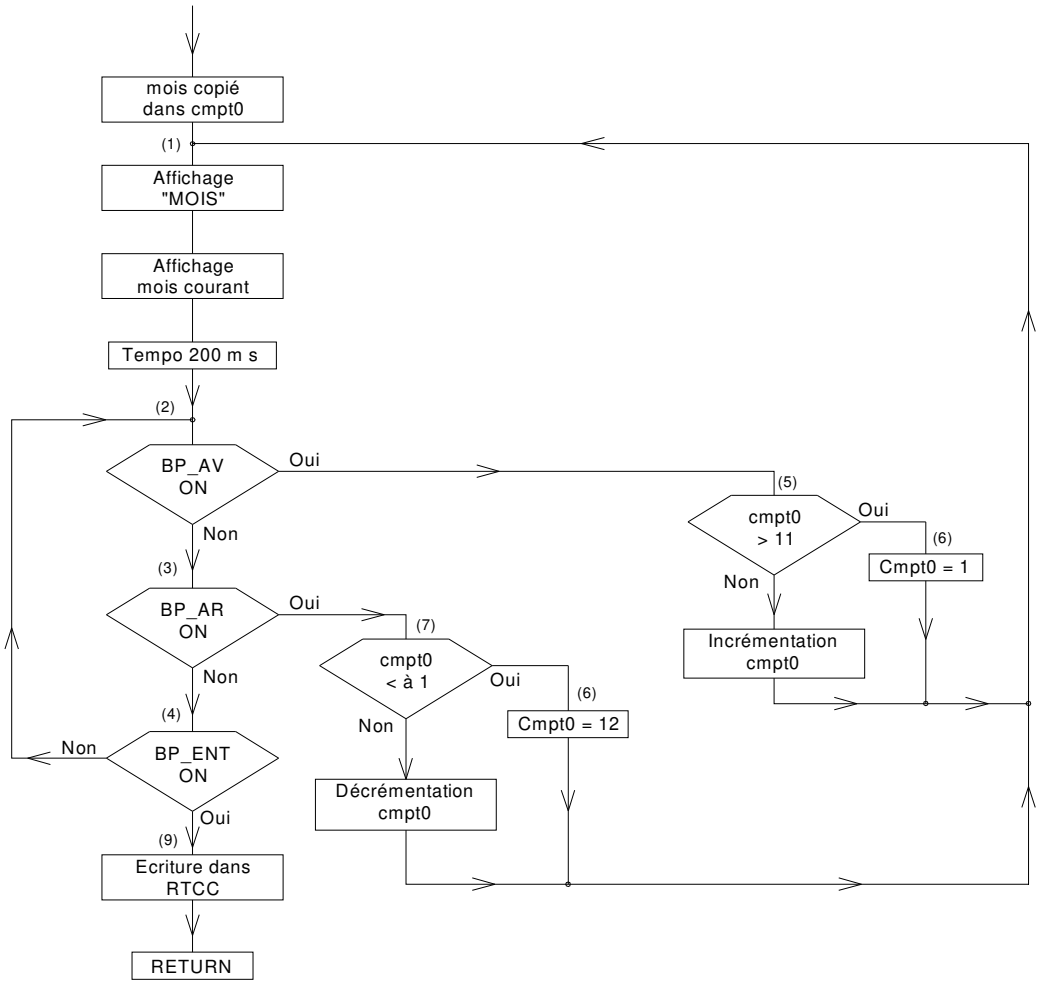



```
REGL_ANN:                       ; Réglage ANNEE

                movf    annee,0
                movwf   cmpt0

        Ann1
                movwf   x+0                     ; Affichage: "ANNEE:" sur la 1ère ligne
                call    LCD_EFFACE
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "A"
                call    LCD_DONNEE
                movlw   "N"
                call    LCD_DONNEE
                movlw   "N"
                call    LCD_DONNEE
                movlw   "E"
                call    LCD_DONNEE
                movlw   "E"
                call    LCD_DONNEE
                movlw   ":"
                call    LCD_DONNEE
                movlw   " "
                call    LCD_DONNEE
                movlw   "2"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movlw   d'10'
                CALLX   AFFICHE_24BS            ; Affichage de l'année courante
                call    Tempo_200ms
```

```
Ann2
        btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
        goto    Ann3
        goto    Ann5

Ann3
        btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
        goto    Ann4
        goto    Ann7

Ann4
        btfsc   BP_VAL                  ; Validation de la valeur pour écriture dans le RTCC
        goto    Ann2
        goto    Ann9

Ann5
        movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
        sublw   d'98'                   ; Pour limitation à 2099 (année de 2010 à 2099)
        btfss   CARRY
        goto    Ann6
        incf    cmpt0
        movf    cmpt0,0
        goto    Ann1

Ann6
        movlw   d'10'
        movwf   cmpt0
        goto    Ann1

Ann7
        movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
        sublw   d'10'                   ; Pour limitation à 2010
        btfsc   CARRY
        goto    Ann8
        decf    cmpt0
        movf    cmpt0,0
        goto    Ann1

Ann8
        movlw   d'99'
        movwf   cmpt0
        goto    Ann1

Ann9
        call    I2C_START_              ; BP_VAL actionné (écriture RTCC)
        movlw   ADRS_RTCC+0
        call    I2C_SEND_BYTE
        movlw   0x06
        call    I2C_SEND_BYTE
        movf    cmpt0,0
        call    conv_DH                 ; Conversion binaire/Hexa en BCD
        movwf   cmpt0
        call    I2C_SEND_BYTE
        call    I2C_STOP_
        call    Tempo_500ms
        return
```

54 - REGL_MOI:

Exécute le paramétrage de la date au niveau du mois.
Les valeurs de paramétrage vont de 1 à 12.

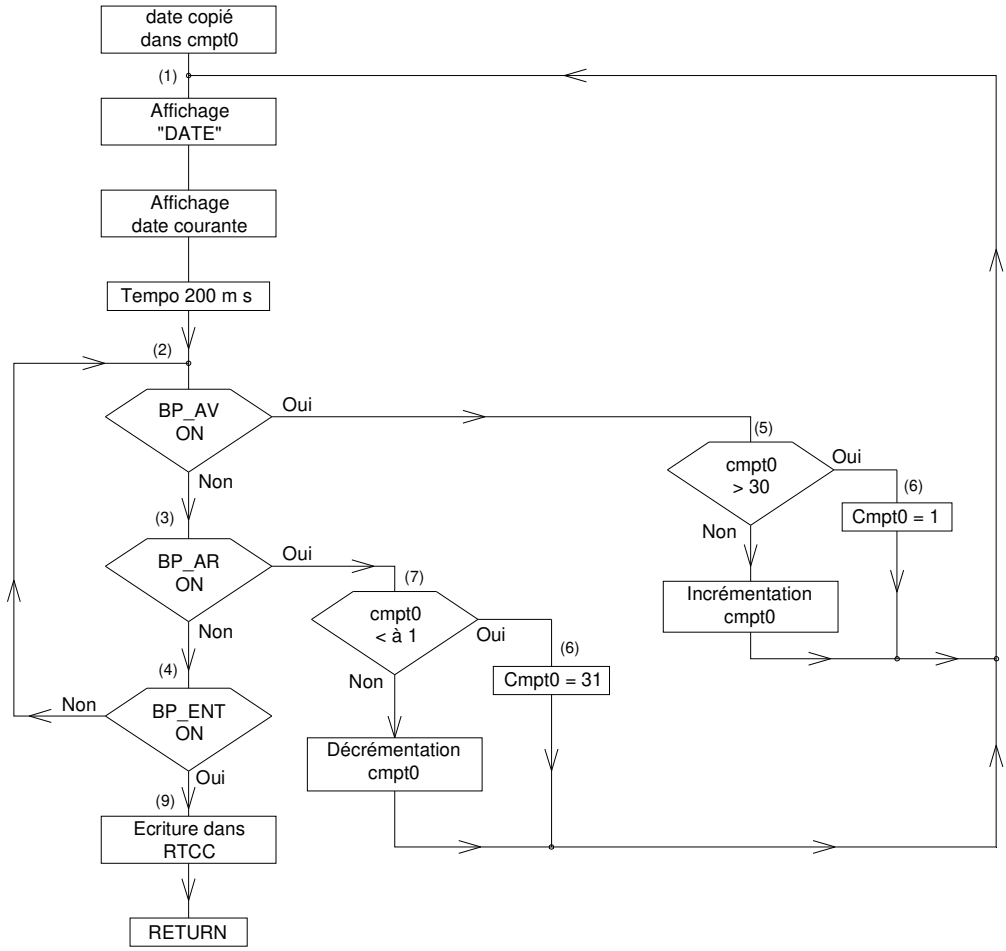



```
REGL_MOI:                               ; Réglage du MOIS

            movf    mois,0
            movwf   cmpt0

      Moi1
            movwf   x+0
            call    LCD_EFFACE
            movlw   d'0'
            call    LCD_LOCATE
            movlw   "M"                         ; Affichage: "MOIS:" sur la 1ère ligne
            call    LCD_DONNEE
            movlw   "O"
            call    LCD_DONNEE
            movlw   "I"
            call    LCD_DONNEE
            movlw   "S"
            call    LCD_DONNEE
            movlw   d'7'
            CALLX   AFFICHE_24BS            ; Affichage du mois courant
            call    Tempo_200ms

      Moi2
            btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
            goto    Moi3
            goto    Moi5

      Moi3
            btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
            goto    Moi4
            goto    Moi7
```

```
Moi4
        btfsc   BP_VAL                  ; Validation de la valeur pour écriture dans le RTCC
        goto    Moi2
        goto    Moi9

Moi5
        movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
        sublw   d'11'                   ; Pour limitation à 12 (mois de 1 à 12)
        btfss   CARRY
        goto    Moi6
        incf    cmpt0
        movf    cmpt0,0
        goto    Moi1

Moi6
        movlw   d'1'
        movwf   cmpt0
        goto    Moi1

Moi7
        movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
        sublw   d'1'                    ; Pour limitation à 1 (mois de 1 à 12)
        btfsc   CARRY
        goto    Moi8
        decf    cmpt0
        movf    cmpt0,0
        goto    Moi1

Moi8
        movlw   d'12'
        movwf   cmpt0
        goto    Moi1

Moi9
        call    I2C_START_              ; BP_VAL actionné (écriture RTCC)
        movlw   ADRS_RTCC+0
        call    I2C_SEND_BYTE
        movlw   0x05
        call    I2C_SEND_BYTE
        movf    cmpt0,0
        CALLX   conv_DH                 ; Conversion binaire/Hexa en BCD
        movwf   cmpt0
        call    I2C_SEND_BYTE
        call    I2C_STOP_
        call    Tempo_500ms
        return
```

55 - REGL_DAT:

Exécute le paramétrage de la date au niveau du jour dans le mois.
Les valeurs possibles vont de 1 à 31.

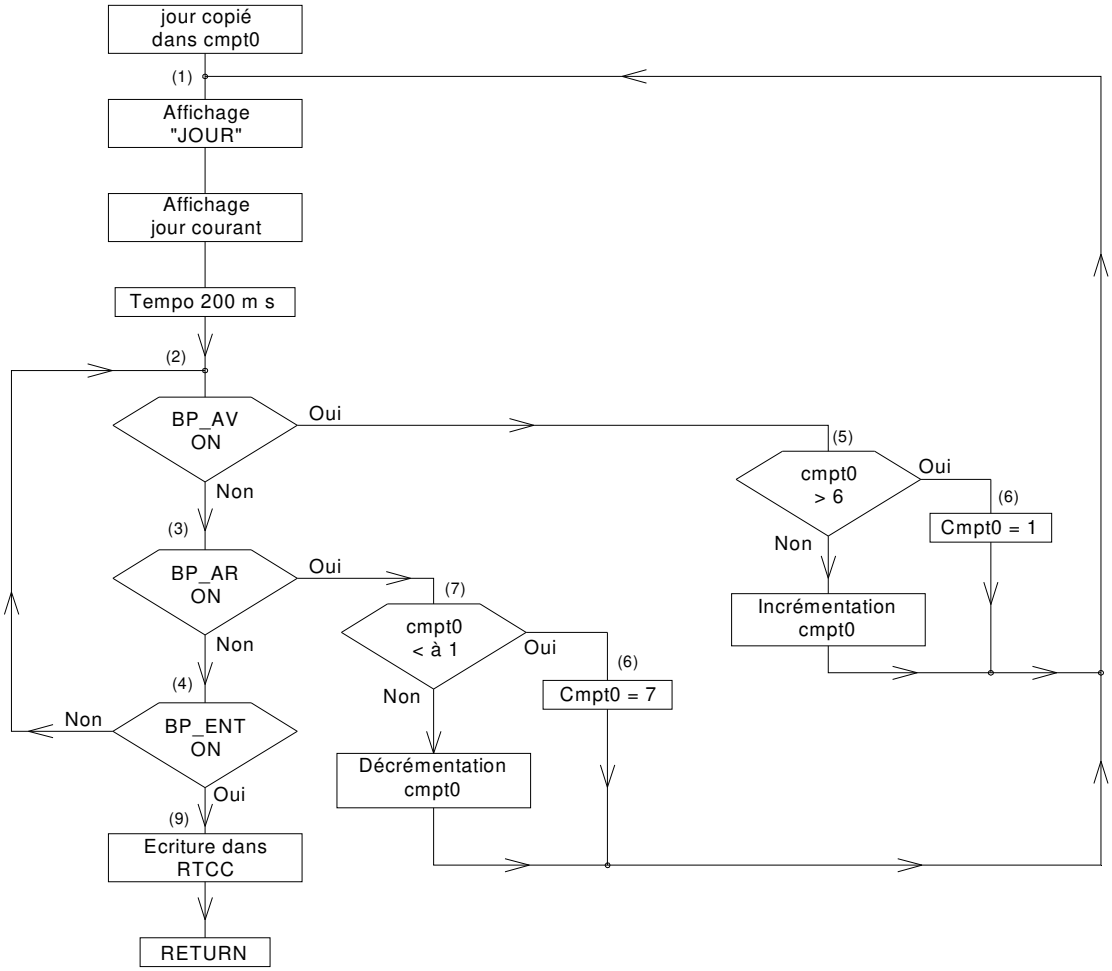



```
REGLAGE_DAT:                        ; Réglage de la DATE

            movf    date,0
            movwf   cmpt0

       Dat1
            movwf   x+0                     ; Affichage: "DATE:" sur la 1ère ligne
            call    LCD_EFFACE
            movlw   d'0'
            call    LCD_LOCATE
            movlw   "D"
            call    LCD_DONNEE
            movlw   "A"
            call    LCD_DONNEE
            movlw   "T"
            call    LCD_DONNEE
            movlw   "E"
            call    LCD_DONNEE
            movlw   ":"
            call    LCD_DONNEE
            movlw   d'7'
            CALLX   AFFICHE_24BS            ; Affichage de la date courante
            call    Tempo_200ms

       Dat2
            btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
            goto    Dat3
            goto    Dat5

       Dat3
            btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
            goto    Dat4
            goto    Dat7
```

```
Dat4
        btfsc   BP_VAL                  ; Validation de la valeur pour écriture dans le RTCC
        goto    Dat2
        goto    Dat9

Dat5
        movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
        sublw   d'30'                   ; Pour limitation à 31 (date de 1 à 31)
        btfss   CARRY
        goto    Dat6
        incf    cmpt0
        movf    cmpt0,0
        goto    Dat1

Dat6
        movlw   d'1'
        movwf   cmpt0
        goto    Dat1
Dat7
        movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
        sublw   d'1'                    ; Pour limitation à 1 (date de 1 à 31)
        btfsc   CARRY
        goto    Dat8
        decf    cmpt0
        movf    cmpt0,0
        goto    Dat1

Dat8
        movlw   d'31'
        movwf   cmpt0
        goto    Dat1

Dat9
        call    I2C_START_              ; BP_VAL actionné (écriture RTCC)
        movlw   ADRS_RTCC+0
        call    I2C_SEND_BYTE
        movlw   0x04
        call    I2C_SEND_BYTE
        movf    cmpt0,0
        call    conv_DH                 ; Conversion binaire/Hexa en BCD
        movwf   cmpt0
        call    I2C_SEND_BYTE
        call    I2C_STOP_
        call    Tempo_500ms
        return
```

56 - REGLAGE_JOU:

Exécute le paramétrage du jour de la semaine.
Les valeurs possibles vont de 1 à 7
Lundi est le premier jour de la semaine : valeur 1 pour lundi

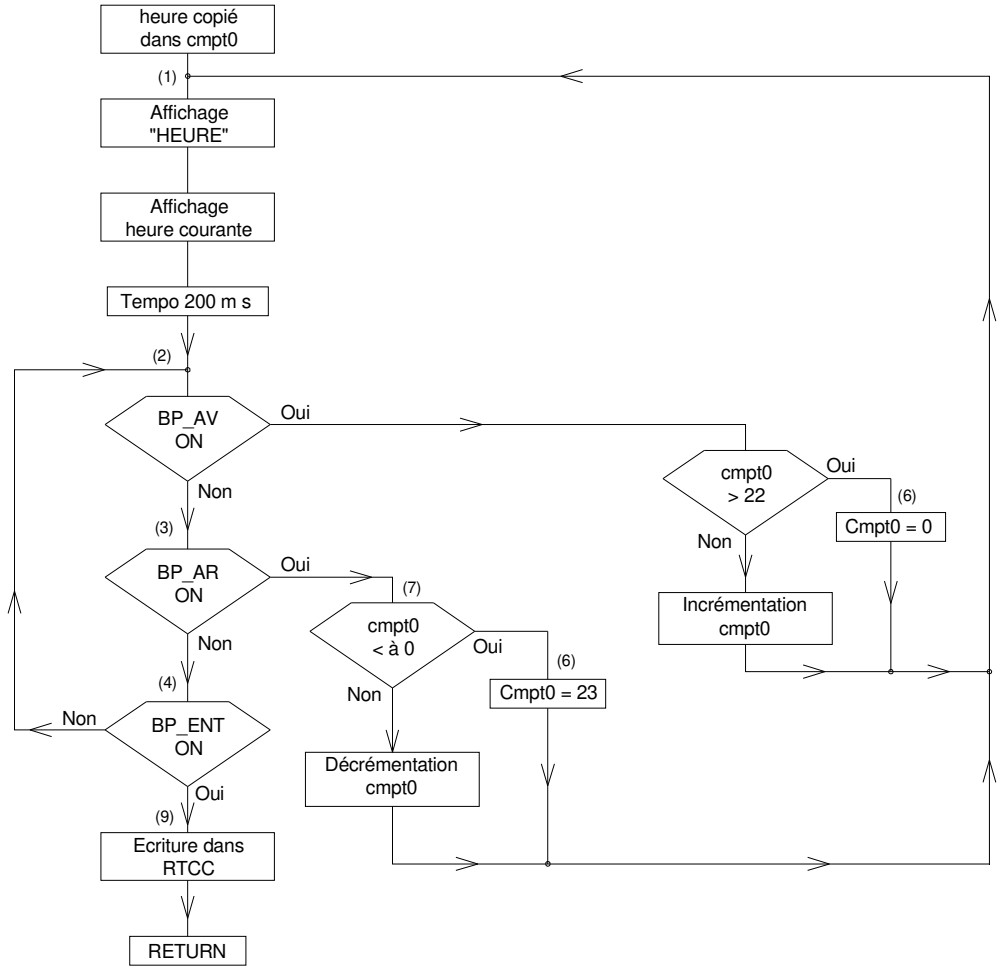



```
REGLAGE_JOU:                        ; Réglage du jour de la semaine

            movf    jour,0
            movwf   cmpt0

      Jou1
            movwf   x+0                     ; Affichage: "JOUR:" sur la 1ère ligne
            call    LCD_EFFACE
            movlw   d'0'
            call    LCD_LOCATE
            movlw   "J"
            call    LCD_DONNEE
            movlw   "O"
            call    LCD_DONNEE
            movlw   "U"
            call    LCD_DONNEE
            movlw   "R"
            call    LCD_DONNEE
            movlw   ":"
            call    LCD_DONNEE
            movlw   d'7'
            CALLX   AFFICHE_24BS            ; Affichage du jour courant de la semaine
            call    Tempo_200ms

      Jou2
            btfsc   BP_AV                   ; Défilement de la valeur en incrémentation
            goto    Jou3
            goto    Jou5

      Jou3
```

```
        btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
        goto    Jou4
        goto    Jou7

Jou4
        btfsc   BP_VAL                  ; Validation de la valeur pour écriture dans le RTCC
        goto    Jou2
        goto    Jou9

Jou5
        movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
        sublw   d'6'                    ; Pour limitation à 7 (jour de 1 à 7)
        btfss   CARRY
        goto    Jou6
        incf    cmpt0
        movf    cmpt0,0
        goto    Jou1

Jou6
        movlw   d'1'
        movwf   cmpt0
        goto    Jou1

Jou7
        movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
        sublw   d'1'                    ; Pour limitation à 1 (jour de 1 à 7)
        btfsc   CARRY
        goto    Jou8
        decf    cmpt0
        movf    cmpt0,0
        goto    Jou1

Jou8
        movlw   d'7'
        movwf   cmpt0
        goto    Jou1

Jou9
        call    I2C_START_              ; BP_VAL actionné (écriture RTCC)
        movlw   ADRS_RTCC+0
        call    I2C_SEND_BYTE
        movlw   0x03
        call    I2C_SEND_BYTE
        bsf             cmpt0,3         ; Broche VBAT du circuit RTCC connectée
        movf    cmpt0,0                 ; (Pile CR2032 opérationnelle si coupure de courant)
        call    I2C_SEND_BYTE
        call    I2C_STOP_
        call    Tempo_500ms
        return
```

57 - REGL_HEU:

Exécute le paramétrage de l'heure.
Les valeurs possibles vont de 0 à 23.

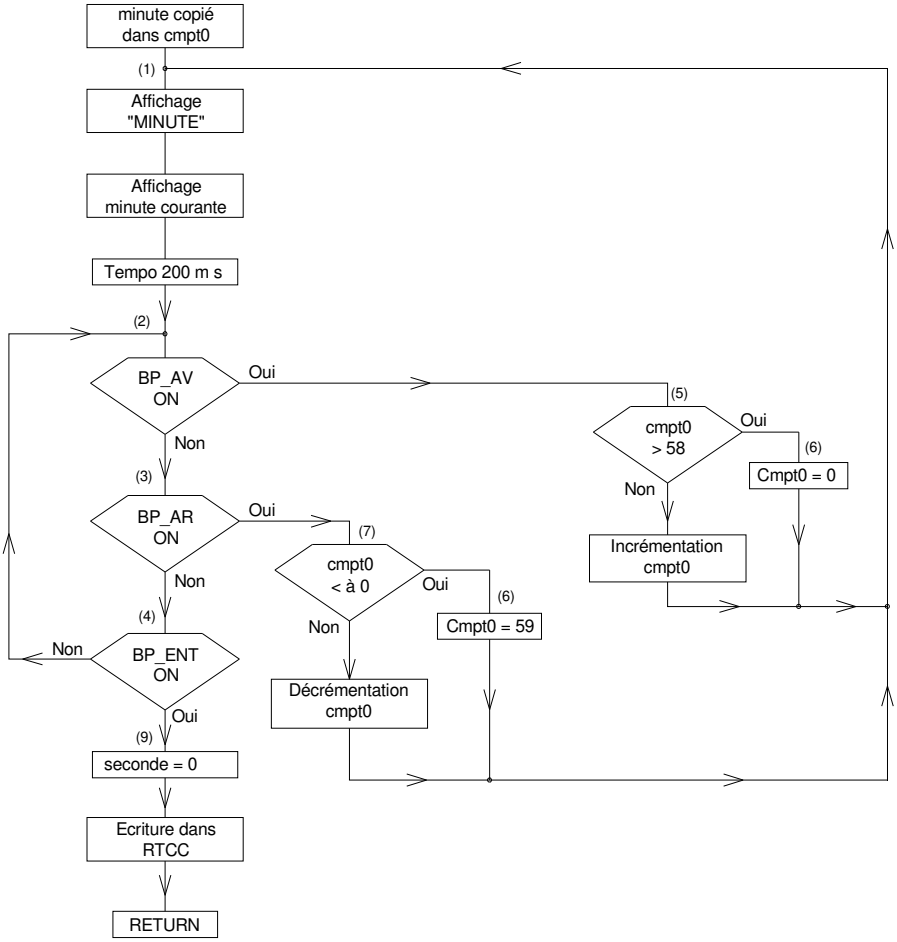



```
REGLAGE_HEU:                        ; Réglage de l'HEURE

            movf    heure,0             ; Heure du soleil (UTC)
            movwf   cmpt0

      Heu1
            movwf   x+0                 ; Affichage: "HEURES:" sur la 1ère ligne
            call    LCD_EFFACE
            movlw   d'0'
            call    LCD_LOCATE
            movlw   "H"
            call    LCD_DONNEE
            movlw   "E"
            call    LCD_DONNEE
            movlw   "U"
            call    LCD_DONNEE
            movlw   "R"
            call    LCD_DONNEE
            movlw   "E"
            call    LCD_DONNEE
            movlw   "S"
            call    LCD_DONNEE
            movlw   ":"
            call    LCD_DONNEE
            movlw   d'9'
            CALLX   AFFICHE_24BS        ; Affichage de l'heure courante
            call    Tempo_200ms

      Heu2
            btfsc   BP_AV               ; Défilement de la valeur en incrémentation
            goto    Heu3
```

```
        goto    Heu5

Heu3
        btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
        goto    Heu4
        goto    Heu7

Heu4
        btfsc   BP_VAL                  ; Validation de la valeur pour écriture dans le RTCC
        goto    Heu2
        goto    Heu9

Heu5
        movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
        sublw   d'22'                   ; Pour limitation à 23 (heure de 0 à 23)
        btfss   CARRY
        goto    Heu6
        incf    cmpt0
        movf    cmpt0,0
        goto    Heu1

Heu6
        movlw   d'0'
        movwf   cmpt0
        goto    Heu1

Heu7
        movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
        sublw   d'0'                    ; Pour limitation à 0 (heure de 0 à 23)
        btfsc   CARRY
        goto    Heu8
        decf    cmpt0
        movf    cmpt0,0
        goto    Heu1

Heu8
        movlw   d'23'
        movwf   cmpt0
        goto    Heu1

Heu9
        call    I2C_START_              ; BP_VAL actionné (écriture RTCC)
        movlw   ADRS_RTCC+0
        call    I2C_SEND_BYTE
        movlw   0x02
        call    I2C_SEND_BYTE
        movf    cmpt0,0
        call    conv_DH                 ; Conversion binaire/Hexa en BCD
        movwf   cmpt0
        call    I2C_SEND_BYTE
        call    I2C_STOP_
        call    Tempo_500ms
        return
```

58 - REGL_MIN:

Exécute le paramétrage des minutes.
Les valeurs possibles vont de 0 à 59.
Le programme règle en même temps la valeur 0 pour les secondes
Ceci sous-entend que la mise à l'heure exacte (à la seconde près) peut être exécuter en pressant la touche « V » (validation) lorsque l'horloge de référence indique précisément : 0 seconde lors du réglage des minutes et secondes.
Si le circuit RTCC est corectement qualibré, la mise à l'heure est exceptionnelle si la pile CR2032 garde sa tension de 3V durant plusieurs années (consommation très faible du circuit MCP79410).

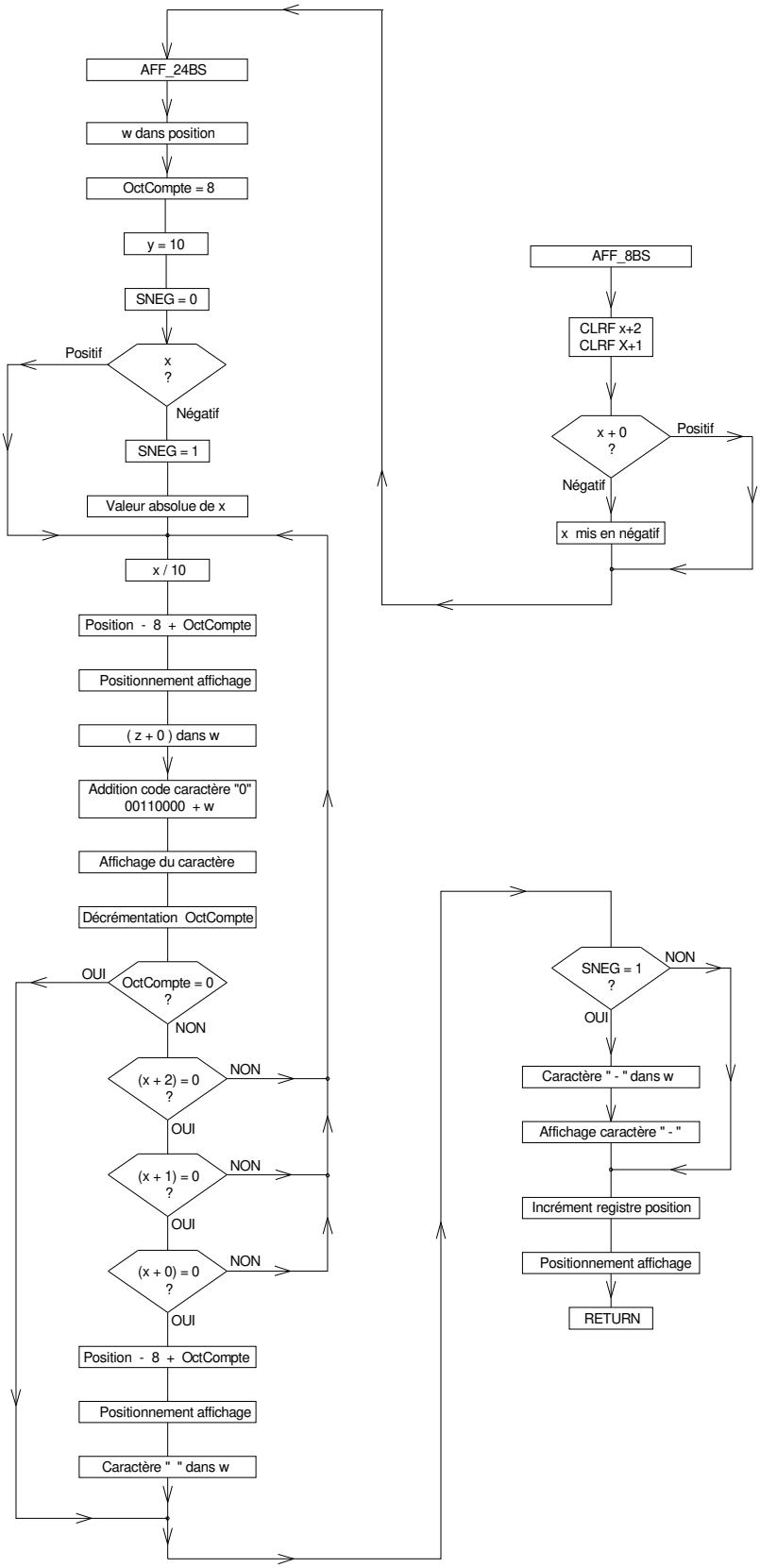



```
REGL_MIN:                          ; Réglage MINUTES ET SECONDES (Secondes=00)

              movf    minute,0
              movwf   cmpt0

       Min1
              movwf   x+0                    ; Affichage: "MINUTES:" sur la 1ère ligne
              call    LCD_EFFACE
              movlw   d'0'
              call    LCD_LOCATE
              movlw   "M"
              call    LCD_DONNEE
              movlw   "I"
              call    LCD_DONNEE
              movlw   "N"
              call    LCD_DONNEE
              movlw   "U"
              call    LCD_DONNEE
              movlw   "T"
              call    LCD_DONNEE
              movlw   "E"
              call    LCD_DONNEE
              movlw   "S"
              call    LCD_DONNEE
              movlw   ":"
              call    LCD_DONNEE
              movlw   d'10'
```

```
        CALLX   AFFICHE_24BS            ; Affichage de la minute courante
        call    Tempo_200ms

Min2
        btfsc   BP_AV                   ; Défilement de la valeur en incrémentation

        goto    Min3
        goto    Min5

Min3
        btfsc   BP_AR                   ; Défilement de la valeur en décrémentation
        goto    Min4
        goto    Min7

Min4
        btfsc   BP_VAL                  ; Validation de la valeur pour écriture dans le RTCC
        goto    Min2
        goto    Min9

Min5
        movf    cmpt0,0                 ; BP_AV actionné (incrémentation)
        sublw   d'58'                   ; Pour limitation à 59 (minutes de 0 à 59)
        btfss   CARRY
        goto    Min6
        incf    cmpt0
        movf    cmpt0,0
        goto    Min1

Min6
        movlw   d'0'
        movwf   cmpt0
        goto    Min1

Min7
        movf    cmpt0,0                 ; BP_AR actionné (décrémentation)
        sublw   d'0'                    ; Pour limitation à 0 (minutes de 0 à 59)
        btfsc   CARRY
        goto    Min8
        decf    cmpt0
        movf    cmpt0,0
        goto    Min1

Min8
        movlw   d'59'
        movwf   cmpt0
        goto    Min1

Min9
        call    I2C_START_              ; BP_VAL actionné (écriture RTCC)
        movlw   ADRS_RTCC+0
        call    I2C_SEND_BYTE
        movlw   0x00
        call    I2C_SEND_BYTE
        movlw   b'10000000'             ; 0 seconde  et bit ST positionné à 1
        call    I2C_SEND_BYTE           ; Envoyé à l'adresse 0x00
        movf    cmpt0,0                 ;
        call    conv_DH                 ; Conversion binaire/Hexa en BCD
        movwf   cmpt0                   ; Minutes
        call    I2C_SEND_BYTE           ; Envoyé à l'adresse suivante (0x01)
        call    I2C_STOP_
        call    Tempo_500ms
        return
```

61 - AFFICHE_8BS :
62 - AFFICHE_24BS:

AFFICHE_8BS réalise l'affichage.d'une valeur codée sur 1 octet sur l'afficheur LCD
AFFICHE_24BS réalise l'affichage.d'une valeur codée sur 3 octets sur l'afficheur LCD



```
AFFICHE_8BS:                ; Affichage d'une valeur sur 1 octet avec signe

            clrf    x+2                 ; Effacement de "x+2"
            clrf    x+1                 ; Effacement de "x+1"
            btfss   x+0,7               ; Test si "x+0" est négatif
            goto    AFFICHE_24BS        ; Si "x+0" est positif, saut vers "AFFICHE_24BS"
            decf    x+2                 ; Si "x+0" est négatif, "x+2" = 0xFF
            decf    x+1                 ; "x+0" négatif, "x+1" = 0xFF

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

AFFICHE_24BS:                       ; Affichage d'une valeur sur 3 octets avec signe

            movwf   Position            ; Transfert de w dans le registre "Position"
            movlw   0x08
            movwf   OctCompte           ; Valeur "8" dans le registre "OctCompte"
            movlw   0x00
            movwf   y+2                 ; Valeur "0" dans le registre "y+2"
            movlw   0x00
            movwf   y+1                 ; Valeur "0" dans le registre "y+1"
            movlw   0x0a
            movwf   y+0                 ; Valeur 10 (0x0A) dans "y+0"
            bcf     SNEG                ; Mise à 0 du bit "SNEG"
            btfss   x+2,7               ; Test si "x" est négatif
            goto    Affiche             ; Si "x" est négatif, saut vers "boucle"
            bsf     SNEG                ; Mise à 1 de "SNEG"
            comf    x+2                 ; Complément à 2 de "x" (x en valeur positive)
            comf    x+1                 ;
            comf    x+0                 ;
            incf    x+0                 ;
            skpnz                       ;
            incf    x+1                 ;
            skpnz                       ;
            incf    x+2                 ;

      Affiche
            CALLX   FXD_2424U           ; Division de x par y (x/y)
            movlw   d'8'
            subwf   Position,w          ; w est soustrait de "position" (Position - w)
            addwf   OctCompte,w         ; "OctCompte" + w ((OctCompte + Position - 8)
            call    LCD_LOCATE          ; Positionnement affichage à (OctCompte + Position - 8)
            movfw   z+0                 ; "z+0" transféré dans w
            addlw   0x30                ; w + d'48'
            call    LCD_DONNEE          ; Affichage du caractère code ASCII de (w + 0x30)
            decf    OctCompte           ; Décrémentation de "OctCompte"
            btfsc   ZERO                ; Test si "OctCompte" est égal à 0
            goto    Signe               ; Si "OctCompe" est égal à à, saut vers Etiq "Signe"
            movfw   x+2                 ; Si "OctCompte" n'est pas égal à 0,....
            btfss   ZERO                ; Test si (x+2) = 0
            goto    Affiche             ; Si (x=2) n'est pas égal à 0, saut vers Etiq "Boucle"
            movf    x+1,0
            btfss   ZERO                ; Test si (x+1) = 0
            goto    Affiche             ; Si (x+1) n'est pas égal à 0, saut vers Etiq "Boucle"
            movf    x+0,0
            btfss   ZERO                ; Test si (x+0) = 0
            goto    Affiche             ; Si (x+0) n'est pas égal à 0, saut vers Etiq "Boucle"
            movlw   d'8'                ; Si x = 0, 8 dans w
            subwf   Position,w          ; w est soustrait de "position" (Position - w)
            addwf   OctCompte,w         ; "OctCompte" + w ((OctCompte + Position - 8)
            call    LCD_LOCATE          ; Positionnement affichage à (OctCompte + Position - 8)

      Signe
            btfss   SNEG                ; Test si bit "SNEG"
            goto    $+3                 ; Si x est positif, saut 3 lignes plus loin
            movlw   "-"                 ; x négatif, caractère "-" dans w
            call    LCD_DONNEE          ; Affichage du caractère "-"
            incf    Position,w          ; Incrémentation du registre "Position" (résultat dans w)
            call    LCD_LOCATE          ; Positionnement de l'affichage à la valeur "Position"
            return
```

63 - AFF_DATEHEURE:

Exécute l'affichage de la date et de l'heure sur la 1ère ligne de l'afficheur LCD.

```
AFF_DATEHEURE                ; Affichage de la DATE et de l'HEURE
                                        ; sur la 1ère ligne de l'afficheur.
                movlw   d'0'
                call    LCD_LOCATE
                movlw   "0"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movfw   date
                movwf   x+0
                movlw   d'1'
                call    AFFICHE_8BS             ; Affichage de la "Date" à la position 1

                movlw   "/"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movfw   mois
                movwf   x+0
                movlw   d'4'
                call    AFFICHE_8BS             ; Affichage du "Mois" à la position 2

                movlw   "/"
                call    LCD_DONNEE
                movlw   "2"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movfw   annee
                movwf   x+0
                movlw   d'9'
                call    AFFICHE_8BS             ; Affichage de l'Année à la position 9

                movlw   " "
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movfw   heure
                movwf   x+0
                incf    x+0                     ; Fuseau horaire de la FRANCE GMT+1
                movlw   d'24'                   ; Affichage de 00H00 au lieu de 24H00
                subwf   x+0,0                   ; heure - w
                btfss   ZERO
                 goto   $+2
                clrf    x+0                     ; Si (heure - 24) = 0 alors valeur "0" dans "x+0"
                btfsc   REG_TRACK,1             ; Lecture de la valeur de REG_TRACK,1
                incf    x+0                     ; Heure d'été (+1 heure) si REG_TRACK,1 = 1
                movlw   d'24'                   ; Affichage de 00H00 au lieu de 24H00
                subwf   x+0,0                   ; heure - w
                btfss   ZERO
                goto    $+2
                clrf    x+0                     ; si (heure - 24) = 0 alors valeur "0" dans "x+0"
                movlw   d'12'                   ; Positionnement de l'affichage
                call    AFFICHE_8BS             ; Affichage des "Heures" à la position 12

                movlw   "H"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movlw   "0"
                call    LCD_DONNEE
                movfw   minute
                movwf   x+0
                movlw   d'15'
                call    AFFICHE_8BS             ; Affichage des "Minutes" à la position 15
                return
```

64 - AFF_HAUTEUR:

Exécute l'affichage de la hauteur du soleil sur la 2ème ligne de l'afficheur LCD.

```
AFF_HAUTEUR:                        ; Affichage de la HAUTEUR
                                    ; sur la 2ème ligne de l'afficheur
               movlw   d'64'
               call    LCD_LOCATE
               movlw   "H"
               call    LCD_DONNEE
               movlw   ":"
               call    LCD_DONNEE
               movfx   RegHaut              ; Angle "Hauteur" multiplié par 10
               movly   d'0',d'0',d'10'
               CALLX   FXD_2424S            ; Valeur entière de l'angle "Hauteur"
               movfw   z+0
               movwf   cmpt1                ; Valeur décimale dans cmpt1
               btfss   SNEGx                ; test si angle négatif
               goto    $+5
               movlw   d'67'                ; Si angle négatif, positionnement à 67
               call    LCD_LOCATE           ;
               movlw   "-"                  ; Affichage du signe"-"
               call    LCD_DONNEE           ;
               movlw   d'68'                ; Positionnement à 68
               call    AFFICHE_24BS         ; Affichage de la partie entière (avec signe)
               movlw   d'69'                ; Positionnement à 69
               call    LCD_LOCATE           ;
               movlw   ","                  ; Affichage du signe ","
               call    LCD_DONNEE           ;
               movfw   cmpt1                ; Valeur décimale de l'angle dans w
               movwf   x+0                  ; Transfert dans "x+0"
               movlw   d'70'                ; Valeur décimale dans cmpt1
               call    AFFICHE_24BS         ; Affichage de la valeur décimale de l'angle
               return
```

65 - AFF_AZIMUT:

Exécute l'affichage de l'azimut du soleil sur la 2ème ligne de l'afficheur LCD.

```
AFF_AZIMUT:                         ; Affichage de l'AZIMUT
                                    ; sur la 2ème ligne de l'afficheur
               movlw   d'72'
               call    LCD_LOCATE
               movlw   "A"
               call    LCD_DONNEE
               movlw   ":"
               call    LCD_DONNEE
               movfx   RegAzim
               movly   h'00',h'07',h'08'    ; Valeur 1800 (180°) ajoutée à RegHaut
               CALLX   FXA_2424S            ; pour affichage azimut de 0° à 360°
                                            ; au lieu de angle négatif à l'EST
                                            ; et angle positif à l'ouest à partir du midi

               movly   d'0',d'0',d'10'      ; Valeur "AZIMUT" multipliée par 10
               CALLX   FXD_2424S            ; Valeur entière de l'angle "AZIMUT"
               movfw   z+0
               movwf   Calcul2+0            ; Valeur décimale de l'angle dans Calcul2+0
               btfss   SNEGx                ; test si angle négatif
               goto    $+5
               movlw   d'76'                ; Si angle négatif, positionnement à 76
               call    LCD_LOCATE           ;
               movlw   "-"                  ; Affichage du signe"-"
               call    LCD_DONNEE           ;
               movlw   d'77'                ; Positionnement à 77
               call    AFFICHE_24BS         ; Affichage de la partie entière (avec signe)
               movlw   d'78'                ; Positionnement à 78
               call    LCD_LOCATE
               movlw   ","                  ; Affichage du signe ","
               call    LCD_DONNEE           ;
               movfw   Calcul2+0
               movwf   x+0                  ; Valeur de Calcul2+0 dans "x+0"
               movlw   d'79'                ; Positionnement à 79
               call    AFFICHE_8BS          ; Affichage de la valeur décimale de l'angle
               return                       ; AZIMUT
```

66 - ARRONDI:

```
ARRONDI:              ; Arrondi à la dizaine pour la valeur contenu dans « x »

               movly  d'0',d'0',d'10'
               call   FXD_2424S
               call   FXM_2424S
               movfw  z+0
               addlw  d'250'
               btfss  CARRY
               return                       ; Pas de débordement (arrondi en dessous)
               movly  d'0',d'0',d'10'       ; Débordement (arrondi au dessus)
               btfss  x+2 , 7
               goto   arron1
               comf   y+2
               comf   y+1
               comf   y+0
               incf   y+0
               skpnz
               incf   y+1
               skpnz
               incf   y+2

       arron1
               call   FXA_2424S
               return
```

67 - LATITUDE:

Ce programme exprime la latitude terrestre sur 4 chiffres (c'est-à-dire en centièmes) à partir des 2 valeurs distinctes stockées dans les registres : « latitude+0 » pour le nombre de degrés et « latitude+1 » pour le nombres de centièmes.

```
LATITUDE:             ; Latitude exprimée en centièmes de degrés

               clrf   x+2
               clrf   x+1
               movf   latitude+0,0
               movwf  x+0
               movly  d'0',d'0',d'100'
               call   FXM_2424S
               clrf   y+2
               clrf   y+1
               movf   latitude+1,0
               movwf  y+0
               call   FXA_2424S
               return
```

68 - LONGITUDE:

Ce programme exprime la longitude terrestre sur 4 chiffres (c'est-à-dire en centièmes) à partir des 2 valeurs distinctes stockées dans les registres : « longitude+0 » pour le nombre de degrés et « longitude+1 » pour le nombre de centièmes.

```
LONGITUDE:            ; Longitude exprimée en centièmes de degrés

               clrf   x+2
               clrf   x+1
               movf   longitude+0,0
               movwf  x+0
               movly  d'0',d'0',d'100'
               call   FXM_2424S
               clrf   y+2
               clrf   y+1
               movf   longitude+1,0
               movwf  y+0
               call   FXA_2424S
               return
```

69 - CALCULS:

CALCUL :

Ce sous-programme est le programme principal de calcul des 2 valeurs principales à déterminer ; en l'occurrence : La Hauteur et l'Azimut du soleil.
Il exécute séquentiellement les sous-programmes suivants.

Jourannee :	Calcul du quantième du jour de l'année
Anomoy :	Calcul de l'anomalie moyenne
EquCen :	Calcul de l'équation du centre
LonEcl :	Calcul de la longitude écliptique
RedEqu :	Calcul de la réduction à l'équateur
Declin :	Calcul de la déclinaison du soleil
Heurjour :	Conversion de l'heure en décimal
AngHor :	Calcul de l'angle horaire
Hauteur :	Calcul de la hauteur
Azimut	Calcul de l'azimut

Nota : toutes ces séquences de calcul font également appel à d'autres routines de sous-programmes : Sinus, Cosinus, ArcSinus, addition, multiplication, divisions, etc,.dont la plupart ont été décrite ci-dessus.

```
Jourannee:                  ; Quantième du jour de l'Année

            movfw   mois
            movwf   MoisAnnee
            clrf    x+2
            clrf    x+1
            movfw   date
            movwf   x+0
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
            movly   d'0',d'0',d'31'             ; + 31 jours (mois de janvier)
            call    FXA_2424S
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
            movly   d'0',d'0',d'28'             ; + 28 jours (mois de février)
            btfsc   BISSEXTILE
            incf    y+0                         ; ou + 29 jours si année bissextile
            call    FXA_2424S
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
            movly   d'0',d'0',d'31'             ; + 31 jours (mois de mars)
            call    FXA_2424S
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
            movly   d'0',d'0',d'30'             ; + 30 jours (mois d'avril)
            call    FXA_2424S
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
            movly   d'0',d'0',d'31'             ; + 31 jours (mois de mai)
            call    FXA_2424S
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
            movly   d'0',d'0',d'30'             ; + 30 jours (mois de juin)
            call    FXA_2424S
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
            movly   d'0',d'0',d'31'             ; + 31 jours (mois de juillet)
            call    FXA_2424S
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
            movly   d'0',d'0',d'31'             ; + 31 jours (mois d'août)
            call    FXA_2424S
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
            movly   d'0',d'0',d'30'             ; + 30 jours (mois de septembre)
            call    FXA_2424S
            decf    MoisAnnee
            btfsc   ZERO
            goto    fin_jourannee
```

```
                movly   d'0',d'0',d'31'              ; + 31 jours (mois d'octobre)
                call    FXA_2424S
                decf    MoisAnnee
                btfsc   ZERO
                goto    fin_jourannee
                movly   d'0',d'0',d'30'              ; + 30 jours (mois de novembre)
                call    FXA_2424S
        fin_jourannee
;               return

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

Anomoy:                 ; Anomalie moyenne (M) = 357,53 + (0,9856 J)

                movly   h'00',h'26',h'80'     ; VALEUR "9856" dans y
                call    FXM_2424S             ; JourAnnee multiplié par 9856
                movxf   Calcul1               ; Sauvegarde dans "Calcul1"
                movly   h'36',h'8e',h'04'     ; 3575300 dans y
                call    FXA_2424S             ; Calcul précédent additionné à y
                movly   h'00',h'00',h'64'     ; Valeur d'100' dans y
                call    FXD_2424SA            ; Division par 100 avec arrondi
;               return

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

Equcen:                 ; Equation du centre (C) = 1,914 sin(M)

                call    Sinus
                movly   0x00,0x03,0xE8        ; d'1000'
                call    FXD_2424SA
                movly   h'00',h'07',h'7a'     ; d'1914'
                call    FXM_2424S
                movxf   RegHaut
;               return

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

Lonecl:                 ; Longitude écliptique (Lon) = 280,5 + (0.9856 J) +C

                movfx   Calcul1               ; (0,9856 J) multiplié par 10000
                movly   h'2a',h'cd',h'08'     ; d'2805000'
                call    FXA_2424S
                movxf   Calcul1
                movfx   RegHaut               ; Equation du Centre
                movly   d'0',d'0',d'100'      ; Equation du Centre multiplié par 10000
                call    FXD_2424SA
                movfy   Calcul1
                call    FXA_2424S
                movxf   Calcul1               ; Sauvegarde dans Calcul1
;               return

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

Redequ:                 ; Réduction à l'Equateur (R) = -2,468 sin(2Lon) + 0,053 sin(4Lon)

                movfx   Calcul1
                movly   h'00',h'00',h'0a'     ; d'10'
                call    FXD_2424SA            ; Longitude multiplié par d'1000'
                movly   h'00',h'00',h'02'     ; d'2'
                call    FXM_2424S             ; Longitude multiplié par d'2000'
                movly   h'00',h'00',h'0a'     ; d'10'
                call    FXD_2424SA            ; Longitude multiplié par d'200'
                call    Sinus
                movly   0x00,0x03,0xE8        ; d'1000'
                call    FXD_2424SA
                movly   h'ff',h'ff',h'09'     ; - 247
                call    FXM_2424S
                movxf   Calcul2               ; Sauvegarde dans Calcul2
                movfx   Calcul1
                movly   d'0',d'0',d'10'       ; d'10'
                call    FXD_2424SA            ; Longitude multiplié par d'1000'
                movly   h'00',h'00',h'04'     ; d'4'
                call    FXM_2424S             ; Longitude multiplié par d'4000'
                movly   d'0',d'0',d'10'       ; d'10'
                call    FXD_2424SA            ; Longitude multiplié par d'400'
                call    Sinus
                movly   0x00,0x27,0x10        ; d'10000'
                call    FXD_2424SA
                movly   h'00',h'00',h'35'     ; d'53'
                movfy   Calcul2
                call    FXA_2424S
```

```
                movxf   Calcul2
;               return

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

Declin:                 ; Déclinaison (Dec) = 22,8 sin(Lon) + (0,6 sin(Lon) au cube)

                movfx   Calcul1
                movly   d'0',d'0',d'100'
                call    FXD_2424SA
                call    Sinus
                movly   0x00,0x03,0xE8          ; d'1000'
                call    FXD_2424SA
                movxf   Calcul1                 ; DataTemp  sin(Lon)
                movly   d'0',d'0',d'228'
                call    FXM_2424S
                movly   0x00,0x00,0x64          ; d'100'
                call    FXD_2424SA
                movxf   Calcul3
                movfx   Calcul1                 ; DataTemp
                movfy   Calcul1                 ; DataTemp
                call    FXM_2424S               ; sin(Lon) au carré
                movly   h'0',h'03',h'E8'        ; / par 1000
                call    FXD_2424SA              ;
                movfy   Calcul1                 ; DataTemp
                call    FXM_2424S
                movly   0x00,0x00,0x0A          ; d'10'
                call    FXD_2424SA
                movly   0x00,0x00,0x06          ; d'6'
                call    FXM_2424S
                movly   h'0',h'27',h'10'        ; / par 10000
                call    FXD_2424SA              ;
                movfy   Calcul3
                call    FXA_2424S
                movxf   Calcul3                 ; Sauvegarde déclinaison
;               return


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

Heurejour:                                      ; Conversion de l'heure en décimales

                clrf    x+2
                clrf    x+1
                movfw   heure
                movwf   x+0
                movly   d'0',d'0',d'100'
                call    FXM_2424S
                movxf   Calcul1
                clrf    x+2
                clrf    x+1
                movfw   minute
                movwf   x+0
                movly   d'0',d'0',d'100'
                call    FXM_2424S
                movly   d'0',d'0',d'60'
                call    FXD_2424SA
                movfy   Calcul1
                call    FXA_2424S
;               return

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

Anghor:                 ; Angle horaire (Ah) = ((Hutc - 12) x 15) - (C+R) + Longitude terrestre
                        ; Exprimé en centièmes de degrés

                movly   h'ff',h'fb',h'50'               ; -1200
                call    FXA_2424S                               ; (Hutc -12) sur 4 chiffres (x par 100)
                movly   d'0',d'0',d'15'
                call    FXM_2424S                               ; ((Hutc -12) sur 4 chiffres) x par 15
                movxf   Calcul1                                 ; Sauvegarde dans Calcul1
                movfx   ArcA                                    ; Equation du centre (C)
                movfy   Calcul2                                 ; Réduction à l'équateur (R)
                call    FXA_2424S                               ; Equation du temps (C+R)
                movly   h'0',h'03',h'e8'                        ; / par 1000
                call    FXD_2424SA                              ; Equation du temps (C+R) (x par 100)
                comf    x+2                                             ; Inversion du signe de (C+R)
                comf    x+1
                comf    x+0
                incf    x+0
                skpnz
```

```
                incf    x+1
                skpnz
                incf    x+2
                movfy   Calcul1                         ; Transferts de (Hutc_12)x15 dans (y)
                call    FXA_2424S                       ; (Hutc-12)x15 - (C+R)
                movxf   Calcul1
                call    LONGITUDE                       ; Longitude terrestre
                btfss   longitude+2,0                   ; Test pour longitude OUEST ou EST
                goto    Angh1
                comf    x+2                             ; Inversion du signe de la longitude
                comf    x+1                             ; si longitude EST (longitude négative)
                comf    x+0                             ; Longitude+2 égal à 0 pour l'OUEST
                incf    x+0                             ; Longitude+2 égal à 1 pour l'EST
                skpnz
                incf    x+1
                skpnz
                incf    x+2
        Angh1
                movfy   Calcul1                         ; Longitude terrestre transférée dans (y)
                call    FXA_2424S                       ; Ajout de la longitude terrestre
                movxf   Calcul1                         ; Sauvergarde Angle horaire dans "Calcul1"
                bcf     SNEGaz                          ; SNEGaz est mis à O
                btfsc   x+2 , 7                         ; Test si Angle horaire est négatif
                bsf     SNEGaz                          ; Résultat négatif; SNEGaz est mis à 1
;               return

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

Hauteur:                        ; sin(H) = sin(Lat)sin(Dec) + cos(Lat)cos(Dec)cos(Ah)

                call    LATITUDE
                call    Sinus                           ; 6 chiffres
                movly   h'0',h'03',h'E8'                ; d'1000'
                call    FXD_2424SA                      ; Sinus de la latitude sur 3 chiffres
                movxf   ArcA                            ; Sauvegarde dans ArcA
                movfx   Calcul3                         ; Déclinaison sur 4 chiffres
                call    Sinus                           ; 6 chiffres
                movly   d'0',d'0',d'100'
                call    FXD_2424SA                      ; Sinus de la déclinaison sur 4 chiffres
                movfy   ArcA                            ; ArcA dans (y), ou Sin(Dec) dans (y)
                call    FXM_2424S                       ; Sin(Lat)sin(Dec)
                movly   h'00',h'00',h'0a'               ; d'10'
                call    FXD_2424SA
                movxf   RegHaut                         ; Sauvegardé dans le registre Hauteur
                movfx   Calcul3                         ; Dec (x100) soit 4 chiffres dans (x)
                call    Cosinus                         ; Cosinus de la déclinaison
                movly   h'0',h'0',h'64'                 ; d'100'
                call    FXD_2424SA                      ; COS(Dec) sur 4 chiffres
                movxf   ArcB                            ; Cos(Dec) sauvegardée dans ArcB
                call    LATITUDE                              ; Latitude
                call    Cosinus                         ; Cos(Lat)
                movly   h'0',h'03',h'E8'                ; d'1000'
                call    FXD_2424SA                      ; COS(Lat) sur 3 chiffres
                movfy   ArcB                            ; ArcA transféré dans (y)
                call    FXM_2424S                       ; Cos(Dec)Cos(Lat)
                movly   h'0',h'27',h'10'                ; d'10000'
                call    FXD_2424SA
                movxf   Calcul2                         ; Sauvegarde dans Calcul2
                movfx   Calcul1                         ; Angle horaire
                call    Cosinus                         ; Cosinus de l'angle horaire
                movly   h'00',h'00',h'64'               ; d'100'
                call    FXD_2424SA
                movfy   Calcul2                         ; Cos(Dec)Cos(Lat) dans y
                call    FXM_2424S                       ; Cos(Dec)Cos(Lat)Cos(Ah)
                movly   h'00',h'00',h'0A'               ; d'10'
                call    FXD_2424SA
                movfy   RegHaut                         ; Sin(Lat)sin(Dec)
                call    FXA_2424S                       ; sin(Lat)sin(Dec) + cos(Lat)cos(Dec)cos(Ah)
                call    ArcSinus
                movxf   RegHaut                 ; Sauvegarde "Hauteur" (sur 4 chiffres) dans "RegHaut"
;               return

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

Azimut:                 ;  cos(A) = (cos(dec)sin(lat)cos(Ah) - sin(dec)cos(lat))/cos(H)

                call    Cosinus                         ; Cos(H) sur 6 chiffres
                movly   0x00,0x03,0xE8                  ; d'1000'
                call    FXD_2424SA
                movxf   Calcul2                         ; Sauvegarde Cos(H) sur 3 chiffres dans Calcul2
                movfx   RegHaut                         ; RegHaut dans x
```

```
                movly   d'0',d'0',d'10'
                call    FXD_2424SA
                movxf   RegHaut                 ; "Hauteur" sur 3 chiffres dans "RegHaut"
                movfx   Calcul1                 ; Angle Horaire (Ah) soit Calcul1 dans x
                call    Cosinus                 ; Cos(Ah)
                movfy   Calcul2                 ; Cos(H) soit Calcul2 dans y
                call    FXD_2424SA              ; Cos(Ah)/Cos(H)
                movxf   RegAzim                 ; Sauvegarde dans RegAzim
                movfx   Calcul3                 ; Dec dans x
                call    Cosinus                 ; Cos(Dec)
                movly   0x00,0x03,0xE8          ; d'1000'
                call    FXD_2424SA              ; Cos(Dec)/1000

                movxf   ArcA                    ; Sauvegarde de x dans ArcA
                call    LATITUDE
                call    Sinus                   ; 6 chiffres
                movly   h'0',h'03',h'E8'        ; d'1000'
                call    FXD_2424SA              ; Sinus de la latitude sur 3 chiffres
                movfy   ArcA                    ; Sin(Lat) dans (y)

;               movly   0x00,0x02,0xAD          ; d'685'  sin(Lat)
                call    FXM_2424S               ; Cos(Ah)Sin(Lat)
                movly   0x00,0x03,0xE8          ; d'1000'
                call    FXD_2424SA              ; Cos(Ah)Sin(Lat)/1000
                movfy   RegAzim                 ; Cos(Ah)/Cos(H) dans y
                call    FXM_2424S               ; Cos(Ah)Sin(Lat)Cos(Dec)/Cos(H)
                movxf   Calcul1                 ; Sauvegarde dans Calcul1
                movfx   Calcul3                 ; Dec dans x
                call    Sinus                   ; Sin(Dec)
                movfy   Calcul2                 ; Cos(H) soit Calcul2 dans y
                call    FXD_2424SA              ; Cos(Dec)/Cos(H)

                movxf   ArcA                    ; Sauvegarde de x dans ArcA
                call    LATITUDE
                call    Cosinus                 ; 6 chiffres
                movly   h'0',h'03',h'E8'      ; d'1000'
                call    FXD_2424SA              ; Cosinus de la latitude sur 3 chiffres
                movfy   ArcA                    ; Cos(Dec)/Cos(H) dans (y)

;               movly   0x00,0x02,0xD9          ; cos(Lat)/1000   ,(729)
                call    FXM_2424S               ; (Sin(Dec)/Cos(H)) * Cos(Lat)
                comf    x+2                     ; Inversion du signe
                comf    x+1                     ;   de (Sin(Dec)/Cos(H)) * Cos(Lat)
                comf    x+0
                incf    x+0
                skpnz
                incf    x+1
                skpnz
                incf    x+2
                movfy   Calcul1                 ; Cos(Dec)Sin(Lat)Cos(Ah)/Cos(H)
                call    FXA_2424S               ; Cos(A)
                btfsc   x+2 , 7                 ; Cos(A) positif, Azimut < 90°
                goto    Cosazneg                ; Cos(A) négatif, Azimut > 90°

        Cosazpos
                call    ArcSinus
                comf    x+2
                comf    x+1
                comf    x+0
                incf    x+0
                skpnz
                incf    x+1
                skpnz
                incf    x+2
                goto    Fincosaz

        Cosazneg
                comf    x+2
                comf    x+1
                comf    x+0
                incf    x+0
                skpnz
                incf    x+1
                skpnz
                incf    x+2
                call    ArcSinus

        Fincosaz
                movly   h'00',h'23',h'28'       ; d'9000'
                call    FXA_2424S
                btfss   SNEGaz
```

```
            goto    Azimut3
            comf    x+2
            comf    x+1
            comf    x+0
            incf    x+0
            skpnz
            incf    x+1
            skpnz
            incf    x+2

      Azimut3
            movly   d'0',d'0',d'10'
            call    FXD_2424SA                  ; "Azimut" (x par 10)
            movxf   RegAzim                     ; Sauvegarde Azimut (sur 3 chiffres) dans "RegAzim"
            return

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

            END                     ; directive fin de programme
```

VII - TABLE DES SINUS

Un programme spécifique a été élaboré pour pour établir la table des sinus et la stocker dans la mémoire EEPROM 24AA256.
Pour rappel, cette table va contenir le sinus de 9000 valeurs de degrés exprimés en centièmes.
Chaque valeur est stockée sur 3 octets ; soit un total de 27000 octets.
La mémoire 24AA256 possède un buffer de 64 octets. Ce qui signifie que la table ne va pouvoir être construite que par plages de 64 octets.
Pour cela, on a établi 36 sous-programmes. Chacun de ces sous-programmes contient 12 séquences qui envoient chacune un ensemble de 64 octets vers la mémoire EEPROM.
Chacun de ces sous-programmes occupent environ 1650 lignes de la mémoire programme.
Le programme complet pour l'élaboration de la table peut donc contenir dans son ensemble dans la BANK0 du 16F886.
Les 36 sous-programmes seront appelés les uns après les autres à l'aide de la directive :
« # include ».
Dans le tableau suivant, les 36 sous-programmes listés font apparaîtrent les 12 séquences avec l'angle concerné au début de l'exécution de la séquence.
Le tableau fait également apparaître la valeur en centièmes de degrés en hexadécimal au début de l'exécution du sous-programme.
Les sous-programmes sont nommés :
Page1.asm, Page2.asm, ……..……jusqu'à..Page36.asm

Aperçu du programme Page1.asm

Les séquences d'envoi de 64 octets pour « Page1.asm » sont séparées par les étiquettes :

```
SIN0000, SIN0021, SIN0042, SIN0064, SIN0085, SIN0106, SIN0128, SIN0149, SIN0170,
SIN0192, SIN0213, SIN000234
```

La numérotation permet de repérer l'angle concerné
Exemple : le début de la séquence : SIN0021 concerne l'angle ayant pour valeur : 2,10°

```
SIN0000
     call       I2C_START_          ; start condition
     movlw      b'10100000'         ; Adressage de la mémoire 24AA256
     call       I2C_SB              ;
     movlw      0x00                ; Pointage vers adresse 0x00
     call       I2C_SB              ;
     movlw      0x00                ;
     call       I2C_SB              ;

     movlw      0x00                ; 1er octet de la 1ère valeur du sinus dans w
     call       I2C_SB              ; Envoi du 1er octet (valeur : 00)
     movlw      0x00                ; 2ème octet de la 1ère valeur du sinus dans w
     call       I2C_SB              ; Envoi du 2ème octet (valeur : 00)
     movlw      0x00                ; 3ème octet de la 1ère valeur du sinus dans w
     call       I2C_SB              ; Envoi du 3ème octet (valeur : 00)

     movlw      0x00                ; 1er octet de la 2ème valeur du sinus dans w
     call       I2C_SB              ; Envoi du 1er octet (valeur : 00)
```

```
      movlw     0x00              ; 2ème octet de la 2ère valeur du sinus dans w
      call      I2C_SB            ; Envoi du 2ème octet (valeur : 00)
      movlw     0xAF              ; 3ème octet de la 2ère valeur du sinus dans w
      call      I2C_SB            ; Envoi du 3ème octet (valeur : AF)
      -
      -
      -
      -         ; On insert ici les autres instructions
      -         ; Chaque séquence comporte 138 instructions au total
      -
      -
      movlw     0x00              ; 1er octet de la 21ème valeur du sinus dans w
      call      I2C_SB            ; Envoi du 1er octet (valeur : 00)
      movlw     0x0D              ; 2ème octet de la 21ère valeur du sinus dans w
      call      I2C_SB            ; Envoi du 2ème octet (valeur : 0D)
      movlw     0xA3              ; 3ème octet de la 21ère valeur du sinus dans w
      call      I2C_SB            ; Envoi du 3ème octet (valeur : A3)

      movlw     0x00              ; 1er octet de la 22ème valeur du sinus dans w
      call      I2C_SB            ; Envoi du 1er octet (valeur : 00)
      call      I2C_STOP_         ; Stop condition
      call      Tempo_10ms        ; Tempo 1à ms

SIN0021
      Call      I2C_START_        ; Start condition
      movlw   b'10100000'         ; Adressage de la mémoire 24AA256
      call      I2C_SB
      movlw     0x00
      call      I2C_SB
      movlw     0x40              ; Pointage vers l'adresse 0x40
      call      I2C_SB
      movlw     0x0E              ; 2ème octet de la 22ème valeur du sinus dans w
      call      I2C_SB            ; Envoi du 2ème octet (valeur : 0E)
      movlw     0x51              ; 3ème octet de la 22ème valeur du sinus dans w
      call      I2C_SB            ; Envoi du 3ème octet (valeur : 51)
      movlw     0x00
      call      I2C_SB
      movlw     0x0F
      call      I2C_SB
      movlw     0x00
      call      I2C_SB
      movlw     0x00
      call      I2C_SB
      movlw     0x0F
      call      I2C_SB
      movlw     0xAE
      call      I2C_SB
      -
      -
      -         ; Le sous-programme continu ici. Il contient 138 x 12 instructions
      -         ; Soit un total de 1656 instructions
      -
```

Chaque séquence a été établi à l'aide du tableur Open Office qui a permis de :

1 – Calculer la valeur du sinus d'un angle
2 – Récupérer la valeur ainsi trouvée avec 6 décimales pour l'exprimer avec un nombre entier de 6 chiffres
3 – Convertir cette valeur en hexadécimal sur 3 octets
4 – Ordonner convenablement toutes les valeurs sur 2 colonnes du tableau ou la 1er colonne représente l'opérande et la 2ème colonne l'argument de l'instruction.
5 – Récupérer les données en format « texte »
6 – Créer le fichier « .asm » à partir du fichier « texte » obtenu précédemment.

Nota :

L'élaboration de cette table a été assez fastidieuse.
Après réflection, elle aurait pu être établie avec une précision de 4 chiffres après la virgule au lieu de 6
Une telle solution aurait permis de codifier les sinus sur 2 octets au lieu de 3 (soit 18000 valeurs au lieu de 27000).
La précision des calculs aurait certainement été très peu affectée.

PLAGES DE LA TABLE DES SINUS

0000		03FF		0800		0C00		1000		1400		1800		1C00		2000	
1		5		9		13		17		21		25		29		33	
0,00	22	10,24	22	20,48	22	30,72	22	40,96	22	51,20	22	61,44	22	71,68	22	81,92	22
0,21	21	10,45	21	20,69	21	30,93	21	41,17	21	51,41	21	61,65	21	71,89	21	82,13	21
0,42	21	10,66	21	20,90	21	31,14	21	41,38	21	51,62	21	61,86	21	72,10	21	82,34	21
0,64	22	10,88	22	21,12	22	31,36	22	41,60	22	51,84	22	62,08	22	72,32	22	82,56	22
0,85	21	11,09	21	21,33	21	31,57	21	41,81	21	52,05	21	62,29	21	72,53	21	82,77	21
1,06	21	11,30	21	21,54	21	31,78	21	42,02	21	52,26	21	62,50	21	72,74	21	82,98	21
1,28	22	11,52	22	21,76	22	32,00	22	42,24	22	52,48	22	62,72	22	72,96	22	83,20	22
1,49	21	11,73	21	21,97	21	32,21	21	42,45	21	52,69	21	62,93	21	73,17	21	83,41	21
1,70	21	11,94	21	22,18	21	32,42	21	42,66	21	52,90	21	63,14	21	73,38	21	83,62	21
1,92	22	12,16	22	22,40	22	32,64	22	42,88	22	53,12	22	63,36	22	73,60	22	83,84	22
2,13	21	12,37	21	22,61	21	32,85	21	43,09	21	53,33	21	63,57	21	73,81	21	84,05	21
2,34	21	12,58	21	22,82	21	33,06	21	43,30	21	53,54	21	63,78	21	74,02	21	84,26	21

0100		04FF		0900		0D00		1100		1500		1900		1D00		2100	
2		6		10		14		18		22		26		30		34	
2,56	22	12,80	22	23,04	22	33,28	22	43,52	22	53,76	22	64,00	22	74,24	22	84,48	22
2,77	21	13,01	21	23,25	21	33,49	21	43,73	21	53,97	21	64,21	21	74,45	21	84,69	21
2,98	21	13,22	21	23,46	21	33,70	21	43,94	21	54,18	21	64,42	21	74,66	21	84,90	21
3,20	22	13,44	22	23,68	22	33,92	22	44,16	22	54,40	22	64,64	22	74,88	22	85,12	22
3,41	21	13,65	21	23,89	21	34,13	21	44,37	21	54,61	21	64,85	21	75,09	21	85,33	21
3,62	21	13,86	21	24,10	21	34,34	21	44,58	21	54,82	21	65,06	21	75,30	21	85,54	21
3,84	22	14,08	22	24,32	22	34,56	22	44,80	22	55,04	22	65,28	22	75,52	22	85,76	22
4,05	21	14,29	21	24,53	21	34,77	21	45,01	21	55,25	21	65,49	21	75,73	21	85,97	21
4,26	21	14,50	21	24,74	21	34,98	21	45,22	21	55,46	21	65,70	21	75,94	21	86,18	21
4,48	22	14,72	22	24,96	22	35,20	22	45,44	22	55,68	22	65,92	22	76,16	22	86,40	22
4,69	21	14,93	21	25,17	21	35,41	21	45,65	21	55,89	21	66,13	21	76,37	21	86,61	21
4,90	21	15,14	21	25,38	21	35,62	21	45,86	21	56,10	21	66,34	21	76,58	21	86,82	21

0200		05FF		0A00		0E00		1200		1600		1A00		1E00		2200	
3		7		11		15		19		23		27		31		35	
5,12	22	15,36	22	25,60	22	35,84	22	46,08	22	56,32	22	66,56	22	76,80	22	87,04	22
5,33	21	15,57	21	25,81	21	36,05	21	46,29	21	56,53	21	66,77	21	77,01	21	87,25	21
5,54	21	15,78	21	26,02	21	36,26	21	46,50	21	56,74	21	66,98	21	77,22	21	87,46	21
5,76	22	16,00	22	26,24	22	36,48	22	46,72	22	56,96	22	67,20	22	77,44	22	87,68	22
5,97	21	16,21	21	26,45	21	36,69	21	46,93	21	57,17	21	67,41	21	77,65	21	87,89	21
6,18	21	16,42	21	26,66	21	36,90	21	47,14	21	57,38	21	67,62	21	77,86	21	88,10	21
6,40	22	16,64	22	26,88	22	37,12	22	47,36	22	57,60	22	67,84	22	78,08	22	88,32	22
6,61	21	16,85	21	27,09	21	37,33	21	47,57	21	57,81	21	68,05	21	78,29	21	88,53	21
6,82	21	17,06	21	27,30	21	37,54	21	47,78	21	58,02	21	68,26	21	78,50	21	88,74	21
7,04	22	17,28	22	27,52	22	37,76	22	48,00	22	58,24	22	68,48	22	78,72	22	88,96	22
7,25	21	17,49	21	27,73	21	37,97	21	48,21	21	58,45	21	68,69	21	78,93	21	89,17	21
7,46	21	17,70	21	27,94	21	38,18	21	48,42	21	58,66	21	68,90	21	79,14	21	89,38	21

0300		0700		0B00		0F00		1300		1700		1B00		1F00		2300	
4		8		12		16		20		24		28		32		36	
7,68	22	17,92	22	28,16	22	38,40	22	48,64	22	58,88	22	69,12	22	79,36	22	89,60	22
7,89	21	18,13	21	28,37	21	38,61	21	48,85	21	59,09	21	69,33	21	79,57	21	89,81	21
8,10	21	18,34	21	28,58	21	38,82	21	49,06	21	59,30	21	69,54	21	79,78	21		
8,32	22	18,56	22	28,80	22	39,04	22	49,28	22	59,52	22	69,76	22	80,00	22		
8,53	21	18,77	21	29,01	21	39,25	21	49,49	21	59,73	21	69,97	21	80,21	21		
8,74	21	18,98	21	29,22	21	39,46	21	49,70	21	59,94	21	70,18	21	80,42	21		
8,96	22	19,20	22	29,44	22	39,68	22	49,92	22	60,16	22	70,40	22	80,64	22		
9,17	21	19,41	21	29,65	21	39,89	21	50,13	21	60,37	21	70,61	21	80,85	21		
9,38	21	19,62	21	29,86	21	40,10	21	50,34	21	60,58	21	70,82	21	81,06	21		
9,60	22	19,84	22	30,08	22	40,32	22	50,56	22	60,80	22	71,04	22	81,28	22		
9,81	21	20,05	21	30,29	21	40,53	21	50,77	21	61,01	21	71,25	21	81,49	21		
10,02	21	20,26	21	30,50	21	40,74	21	50,98	21	61,22	21	71,46	21	81,70	21		

VIII - CONCLUSION, REMERCIEMENTS:

Je tiens à remercier particulièrement mon fils Sébastien qui m'a mis le pied à l'étrier quant à l'utilisation et à la programmation des microcontrôleurs.

Lorsque j'effectuais mes Etudes dans des années lointaines, les premiers microprosseurs voyaient à peine le jour et les microcontrôleurs n'existaient pas encore.
Je vous parle d'un temps que les moins de 50 ans ne peuvent pas connaître.
L'électronique en ce temps là, se limitait à quelques circuits intégrés rassemblant de simples portes « nand ».

Ainsi, avec lui, j'ai pu m'initier à la programmation en assembleur et à la manipulation des principales fonctions offertes par les microcontrôleurs et dont j'avais essentiellement besoin pour démarrer les projets que j'avais dans la tête ; dont principalement celui décrit ici dans ces pages :

- Fonctions arithmétiques sur 2 ou 3 octets (additions, sousctractions, multiplications, divisions)
- Fonction affichage avec notamment l'utilisation d'un afficheur 16 caractères sur 2 lignes
- Utilisation du bus I2C et interconnexion avec une mémoire de type 24AA256
- Utilisation d'un circuit spécialisé RTCC MCP79410 (utilisant une interface I2C)

Je veux aussi remercier « BIGONOFF » pour ses fameux cours sur la programmation des PICs (dont le 16F84) et tous ceux qui comme lui, mettent leurs connaissances gracieusement sur le WEB à la disposition de tous.