Rapport de Stage I3
Septembre octobre 2007

IATRIDES Clément

8 novembre 2007

"Additional Photometer Project"

au

CETP : Centre d'étude des Environnements Terrestre et Planétaires

Centre d'étude des Environnements Terrestre et Planétaires
CETP

CNRS CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Table des matières

1 Remerciements

Je remercie tout particulièrement mon tuteur Michel GODEFROY, de m'avoir accompagné tout au long de ce stage dans les difficultés que j'ai rencontrées et pour ses explications en électronique, mais aussi Elena SERAN, pour m'avoir transmis ses connaissances sur les phénomènes physiques étudiés et pour son aide en informatique.

Je remercie l'ensemble du personnel du CETP Saint Maur pour l'accueil qu'ils m'ont réservé et pour leur disponibilité.

Je remercie également les fidèles sportifs avec lesquels j'ai pu me détendre en jouant au tennis de table lors des pauses déjeuner.

2 Introduction

2.1 CETP - Centre d'études des Environnements Terrestre et Planétaires

2.1.1 Présentation

Le CETP est une Unité Mixte de Recherche du CNRS et de l'Université de Versailles Saint-Quentin-En-Yvelines. C'est l'un des cinq laboratoires de l'Institut Pierre Simon Laplace. Il regroupe environ 100 permanents, chercheurs, ingénieurs et administratifs et accueille régulièrement une trentaine d'étudiants en thèse, visiteurs ou stagiaires. Il est implanté sur 2 sites, une centaine de personnes travaillant sur le site du Centre Universitaire de Vélizy, le reste du laboratoire étant implanté sur le site de l'Observatoire du Parc de Saint-Maur, lieu ou j'ai effectué mon stage.

Le CETP s'intéresse principalement à trois grandes thématiques. La première est centrée sur le cycle de l'eau, l'interaction et les échanges entre l'atmosphère et les surfaces continentales et ocaniques, ainsi que sur la formation, la structure, et l'évolution des systèmes précipitants et des nuages en couche mince. La seconde thématique porte sur l'étude de l'interaction du vent solaire (expansion de l'atmosphère ionisée du soleil) et les objets du système solaire : les planètes, dont la Terre, et les comètes. Elle inclut des travaux en physique des plasmas et en plantologie. L'ensemble de ces travaux fournit des résultats importants pour l'étude du climat et des processus météorologiques, ainsi que pour la compréhension des interactions entre le soleil et la Terre.

Pour ces recherches, le CETP développe des approches exprimentales originales, principalement basées sur des techniques radioélectriques : mesures de champs électriques et magnétiques dans les plasmas, observations par télédétection active (radars) et passive (radiomètres, micro-ondes) de l'atmosphère, des surfaces et du sous-sol. En complément des recherches thématiques, le CETP s'intéresse à un troisième volet de recherche, en lien avec ces techniques et de manière transversale aux deux premiers volets : il développe des recherches en modélisation électromagntique et en analyse et traitement du signal, avec notamment des travaux sur le traitement des images produites par télédétection.

2.1.2 Organigramme

Direction

- Directeur : Hervé de FERAUDY
- Directrice Adjointe : Danièle HAUSER
- Administratrice : Minh Trang BUI

Dèpartements Scientifiques

- **ABM** Atmosphère Basse et Moyenne
  Responsable : Yvon LEMAITRE

- **EMI** Electromagnétisme des Milieux Ionisés
  Responsable : Dominique DELCOURT

- **EMA** Electromagnétisme et Méthodes d'Analyse
  Responsable : Valrie CIARLETTI

- **IOTA** Interactions Terre Océan Atmosphère
  Responsable : Alain WEILL

- **OPN** Ondes dans le Plasma Naturel
  Responsable : Patrick CANU

Départements Techniques

- **IDA** Informatique Distribuée et Application
  Responsable : Francis VIVAT

- **ITS** Instrumentation Terrestre et Spatiale
  Responsable : Monique DECHAMBRE

Département Administratif

- **MGC** Moyens Généraux Communs
  Responsable : Minh Trang BUI

2.2 Le laboratoire CETP de Saint Maur des Fossés

Le site de Saint Maur des Fossés regroupe plusieurs laboratoires en plus du CETP, dont l'IPGP ce qui représente une centaine de chercheurs, ingénieurs de recherches et techniciens. Actuellement 43 personnes du CETP travaillent sur ce site, principalement des chercheurs du département EMI et des ingénieurs du département ITS, qui m'ont accueillis dans leur département pour la durée de mon stage. Ce département étant spécialisé dans les techniques spatiales, la majeure partie des équipements développés sont des équipements embarqués sur satellites.

Missions déjà lancées, où les membres de ce laboratoire ont participé :

- **Demeter** Detection of Electro-Magnetic Emissions Transmitted from Earthquake Regions :

  Lancé le 29 juin 2004 autour de la terre, ce satellite a pour premier objectif de rechercher l'existence de signaux électriques et magnétiques dans la haute atmosphère, associés aux crises sismiques et/ou volcaniques et plus particulirement des signaux associés à leurs phases pérparatoires. Mais aussi de déterminer les caractéristiques des perturbations de l'atmosphère neutre et de l'ionosphère liées à cette activité sismique. Ce satellite regroupe 7 instruments dont IMSC, un ensemble triaxial de 3 capteurs magnétiques (Search Coil), ICE, un système de 4 capteurs électriques, IAP, un analyseur plasma, ISL, une sonde de Langmuir, et IDP, un détecteur de particules.

- **Cassini-Huygens** :

  Cette sonde spatiale fut lancée le 15 octobre 1997 et a atteint sa destination finale, Saturne, dans le courant de l'année 2006. La mission Cassini-Huygens est une mission conjointe de la NASA, de l'Agence Spatiale Européenne (ESA) et de l'Agence Spatiale Italienne. Elle a pour objectifs de déterminer la structure en 3D et la dynamique du comportement des anneaux de Saturne, la composition de la surface des satellites et leur histoire géologique, ainsi que la nature et l'origine de la matière sombre prsente sur Japet. Elle a également pour objectif de mesurer la structure en 3D et le comportement de la magntosphre de Saturne, d'étudier la dynamique du comportement de l'atmosphre de Saturne au niveau de ses nuages, mais aussi d'étudier le comportement météorologique de Titan et sa surface.

– **Rosetta** :

La Mission ROSETTA, lancée le 2 Mars 2004 par ARIANE-5, de l'Agence Spatiale Europenne (ESA) a pour objectif l'étude de la comète Churyumov Gerasimenko avec laquelle la sonde a rendez-vous en Août 2014. Elle analysera la structure interne du noyau, ainsi que sa nature et composition minéralogique, chimique et isotopique.Cette mission s'intéresse notamment à la composante organique de la comète et à l'interaction du noyau avec le vent et la pression de radiation solaire.

Les Missions en cours de développement :

– **Taranis** :

La mission Taranis dont le lancement autour de la terre est pérvu en 2011 a pour objectifs scientifiques, l'estimation du taux d'occurrence des TLEs (Transiant light events) et des émissions associées, la mise en évidence des acteurs déclenchants. Elle s'intéresse aussi à la caractérisation des faisceaux d'électrons ("runaway electrons") accélérs, la mise en évidence des effets des TLEs et du rôle des électrons précipités sur le couplage entre les couches de l'atmosphère.

– **Bepicolombo** :

Cette mission dont le lancement est pérvu en 2013 devrait se mettre en orbite autour de Mercure en 2019. Elle a pour principaux objectifs de comprendre la structure interne de la planète en comparant le champ magnétique et la magnétosphère de Mercure à ceux de la Terre, et de contribuer à la compréhension de la formation et de l'évolution des planètes analogues à la Terre.

– **ExoMars** :

La charge utile proposée d'ExoMars, qui devrait être lancée en 2013, inclut un rover et un ensemble d'instruments fixes dédiés à l'étude de Mars (GEP). Ce dernier mesurera les propriétés géophysiques de première importance pour comprendre Mars et son habitabilité à long terme. Des mesures telles que l'activité sismique, tectonique et volcanique, mais aussi la mesure du flux de chaleur interne, seront effectuées. Il s'intéressera aussi au rayonnement UV, à l'étude de la poussière, de l'humidité et de la météorologie de la planète. Il devrait survivre plusieurs années sur Mars, offrant alors la possibilité de mesurer des variations sur le long terme de l'environnement, et permettra d'initier un futur réseau de stations scientifiques à la surface de Mars.

2.3 Le projet ALFA

2.3.1 Objectifs scientifiques

Le projet ALFA (Auroral Light Fine Analysis) a comme objectif la mise en oeuvre d'instruments optiques pour effectuer des mesures fines des émissions lumineuses dans le visible (400 - 900 nm) engendrées dans l'ionosphère par des électrons énergétiques accélérés sur les frontières magnétosphériques de la Terre. En effet ces électrons, issuent du vent solaire, vont en arrivant dans les régions polaires être accélérées par le bouclier magnétique de la terre et leur énergie cinétique va ioniser les atomes rencotrées sur leur passage. Ce changement de niveau d'énergie des atomes est accompagé de l'émission d'un photon dont la longeur d'onde est spécifique à l'atome exitée. Dans notre cas les atomes axitées sont principalement O, qui emmetra en fonction de l'énergie ressut soit à 630 nm soit à 558 nm, et $2N^+$ qui emmet à 428 nm.

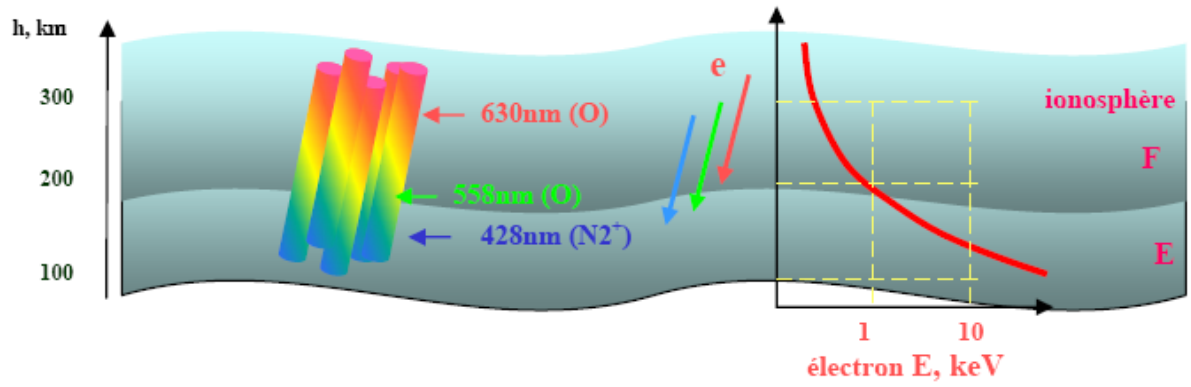

FIG. 1 – Ionosphere

L'instrument dans sa version "Dôme C" est composé d'une caméra "All-Sky" et de photomètres. "All-Sky" couvrira une région de 3000 km à l'altitude de 200 km, aura une résolution de 200 m au zénith et de 7 km en azimut à l'horizon et temporelle de $\approx$ 1 RGB image / minute. "All-Sky" effectuera une surveillance de l'intensité, de la distribution spatiale et de la variation temporelle des radiations lumineuses dans les bandes RGB. Tandit que les photomètres mesureront les variations temporelles des intensités lumineuses dans une large bande de fréquence. Le premier photomètre sera utilisé pour déterminer en temps réel les paramètres de contrôle automatique du "All-Sky", comme la sensibilité, le temps d'exposition, la résolution temporelle, etc. Deux autres photomètres seront utilisés pour les observations des émissions spécifiques dans la mésosphère et dans la ionosphère.

2.3.2 Moyens mis en oeuvres

Dans le cadre de l'étude de la magnétosphère, plusieurs autres équipements ont déjà été développés. C'est le cas du réseau de radar SuperDARN (Dual Auroral Radar Network), dont les quelques 9 radars installés au pôle sud et au pôle nord permettent de suivre en temps réel les flux de convection des plasmas de l'ionosphère. C'est également le cas de la mission DEMETER,"Detection of Electro-Magnetic Emissions Transmitted from Earthquake Regions", dont le satellite d'observation permet l'étude des perturbations de l'ionosphère et la mesure de l'environnement électromagnétique de la terre.

Dans le cadre du projet ALFA, a été développé le module "All-Sky" qui couvre, avec une résolution spatiale de 200 m, 3000 km de ciel grâce a une optique "fish-eye" et un capteur CCD (Charge-Coupled Device) de 6 Mpixels. Ce module est composé d'un appareil photo, le FUJI S3 pro dont le capteur nous permet d'avoir une sensibilité de 1600 ISO pour une résolution de 3024x2016 pixels, d'un GPS pour la datation et la localisation des clichés, mais également d'une batterie et d'un système de chauffage car les conditions de températures dans les régions polaires endicapent l'électronique et nous obligent à élever artificiellement la température pour avoir un comportement nominal. La récupération des photos et le contrôle du système se font par l'intermédiaire d'une connexion "FireWire" sur un ordinateur portable à l'aide de la bibliothèque SDK (Software Development Kit) de Fuji.

3 "Additional photometer project"

3.1 Objectif du stage

En complément du module "All-Sky", un deuxième instrument optique est en cours de développement. Celui-ci met en oeuvre deux photomètres et permet d'acquérir, de visualiser et d'enregistrer en temps réel l'intensité lumineuse mesurée, ainsi que plusieurs autres paramètres physiques tels que la température ou les "offsets" de courants des photomètres. La conception de ce système s'est naturellement décomposée en trois parties distinctes. Une partie électronique d'amplification, car on souhaite effectuer des mesures avec une sensibilitée de quelques pA sur une dynamique de 5 10^5. La seconde partie sera consacrée au developpement des servitudes, soit les mesures de températures, l'utilisation d'une centrale inertielle et d'un GPS. La troisième partie de ce projet est une partie de programmation pour le contrôle des acquisitions, le traitement, la visualisation et le stockage des données.

Pour conserver une homogénéité avec l'interface développée pour le module «All - Sky", la programmation se fera sous LabWindows 8.5, un logiciel de National Instruments dédié à la conception d'interfaces d'acquisitions. Il permet de créer simplement des interfaces en choisissant les composants dans une liste et de les placer sur l'écran. Il permet également, et c'est ce qui fait la force de cet environnement de développement, de programmer en langage C les fonctions de notre logiciel. LabWindows offrant des bibliothèques de fonctions simples à utiliser, on a fait le choix d'un module d'acquisition compatible avec cet environnement, ce qui nous a amené à nous pencher sur le USB-6008 de National Instrument, qui fut retenu pour des raisons de faible coût.

Fig. 2 – All-Sky module

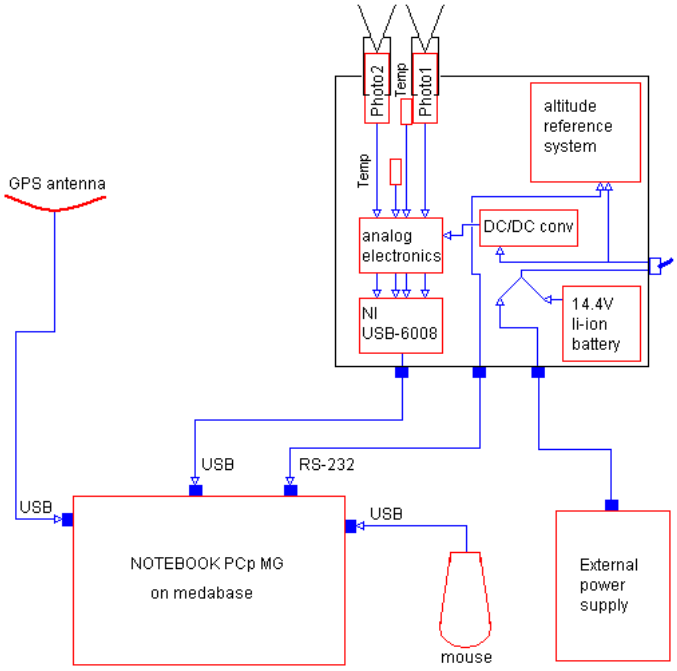

FIG. 3 – Diagramme

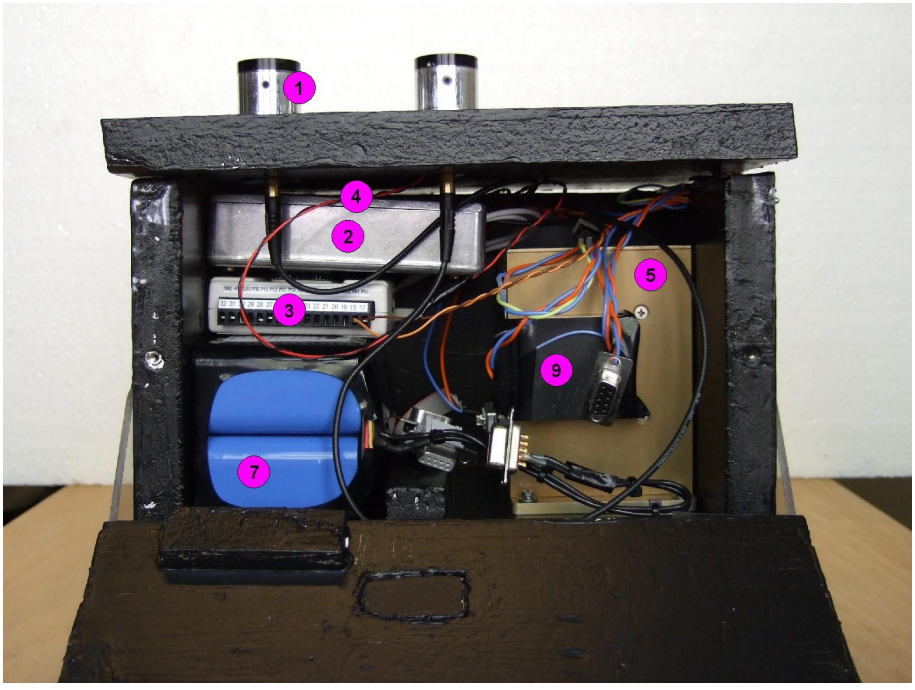

FIG. 4 – Photometer module

Composans du systèmes :

1. Les photomètres : Ce sont des photodiodes au silicium, S2281 de chez Hamamatsu, qui délivreront un courant en fonction de la luminosité. Elles détectent les radiations lumineuses dans un spectre de 190 à 1000 nm avec un pic de sensibilité à 720 nm et délivre un courant, appelé "Dark current", de l'ordre de 6 pA dans l'obscurité totale.

2. L'électronique analogique : Contiendra l'électronique analogique du système.

3. Le USB-6008 : Ce "multifunction I/O" de National Instruments possède quatre entrées analogiques multiplexées sur un ADC ayant une résolution de 12 bits pour les 8 dynamiques possibles, appelées "range" (± 1, ± 1.5, ± 2, ± 2.5, ± 4, ± 5, ± 10, ± 20), et une fréquence d'échantillonnage maximum de 10 kHz, deux sorties analogiques avec deux DAC (Digital to Analog Converter) ayant une dynamique de 0 à 5 V, 12 entrées sorties numériques, CMOS (Complementary Metal Oxide Semi-conductor) et TTL (Transistor-Transistor Logic) compatible, ainsi que deux alimentations stabilisées, 5 V et 2.5 v. Ce boîtier n'est, contrairement aux autres composants du système, validé que pour une température entre 0 C et 55 C.

4. Les capteurs de température : On utilise des AD590 d'Analog Device qui une fois alimentés par une tension comprise entre 4 et 30 V fournissent un courant proportionnel à la température absolue, 1 μA/ K.

5. La centrale inertielle : L'ARHS 400 de chez Crossbow est une centrale inertielle avec un contrôleur intégré ce qui lui permet à la fois de dialoguer avec un ordinateur via une connexion RS-232, qui standardise les communications de type série, mais aussi de corriger les mesures en fonction d'un de système calibration interne.

6. Le GPS (non montré) : Le récepteur GPS WS5011 de Wi-Sys connecté sur interface USB se pilote à travers un port RS-232 virtuel, ce qui facilite son utilisation.

7. La batterie : Elle est du type Li-Ion et fournit une tension de 14.4 V pendant, en estimant les consommations électriques des divers composants, 36 h. Ce qui permet d'avoir une grande autonomie.

8. La source de tension externe (non montrée) : C'est un adaptateur secteur qui va fournir du 12 V à partir de la tension secteur. Il permet de s'affranchir des batteries dans le cas ou l'environnement des observations possède une alimentation électrique.

9. Le convertisseur DC/DC : Il admet comme tension d'entrée une tension comprise entre 9 et 36 V et délivre une tension de 5 V avec un courant maximum de 4 A.

3.2 Développement du logiciel d'acquisition, de pilotage et de visualisation

3.2.1 Le boitier NI USB-6008

Le NI USB-6008 est un système comprenant plusieurs interfaces d'entrées - sorties. Il possède 4 entrées analogiques multiplexées , ce qui nous impose de faire les acquisitions successivement sur chaque voie, mais aussi de diviser la fréquence d'échantillonage maximum, qui est de 10 kHz, par le nombre de voies utilisées, ce qui nous ramène à une fréquence maximale de 2.5 kHz pour 4 voies. Ce système comporte également deux sorties analogiques et 12 entrées - sorties numériques. La présence d'une mémoire tampon interne permet à ce boîtier d'offrir 3 modes d'acquisitions et la compatibilité du NI USB-6008 avec LabWindows nous permet d'utiliser la large bibliothèque de fonctions de contrôles pour l'utilisation des entrées - sorties fournit par le constructeur.

Il existe trois modes d'acquisitions :

- Le mode "one - shot" :
  Ce mode permet de demander l'acquisition d'une ou plusieurs valeurs par voie sélectionnée et de lire ces valeurs.

```
 /* Configuration des entr\'ees */
DAQmxErrChk(DAQmxCreateTask("",&taskHandle));
// cr\'ee une tache sur le p\'eriph\'erique

DAQmxErrChk(DAQmxCreateAIVoltageChan(taskHandle,"Dev1/ai0","",DAQmx_Val_Cfg_Default,-1
    *range,range,DAQmx_Val_Volts,NULL));
// ajoute \‘a la tache une acquisition de signal analogique avec range l'une des valeur
     de configuration

DAQmxErrChk(DAQmxCfgSampClkTiming(taskHandle,"",sampling_rate,DAQmx_Val_Rising,
    DAQmx_Val_ContSamps,n));
// configure la fr\'equance des acquisition ainsi que le nombre d'acquisition luent par
     lecture du buffer

/* D\'emarrage des acquisitions */
DAQmxErrChk (DAQmxStartTask(taskHandle));
// d\'emarre l'acquisition

/* lecture */
DAQmxErrChk(DAQmxReadAnalogF64(taskHandle,n,10.0,DAQmx_Val_GroupByScanNumber,data_arrey
    ,arrey_size,&nombre_lu,NULL));
// lit les n acquisition et les stock dans une arrey
```

```
18
19 traitement(data_arrey,nombre_lu);
20 // traitement quelconque des acquisitions
21
22 /* fin de la tache */
23 DAQmxStopTask(taskHandle);
24 // arrete la tache
25
26 DAQmxClearTask(taskHandle);
27 // \'efface la configuration de la tache du p\'eriph\'erique
28
29 taskHandle = 0;
```

– Le mode continu :
Ce mode permet de demander l'acquisition continue de valeurs par voies sélectionnées et de lire ces valeurs.

```
1  /* Configuration des entr\'ees */
2      .......
3
4  /* D\'emarrage des acquisitions */
5      .......
6
7  gRunning = 1;
8  // nous permet de controler la boucle il est modifi\'e dans un autre thread et donc d\'
       eclar\'e en tant que variable globale et volatile
9
10 /* lecture continue */
11 while(gRunning)
12 {
13   DAQmxErrChk(DAQmxReadAnalogF64(taskHandle,n,10.0,DAQmx_Val_GroupByScanNumber,
         data_arrey,arrey_size,&nombre_lu,NULL));
14   // lit les n acquisition et les stock dans une arrey
15
16   traitement(data_arrey,nombre_lu);
17   // traitement quelconque des acquisitions
18
19 }
20
21 /* fin de la tache */
22     .......
```

– Le mode continu avec interruption :
Ce mode permet de demander l'acquisition continue de valeurs par voies sélectionnées et de lire ces valeurs avec une routine d'interruption.

```
1
2  int taskHandle=0;
3  volatile int gRunning=0;
4
5  int debut(void)
6  {
7   /* Configuration des entr\'ees */
8       .......
9
10 DAQmxErrChk (DAQmxRegisterEveryNSamplesEvent(taskHandle,DAQmx_Val_Acquired_Into_Buffer,
       config->n_sbloc,0,EveryNSamplesCallback,NULL));
11 // configure le p\'eriph\'erique pour qu'une interuption soit g\'en\'eri\'e \'a la fin
       de chaque bloc
12
13 /* D\'emarrage des acquisitions */
14      .......
15
16 gRunning = 1;
17
18   return 0;
19 }
20
21 int32 CVICALLBACK EveryNSamplesCallback(TaskHandle taskHandle, int32
       everyNsamplesEventType, uInt32 nSamples, void *callbackData)
22 {
23   float64 *data_arrey;
24   int nombre_lu,data_size;
25
26   /* allocation et initialisation des variables */
27   ......
28   ......
29   ......
30
31   /* lecture */  DAQmxErrChk(DAQmxReadAnalogF64(taskHandle,nSamples,10.0,
         DAQmx_Val_GroupByScanNumber,data_arrey,arrey_size,&nombre_lu,NULL));
32   // lit les n acquisition et les stock dans une arrey
33
34   traitement(data_arrey,nombre_lu);
35   // traitement quelconque des acquisitions
36
37   /* test de fin */
```

```
38   if(gRunning == 0) fin();
39
40   return 0;
41 }
42
43 int fin(void)
44 {
45   /* fin de la tache */
46        .......
47
48   return 0;
49 }
```

Une des subtilités de ce périphérique est que seul un type d'entrées - sorties et un seul mode entrée ou sortie peuvent être choisit au sein d'une même tache. On ne peut donc utiliser une même tache pour piloter des entrés numériques ou analogiques. De plus les quatre entrées analogiques étant multiplexées, on ne peut définir qu'une seule gamme de tension dans leurs configurations et la fréquence d'échantillonnage sera divisée par le nombre d'entrées.

Après avoir testé les différentes possibilités, la lecture continue avec interruption est celle qui fut retenue car c'est elle qui va, avec le système des interruptions, être la moins gourmande en ressources système car on n'exécute aucun code tant qu'un "événement", ici un bloc d'acquisition complet, ne s'est pas produit. On utilisera comme configuration par défaut pour les acquisitions, la gamme de tension de ± 5 V, une fréquence d'acquisition de 1000 Hz ainsi qu'un nombre de 5 sous blocs (de 20 acquisitions chacun)par bloc, ce qui entraîne une périodicité de 0.1 s pour les mesures.

3.2.2 Definition de l'interface de visualisation

Une interface utilisateur est comme son nom l'indique ce qui permet à l'utilisateur d'interagir avec le NI USB-6008, elle doit donc être composée de deux types d'éléments. Des éléments de visualisation des données, qui permettront dans notre cas de visualiser les intensités mesurées par les photomètres et les thermomètres, mais aussi des éléments qui permettent à l'utilisateur d'agir sur le système, comme changer la gamme de tension de l'interface d'acquisition, la fréquence d'échantillonage ou démarrer - arrêter l'acquisition.

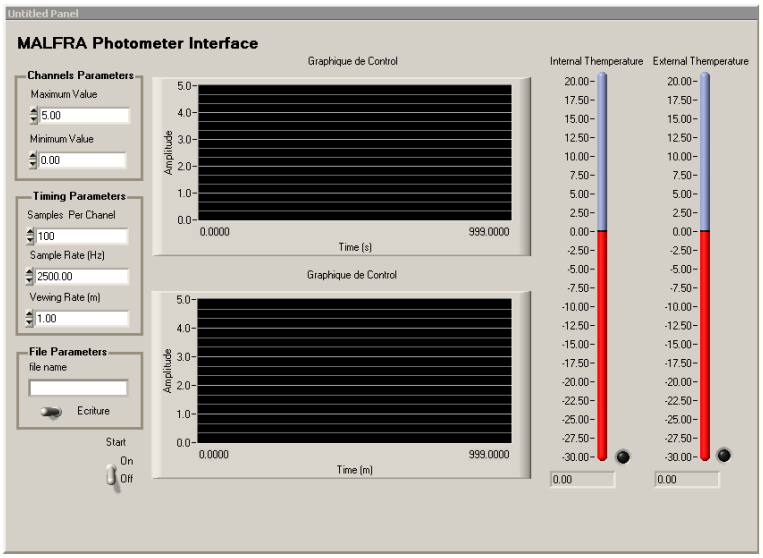

FIG. 5 – Premiere interface réalisée.

Cette interface comporte en plus du simple affichage des valeurs courantes, un affichage long terme, spécifié dans le cahier des charges, qui permet à l'utilisateur de visualiser l'évolution globale des phénomènes au cours de la nuit de mesures.

Après les tests effectués par Elena et Michel, plusieurs problèmes sont apparut. L'un de ces problèmes était la lisibilité des graphes, ce à quoi je pu rapidement remédier en séparant les graphiques de visualisation et en ajoutant des afficheurs numériques. Plusieurs nouvelles fonctions furent ajoutées pour mieux s'adapter aux besoins des chercheurs, tel que la possibilité de demander les mesures de luminosités particulières que sont le "dark courant", on couvre le capteur et on mesure le courant délivré, et le "sky background courant", on mesure le courant délivré par le capteur lorsqu'il n'y a pas de phénomènes. Ces deux mesures nous conduisent à un offset qui nous permet d'étalonner nos mesures, car ces valeurs de courants peuvent dériver en température et dans le temps, ce qui fausserait les mesures de luminosité des phénomènes.

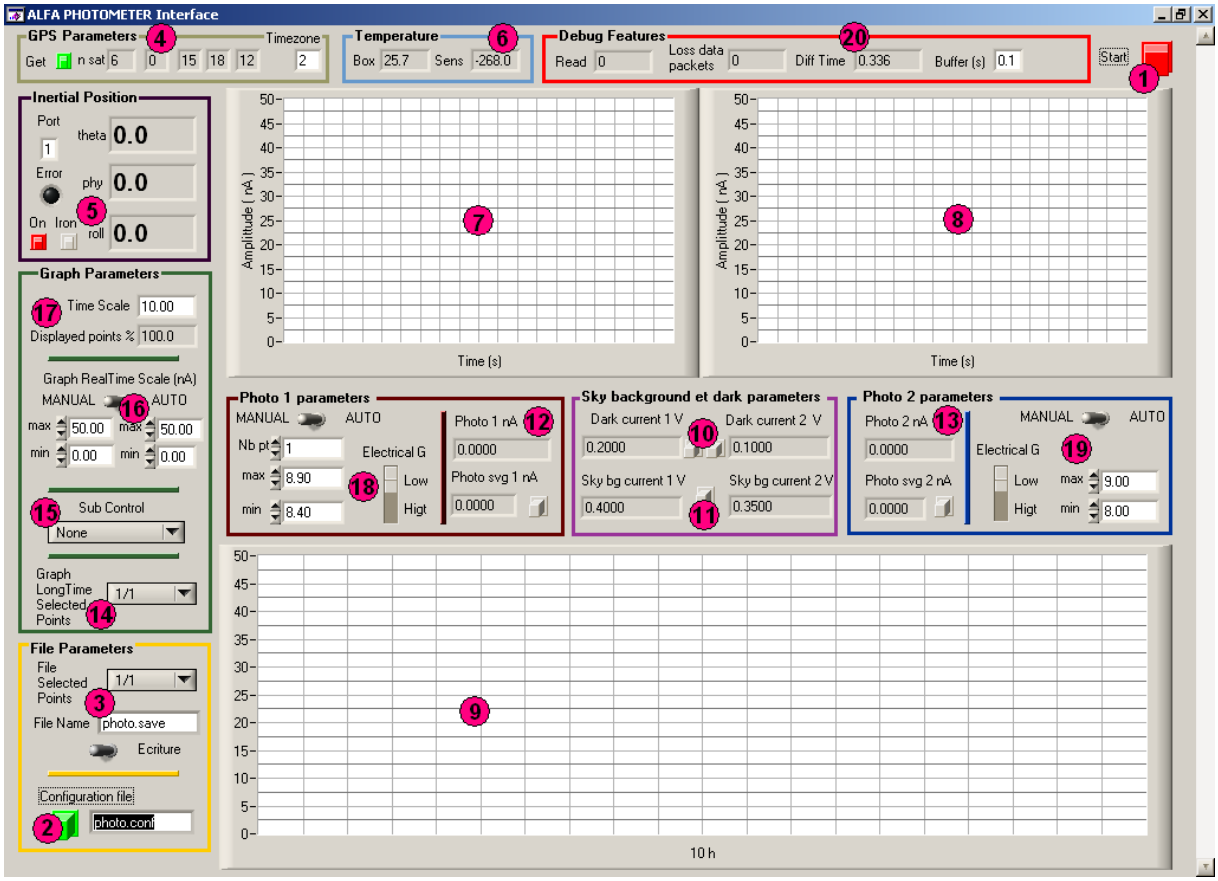

FIG. 6 – Seconde interface réalisée.

Composants de cette interface :

1. Ce bouton permet de démarrer et arrêter l'acquisition des valeurs des photomètres ainsi que des températures.

2. Ce bouton permet de sélectionner et charger les informations contenus dans le fichier de configuration, par défaut : "photo.conf". Si la configuration à été correctement chargée, l'indicateur lumineux passe du rouge au vert.

3. Ces trois éléments de l'interface permettent pour le premier de choisir le nombre de points à enregistrer dans le fichier de sauvegarde, ici "1/1" ce qui revient à enregistrer 1 point sur 1 soit tout les points, le second de choisir le répertoire et le nom du fichier de sauvegarde, par défaut "photo.save", qui lorsqu'aucun chemin n'est spécifié ce situe dans le même répertoire que le logiciel. Et en dernier le bouton qui active la sauvegarde lorsqu'il est placé du côté "Write".

4. Cette zone de dialogue est dédiée au GPS. Le bouton "Get" permet de déclencher la création d'une référence de temps et ne peut être activé que si l'acquisition des photomètres et des températures ne l'est pas. Le champ "n sat" affiche le nombre de satellites utilisés pour la dernière référence de temps si ce nombre est inférieur a 2 alors la référence de temps à été réalisée avec l'heure de l'ordinateur à laquelle on a soustrait la valeur du champ "Timezone" pour avoir une heure en UTC. L'heure de la dernière référence de temps est affichée dans les trois champs centraux.

5. Cette partie de l'interface concerne l'interaction avec la centrale inertielle, theta, phy et roll sont les trois angles qui nous permettent de définir dans quelle direction sont pointés les capteurs. Le bouton "On" permet de déclancher les acquisitions des valeurs des angles et le bouton "Iron" permet lorsque les acquisitions sont désactivées d'effectuer une calibration de la centrale inertielle. Le port de communication sur lequel est connecté la centrale inertielle peut être spécifié dans le champ "port". Pour ne pas pénaliser l'acquisition des photomètres, lorsque celle-ci est activée on passe d'une fréquence de rafraîchissement de 10 Hz à 0.2 Hz.

6. Ces deux valeurs correspondent aux températures de la boite électronique et des photomètres en C.

7. Graphique correspondant aux valeurs acquises pour le photomètre 1.

8. Graphique correspondant aux valeurs acquises pour le photomètre 2.

9. Graphique correspondant aux valeurs acquises pour les photomètres 1 et 2 pendant les 10 dernières heures.

10. Ces deux contrôles permettent de déclencher respectivement l'acquisition des valeurs de "Dark current" pour les photomètres 1 et 2, lorsque l'acquisition continue est arrêtée. Ces mesures sont comme expliquées plus tôt un peu particulières car elles nécessitent une intervention de l'opérateur qui doit aller boucher les photomètres pour qu'ils soient en condition d'obscurité totale.

11. Ce bouton permet de sauvegarder les deux valeurs courantes des photomètres 1 et 2 lorsqu'il n'y a pas de phénomène et ainsi relever la valeur du bruit de fond du ciel.

12. Le premier champ affiche en permanente la valeur courante en nA du photomètre 1. Le deuxième champ correspond à la valeur précédemment sauvegardée de ce photomètre lors de l'activation du bouton placé à ces cotés.

13. Identique à (12) mais pour le photomètre 2.

14. Ce contrôle permet de choisir le nombre de points affiché par le graph (9), ici "1/1".

15. Ce contrôle permet de soustraire ou non la valeur du "Dark current" (10) ou bien du "Sky background current" (11) aux valeurs des courants affichées par les graph (7) et (8).

16. Cette zone de l'interface permet d'agir séparément sur l'échelle d'amplitude des graph (7) et (8). Le bouton "MANUAL"/"AUTO" permet de définir soit une échelle automatique soit d'utiliser les valeurs minimums et maximums pour chaque graph, qui ont été spécifiées dans les champs ci-dessous.

17. Ces champs permettent d'agir sur l'échelle temporelle des graph (7) et (8). Le champ "Time Scale" permet de définir la durée affichée sur les graphs, ici chaque graph permet de visualiser les 10 dernières secondes de données. Le deuxième champ indique simplement le pourcentage de points affichés par les graphs car lorsque l'on augmente la durée, on n'affiche volontairement pas chaque point pour des raisons de lisibilité.

18. Ces 5 composants d'interface ont pour rôle de permettre le contrôle du gain analogique de l'électronique, qui permet d'augmenter la résolution des mesures des photomètres. Le bouton "MANUAL"/"AUTO" permet de choisir le mode de contrôle du gain désiré, dans le cas du contrôle manuel, le switch "Electrical G" permet de sélectionner le gain, soit "Low" qui est de 3.3 soit "High" qui est de 17. Dans le mode automatique, le switch "Electrical G" n'est que l'indicateur du gain courant, car celui-ci est déterminé selon l'algorithme suivant.

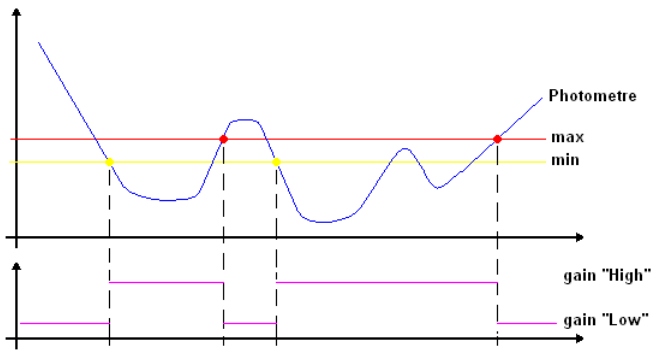

FIG. 7 – Illustration des changements de gains succesifs

19. Identique à (18) mais pour le photometre 2.
20. Ces champs affichent en temps réel les informations de debugage.

3.2.3 Chargement des paramètres de configuration et sauvegarde des données

La souplesse d'un logiciel comprent la facilité avec laquelle on peut modifier les paramètres ainsi que le nombre de ces paramètres modifiables, il était indispensable de pouvoir charger une configuration par défaut. Outre la configuration du USB NI-6008, les gammes de tension, la fréquence d'échantillonnage et le nombre de blocs de données, on doit absolument charger les fonctions de transfert tension / courant calculer les valeurs en nA des deux photomètres et de tension en C pour les sondes de températures. Ces données sont contenues dans un ficher texte et un " parseur " viendra récupérer les valeurs des quelques 25 lignes de configuration (Voir annexe). Le format simple du fichier texte a également été retenu pour la sauvegarde des acquisitions, car même si on ne réalise aucune compression sur les données, l'espace disque occupé par celles-ci n'excède pas 50 Mo pour 15 h de mesures. De plus en écrivant ligne par ligne les informations à sauvegarder, cela permet une relecture sans traitement des fichiers. Après plusieurs discussions sur le choix des informations à sauvegarder il fut convenu du format suivant :

heure(UTC) / photomètre-1 photomètre-2 temperature-photomèrtres temperature-boite (en V) / gains / dark courant photomètre 1 et 2 (en nA) / sky background courant photomètre 1 et 2 (en nA) / Anlge X / Angle Y / Angle Z / nombre de mesures acquisent pour le calcul des valeurs de la lignes

3.2.4 Division du programme en plusieurs "Threads"

L'acquisition de données en temps réel peut se diviser en 3 types de taches, l'acquisition proprement dite, qui est dans notre cas gérée par le NI USB-6008, le traitement mais aussi la visualisation et sauvegarde de ces données et la gestion de la centrale inertielle. Ces trois "fonctions" sont liées car on ne peut les exécuter dans n'importe quel ordre. Une des façons simples de concevoir la séquence d'acquisition serait d'appeler en boucle les fonctions les unes à la suite des autres. Cependant cela induit que le temps de traitement et de visualisation/sauvegarde est très inférieur à la période de récupération des données, ce qui n'est pas forcément toujours le cas. Par exemple l'écriture de données sur le disque dur de l'ordinateur est une opération lente et de durée non fixe, tout comme la lecture sur un port RS-232, pour la communication entre l'ordinateur et la centrale inertielle et le GPS, qui peut durer jusqu'a quelques dizaines de ms. Le temps de la boucle pourrait devenir supérieur à la période d'acquisition et donc entraîner un décalage dans les données ou pire une perte de données. Pour empêcher cette situation, le plus simple est de séparer le programme en plusieurs sous taches indépendantes. Ainsi on peut utiliser un système de pile pour " bufférisér " les éléments continuellement lus puis les traiter dans un autre "thread". Ce système nous permet de traiter les données en différé, de ce fait si le temps de traitement devient ponctuellement trop long on pourra rattraper ce retard par la suite.

Un "thread" est une fonction dont l'exécution est indépendante de la fonction qui l'a appelé, car les "threads" s'exécutent en parallèle et non de façon séquentielle, selon un plan de découpage du temps du processeur, qui ne peut exécuter qu'un "thread" à la fois. Un "thread" est donc une fonction particulière dont le comportement est lié à des changements d'états de variables globales, sur lesquelles le "Main thread" va agir. Dans la plupart des cas un thread est composé d'une boucle "while" qui teste la variable globale d'arrêt du "thread" et à l'intérieur de cette boucle une instruction "if" va déclencher une section de code à exécuter ou mettre en "pause" le "thread" et ainsi libérer le processeur.

3.2.5 Difficultés rencontrées dans cette premiere phase de développement

L'une des difficultés que j'avais envisagé était le pilotage d'un périphérique USB avec LabWindows, cependant la bibliothèque de fonctions et les exemples fournit par le constructeur m'ont permit de programmer rapidement une petite interface pour tester les différentes fonctions du NI USB-6008. L'exploitation des fonctions offertes par LabWindows devint difficile lorsque je fus amené à créer plusieurs " threads ". En effet même si toutes les fonctions des différentes bibliothèques sont accompagnées d'une aide, celle-ci étant parfois trop succincte et des recherches sur Internet ainsi que dans des ouvrages de références de programmation en C ont été nécessaires pour comprendre les notions de "processus" et de "threads", qui sont nécessaires pour mettre en place efficacement ce type de programme "temps réel".

3.3 Développement de l'électronique d'amplification

3.3.1 Conditions Physiques des mesures

Ce projet ayant pour but la caractérisation de phénomènes lumineux se déroulant dans des zones spécifiques et limitées du globe, notre électronique de transformation doit pouvoir fonctionner de façon nominale dans les conditions liées à ces environnements. Les aurores boréales surviennent comme leur nom l'indique dans les régions arctiques et antarctiques de notre planète, lieux où règnes de basses températures, de 10 C à -70 C en extérieur, et s'observent de nuit, moment où la luminosité est minimum permettant ainsi de ne pas saturer l'électronique d'amplification. On fait le choix de prendre comme maximum de dynamique une valeur légèrement supérieure au courant délivré par le photomètre lors d'une nuit de pleine lune, qui est d'environ 38 nA, soit 50 nA qui correspondra à 5V d'amplitude. Un des autres éléments critiques est l'alimentation électrique, car les conditions de mesures peuvent amener les chercheurs à utiliser des batteries Li-ion 14.4V, ce qui engendre des contraintes d'autonomie ainsi que de tensions d'alimentation et de puissance disponible.

3.3.2 Choix d'un gain sélectionnable numériquement

L'électronique de transformation permet d'adapter les signaux émis par les différents capteurs aux spécifications électriques de l'ADC contenu dans le NI USB-6008. En effet comme nous l'avons évoqué plus tôt les très faibles courants délivrés par les photomètres, de l'ordre du pA, devront être convertis en tension et amplifiés pour que la dynamique de notre circuit soit adaptée aux gammes de tensions prisent en charge par l'ADC. Dans un premier temps il avait été envisagé d'utiliser un gain électronique unique et de changer continuellement de gamme de tensions d'acquisitions pour avoir deux dynamiques et ainsi augmenter la dynamique globale d'acquisition. Cependant le temps de configuration ou de rappel de configuration d'une acquisition sur les voies analogiques du NI USB-6008 est de 25 ms, ce qui entraîne que pour une périodicité de mesure de 100 ms, seul 50 ms sont utilisés pour des acquisitions soit 50% du temps. Mais la raison principale qui nous fit choisir un gain analogique sélectionné était la mobilisation importante du système pour écrire continuellement sur le périphérique USB rendant ainsi le reste de l'ordinateur extrêmement lent à réagir aux interactions avec l'utilisateur. On pris donc le parti d'utiliser deux des entrées - sorties numériques du NI USB-6008 pour piloter deux interrupteurs analogiques qui permettent d'insérer ou de retirer une résistance du schéma électrique du second étage d'amplification des photomètres permettant ainsi de basculer les gains de ce second étage de 17 à 3.3. Ce qui nous permet d'avoir une dynamique allant jusqu'à 50 nA avec une résolution de 25 pA / bit et une dynamique allant jusqu'à 10 nA avec une résolution de 5 pA / bit.

3.3.3 Les photodiodes SI S2281

L'SI2281-01 de chez Hamamatsu est une photodiode au silicium qui délivrera un courant en fonction de la luminosité. Elle détecte les radiations lumineuses dans un spectre de 190à 1000 nm avec un pic de sensibilité à 720 nm et délivre un courant, appelé "Dark current", de l'ordre de 6 pA dans l'obscurité totale. La connexion se fait via un câble BNC, pour réduire la sensibilité au bruit.

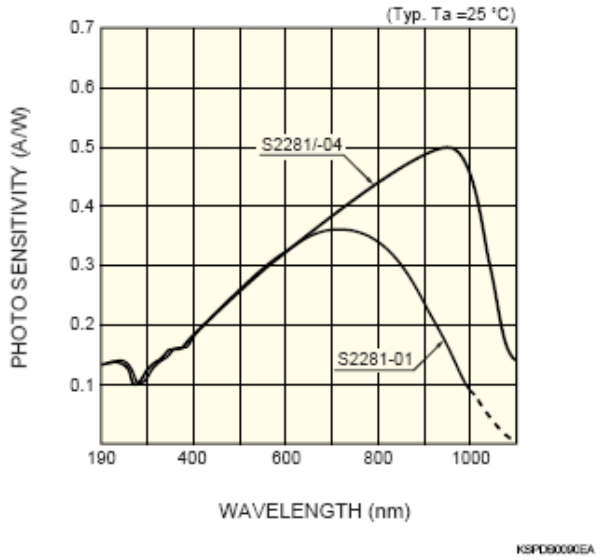

FIG. 8 – Réponse spectrale

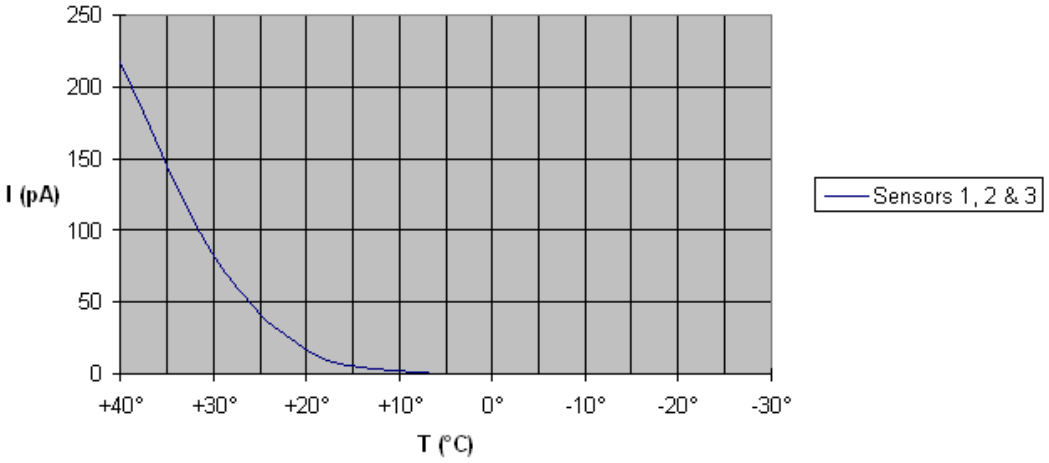

FIG. 9 – Dark current en fonction de la temperature

3.3.4 Les capteurs de temperature AD590

L'AD590 est une sonde de température délivrant un courant en fonction de la température absolue, pour une alimentation entre +4 V et +30 V le composant délivre 1 uA/K. Ce composant respecte donc les contraintes d'alimentation de la carte mais aussi les contraintes en température puisqu'il peut fonctionner jusqu'à -55C. Les montages présents dans la documentation technique du composant montrent que l'ajout d'une simple résistance de $1k\Omega$ permet de convertir le courant délivré en une tension. $1uA \Longleftrightarrow 1K$, or U = R * I et R = $1k\Omega$, donc $1K \Longleftrightarrow 1mV$

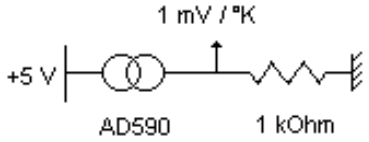

FIG. 10 – Montage 1 AD590

Ma première proposition fut donc d'utiliser une résistance de $10k\Omega$ pour obtenir $1K \Longleftrightarrow 10mV$ et ainsi adapter le signal du capteur à la dynamique et à la résolution de l'ADC, cependant ce circuit électrique si simple comportait un élément qui le rendait irréalisable. Cet élément était l'alimentation électrique +5 V de l'AD590, car avec une résistance de $10k\Omega$, on a 2.7315 V à 0C aux bornes de la résistance et $5-2.7315 = 2.2685V$ aux bornes de l'AD590 ce qui est inférieur à sa tension d'alimentation minimale. Nous avons donc du utiliser un amplificateur opérationnel avec un gain de 10 pour adapter le montage 1 :

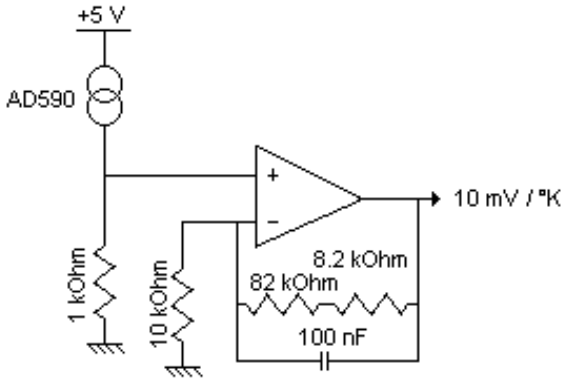

FIG. 11 – Montage 2 AD590

3.3.5 Réalisation de la carte électronique

La réalisation d'une carte électronique est souvent une tache sous-traitée par les développeurs car les opérations de routage, mise en place des composants et traçage des pistes électriques, et de soudure constituent une perte de temps non négligeable. Cependant dans le cadre de ce projet, nous avons choisi de réaliser nous même la carte pour des raisons de coût et à cause de la simplicité du schéma électrique à réaliser. Je me suis donc procuré un logiciel de Conception Assisté par Ordinateur, Eagle Lite, qui est une version gratuite de Eagle, mais aux fonctions réduites. Le développement de circuits imprimés avec ce type de logiciel se décompose en 2 phases. Dans un premier temps il faut saisir le schéma électrique du circuit en choisissant les composants dans des bibliothèques ou si ils ne sont pas disponibles en créant nous même les composants. Notre circuit peut se diviser en trois parties distinctes :

1. Les deux sondes de température et leurs montages amplificateurs.

   Dans cette partie du circuit on réalise comme expliqué plus tôt un gain de 10 avec un montage d'amplification non inverseur.

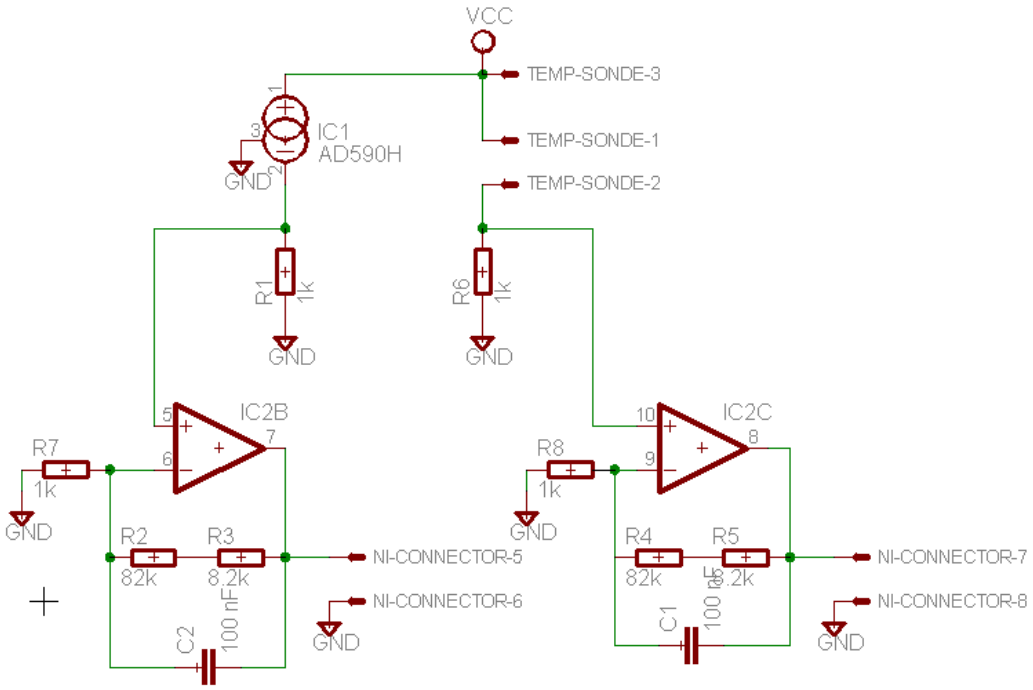

Fig. 12 – Schéma électrique

2. Les deux photomètres et leurs montages amplificateurs.

   Etant la chaîne d'amplification des capteurs de luminosité, elle constitue la partie critique du montage. Le première étage d'amplification est un convertisseur courant tension avec un fort gain, 30 mV / nA , et dont la fréquence de coupure sera de 25 Hz. La résistance de $30M\Omega$ déterminant à elle seule la valeur du gain, on utilisera des résistances calibrées, donc avec une valeur très proche ce celle recherchée. Pour réduires les "offsets" de l'électronique on utilisera une référence de tension 2.5 V et un pont de résistance et ainsi génerer un offset stable de 2.5 mV. Le second étage d'amplification est un montage non inverseur car le courant délivré par les photomètres étant négatif, il ne faut l'inverser qu'une seule fois pour obtenir la dynamique recherchée. La sélection de gain se fera par l'intermédiaire de l'ADG719 qui est un "switch analogique". Il sélectionnera, sur un niveau logique 1 de l'entrée IN, l'entrée analogique S2 mettant ainsi la résistance de $27k\Omega$ en parallèle avec celle de $160k\Omega$ ce qui modifie la valeur de la résistance qui intervient dans le calcul du gain et le fera passer de 17 à 3.3.

$$R_{eq} = \frac{27k\Omega * 160k\Omega}{27k\Omega + 160k\Omega} \approx 23k\Omega$$

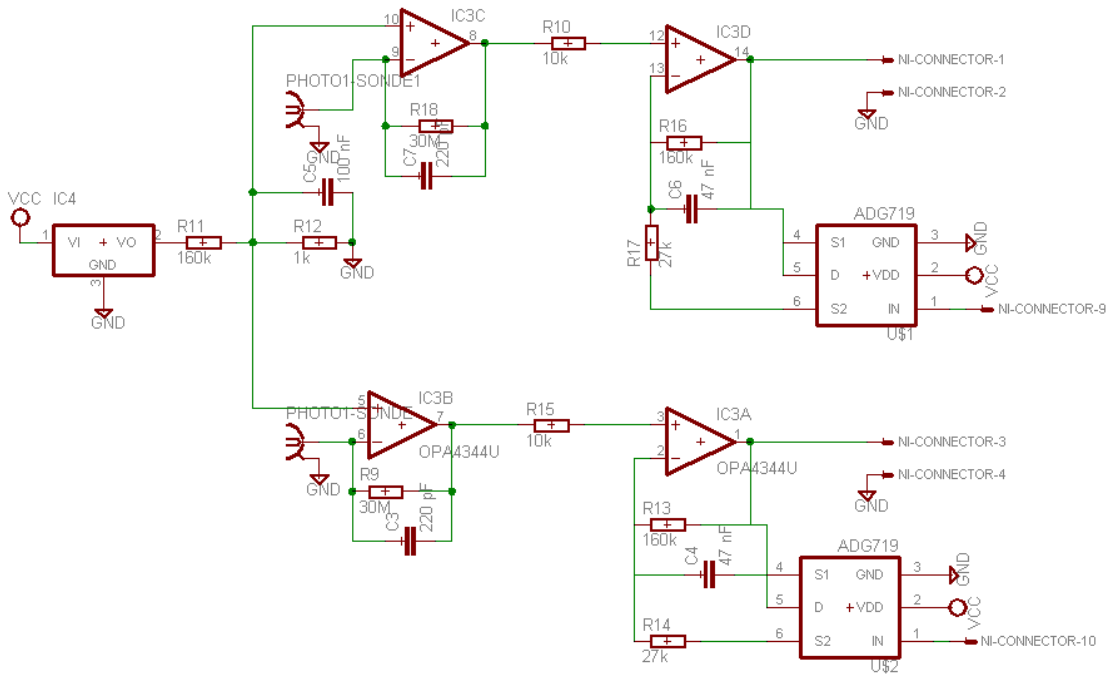

FIG. 13 – Schéma électrique

3. L'alimentation électrique du circuit et les capacités de découplage.

   Les capacités de découplages sont un élément importants du circuit car elle permettent de filtrer les tensions d'alimentations ce qui contribue à diminuer le niveau de bruit du signal. On a choisit de placer deux types de capacités, une capacité de grande valeur qui filtrera les basses fréquences à l'entrée de la carte et une capacité de faible valeur au plus prés des amplificateurs pour filtrer les hautes fréquences.

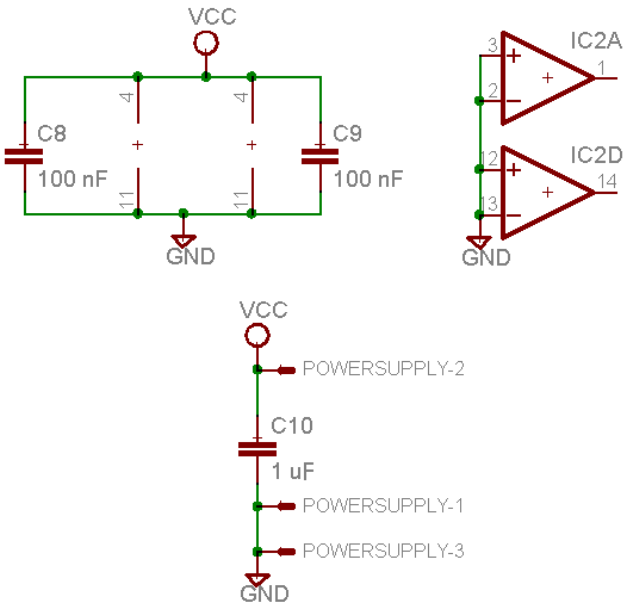

FIG. 14 – Schéma électrique

Après avoir saisi le schéma, il faut définir l'emplacement des composants sur la carte pour que le logiciel puisse tracer les pistes. Cette étape est longue car plusieurs paramètres nous empêchent de simplement disposer "en tas" les composants sur la plaquette. Il faut tenir compte de l'orientation des connecteurs pour ne pas avoir de problème surtout lors de la connection des câbles coaxiaux des photomètres, mais aussi tenter de placer les composants les plus près les uns des autres pour diminuer la sensibilité au bruit. Après plusieurs propositions, il fut convenu de placer les éléments comme suit et ainsi faire sortir tous les câbles par l'un des petits cotés de la boite, car cette disposition des éléments permettait également de minimiser la longueur des pistes ainsi que de faciliter le montage et le démontage des cables coaxiaux.

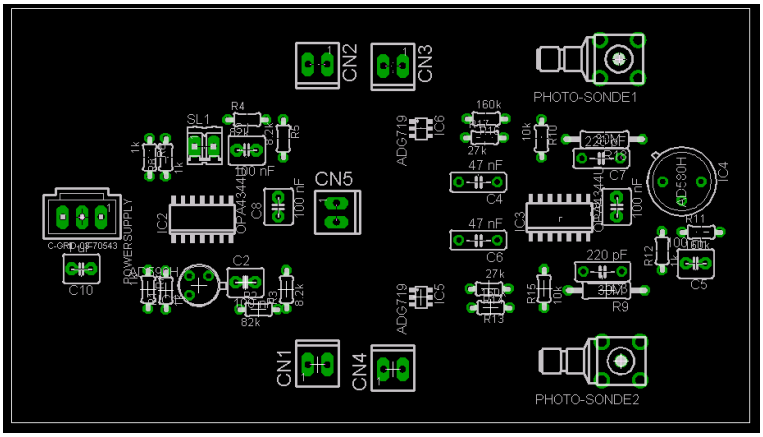

FIG. 15 – Schéma de placement des composants

3.3.6 Étalonnage du système et fonctions de transferts

L'étalonnage d'un système permet à l'utilisateur de le corriger en fonction des imprécisions et des écarts aux valeurs nominales observées dans un environnement contrôlé. Pour remédier aux différences entres les valeurs de gains théoriques et réels, ainsi que pour déterminer les offsets des amplificateurs, on a procédé à l'étalonnage de notre électronique d'amplification.

Pour injecter dans les voies des photomètres des courants contrôlés, on a utilisé une source de tension variable et une résistance de 200 $M\Omega$, nous obtenons donc 5 nA / V. La tension d'entrée est mesurée par un voltmètre et en mesurant, pour une série de valeurs de courants d'entrés en gain "Low" et en gain "High", les tensions à la sortie de notre électronique d'amplification, on a pu déterminer les fonctions de transferts réelles de cette électronique.

- Voie photomètre 1 :
  - Gain "Low" : 9.81823 x + -0.587407
  - Gain "High" : 1.96960 x + -0.518342
- Voie photomètre 2 :
  - Gain "Low" : 9.75791 x + -0.591995
  - Gain "High" : 1.95426 x + -0.522816

Pour étalonner l'amplification des voies de température, ne pouvant la faire varier de façon contrôlée, on a simplement utilisé un thermomètre pour déterminer la température ambiante lorsque la fenêtre était ouverte ou non et avons utilisé ces deux points pour calculer les fonctions de transferts.

- Voie termometre 1 : 98.5221675 x + -273.53
- Voie termometre 2 : 98.5221675 x + -273.43

3.4 Développement du logiciel de relecture des données

Le format de sauvegarde étant un fichier texte, bien que celui-ci permette de relire rapidement les données et de les exporter facilement, il ne permet pas de visualiser les données. Un logiciel de relecture fut donc défini en se basant sur le principe de l'oscilloscope, on charge l'ensemble des données en mémoire et des commandes permettent de sélectionner les courbes à afficher ainsi que de définir les échelles temporelles. Le logiciel va extraire de chaque ligne du fichier de sauvegarde les données et autres informations utiles, en utilisant une boucle ayant comme condition d'arrêt le nombre d'octets lus dans le fichier. En effet pour tester la fin du fichier, on récupère la taille du fichier que l'on va décrémenter de la taille de chaque ligne lue, car celle-ci est exclusivement composée de caractères occupant tous *sizeof(char)* octets. Une fois que cette "taille restante" à atteint 0, je complète le tableau en recopiant les dernières valeurs lues.

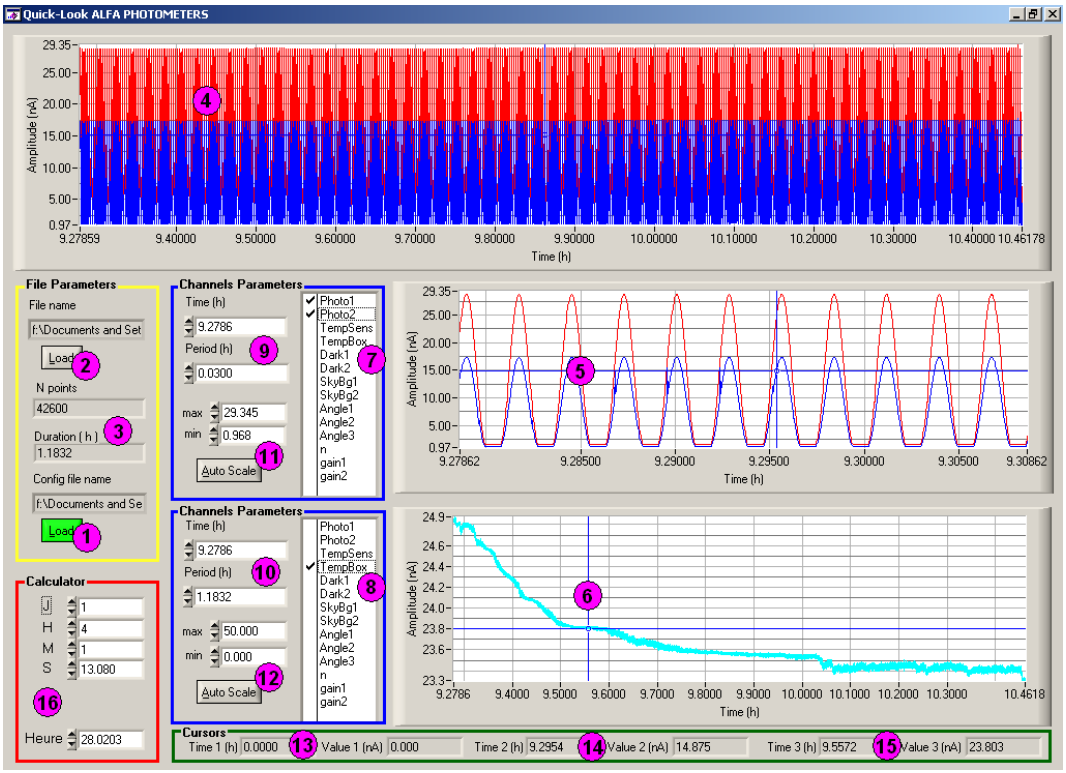

Fig. 16 – Interface du programme de relecture

Composants de cette interface :

1. Ce bouton permet de sélectionner et de charger le fichier de configuration, débloquant ainsi le bouton (2), car le chargement de cette configuration est obli-

gatoire, les données enregistrées sont en V tandis que les données affichées sont en nA.

2. Ce bouton permet de sélectionner et de charger le fichier de sauvegarde. Attention ci le format du fichier n'est pas bon il en résultera un " plantage " du logiciel, de plus cette opération est très longue et peut durer plusieurs dizaines de minutes dans le cas d'une nuit entière de mesures.

3. Ces deux champs indiquent le nombre de points trouvés dans le fichier et la durée d'acquisition correspondante.

4. Ce graph affiche l'ensemble des mesures pour les voies des deux photomètres pour toute la durée de l'enregistrement.

5. Ce graphe est l'écran paramétrable de visualisation de données.

6. Identique à (5).

7. La liste permet de sélectionner la ou les courbes à afficher sur le graph (5).

8. Identique à (7) mais pour le graph (6).

9. Ces deux champs permettent de contrôler l'échelle temporelle du graph (5). "Time" correspondant à l'heure de début du graph et "Period" est la durée affichée, ils sont exprimés en fractions d'heures.

10. Identique à (9) mais pour le graph (6).

11. Le contrôle "Auto Scale" permet de passer l'échelle d'amplitude du graph (5) en mode automatique tandit que les deux champs "Min" et "Max" permettent de spécifier les valeurs minimums et maximums à afficher.

12. Identique à (11) mais pour le graph (6).

13. Affiche la position du curseur sur le graph (4).

14. Identique à (13) mais pour le graph (5).

15. Identique à (14) mais pour le graph (6).

16. Convrtion de l'heure en jours heures minutes secondes et inversemment.

3.5 La centrale inertielle

3.5.1 Raison de cet ajout

Pour exploiter les résultats issus des observations il devient indispensable de pouvoir déterminer vers quelle région du ciel nos instruments sont pointés et fournir une référance aux images réalisées avec le module "All-Sky". Pour ce faire nous devons déterminer les angles d'orientations de notre appareil.

3.5.2 La centrale inertielle AHRS400

La centrale inertielle AHRS400 possède une interface RS-232 pour dialoguer avec l'ordinateur portable, peut être alimentée entre 9 V et 30 V et fonctionne de -40 C à 70 C. En mode Angle, cette centrale inertielle peut fournir à la demande les trois angles recherchés ainsi que des mesures d'accélération et de champs magnétiques sur ces trois mêmes axes.

FIG. 17 – AHRS400

3.5.3 Modification de l'interface et ajout de cette fonctionnalité

L'intégration de cette fonctionnalité me fut grandement facilitée par les travaux d'un autre stagiaire qui a réalisé une interface pour l'AHRS400 sous LabWindows peu de temps avant mon stage. En me basant donc sur son rapport de stage et le programme crée, j'ai pu rapidement communiquer avec la centrale inertielle, cependant j'ai constaté que l'environnement actuel avait une grande influence sur les angles recherchés car le délai de

stabilisation de la centrale après un changement d'inclinaison était au début très grand. Je fut donc contraint de moyenner plusieurs mesures pour stabiliser les valeurs des angles, ce qui a eu un impact direct sur la rapidité d'exécution de mon programme, car les communication avec le protocole RS-232 peuvent pour chaque cycle demande réception durer quelques dizaine de ms. Pour ne pas ralentir la récupération des valeurs de tensions analogiques du NI USB-6008, j'ai choisi de limiter la fréquence de rafraîchissement des angles à 0.2 Hz lorsque les acquisitions sont activées et à 10 Hz dans le cas contraire ce qui correspond a un bon compromis car une fois la mise au point effectuée, les angles sont supposés être fixes.

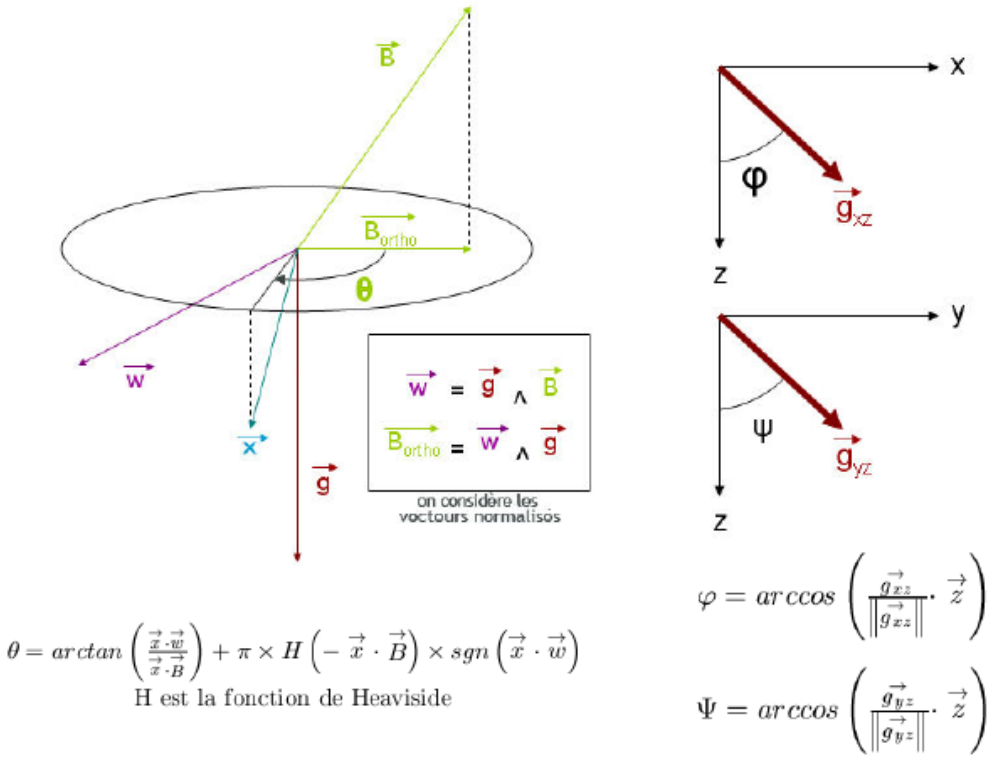

FIG. 18 – Schéma utilisé pour la détermination des angles à partir des mesures d'accélérations, extrait du rapport de Mr Clement MOUSSU

La fonction $get\_fields()$, contenu dans le fichier centralIn.c, est la transcription des calculs vectoriels présentés ci-dessus et nous permet de choisir entre les angles calculés par la centrale inertielle, que nous privilégierons car ils sont corrigés pour minimiser les erreurs dues aux perturbations magnétiques, et ceux que nous calculons avec les mesures d'accélérations et les mesures de champ magnétique faites par la centrale inertielle.

3.6 Le récepteur GPS

3.6.1 Raison de cet ajout

Dans un premier temps la datation des acquisitions se faisait en utilisant l'heure de l'ordinateur, mais celle-ci dérive en fonction du temps, introduisant un offset entre le temps de l'ordinateur et le temps UTC. On utilisera donc un GPS, qui fournit l'heure avec une précision meilleur que la ms pour dater les acquisitions avec une précision de la ms.

La datation avec le temps de l'ordinateur s'effectue par la fonction $GetsystemTime(int* h, int * m, int * s)$, cependant cette fonction ne retourne qu'un nombre entier de secondes il faut donc ajouter l'utilisation de la fonction $GetTickCounts()$, qui retourne le nombre de ms qui se sont écoulées depuis que l'ordinateur est allumé (dans une limite de 49.7 jours), pour créer une référence de temps. Le calcul de la date d'un échantillon se fait en soustrayant la valeur obtenue avec un deuxième $GetTickCounts()$ et la valeur stockée dans la référence de temps pour obtenir le nombre se ms qui se sont écoulée depuis la référence, en y ajoutant l'heure de la référence, on obtient l'heure où l'on a fait l'acquisition de cet échantillon. Mais comme expliqué plus tôt, il existe un Δt en ms d'offset entre l'heure UTC et l'heure ainsi calculée à cause de la dérive du circuit d'horloge de l'ordinateur, car une seconde ne dure pas exactement une seconde. Ce qui n'est pas le cas sur un GPS, qui permetra de créer sur demande de l'opérateur une référence de temps précise, chose qui devra être faite plusieurs fois par nuit pour éviter que les imprécisions de temps auquel $GetTickCounts()$ est également sujet ne faussent la datation des mesures.

3.6.2 Le récepteur GPS WS5011

Le capteur GPS WS5011 se compose d'une antenne, d'un contrôleur GPS et d'un câble USB. Un des atouts de ce capteur est la facilitée d'interaction avec l'utilisateur, car même si c'est un périphérique USB l'utilisateur communique en utilisant un port série virtuel, ce qui élimine les possibles difficultés de communications avec le protocole USB. Ce périphérique envoie par ce port série virtuel plusieurs informations en utilisant la norme NMEA, cette norme de communication définit un certain nombre de "phrases" que le "Talker", ici le GPS, va envoyer pour transmettre la latitude, la longitude et l'altitude du capteur, ainsi que l'heure UTC, mais aussi le nombre de satellites que le périphérique capte et leurs positions.

Fig. 19 – Le GPS WS5011

3.6.3 La norme NMEA 183

La norme NMEA 183 définit un ensemble de messages qui sont soit émis par le récepteur GPS soit par les autres périphériques connectés, ces messages sont composés de caractères ASCII transmis par port série (RS-232). Le message commence par le caractere $, puis vient l'identifiant de l'émetteur sur deux caractères et 3 caractères formant le mnémonique du message. Ensuite viennent les paramètres du message, tous séparés par des virgules sans espaces, des paramètres pouvant être vide mais on conserve les virgules, et pour finir viennent le caractère * qui termine le message puis le "checksum" qui est un XOR de tous les caractères du message. Les caractères de fin de ligne et de retour chariot sont ajoutés en fin de ligne pour faciliter la visualisation des données.

Dans notre cas, on s'intéresse à la trame GGA, car elle fournit en plus de l'heure UTC, le nombre de satellites que le récepteur utilise. Elle est de la forme :

$GPGGA,HHMMSS.SSS,XXXX.XXX,N,XXXXX.XXX,W,X,XX,XX,XXXX,M,XXXX,M,SSSS,XXXX*ZZ[\CR][\LF]

- GP : Message vennant du GPS
- GGA : Données du point du système de positionnement mondial
- HHMMSS.SSS : L'heure avec une précision de la ms. Exemple 100703.000 il est 10 heure 7 minutes et 3 secondes
- XXXX.XXX,N : Latitune Nord ou Sud
- XXXX.XXX,W : Longitude Est ou Ouest
- X : 0 = point non calé, 1 = point calé, 2 = point calé en mode différentiel, 6 point estimé
- XX : Nombre de satélites utilisées
- XX : Dilution horizontale de la précision
- XXXX, M : Altitude en mètres de l'antenne au dessus du niveau de la mer
- XXXX, M : Différence en mètres entre l'ellipsoïde WGS84 et le niveau moyen de la mer
- SSSS : Age des données différentielles en secondes
- XXXX : Numéro de la station différentielle
- ZZ : Checksum

3.6.4 Modification de l'interface et ajout de cette fonctionnalité

L'intégration de cette fonctionnalité fut, malgré un nombre de lignes de code restreint, assez difficile car le récepteur GPS ne fournit pas d'heure à la demande, il fournit l'heure, à travers la phrase NMEA "GGA", toutes les secondes. La récupération d'une heure doit se faire avec le minimum de données « bufférisées », car sinon il existerait un delta de temps entre l'arrivée des données sur le port série et la lecture de celui-ci par le logiciel. On va donc tester dans une boucle la condition de retour du « parseur » à quelques ms d'intervalle, qui renverra -1 si il n'a pas détecté la trame recherchée, dans les données de la mémoire tampon et 0 si il a bien récupéré l'heure et le nombre de satellites. Et si cette heure n'a pas été récupérée au bout de N tentatives ou si le nombre de satellites est inférieur à 2, on utilisera l'heure de l'ordinateur.

3.7 Une carte de contrôle d'alimentation en option

3.7.1 Raisons du développement de cette carte

L'alimentation électrique des différents éléments du projet étant une batterie, on doit prendre en compte des questions de consommation électrique et d'autonomie. Les caractéristiques d'une batterie varient au cours du temps, dans le cas d'une batterie Li-Ion la tension décroît lentement jusqu'à atteindre la valeur de tension nominale, ici 14.4 V, puis s'effondre en quelques secondes. Or lorsque cette tension s'effondre à zéro, les composants ne sont plus alimentés, ils ne fonctionnent plus, il nous faut donc détecter l'état de la batterie pour brancher ou non une source de tension secondaire. En calculant la consommation électrique des diverses parties du système, on observe que la plus grande partie de l'énergie consommée l'est par la centrale inertielle qui consomme 4 W alors que l'électronique d'amplification consomme quelques mW. Donc en plus de détecter la charge de la batterie, on a tout intérêt à pouvoir couper ou non l'alimentation électrique de chaque composant séparément pour ne pas alimenter ceux qui ne sont pas utilisés et ainsi prolonger la durée d'utilisation de la batterie.

3.7.2 Réalisation de la carte électronique

La détection de la charge de la batterie est assez simple, on utilise un comparateur et deux ponts de résistances pour avoir une image de la tension de la batterie et réaliser un montage de comparateur à hystérésis.

Pour réaliser la fonction d'interrupteur commandé, je me suis d'abord penché sur des composants existants mais aucun de ceux trouvés ne convenaient, car je ne dispose pas d'une alimentation pour ces composants qui soit égale ou supérieure à la tension que je veux contrôler. Après discussions avec l'un des ingénieurs du laboratoire, il m'expliqua le fonctionnement d'un montage à base de transistors MOSFET qui permet de contrôler une tension quelconque avec des niveaux logiques.

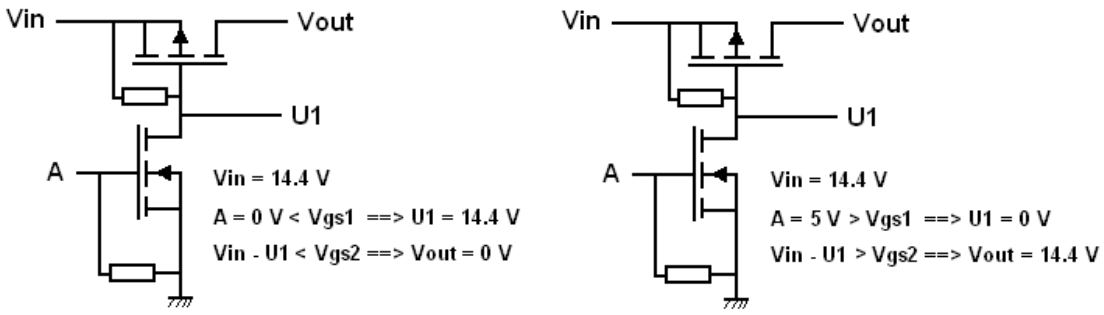

Fig. 20 – Fonctionnement du contrôleur à MOSFET

En utilisant cette fonction d'interupteur analogique pour contrôler l'alimentation du circuit à partir de la source d'alimentation externe et l'alimentation de le centrale inertielle ainsi que du convertisseur DC/DC, on aboutit au schema suivant. On ajoute également deux diodes pour isoler les différentes sources de tensions.

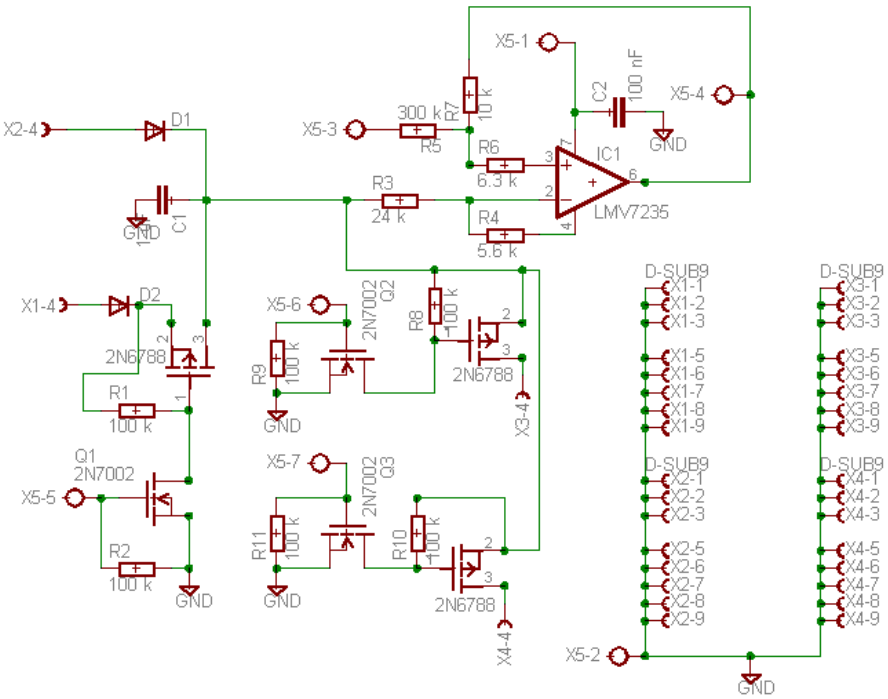

FIG. 21 – Shema de la carte

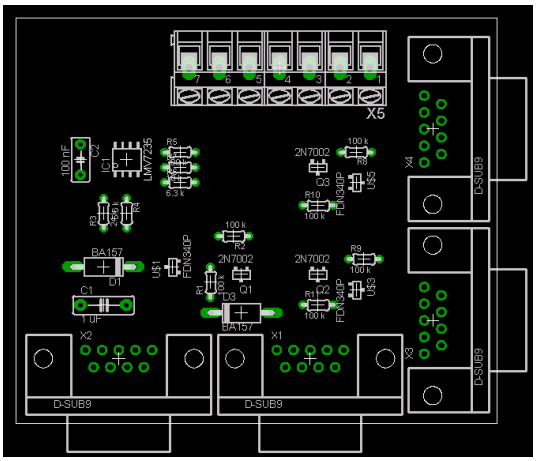

FIG. 22 – Placement des composants

4 Bilan Personnel

Ces deux mois de stage m'ont permis de découvrir les compétences nécessaires à la prise en charge, au sein d'une équipe, d'un projet. En effet le respect des contraintes imposés par le cahier des charges et le planning d'avancement d'un projet, met en oeuvre des compétences spécifiques : organisation personnelle du travail, recherche et utilisation des ressources disponibles, approfondissement des compétences et du savoir faire technique liés au développement aussi bien logiciel qu'électronique. J'ai appris au contact de cette équipe d'ingénieurs/chercheurs, comment organiser les phases de développement et mon travail personnel pour être le plus efficace possible, mais aussi une organisation du travail avec les responsables techniques et les chercheurs dans le cadre d'un projet. Les équipes techniques ont pour rôle de concevoir et réaliser les équipements nécessaires aux expérimentations, ainsi que les méthodes associées. Ce travail implique une discussion permanente des deux parties, car le cahier des charges évolue en permanence pour s'adapter aux spécifications des chercheurs. Cela m'a permit d'apprendre à anticiper les différentes orientations que le projet pouvait prendre et donc à constamment me documenter sur les différentes fonctions offertes par l'environnement de développement dans le cas du logiciel mais aussi les composants et architectures utilisables pour réaliser des fonction électroniques. La création de l'interface logicielle fut la phase de développement la plus longue, car créer une interface qui permet à l'utilisateur de visualiser les données et de contrôler de façon simple et efficace le système était plus complexe que ce que j'avais imaginé. En effet entre la première interface très simple créée et la version finale on remarque en plus de l'ajout de fonctionnalités un changement dans la façon d'afficher les données pour que celles-ci soient directement accessibles et compréhensibles par les chercheurs. J'ai également pu observer comment, dans le cadre de projets plus importants, tel que les missions embarquées sur satellites, ces équipes organisent leur travail pour respecter les importantes contraintes qui leurs sont imposées.

Ce stage fut également l'occasion, au travers du projet réalisé, de confirmer mon intérêt pour des projets qui mixent de l'électronique et de l'informatique, car ils mettent en oeuvre des compétences de programmation mais aussi d'architecture de fonctions électroniques. Cet intérêt, déjà révélé au cour du projet d'électronique numérique du second semestre de I2, dans lequel j'avais participé à l'interfaçage d'un clavier PS/2 et d'un écran LCD avec un FPGA, conforte dans mon orientation professionnelle et me pousse à intégrer la majeur "Systèmes Embarquées" lors de ma quatrième année.

5 Bibliographie

Références

[1] Claude DELANNOY, *Langage C, Eyrolles, 2005*

[2] Site Internet du CETP : http ://www.cetp.ipsl.fr

[3] Site Internet de Superdarn : http ://superdarn.cetp.ipsl.fr/

[4] L'encyclopédie en ligne Wikipedia : http ://fr.wikipedia.org

[5] La bibliotheque de documentation composants en ligne : http ://alldatasheet.com

6 Annexes

.1 Photos de l'équipe MALFRA

FIG. 23 – Elena SERAN

FIG. 24 – Michel GODEFROY

FIG. 25 – Jean-Claude CERISIER

FIG. 26 – Fabrice BERTRAND

.2 Photos du système

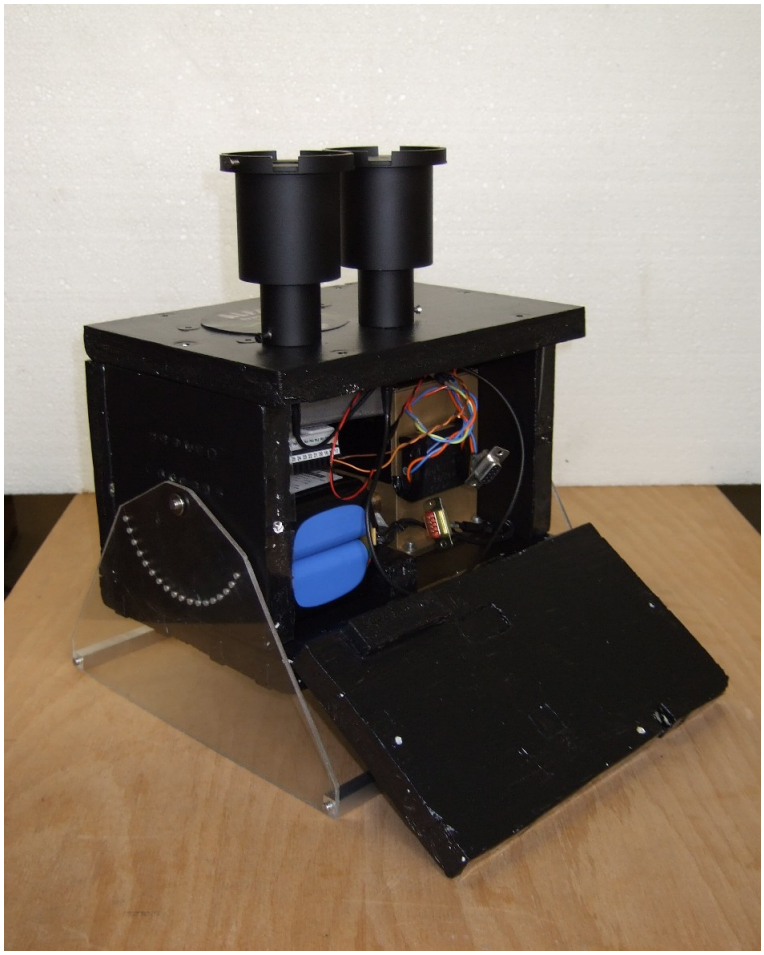

FIG. 27 – Le module complet de " l'additional photometer project "

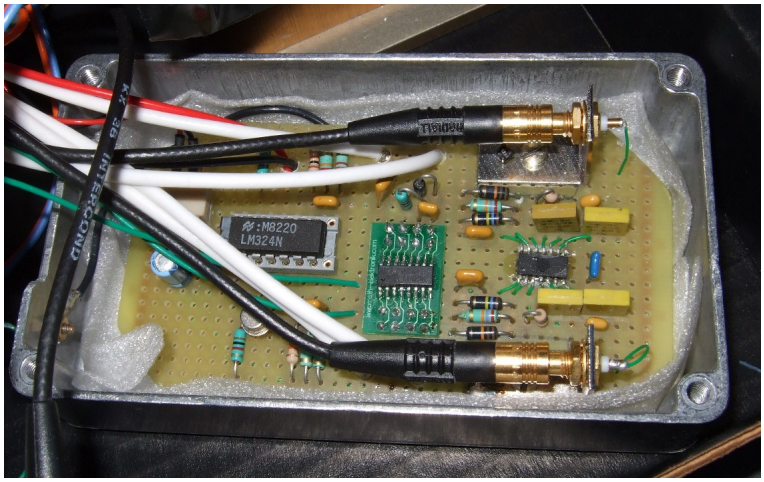

FIG. 28 – Carte électronique d'amplification réalisée

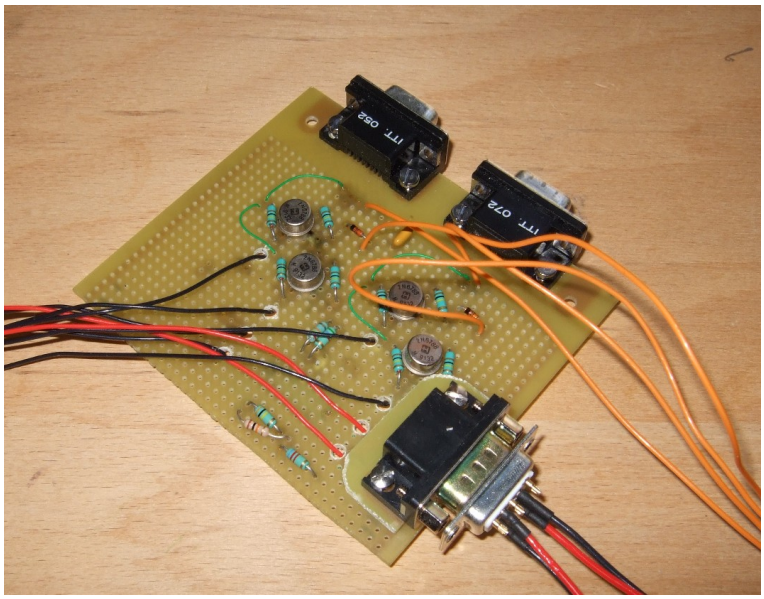

FIG. 29 – Carte du contrôleur d'alimentation réalisée

.3 Code source du projet

.3.1 PHOTO-acq.exe

```
//==============================================================================
//
// Title:      types.h
// Purpose:    Functions déclarations.
//
// Created on: 14/09/2007 at 14:34:50 by IATRIDES Clément.
// Copyright:  cnrs. All Rights Reserved.
//
//==============================================================================

#ifndef __types_H__
#define __types_H__
#ifdef __cplusplus
    extern "C" {
#endif

#include "cvidef.h"
#include "NIDAQmx.h"

struct Donnees {

  float64 *data;
  double  time;
  int32   numRead;
  struct Donnees *next;
  int gain1,gain2;

};

struct Transfert {

  double a,b;
};

struct Config {

  int     mm_n,manual_auto1,mm_sw1;
  double  g_max1,g_min1;
  int     manual_auto2,mm_sw2;
  double  g_max2,g_min2;
  double  sky_bg_c1,sky_bg_c2,dark_c1,dark_c2;
```

```
  double  range;
  int    sampling_rate,n_sbloc,n_bloc;

  struct Transfert photo1F,photo1f,photo2F,photo2f,therm1,therm2;

  int linebufferlength;
  double buffertime;
};

struct File {

  int   fi;
  char  *fname,*fname2;

};

struct Graph {

  int     gr;
  int    sub_sky_bg,sub_dc;
  double  min,max,min2,max2;
  int    manual_auto;
  double  scale,points;
};

struct CentralIn {

  int mode,n;
  double val[3];

};

struct Temps {

  int mode,n,satellites;
  long int time1,time2;
  int j,h,m,s;
  int timezone;

};

#ifdef __cplusplus
    }
#endif
#endif /* ndef __types_H__ */
```

```
/**************************************************************************/
/* LabWindows/CVI User Interface Resource (UIR) Include File              */
/* Copyright (c) National Instruments 2007. All Rights Reserved.          */
/*                                                                        */
/* WARNING: Do not add to, delete from, or otherwise modify the contents */
/*          of this include file.                                         */
/**************************************************************************/
#include <userint.h>
#ifdef __cplusplus
    extern "C" {
#endif

     /* Panels and Controls: */
#define PANEL                           1       /* callback function: PanelCallback */
#define PANEL_TEXTMSG_9                 2
#define PANEL_TEXTMSG_7                 3
#define PANEL_TEXTMSG_8                 4
#define PANEL_TEXTMSG_13                5
#define PANEL_TEXTMSG_12                6
#define PANEL_TEXTMSG_11                7
#define PANEL_TEXTMSG_10                8
#define PANEL_TEXTMSG_5                 9
#define PANEL_TEXTMSG_3                 10
#define PANEL_DECORATION_BLUE_6         11
#define PANEL_DECORATION_BLUE_5         12
#define PANEL_FNAME_2                   13      /* callback function: FileCallback */
#define PANEL_DECORATION_BLUE_4         14
#define PANEL_FNAME                     15      /* callback function: FileCallback */
#define PANEL_DECORATION_BLUE_3         16
#define PANEL_SAVE                      17      /* callback function: FileCallback */
#define PANEL_DECORATION_BLUE_2         18
#define PANEL_DECORATION_GREEN_2        19
#define PANEL_DECORATION_GREEN_3        20
#define PANEL_DECORATION_GREEN          21
#define PANEL_DECORATION_BLUE           22
#define PANEL_C_PHOTO_2                 23
#define PANEL_C_PHOTO_SVG_2             24
#define PANEL_C_PHOTO_SVG               25
#define PANEL_C_PHOTO                   26
#define PANEL_SKYBG_PHOTO_2             27
#define PANEL_SKYBG_PHOTO               28
#define PANEL_DC_PHOTO_2                29
#define PANEL_DC_PHOTO                  30
#define PANEL_FI                        31      /* callback function: FileCallback */
#define PANEL_GR                        32      /* callback function: GraphCallback */
```

```
46 #define PANEL_LTGRAPH               33
47 #define PANEL_G_MIN_2               34      /* callback function: ConfCallback */
48 #define PANEL_G_MAX_2               35      /* callback function: ConfCallback */
49 #define PANEL_MANUAL_AUTO_2         36      /* callback function: ConfCallback */
50 #define PANEL_RTGRAPH_2             37
51 #define PANEL_RTGRAPH               38
52 #define PANEL_GRAPH_MIN_2           39      /* callback function: GraphCallback */
53 #define PANEL_GRAPH_MAX_2           40      /* callback function: GraphCallback */
54 #define PANEL_GRAPH_MIN             41      /* callback function: GraphCallback */
55 #define PANEL_GRAPH_MAX             42      /* callback function: GraphCallback */
56 #define PANEL_G_MIN                 43      /* callback function: ConfCallback */
57 #define PANEL_G_MAX                 44      /* callback function: ConfCallback */
58 #define PANEL_GRAPH_MANUAL_AUTO     45      /* callback function: GraphCallback */
59 #define PANEL_MANUAL_AUTO           46      /* callback function: ConfCallback */
60 #define PANEL_MM_N                  47      /* callback function: ConfCallback */
61 #define PANEL_DARKC_2               48      /* callback function: DarkcCallback */
62 #define PANEL_DARKC                 49      /* callback function: DarkcCallback */
63 #define PANEL_SKYBGC                50      /* callback function: SkybgCallback */
64 #define PANEL_MM_SW                 51      /* callback function: GainCallback */
65 #define PANEL_MM_SW_2               52      /* callback function: GainCallback */
66 #define PANEL_TIME_2                53
67 #define PANEL_READ                  54
68 #define PANEL_BLOCSSKP              55
69 #define PANEL_SVG_C_2               56      /* callback function: Svgc2Callback */
70 #define PANEL_SVG_C                 57      /* callback function: Svgc1Callback */
71 #define PANEL_GPS_N                 58
72 #define PANEL_GPS_S                 59
73 #define PANEL_GPS_M                 60
74 #define PANEL_GPS_J                 61
75 #define PANEL_GPS_H                 62
76 #define PANEL_CI_Z                  63
77 #define PANEL_CI_Y                  64
78 #define PANEL_CI_X                  65
79 #define PANEL_THEMP_SENS            66
80 #define PANEL_THEMP_BOX             67
81 #define PANEL_GO                    68      /* callback function: Callback */
82 #define PANEL_GRAPH_SCALE_POINTS    69
83 #define PANEL_GRAPH_SCALE           70      /* callback function: GraphCallback */
84 #define PANEL_RS_N                  71      /* callback function: CICallback */
85 #define PANEL_IRON                  72      /* callback function: IronCallback */
86 #define PANEL_CI                    73      /* callback function: CICallback */
87 #define PANEL_GPS_ON                74      /* callback function: GpsCallback */
88 #define PANEL_TIMEZONE              75      /* callback function: TZCallback */
89 #define PANEL_SPLITTER_4            76
90 #define PANEL_SPLITTER_3            77
```

```
91  #define PANEL_SPLITTER_2            78
92  #define PANEL_SPLITTER              79
93  #define PANEL_SPLITTER_6            80
94  #define PANEL_SPLITTER_5            81
95  #define PANEL_SUB                   82      /* callback function: GraphCallback */
96  #define PANEL_BUFFER                83
97  #define PANEL_LOAD_BUTTON           84      /* callback function: FileCallback */
98  #define PANEL_SAVE_BUTTON           85      /* callback function: FileCallback */
99  #define PANEL_CONF_LED              86
100
101     /* Menu Bars, Menus, and Menu Items: */
102          /* (no menu bars in the resource file) */
103
104     /* Callback Prototypes: */
105 int  CVICALLBACK Callback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
106 int  CVICALLBACK CICallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
107 int  CVICALLBACK ConfCallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
108 int  CVICALLBACK DarkcCallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
109 int  CVICALLBACK FileCallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
110 int  CVICALLBACK GainCallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
111 int  CVICALLBACK GpsCallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
112 int  CVICALLBACK GraphCallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
113 int  CVICALLBACK IronCallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
114 int  CVICALLBACK PanelCallback(int panel, int event, void *callbackData, int eventData1, int
        eventData2);
115 int  CVICALLBACK SkybgCallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
116 int  CVICALLBACK Svgc1Callback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
117 int  CVICALLBACK Svgc2Callback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
118 int  CVICALLBACK TZCallback(int panel, int control, int event, void *callbackData, int
        eventData1, int eventData2);
119
120 #ifdef __cplusplus
121     }
```

```
122 #endif
```

```c
#include <userint.h>

//==============================================================================
//
// Title:      projet.c
// Purpose:  Gestion de l'interface utilisateur
//
// Created on: 27/08/2007 at 16:08:31 by IATRIDES Clément.
// Copyright:  cnrs. All Rights Reserved.
//
//==============================================================================


#include "projet.h"
#include "functions.h"
#include "centralIn.h"
#include "Daq.h"
#include "Gps.h"

volatile int gRunning=0,gSave=0,gRead=0,gCin=0,gGps=0,defaultload=0,gIgnore = 0;
// variables globales controlant l'execution du programme d'acquisition et de sauvegarde

int panelHandle;
// index du panel ( interface )

struct Donnees *donneesStart=NULL;
// debut de la chaine de données

struct Donnees *donneesCurrent=NULL;
// donnée courrente de la chaine de données

volatile struct Config *config=NULL;
// configuration des acquisitions

volatile struct File *file=NULL;
// configuration fichier

volatile struct Graph *graph=NULL;
// configuration des graph

volatile struct CentralIn *centralin=NULL;

static int ghPool;
// index du "pool" de thread

```

```
46 int itest1,itest2,itest3,itest4;
47 // index identifiants les threads
48
49 volatile double lasttime=0,lasttime2=0;
50 volatile int blocsskp=0;
51 // base de temps en secondes et deux variables permétant de calculer des deltas de temps
52
53 struct Temps *temps=NULL;
54 // variables utilisé pour dater les acquisitions
55
56 char logg[512] = {'p','h','o','t','o','.','l','o','g','\0'};
57
58 /// HIFN Fonction initialisant les variables globales et l'interface utilisateur
59 /// HIRET Retourne un int correspondant au bon déroulement ou non du programe
60 int __stdcall WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
61                        LPSTR lpszCmdLine, int nCmdShow)
62 {
63   logappend( logg , "Démarrage de l'application");
64
65
66
67   if( InitCVIRTE(hInstance,0,0)==0 )
68     return -1; /* out of memory */
69   if( (panelHandle=LoadPanel(0,"projet.uir",PANEL))<0 )
70     return -1; /* panel non trouvé */
71
72   DisplayPanel(panelHandle);
73
74   CmtNewThreadPool (4, &ghPool);
75   //CmtSetThreadPoolAttribute (ghPool, ATTR_TP_PROCESS_EVENTS_WHILE_WAITING, TRUE);
76
77   /* affectation des variables globales */
78   if( (temps=malloc(sizeof(struct Temps)))==NULL ||
79     (centralin=malloc(sizeof(struct CentralIn)))==NULL ||
80     (donneesStart=malloc(sizeof(struct Donnees)))==NULL ||
81     (config=malloc(sizeof(struct Config)))==NULL ||
82     (file=malloc(sizeof(struct File)))==NULL ||
83     (file->fname=malloc(MAX_PATHNAME_LEN *sizeof(char)))==NULL ||
84     (file->fname2=malloc(MAX_PATHNAME_LEN *sizeof(char)))==NULL ||
85     (graph=malloc(sizeof(struct Graph)))==NULL )
86   {
87       MessagePopup("Error","Not enough memory");
88       // message d'erreur si mémoire insufisante
89   }
90   else
```

```
91   {
92
93     /* deffinition du temps de référance */
94     temps->j = -1;
95     system_time(temps);
96
97
98     donneesCurrent=donneesStart;
99     // début de la chaine de données
100
101    /* démarage de l'interface */
102    SetPanelAttribute (panelHandle, ATTR_WINDOW_ZOOM, VAL_MAXIMIZE);
103    SetCtrlAttribute (panelHandle, PANEL_GO , ATTR_ON_COLOR, VAL_GREEN);
104    SetCtrlAttribute (panelHandle, PANEL_GO , ATTR_OFF_COLOR, VAL_RED);
105    SetCtrlAttribute (panelHandle, PANEL_CI , ATTR_ON_COLOR, VAL_GREEN);
106    SetCtrlAttribute (panelHandle, PANEL_CI , ATTR_OFF_COLOR, VAL_RED);
107    SetCtrlAttribute(panelHandle,PANEL_CONF_LED, ATTR_OFF_COLOR,VAL_RED);
108    SetCtrlAttribute(panelHandle,PANEL_CONF_LED, ATTR_ON_COLOR,VAL_GREEN);
109    SetCtrlAttribute (panelHandle, PANEL_GPS_ON , ATTR_ON_COLOR, VAL_CYAN);
110
111    RunUserInterface();
112    DiscardPanel(panelHandle);
113
114    logappend( logg , "Arret de l'application");
115  }
116
117  return 0;
118 }
119
120 /// HIFN Fonction permetant de fermer l'interface utilisateur et d'interompre les
       acquisition
121 int CVICALLBACK PanelCallback(int panel, int event, void *callbackData, int eventData1, int
       eventData2)
122 {
123   if( event==EVENT_CLOSE )
124   {
125     if( gRunning) MessagePopup("Warning","Veuillez étindre l'acquisition des photometres
          avant de quitter");
126     else if( gCin) MessagePopup("Warning","Veuillez étindre l'acquisition de la centrale
          inertielle avant de quitter");
127     else QuitUserInterface(0);
128     // ferme l'interface utilisateur
129   }
130
131   if( defaultload == 0 )
```

```
132   {
133
134     FileCallback( panel, 0, EVENT_COMMIT,callbackData, eventData1,eventData2);
135     // met à jour la configuration fichier et si c'est le premier démarage des acquisitions
          charge la configuration du ficher de base
136
137     ConfCallback( panel, 0, EVENT_COMMIT,callbackData, eventData1,eventData2);
138     // met à jour la configuration des acquisitions
139
140     GraphCallback( panel, 0, EVENT_COMMIT,callbackData, eventData1,eventData2);
141     // met à jour la configuration des graphs
142
143     centralin->n = 1;
144     temps->n = 3;
145     temps->timezone = 2;
146     temps->j = -1;
147
148     SetCtrlVal(panel,PANEL_GPS_ON,1);
149     GpsCallback (panel, 0, EVENT_COMMIT , callbackData, eventData1, eventData2);
150
151   }
152   return 0;
153 }
154
155
156 /// HIFN Fonction permettant de déclancher le démarrage et l'arret des acquisitions
157 int CVICALLBACK Callback(int panel, int control, int event, void *callbackData, int
      eventData1, int eventData2)
158 {
159   int start=0;
160
161   if( event==EVENT_COMMIT )
162   {
163
164     GetCtrlVal(panel,PANEL_GO,&start);
165     // Récupere la valeur du start switch
166
167
168     if( start == 1 )
169     {
170       gRunning = 1;
171
172       CmtScheduleThreadPoolFunction (ghPool, startAcc, NULL, &itest1);
```

```
173       //CmtScheduleThreadPoolFunctionAdv (ghPool, startAcc , NULL,
              THREAD_PRIORITY_TIME_CRITICAL, NULL,EVENT_TP_THREAD_FUNCTION_END, NULL,
              CmtGetCurrentThreadID(), &itest1);
174       // démarre les acquisitions
175
176       CmtScheduleThreadPoolFunction (ghPool, traitement, NULL, &itest2);
177       //CmtScheduleThreadPoolFunctionAdv (ghPool, traitement , NULL,
              THREAD_PRIORITY_TIME_CRITICAL, NULL,EVENT_TP_THREAD_FUNCTION_END, NULL,
              CmtGetCurrentThreadID(), &itest2);
178       // lance le traitement des acquisition
179
180       logappend( logg , "Début des acquisitions photometre");
181
182     }
183     else
184     {
185       gRunning = 0;
186
187       CmtWaitForThreadPoolFunctionCompletion (ghPool, itest1,
              OPT_TP_PROCESS_EVENTS_WHILE_WAITING);
188       CmtWaitForThreadPoolFunctionCompletion (ghPool, itest2,
              OPT_TP_PROCESS_EVENTS_WHILE_WAITING);
189       // patiente jusqu'a l'arret des threads d'acquisition et de traitement
190
191       logappend( logg , "Fin des acquisitions photometre");
192     }
193
194   }
195   return 0;
196 }
197
198 /// HIFN Fonction permettant de déclancher le démarrage et l'arret de la sauvegarde des
      acquisitions
199 int CVICALLBACK SaveCallback(int panel, int control, int event, void *callbackData, int
      eventData1, int eventData2)
200 {
201   if( event==EVENT_COMMIT )
202   {
203     GetCtrlVal(panel,PANEL_SAVE,&gSave);
204     // recupere la valeur du switch de sauvguarde
205     if(gSave)
206     {
207       SetCtrlAttribute(panelHandle,PANEL_SAVE,ATTR_BINARY_SWITCH_COLOR,VAL_PANEL_GRAY);
208       logappend( logg , "Début de la sauvgarde");
209     }
```

```
210     else
211     {
212       logappend( logg , "Fin de la sauvgarde");
213     }
214
215   }
216   return 0;
217 }
218
219
220 /// HIFN Fonction permettant de metre à jour des parametres de configuration du programme
221 int CVICALLBACK ConfCallback (int panel, int control, int event, void *callbackData, int
      eventData1, int eventData2)
222 {
223
224   char buffer[2048]={'\0'};
225
226   if( event==EVENT_COMMIT )
227   {
228
229       GetCtrlVal(panel,PANEL_MANUAL_AUTO,&(config->manual_auto1));
230       GetCtrlVal(panel,PANEL_MM_N,&(config->mm_n));
231       GetCtrlVal(panel,PANEL_G_MAX,&(config->g_max1));
232       GetCtrlVal(panel,PANEL_G_MIN,&(config->g_min1));
233       GetCtrlVal(panel,PANEL_MM_SW,&(config->mm_sw1));
234
235       GetCtrlVal(panel,PANEL_MANUAL_AUTO_2,&(config->manual_auto2));
236       GetCtrlVal(panel,PANEL_G_MAX_2,&(config->g_max2));
237       GetCtrlVal(panel,PANEL_G_MIN_2,&(config->g_min2));
238       GetCtrlVal(panel,PANEL_MM_SW_2,&(config->mm_sw2));
239
240       trigerrange(panel);
241
242       sprintf(buffer,"Changement de configuration Acquisition \n Auto1: %d N1: %d Max1: %f
            Min1: %f Swich1: %d Auto2: %d Max2: %f Min2: %f Swich2: %d", config->manual_auto1,
            config->mm_n,config->g_max1,config->g_min1,config->mm_sw1,config->manual_auto2,
            config->g_max2,config->g_min2,config->mm_sw2);
243
244       logappend( logg , buffer);
245
246   }
247   return 0;
248 }
249
250
```

```
251
252 /// HIFN Fonction permettant de metre à jour des parametres graphiques du programme
253 int CVICALLBACK GraphCallback (int panel, int control, int event,
254     void *callbackData, int eventData1, int eventData2)
255 {
256   double min,max,min2,max2;
257   int manual_auto,sm,sub;
258   double points=0,pourcent = 0;
259   char buffer[2048]={'\0'};
260   double scale=0;
261
262   if( event==EVENT_COMMIT )
263   {
264     GetCtrlVal(panel,PANEL_GR,&(graph->gr));
265     GetCtrlVal(panel,PANEL_GRAPH_SCALE,&scale);
266     GetCtrlVal(panel,PANEL_SUB,&sub);
267     GetCtrlVal(panel,PANEL_GRAPH_MAX,&max);
268     GetCtrlVal(panel,PANEL_GRAPH_MIN,&min);
269     GetCtrlVal(panel,PANEL_GRAPH_MAX_2,&max2);
270     GetCtrlVal(panel,PANEL_GRAPH_MIN_2,&min2);
271     GetCtrlVal(panel,PANEL_GRAPH_MANUAL_AUTO,&manual_auto);
272
273     switch(sub)
274     {
275       default :
276
277         graph->sub_dc = 0;
278         graph->sub_sky_bg = 0;
279
280         trigerrange(panel);
281
282         break;
283
284       case 1 :
285
286         graph->sub_dc = 1;
287         graph->sub_sky_bg = 0;
288
289         trigerrange(panel);
290
291         break;
292
293       case 2 :
294
295         graph->sub_dc = 0;
```

```
296         graph->sub_sky_bg = 1;
297
298         trigerrange(panel);
299
300         break;
301     }
302
303     if(scale != graph->scale)
304     {
305
306       graph->scale = scale;
307
308       /* calcule et affichage du nombre de points affiché sur chaque graph */
309       points = (graph->scale / ( config->n_sbloc * config->n_bloc * ( 1.0 / config->
              sampling_rate) ));
310
311       graph->points = points / 100 ;
312
313       pourcent = 10000 / points;
314
315       if( pourcent > 100) pourcent = 100;
316
317       SetCtrlVal(panel,PANEL_GRAPH_SCALE_POINTS, pourcent );
318
319       DeleteGraphPlot (panel, PANEL_RTGRAPH , -1, VAL_IMMEDIATE_DRAW);
320       DeleteGraphPlot (panel, PANEL_RTGRAPH_2 , -1, VAL_IMMEDIATE_DRAW);
321
322       SetAxisScalingMode (panel, PANEL_RTGRAPH, VAL_BOTTOM_XAXIS, VAL_MANUAL,-2 , -1);
323       SetAxisScalingMode (panel, PANEL_RTGRAPH_2, VAL_BOTTOM_XAXIS, VAL_MANUAL,-2 , -1);
324
325     }
326
327     if(graph->manual_auto && !manual_auto)
328     {
329       GetAxisScalingMode (panel, PANEL_RTGRAPH, VAL_LEFT_YAXIS, &sm, &min, &max);
330       SetCtrlVal(panel,PANEL_GRAPH_MAX,max);
331       SetCtrlVal(panel,PANEL_GRAPH_MIN,min);
332       GetAxisScalingMode (panel, PANEL_RTGRAPH_2, VAL_LEFT_YAXIS, &sm, &min2, &max2);
333       SetCtrlVal(panel,PANEL_GRAPH_MAX_2,max2);
334       SetCtrlVal(panel,PANEL_GRAPH_MIN_2,min2);
335     }
336
337     graph->manual_auto = manual_auto;
338
339     if(graph->manual_auto)
```

```
340     {
341       SetAxisScalingMode (panel, PANEL_RTGRAPH, VAL_LEFT_YAXIS, VAL_AUTOSCALE, 0, 0);
342       SetAxisScalingMode (panel, PANEL_RTGRAPH_2, VAL_LEFT_YAXIS, VAL_AUTOSCALE, 0, 0);
343     }
344     else
345     {
346       graph->max = max;
347       graph->min = min;
348       graph->max2 = max2;
349       graph->min2 = min2;
350 
351       if( max > min)
352       {
353         SetAxisScalingMode (panel, PANEL_RTGRAPH, VAL_LEFT_YAXIS, VAL_MANUAL, min, max);
354       }
355       if( max2 > min2)
356       {
357         SetAxisScalingMode (panel, PANEL_RTGRAPH_2, VAL_LEFT_YAXIS, VAL_MANUAL, min2, max2);
358       }
359     }
360 
361     sprintf(buffer,"Changement de configuration Graph \n Gr: %d Scale: %f Auto: %d Max1: %f
          Min1: %f Max2: %f Min2: %f",graph->gr,graph->scale, graph->manual_auto,graph->max ,
          graph->min ,graph->max2,graph->min2);
362     logappend( logg , buffer);
363 
364 
365   }
366   return 0;
367 }
368 
369 /// HIFN Fonction permettant de metre à jour des parametres fichiers du programme
370 int CVICALLBACK FileCallback (int panel, int control, int event,
371     void *callbackData, int eventData1, int eventData2)
372 {
373   char buffer[2048]={'\0'};
374 
375 
376   if( event==EVENT_COMMIT )
377   {
378 
379     GetCtrlVal(panel,PANEL_FI,&(file->fi));
380     GetCtrlVal(panel,PANEL_SAVE,&gSave);
381 
382 
```

```
383     if((defaultload == 0 || control == PANEL_LOAD_BUTTON) && gRunning == 0)
384     {
385
386       if(defaultload == 0) GetCtrlVal(panel,PANEL_FNAME_2,file->fname2);
387       else FileSelectPopup ("Data", "photo.conf", "*.conf", "", VAL_LOAD_BUTTON, 0, 0, 0, 1,
              file->fname2);
388       SetCtrlVal(panel,PANEL_FNAME_2,file->fname2);
389
390       if( readConf() == -1 )
391       {
392         SetCtrlVal(panel,PANEL_CONF_LED,0);
393       }
394       else
395       {
396         SetCtrlVal(panel,PANEL_CONF_LED,1);
397
398         defaultload = 1;
399
400         SetCtrlVal(panel,PANEL_GR,(graph->gr));
401         SetCtrlVal(panel,PANEL_FI,(file->fi));
402         SetCtrlVal(panel,PANEL_BUFFER,config->buffertime);
403
404         SetCtrlVal(panel,PANEL_MANUAL_AUTO,(config->manual_auto1));
405         SetCtrlVal(panel,PANEL_MM_N,(config->mm_n));
406         SetCtrlVal(panel,PANEL_G_MAX,(config->g_max1));
407         SetCtrlVal(panel,PANEL_G_MIN,(config->g_min1));
408         SetCtrlVal(panel,PANEL_MM_SW,(config->mm_sw1));
409
410         SetCtrlVal(panel,PANEL_MANUAL_AUTO_2,(config->manual_auto2));
411         SetCtrlVal(panel,PANEL_G_MAX_2,(config->g_max2));
412         SetCtrlVal(panel,PANEL_G_MIN_2,(config->g_min2));
413         SetCtrlVal(panel,PANEL_MM_SW_2,(config->mm_sw2));
414
415         SetCtrlVal(panel,PANEL_DC_PHOTO,config->dark_c1);
416         SetCtrlVal(panel,PANEL_DC_PHOTO_2,config->dark_c2);
417
418         SetCtrlVal(panel,PANEL_SKYBG_PHOTO,config->sky_bg_c1);
419         SetCtrlVal(panel,PANEL_SKYBG_PHOTO_2,config->sky_bg_c2);
420
421       }
422
423     }
424     else if( control == PANEL_SAVE_BUTTON && gSave == 0 )
425     {
```

```
426       if(VAL_NO_FILE_SELECTED == FileSelectPopup ("Data", "photo.save", "*.save", "",
              VAL_OK_BUTTON, 0, 0, 1, 1, file->fname))
427           file->fname[0] = '\0';
428
429       SetCtrlVal(panel,PANEL_FNAME,file->fname);
430     }
431     else if( strlen(file->fname) == 0)
432     {
433         gSave = 0;
434         SetCtrlVal(panel,PANEL_SAVE,gSave);
435         MessagePopup("Error","Nom du fichier de sauvgarde invalide");
436     }
437
438     sprintf(buffer,"Changement de configuration Fichier \n ficher de configuration : %s
            fichier de sauvgarde : %s", file->fname2,file->fname);
439     logappend( logg , buffer);
440   }
441
442
443
444   return 0;
445 }
446
447
448
449 /// HIFN Fonction permettant de selectionner le guain désiré
450 int CVICALLBACK GainCallback (int panel, int control, int event, void *callbackData, int
      eventData1, int eventData2)
451 {
452   int gain1=0,gain2=0;
453
454   if( event==EVENT_COMMIT )
455   {
456
457     GetCtrlVal(panel,PANEL_MM_SW,&gain1);
458
459     GetCtrlVal(panel,PANEL_MM_SW_2,&gain2);
460
461     if(config->manual_auto1 == 0 && gain1 != config->mm_sw1)
462     {
463       config->mm_sw1=gain1;
464       writeGainDigit(0, 0 ,(uInt8)config->mm_sw1);
465     }
466     else SetCtrlVal(panel,PANEL_MM_SW,config->mm_sw1);
467
```

```
468     if(config->manual_auto2 == 0 && gain2 != config->mm_sw2)
469     {
470       config->mm_sw2=gain2;
471       writeGainDigit(0, 1 ,(uInt8)config->mm_sw2);
472     }
473     else SetCtrlVal(panel,PANEL_MM_SW_2,config->mm_sw2);
474
475   }
476   return 0;
477 }
478
479 /// HIFN Fonction permettant de conserver et afficher les derniere valeur acquise
480 int CVICALLBACK SkybgCallback (int panel, int control, int event,
481     void *callbackData, int eventData1, int eventData2)
482 {
483   //float64 *tmp;
484   double photo1,photo2;
485
486   if( event==EVENT_COMMIT )
487   {
488
489
490     GetCtrlVal(panel,PANEL_C_PHOTO,&photo1);
491     GetCtrlVal(panel,PANEL_C_PHOTO_2,&photo2);
492
493     config->sky_bg_c1 = photo1;
494     config->sky_bg_c2 = photo2;
495
496     SetCtrlVal(panel,PANEL_SKYBG_PHOTO,config->sky_bg_c1);
497     SetCtrlVal(panel,PANEL_SKYBG_PHOTO_2,config->sky_bg_c2);
498
499     // on utilise ce code si on veut prendre des valeurs hors acquisitions
500   /*  if(gRunning == 0)
501     {
502       writeGainDigit(0,1);
503       writeGainDigit(1,1);
504
505       tmp = moyNread2(1000,100,-5,5);
506
507       if(tmp != NULL)
508       {
509         config->sky_bg_c1=tmp[0];
510         config->sky_bg_c2=tmp[1];
511
512         SetCtrlVal(panel,PANEL_SKYBG_PHOTO,config->sky_bg_c1);
```

```
513         SetCtrlVal(panel,PANEL_SKYBG_PHOTO_2,config->sky_bg_c2);
514       }
515     }
516     else MessagePopup("Error","Veuillez éteindre l'acquisition continue avant de proceder à
          des mesures de Sky Background current");
517     */
518   }
519
520   SetCtrlVal(panel,PANEL_SKYBGC,0);
521
522   return 0;
523 }
524
525 /// HIFN Fonction permettant de conserver et afficher les derniere valeur acquise
526 int CVICALLBACK DarkcCallback (int panel, int control, int event,
527     void *callbackData, int eventData1, int eventData2)
528 {
529   double photo=-1;
530   int voie=-1,button1=0,button2=0;
531
532   if( event==EVENT_COMMIT )
533   {
534     if(gRunning == 0)
535     {
536
537       GetCtrlVal(panel,PANEL_DARKC,&button1);
538       GetCtrlVal(panel,PANEL_DARKC_2,&button2);
539
540       if(button1 == 1) voie = 0;
541       else if(button2 == 1) voie = 1;
542
543       writeGainDigit(0,voie,1);
544
545       photo = moyNread2(voie,1000,100,-5,5);
546
547       if(photo != -1)
548       {
549         if(voie == 0) config->dark_c1=photo;
550         else if(voie == 1) config->dark_c2=photo;
551
552         SetCtrlVal(panel,PANEL_DC_PHOTO,config->dark_c1);
553         SetCtrlVal(panel,PANEL_DC_PHOTO_2,config->dark_c2);
554       }
555     }
```

```
556     else MessagePopup("Error","Veuillez éteindre l'acquisition continue avant de proceder à
            des mesures de Dark current");
557
558     SetCtrlVal(panel,PANEL_DARKC,0);
559     SetCtrlVal(panel,PANEL_DARKC_2,0);
560   }
561
562
563
564
565   return 0;
566 }
567
568 /// HIFN Fonction permettant de conserver et afficher les derniere valeur acquise
569 int CVICALLBACK Svgc1Callback (int panel, int control, int event,
570     void *callbackData, int eventData1, int eventData2)
571 {
572   double photo1;
573
574   if( event==EVENT_COMMIT )
575   {
576     GetCtrlVal(panel,PANEL_C_PHOTO,&photo1);
577     SetCtrlVal(panel,PANEL_C_PHOTO_SVG,photo1);
578     SetCtrlVal(panel,PANEL_SVG_C,0);
579
580   }
581   return 0;
582 }
583
584 /// HIFN Fonction permettant de conserver et afficher les derniere valeur acquise
585 int CVICALLBACK Svgc2Callback (int panel, int control, int event,
586     void *callbackData, int eventData1, int eventData2)
587 {
588   double photo2;
589
590   if( event==EVENT_COMMIT )
591   {
592     GetCtrlVal(panel,PANEL_C_PHOTO_2,&photo2);
593     SetCtrlVal(panel,PANEL_C_PHOTO_SVG_2,photo2);
594     SetCtrlVal(panel,PANEL_SVG_C_2,0);
595   }
596   return 0;
597 }
598
599 int CVICALLBACK CICallback (int panel, int control, int event,
```

```
600     void *callbackData, int eventData1, int eventData2)
601 {
602   int state=0;
603   int n;
604 
605   if( event==EVENT_COMMIT )
606   {
607     GetCtrlVal(panel,PANEL_CI,&state);
608     GetCtrlVal(panel,PANEL_RS_N,&n);
609 
610     if( n != centralin->n )
611     {
612 
613       if(gCin == 1 )
614       {
615         gCin = 0;
616         CmtWaitForThreadPoolFunctionCompletion (ghPool, itest3,
              OPT_TP_PROCESS_EVENTS_WHILE_WAITING);
617         stop_rs();
618       }
619 
620       centralin->n = n;
621 
622 
623     }
624     else if(state)
625     {
626 
627       if( start_rs() == 0)
628       {
629         gCin = 1;
630         CmtScheduleThreadPoolFunction (ghPool, centrale, NULL, &itest3);
631       }
632 
633     }
634     else
635     {
636       gCin = 0;
637       CmtWaitForThreadPoolFunctionCompletion (ghPool, itest3,
            OPT_TP_PROCESS_EVENTS_WHILE_WAITING);
638       stop_rs();
639     }
640 
641     SetCtrlVal(panel,PANEL_CI,gCin);
642   }
```

```
643
644   return 0;
645 }
646
647 int CVICALLBACK GpsCallback (int panel, int control, int event,
648     void *callbackData, int eventData1, int eventData2)
649 {
650   struct Temps *tempsnew=NULL;
651
652   if( event==EVENT_COMMIT )
653   {
654
655     ProcessDrawEvents();
656
657     if(gRunning == 0)
658     {
659
660       tempsnew = malloc(sizeof(struct Temps));
661       tempsnew->timezone = temps->timezone;
662       tempsnew->n = temps->n;
663
664       if( gps_start() == 0 )
665       {
666         calibrate_time(tempsnew);
667         gps_stop();
668       }
669
670       if( tempsnew->satellites < 0 ) tempsnew->satellites = 0;
671
672       if(tempsnew->satellites < 2)
673       {
674         system_time(tempsnew);
675         tempsnew->mode = 0;
676         SetCtrlAttribute (panelHandle, PANEL_GPS_ON , ATTR_OFF_COLOR, VAL_YELLOW);
677
678       }
679       else
680       {
681         tempsnew->mode = 1;
682         SetCtrlAttribute (panelHandle, PANEL_GPS_ON , ATTR_OFF_COLOR, VAL_GREEN);
683       }
684
685       changement_jour(tempsnew);
686
687       free(temps);
```

```
688
689       temps = tempsnew;
690
691       SetCtrlVal(panel,PANEL_GPS_J,temps->j);
692       SetCtrlVal(panel,PANEL_GPS_H,temps->h);
693       SetCtrlVal(panel,PANEL_GPS_M,temps->m);
694       SetCtrlVal(panel,PANEL_GPS_S,temps->s);
695       SetCtrlVal(panel,PANEL_GPS_N,temps->satellites);
696
697
698     }
699     else MessagePopup("Warning","Veuillez étindre l'acquisition des photometre pour mettre à
              jour la référance de temps");
700
701
702     SetCtrlVal(panel,PANEL_GPS_ON,0);
703   }
704
705
706   return 0;
707 }
708
709 int CVICALLBACK TZCallback (int panel, int control, int event,
710     void *callbackData, int eventData1, int eventData2)
711 {
712
713   if( event==EVENT_COMMIT )
714   {
715     GetCtrlVal(panel,PANEL_TIMEZONE,&(temps->timezone));
716   }
717   return 0;
718 }
719
720
721 int CVICALLBACK IronCallback (int panel, int control, int event,
722     void *callbackData, int eventData1, int eventData2)
723 {
724
725   if( event==EVENT_COMMIT )
726   {
727
728     if(gRunning == 0 && gCin == 0)
729     {
730       ironCI();
731     }
```

```
732     else MessagePopup("Central Inertial Managment","Veillez arreter l'acquisition photometre
            et inertiel pour la calibration.");
733
734     SetCtrlVal(panel,PANEL_IRON,0);
735   }
736   return 0;
737 }
738
739
740 int CVICALLBACK BufferCallback (int panel, int control, int event,
741     void *callbackData, int eventData1, int eventData2)
742 {
743   if( event==EVENT_COMMIT )
744   {
745
746   }
747   return 0;
748 }
```

```c
//==============================================================================
//
// Title:      functions.h
// Purpose:    Functions déclarations
//
// Created on: 28/08/2007 at 10:44:43 by IATRIDES Clément.
// Copyright:  cnrs. All Rights Reserved.
//
//==============================================================================

#ifndef __functions_H__
#define __functions_H__
#ifdef __cplusplus
    extern "C" {
#endif

#include <windows.h>
#include <ansi_c.h>
#include <formatio.h>
#include <userint.h>
#include <math.h>
#include <utility.h>
#include "toolbox.h"
#include "types.h"

int CVICALLBACK traitement (void *functionData);
double val(int channel ,float64 *datablock, int32 numReadblock,int gain1,int gain2);
float64 add( int chanel, int numChannels,struct Donnees *donnees);
int affRT(int panel,double *valeurs,double time);
int affLT(int panel,double *valeurs,double time);
int save(char *linebuffer,float64 *datablock, int32 numReadblock , double time,int gain1,int
     gain2);
int writetofile(char *linebuffer);
int readConf(void);
int logappend( char *fichier, char *line);
int autoGain(int panel, double *valeurs);
int timestring(char *line, double time);
int trigerrange(int panel) ;

#ifdef __cplusplus
    }
#endif
#endif /* ndef __functions_H__ */
```

```
1  //==============================================================================
2  //
3  // Title:      functions.c
4  // Purpose:    Fonctions de traitements, de calcules numériques et de lecture / écruture
       fichiers
5  //
6  // Created on: 14/09/2007 at 13:48:50 by IATRIDES Clément.
7  // Copyright:  cnrs. All Rights Reserved.
8  //
9  //==============================================================================
10
11 #include "projet.h"
12 #include "functions.h"
13 #include "Daq.h"
14
15 /* definitions des variables externes */
16 extern volatile struct Config *config;
17 extern volatile struct Graph *graph;
18 extern volatile struct File *file;
19 extern volatile struct CentralIn *centralin;
20 extern volatile double lasttime,lasttime2;
21 extern volatile int gRunning,gRead,gSave,gIgnore;
22 extern struct Donnees *donneesStart;
23 extern struct Donnees *donneesCurrent;
24 extern int panelHandle;
25 extern struct Temps *temps;
26
27
28 /// HIFN Fonction permettant de traiter les acquisition
29 int CVICALLBACK traitement (void *functionData)
30 {
31   struct Donnees *donneesNext=NULL;
32   // pointeur pour la structure suivante à traiter
33
34   float64 datablock[4]={0,0,0,0},valeursmoy[4]={0,0,0,0};
35   int32 numReadblock=0;
36   int gain1=-1,gain2=-1;
37   // variables de constructions des blocs
38
39   double valeurs[4];
40   // valeurs du blocs de mesures
41
42   int lastline=0 , lastLTgraph=0 , lastbloc = 0 , lastdiff = 0, lastRTgraph=0, lastgain=0,
       lastlinebuffer=0;
43   // variables des compteurs pour la construction, l'affichage et la sauvguarde des blocs
```

```
44
45  char *linebuffer=NULL;
46
47  lasttime = temps->h*3600+temps->m*60+temps->s;
48
49
50
51  while(gRunning)
52  {
53
54    GetCtrlVal(panelHandle,PANEL_BUFFER,&(config->buffertime));
55    Delay(config->buffertime);
56    // on charge la mémoire tempon avec buffertime secondes d'acquisitions
57
58    while(gRead > 0)
59    {
60
61      gRead--;
62      SetCtrlVal(panelHandle,PANEL_READ, gRead);
63      // On décrémente de compteur de sous-blocs
64
65      /* Construction du block */
66      if( donneesStart->numRead >0 )
67      // si il y a des données dans ce sous-bloc
68      {
69        if( gain1 == -1) gain1 = donneesStart->gain1;
70        if( gain2 == -1) gain2 = donneesStart->gain2;
71
72        if(gain1 == donneesStart->gain1 && gain2 == donneesStart->gain2)
73        {
74          /* Calcule des sommes pour reconstituer le bloc */
75          datablock[0] += add(0,4,donneesStart);
76          datablock[1] += add(1,4,donneesStart);
77          datablock[2] += add(2,4,donneesStart);
78          datablock[3] += add(3,4,donneesStart);
79          numReadblock += donneesStart->numRead;
80        }
81      }
82
83      lastbloc++;
84
85      if( lastbloc >= config->n_bloc)
86      // si le bloc est complet
87      {
88
```

```
89        /* calcule des valeurs numériques */
90        valeurs[0]=val(0,datablock, numReadblock,gain1,gain2);
91        valeurs[1]=val(1,datablock, numReadblock,gain1,gain2);
92        valeurs[2]=val(2,datablock, numReadblock,gain1,gain2);
93        valeurs[3]=val(3,datablock, numReadblock,gain1,gain2);
94
95        /* calcule des valeurs moyennées */
96
97        valeursmoy[0] = valeurs[0];
98          valeursmoy[1] = valeurs[1];
99
100       /* Affichage des données temps réel */
101       if( lastRTgraph >= graph->points )
102       {
103
104         affRT(panelHandle,valeurs,donneesStart->time);
105
106         lastRTgraph = 0;
107       }
108
109       lastRTgraph++;
110
111
112       /* Affichage des données long terme */
113       if( lastLTgraph >= graph->gr )
114       // si ce bloc doit etre affiché
115       {
116         affLT(panelHandle,valeurs,donneesStart->time);
117
118         lastLTgraph = 0;
119       }
120
121       lastLTgraph++;
122
123       /* Sauvgarde des block */
124       if(gSave)
125       // si la sauvguarde est activée
126       {
127         if(lastlinebuffer == 0 )
128         {
129           linebuffer = malloc(config->linebufferlength*1024*sizeof(char));
130           linebuffer[0] = '\0';
131         }
132
133         if(lastline >= file->fi)
```

```
134       // si ce bloc doit etre sauvguardé
135       {
136         save(linebuffer,datablock, numReadblock , donneesStart->time,gain1,gain2);
137         lastlinebuffer++;
138
139         if(lastlinebuffer >= config->linebufferlength / file->fi)
140         {
141           writetofile(linebuffer);
142           free(linebuffer);
143           lastlinebuffer = 0;
144         }
145         lastline = 0;
146       }
147
148       lastline++;
149     }
150     else if(lastlinebuffer > 0)
151     {
152       writetofile(linebuffer);
153       lastlinebuffer = 0;
154       lastline = 0;
155     }
156
157     if(lastdiff >= 100 )
158     {
159       /* calcule et affichage du temps entre deux blocs utilise 1 bloc tout les 100 blocs
              */
160       SetCtrlVal(panelHandle,PANEL_TIME_2, (donneesStart->time - lasttime)/100);
161       lasttime = donneesStart->time;
162
163       lastdiff = 0;
164     }
165
166     lastdiff++;
167
168     if(lastgain >= config->mm_n )
169     {
170       valeursmoy[0] = valeursmoy[0] / lastgain;
171       valeursmoy[1] = valeursmoy[1] / lastgain;
172
173
174       /* Control automatique du gain */
175       autoGain(panelHandle,valeursmoy);
176
177       valeursmoy[0] = 0;
```

```
178           valeursmoy[1] = 0;
179
180           lastgain = 0;
181
182         }
183
184         lastgain++;
185
186         /* Réinitialisation du bloc */
187         datablock[0] = 0;
188         datablock[1] = 0;
189         datablock[2] = 0;
190         datablock[3] = 0;
191         numReadblock = 0 ;
192         gain1 = -1;
193         gain2 = -1;
194
195         lastbloc = 0;
196
197       }
198
199       donneesNext=donneesStart->next;
200       // on met à jour le pointeur vers la structure suivante
201
202       /* liberation de la mémoire */
203       free(donneesStart->data);
204       free(donneesStart);
205
206       donneesStart = donneesNext;
207       // on met à jour le pointeur de début de chaine
208     }
209
210   }
211
212
213
214   return 0;
215 }
216
217 /// HIFN Fonction permettant d'afficher les données d'un bloc sur le graph et les afficheurs
      real time
218 int affRT(int panel,double *valeurs,double time)
219 {
220   int sm;
221   double min,max;
```

```
222   char timename1[256]={'\0'};
223
224   /* graphs photo 1 & 2 */
225   PlotPoint (panel, PANEL_RTGRAPH , time, valeurs[0], VAL_DOTTED_SOLID_DIAMOND, VAL_RED);
226   PlotPoint (panel, PANEL_RTGRAPH_2 , time, valeurs[1], VAL_DOTTED_SOLID_DIAMOND, VAL_BLUE);
227
228   /* mise à jour de l'échelle temporelle du graph */
229   GetAxisScalingMode (panel, PANEL_RTGRAPH, VAL_BOTTOM_XAXIS, &sm, &min, &max);
230   if(time > max)
231   {
232     SetAxisScalingMode (panel, PANEL_RTGRAPH, VAL_BOTTOM_XAXIS, sm, time , time + graph->
          scale);
233     SetAxisScalingMode (panel, PANEL_RTGRAPH_2, VAL_BOTTOM_XAXIS, sm, time , time + graph->
          scale);
234     DeleteGraphPlot (panel, PANEL_RTGRAPH , -1, VAL_IMMEDIATE_DRAW);
235     DeleteGraphPlot (panel, PANEL_RTGRAPH_2 , -1, VAL_IMMEDIATE_DRAW);
236     ClearAxisItems (panel, PANEL_RTGRAPH , VAL_BOTTOM_XAXIS);
237     ClearAxisItems (panel, PANEL_RTGRAPH_2 , VAL_BOTTOM_XAXIS);
238
239     timestring(timename1,time);
240
241     InsertAxisItem (panel, PANEL_RTGRAPH , VAL_BOTTOM_XAXIS, -1, timename1, time);
242     InsertAxisItem (panel, PANEL_RTGRAPH_2 , VAL_BOTTOM_XAXIS, -1, timename1, time );
243   }
244
245   /* photo 1 & 2 digital displays */
246   SetCtrlVal(panel,PANEL_C_PHOTO,valeurs[0]);
247   SetCtrlVal(panel,PANEL_C_PHOTO_2,valeurs[1]);
248
249   /* therm 1 & 2 digital displays */
250   SetCtrlVal(panel,PANEL_THEMP_SENS,valeurs[2]);
251   SetCtrlVal(panel,PANEL_THEMP_BOX,valeurs[3]);
252
253   return 0;
254 }
255
256 /// HIFN Fonction permettant d'afficher les données d'un bloc sur le graph long time
257 int affLT(int panel,double *valeurs,double time)
258 {
259   int sm;
260   double min,max;
261   char timename1[256]={'\0'};
262
263   /* graph photo 1 & 2 */
264   PlotPoint (panel, PANEL_LTGRAPH , time, valeurs[0], VAL_SIMPLE_DOT, VAL_RED);
```

```
265   PlotPoint (panel, PANEL_LTGRAPH , time, valeurs[1], VAL_SIMPLE_DOT, VAL_BLUE);
266
267   /* mise à jour de l'échelle temporelle du graph */
268   GetAxisScalingMode (panel, PANEL_LTGRAPH, VAL_BOTTOM_XAXIS, &sm, &min, &max);
269   if(time > max)
270   {
271     SetAxisScalingMode (panel, PANEL_LTGRAPH, VAL_BOTTOM_XAXIS, sm, time , time + 3600*10);
272     // on décale le graph d'une heure pour afficher la suite des valeurs
273
274     ClearAxisItems (panel, PANEL_LTGRAPH , VAL_BOTTOM_XAXIS);
275
276     timestring(timename1,time);
277
278     InsertAxisItem (panel, PANEL_LTGRAPH , VAL_BOTTOM_XAXIS, -1, timename1, time);
279   }
280
281   return 0;
282 }
283
284 /// HIFN Fonction permettant de convertir l'heure en string
285 int timestring(char *line, double time)
286 {
287   double htemp,mtemp,jtemp,stemp;
288
289   jtemp = (int)time / (3600*24);
290
291   htemp = (int)(time - jtemp*3600*24) / 3600;
292
293   mtemp = (int)(time - jtemp*3600*24 - htemp*3600) / 60;
294
295   stemp = time - jtemp*3600*24 - htemp*3600 - mtemp*60;
296
297   sprintf(line,"%2.0fj%2.0fh%2.0fm%2.3fs",jtemp,htemp,mtemp,stemp);
298
299   return 0;
300 }
301
302 /// HIFN Fonction permettant de déterminer le gain aproprié pour les lignes analogiques
303 int autoGain( int panel, double *valeurs )
304 {
305   int  gain1 =0 , gain2 = 0;
306
307   if(  valeurs[0] > config->g_max1) gain1 = 0;
308   else if( valeurs[0] < config->g_min1) gain1 = 1;
309   else gain1 = config->mm_sw1;
```

```
310
311   if(  valeurs[1] > config->g_max2) gain2 = 0;
312   else if( valeurs[1] < config->g_min2) gain2 = 1;
313   else gain2 = config->mm_sw2;
314
315
316   if(gain1 != config->mm_sw1 && config->manual_auto1)
317   {
318      config->mm_sw1=gain1;
319
320      gIgnore = config->n_bloc/2;
321      // lors d'un changement de gain on ignore les n_bloc / 2 sous-blocs suivants pour
          laisser le gain se stabiliser
322
323      writeGainDigit(0, 0 ,(uInt8)config->mm_sw1);
324
325      SetCtrlVal(panel,PANEL_MM_SW,config->mm_sw1);
326   }
327
328   if( gain2 != config->mm_sw2 && config->manual_auto2)
329   {
330       config->mm_sw2=gain2;
331
332     gIgnore = config->n_bloc/2;
333     // lors d'un changement de gain on ignore les n_bloc / 2 sous-blocs suivants pour
          laisser le gain se stabiliser
334
335     writeGainDigit( 0,1 ,(uInt8) config->mm_sw2);
336
337     SetCtrlVal(panel,PANEL_MM_SW_2,config->mm_sw2);
338   }
339
340   return 0;
341 }
342
343 /// HIFN Fonction permettant de calculer les valeurs numériques correspondantes aux
      acquisitions
344 double val(int channel ,float64 *datablock, int32 numReadblock, int gain1, int gain2)
345 {
346    double valeur=0;
347
348    switch(channel)
349    {
350      default : break;
351
```

```
352     case 0:
353
354       valeur = (datablock[0]/numReadblock);
355       if(gain1) valeur = valeur * config->photo1f.a + config->photo1f.b ;
356       else valeur = valeur * config->photo1F.a + config->photo1F.b ;
357       if(graph->sub_dc) valeur = valeur - config->dark_c1;
358       if(graph->sub_sky_bg) valeur = valeur - config->sky_bg_c1;
359
360       if(valeur < 0) valeur = 0;
361       // on masque les valeurs négatives var elles n'ont aucun sens
362
363       break;
364
365     case 1:
366
367       valeur = (datablock[1]/numReadblock);
368       if(gain2) valeur = valeur * config->photo2f.a + config->photo2f.b ;
369       else valeur = valeur * config->photo2F.a + config->photo2F.b ;
370       if(graph->sub_dc) valeur = valeur - config->dark_c2;
371       if(graph->sub_sky_bg) valeur = valeur - config->sky_bg_c2;
372
373       if(valeur < 0) valeur = 0;
374       // on masque les valeurs négatives var elles n'ont aucun sens
375
376       break;
377
378     case 2:
379
380       valeur = (datablock[2]/numReadblock) * config->therm1.a + config->therm1.b ;
381       break;
382
383     case 3:
384
385       valeur = (datablock[3]/numReadblock) * config->therm2.a + config->therm2.b ;
386       break;
387   }
388
389   return valeur;
390 }
391
392 /// HIFN Fonction permettant de calculer la somme des donnes du canal channel
393 float64 add( int chanel, int numChannels, struct Donnees *donnees)
394 {
395   float64 valeur=0;
396   int i;
```

```
397
398     for(i = 0; i < donnees->numRead; i++)
399     {
400       valeur += donnees->data[chanel+i*numChannels];
401     }
402
403     return valeur;
404 }
405
406 /// HIFN Fonction permettant de sauvgarder dans un fichier texte les valeurs brutes des
        acquisitions et plusieurs autres informations
407 int save(char *linebuffer,float64 *datablock, int32 numReadblock , double time,int gain1,int
        gain2)
408 {
409
410   char line[1024]={'\0'};
411   char timename1[256]={'\0'};
412
413   if(linebuffer != NULL)
414   {
415     timestring(timename1,time);
416
417     sprintf(line, "%s %2.4f %2.4f %2.4f %2.4f %d %d %2.4f %2.4f %2.4f %2.4f %3.1f %3.1f %3.1
          f %d\n", timename1, datablock[0]/numReadblock, datablock[1]/numReadblock, datablock
          [2]/numReadblock, datablock[3]/numReadblock,gain1,gain2,config->dark_c1, config->
          dark_c2, config->sky_bg_c1, config->sky_bg_c2,centralin->val[0],centralin->val[1],
          centralin->val[2],numReadblock);
418
419     strcat (linebuffer,line);
420   }
421
422   return 0;
423 }
424
425 int writetofile(char *linebuffer)
426 {
427
428   int filehandle;
429   int error = -1;
430
431   if(linebuffer != NULL && strlen(file->fname) !=0 && FindPattern (file->fname, 0, strlen(
        file->fname), ".conf", 0, 1) == -1 )
432   {
433     filehandle = OpenFile (file->fname, VAL_WRITE_ONLY, VAL_APPEND, VAL_ASCII);
```

```
434     // on ne teste pas l'existance du fichier car on ouvre celui-ci en écriture et car on
            écrit toujours à partir de la fin de celui-ci
435
436     WriteFile(filehandle,linebuffer,strlen(linebuffer));
437     // écrit la ligne dans le fichier et passe à la ligne
438
439     CloseFile(filehandle);
440     // ferme le fichier
441
442     error = GetFmtIOError();
443   }
444
445   return error;
446 }
447
448 /// HIFN Fonction permettant de lire la configuration contenu dans un fichier texte
449 int readConf(void)
450 {
451
452   int filehandle;
453   int  error=0,i=0,size=0;
454   char line[512]={'\0'};
455
456   if(FileExists(file->fname2,&size) == 0 || FindPattern (file->fname2, 0, strlen(file->
          fname2), ".conf", 0, 1) == -1)
457   {
458     error = -1;
459   }
460   else
461   {
462     filehandle = OpenFile (file->fname2, VAL_READ_ONLY, VAL_OPEN_AS_IS, VAL_ASCII);
463
464     for(i = 0; i <= 26 && error != -1; i++)
465     {
466       do
467       {
468         error = ReadLine(filehandle,line,510);
469
470       }
471       while( (line[0] == '\0' || line[0] == '#' || line[0] == '\n' ) && error != -1);
472
473
474       if( error != -1 )
475       {
476         switch(i)
```

```
477         {
478           case 0: Scan(line, "%d",&(graph->gr)); break;
479           case 1: Scan(line, "%d",&(file->fi)); break;
480           case 2: Scan(line, "%f",&(config->range)); break;
481           case 3: Scan(line, "%d",&(config->sampling_rate)); break;
482           case 4: Scan(line, "%d",&(config->n_sbloc)); break;
483           case 5: Scan(line, "%d",&(config->n_bloc)); break;
484           case 6: Scan(line, "%f",&(config->buffertime)); break;
485           case 7: Scan(line, "%d",&(config->linebufferlength)); break;
486
487           case 8: Scan(line, "%f",&(config->dark_c1)); break;
488           case 9: Scan(line, "%f",&(config->sky_bg_c1)); break;
489           case 10: Scan(line, "%f x + %f",&(config->photo1F.a),&(config->photo1F.b)); break;
490           case 11: Scan(line, "%f x + %f",&(config->photo1f.a),&(config->photo1f.b)); break;
491           case 12: Scan(line, "%d",&(config->manual_auto1)); break;
492           case 13: Scan(line, "%d",&(config->mm_n)); break;
493           case 14: Scan(line, "%f",&(config->g_max1)); break;
494           case 15: Scan(line, "%f",&(config->g_min1)); break;
495           case 16: Scan(line, "%d",&(config->mm_sw1)); break;
496
497           case 17: Scan(line, "%f",&(config->dark_c2)); break;
498           case 18: Scan(line, "%f",&(config->sky_bg_c2)); break;
499           case 19: Scan(line, "%f x + %f",&(config->photo2F.a),&(config->photo2F.b)); break;
500           case 20: Scan(line, "%f x + %f",&(config->photo2f.a),&(config->photo2f.b)); break;
501           case 21: Scan(line, "%d",&(config->manual_auto2)); break;
502           case 22: Scan(line, "%f",&(config->g_max2)); break;
503           case 23: Scan(line, "%f",&(config->g_min2)); break;
504           case 24: Scan(line, "%d",&(config->mm_sw2)); break;
505
506           case 25: Scan(line, "%f x + %f",&(config->therm1.a),&(config->therm1.b));
507           break;
508
509           case 26: Scan(line, "%f x + %f",&(config->therm2.a),&(config->therm2.b));
510           break;
511
512           default : error = -1; break;
513         }
514
515         if( NumFmtdBytes() <= 0 ) error = -1;
516
517       }
518
519     }
520
521     CloseFile(filehandle);
```

```
522   }
523
524   return error;
525 }
526
527 int logappend( char *fichier, char *line)
528 {
529   int filehandle;
530   int error = -1;
531   char tmp1[512],tmp2[512];
532   char tmp3[1]={' '};
533
534   if(strlen(fichier)!=0 )
535   {
536     filehandle = OpenFile (fichier, VAL_WRITE_ONLY, VAL_APPEND, VAL_ASCII);
537
538     strcpy (tmp1, DateStr());
539     strcpy (tmp2, TimeStr());
540
541     WriteLine(filehandle,tmp1,strlen(tmp1));
542     WriteLine(filehandle,tmp2,strlen(tmp2));
543     WriteLine(filehandle,line,strlen(line));
544     WriteLine(filehandle,tmp3,1);
545
546     CloseFile(filehandle);
547
548     error = GetFmtIOError();
549   }
550
551   return error;
552 }
553
554 int trigerrange(int panel)
555 {
556   double tmp1=9.5,tmp2=9.5;
557
558   if(graph->sub_dc)
559   {
560     tmp1 -= config->dark_c1;
561     tmp2 -= config->dark_c2;
562   }
563   if(graph->sub_sky_bg)
564   {
565     tmp1 -= config->sky_bg_c1;
566     tmp2 -= config->sky_bg_c2;
```

```
567   }
568
569   if(config->g_max1 > tmp1 || config->g_max1 < 0)
570   {
571     config->g_max1 = tmp1;
572     SetCtrlVal(panel,PANEL_G_MAX,config->g_max1);
573     config->g_min1 = tmp1-0.5;
574     SetCtrlVal(panel,PANEL_G_MIN,config->g_min1);
575   }
576
577   if(config->g_max2 > tmp2 || config->g_max2 < 0)
578   {
579     config->g_max2 = tmp2;
580     SetCtrlVal(panel,PANEL_G_MAX_2,config->g_max2);
581     config->g_min2 = tmp2-0.5;
582     SetCtrlVal(panel,PANEL_G_MIN_2,config->g_min2);
583   }
584
585   return 0;
586 }
```

```c
//==============================================================================
//
// Title:       Daq.h
// Purpose:     Functions déclarations .
//
// Created on: 14/09/2007 at 14:30:13 by IATRIDES Clément.
// Copyright:  cnrs. All Rights Reserved.
//
//==============================================================================

#ifndef __Daq_H__
#define __Daq_H__

#ifdef __cplusplus
    extern "C" {
#endif

#define DAQmxErrChk(functionCall) if( DAQmxFailed(error=(functionCall)) ) goto Error; else

#include <windows.h>
#include <ansi_c.h>
#include "NIDAQmx.h"
#include <DAQmxIOctrl.h>
#include "types.h"

int CVICALLBACK startAcc (void *functionData);
int cleanupAcc (void);

int32 CVICALLBACK EveryNSamplesCallback(TaskHandle taskHandle, int32 everyNsamplesEventType,
    uInt32 nSamples, void *callbackData);
double moyNread2(int voie,int hz,int n, double min , double max);

int writeGainDigit(int port, int line, uInt8 gain);


#ifdef __cplusplus
    }
#endif

#endif /* ndef __Daq_H__ */
```

```
1  //==============================================================================
2  //
3  // Title:      Daq.c
4  // Purpose:    Gestion du NI USB-6008.
5  //
6  // Created on: 14/09/2007 at 14:30:13 by IATRIDES Clément.
7  // Copyright:  cnrs. All Rights Reserved.
8  //
9  //==============================================================================
10
11 #include "projet.h"
12 #include "functions.h"
13 #include "Daq.h"
14
15 static TaskHandle taskHandle=0;
16 // index de la tache du périphérique d'acquisition
17
18 /* definitions des variables externes */
19 extern volatile struct Config *config;
20 extern volatile int gRunning,gRead,gIgnore;
21 extern struct Donnees *donneesStart;
22 extern struct Donnees *donneesCurrent;
23 extern volatile int blocsskp;
24 extern struct Temps *temps;
25 extern int panelHandle;
26
27 /// HIFN Fonction configurant et démarrant l'acquisition continue d'échantillons sur les
       quatres entrées analogiques avec interuptions
28 int CVICALLBACK startAcc (void *functionData)
29 {
30   int32   error=0;
31   char    errBuff[2048]={'\0'};
32   // variables pour la gestion des erreurs
33
34   /*********************************************/
35   // DAQmx Configure Code
36   /*********************************************/
37   DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
38   // crée une tache sur le périphérique
39
40   DAQmxErrChk (DAQmxCreateAIVoltageChan(taskHandle,"Dev1/ai0","",DAQmx_Val_Cfg_Default,-1 *
       config->range,config->range,DAQmx_Val_Volts,NULL));
41   // ajoute à la tache une acquisition de signal analogique
42
```

```
43    DAQmxErrChk (DAQmxCreateAIVoltageChan(taskHandle,"Dev1/ai1","",DAQmx_Val_Cfg_Default,-1 *
          config->range,config->range,DAQmx_Val_Volts,NULL));
44
45    DAQmxErrChk (DAQmxCreateAIVoltageChan(taskHandle,"Dev1/ai2","",DAQmx_Val_Cfg_Default,-1 *
          config->range,config->range,DAQmx_Val_Volts,NULL));
46
47    DAQmxErrChk (DAQmxCreateAIVoltageChan(taskHandle,"Dev1/ai3","",DAQmx_Val_Cfg_Default,-1 *
          config->range,config->range,DAQmx_Val_Volts,NULL));
48
49
50    DAQmxErrChk (DAQmxCfgSampClkTiming(taskHandle,"",config->sampling_rate,DAQmx_Val_Rising,
          DAQmx_Val_ContSamps,config->n_sbloc));
51    // configure la fréquance des acquisition ainsi que le nombre d'acquisition par bloc d'
          acquisition
52
53    DAQmxErrChk (DAQmxRegisterEveryNSamplesEvent(taskHandle,DAQmx_Val_Acquired_Into_Buffer,
          config->n_sbloc,0,EveryNSamplesCallback,NULL));
54    // configure le périphérique pour qu'une interuption soit générié à la fin de chaque bloc
55
56    /* configure l'amplification externe avec le guain par default */
57    writeGainDigit(0,0,(uInt8)config->mm_sw1);
58    writeGainDigit(0,1,(uInt8)config->mm_sw2);
59
60
61    /*********************************************/
62    // DAQmx Start Code
63    /*********************************************/
64
65    DAQmxErrChk (DAQmxStartTask(taskHandle));
66    // démarre l'acquisition
67
68    gRunning = 1;
69
70  /* gestion des érreurs */
71  Error:
72    SetWaitCursor(0);
73    if( DAQmxFailed(error) )
74    {
75      DAQmxGetExtendedErrorInfo(errBuff,2048);
76      cleanupAcc ();
77      MessagePopup("DAQmx Error",errBuff);
78    }
79
80    return 0;
81  }
```

```
82
83  /// HIFN Routine d'interuption récuperant les nSamples acquisitions
84  int32 CVICALLBACK EveryNSamplesCallback(TaskHandle taskHandle, int32 everyNsamplesEventType,
        uInt32 nSamples, void *callbackData)
85  {
86    int32       error=0;
87    char        errBuff[2048]={'\0'};
88    // variables pour la gestion des erreurs
89
90    struct Donnees *donneesNext=NULL;
91    // pointeur pour la structure suivante
92
93    donneesCurrent->time = (double)(GetTickCount() - temps->time1) / 1000 + temps->j*3600*24+
          temps->h*3600+temps->m*60+temps->s;
94    // datation de la récupération du bloc
95
96    /* affectation de la mémoire pour le bloc de donnees et pour la structure qui contiendra
          le block suivant */
97    if((donneesNext=malloc(sizeof(struct Donnees)))==NULL || (donneesCurrent->data=malloc(
          config->n_sbloc * 4 * sizeof(float64)))==NULL)
98    {
99      MessagePopup("Error","Not enough memory");
100     // message d'erreur si mémoire insufisante
101     goto Error;
102   }
103
104   /*******************************************/
105   // DAQmx Read Code
106   /*******************************************/
107   donneesCurrent->gain1 = config->mm_sw1;
108   donneesCurrent->gain2 = config->mm_sw2;
109
110   DAQmxErrChk (DAQmxReadAnalogF64(taskHandle,nSamples,10.0,DAQmx_Val_GroupByScanNumber,
          donneesCurrent->data,config->n_sbloc*4,&(donneesCurrent->numRead),NULL));
111   // lit les acquisition du bloc et les stoquent dans la structure courrente
112
113   if(gRead >= ((config->buffertime+1) * config->sampling_rate / config->n_sbloc))
114   {
115     SetCtrlVal(panelHandle,PANEL_BLOCSSKP,blocsskp++);
116     gIgnore = 0;
117     donneesCurrent->numRead = 0;
118     if(config->buffertime < 10)
119     {
120       config->buffertime += 0.1;
121       SetCtrlVal(panelHandle,PANEL_BUFFER,config->buffertime);
```

```
122     }
123   }
124   else if(gIgnore != 0)
125   {
126     gIgnore--;
127     donneesCurrent->numRead = 0;
128   }
129 
130   donneesCurrent->next = donneesNext;
131   // affectation du pointeur de la donnée courrente pour qu'il pointe vers la donnée
        suivante
132 
133   donneesCurrent=donneesNext;
134   // la structure suivante devient la structure courrente
135 
136   gRead++;
137   SetCtrlVal(panelHandle,PANEL_READ, gRead);
138 
139 /* gestion des érreurs */
140 Error:
141   if( DAQmxFailed(error) )
142   {
143     DAQmxGetExtendedErrorInfo(errBuff,2048);
144     cleanupAcc ();
145     MessagePopup("DAQmx Error",errBuff);
146   }
147 
148   if(gRunning == 0 ) cleanupAcc();
149 
150   return 0;
151 }
152 
153 
154 /// HIFN Fonction permettant de stoper les acquisitions
155 int cleanupAcc (void)
156 {
157   if( taskHandle!=0 )
158   // test si la tache existe sur le périphérique
159   {
160 
161     /*********************************************/
162     // DAQmx Stop Code
163     /*********************************************/
164     DAQmxStopTask(taskHandle);
165     // arrete la tache
```

```
166
167     DAQmxClearTask(taskHandle);
168     // éfface la configuration de la tache du périphérique
169
170     taskHandle = 0;
171
172
173   }
174
175   return 0;
176 }
177
178 /// HIFN Fonction permettant moyener n acquisitions a hz HZ sur 1 vois dans le calibre min-
      max
179 double moyNread2(int voie,int hz,int n, double min , double max)
180 {
181   int32       error=0;
182   char        errBuff[2048]={'\0'};
183   // variables pour la gestion des erreurs
184
185   struct Donnees *donnees=NULL;
186   double tmp=-1;
187   char port[64] = {'\0'};
188   double databloc[4] = {0,0,0,0};
189
190   if(taskHandle == 0 && ( voie == 0 || voie == 1 || voie == 2 || voie == 3))
191   {
192
193     sprintf(port,"Dev1/ai%d",voie);
194
195     /*********************************************/
196     // DAQmx Configure Code
197     /*********************************************/
198     DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
199
200     DAQmxErrChk (DAQmxCreateAIVoltageChan(taskHandle,port,"",DAQmx_Val_Cfg_Default,min,max,
          DAQmx_Val_Volts,NULL));
201
202     DAQmxErrChk (DAQmxCfgSampClkTiming(taskHandle,"",hz,DAQmx_Val_Rising,DAQmx_Val_ContSamps
          ,n));
203
204     /*********************************************/
205     // DAQmx Start Code
206     /*********************************************/
207
```

```
208     DAQmxErrChk (DAQmxStartTask(taskHandle));
209
210     if((donnees=malloc(sizeof(struct Donnees)))==NULL || (donnees->data=malloc(n*sizeof(
            float64)))==NULL )
211     {
212       MessagePopup("Error","Not enough memory");
213       goto Error;
214     }
215
216     /*********************************************/
217     // DAQmx Read Code
218     /*********************************************/
219     DAQmxErrChk (DAQmxReadAnalogF64(taskHandle,n,10.0,DAQmx_Val_GroupByScanNumber,donnees->
            data,n,&(donnees->numRead),NULL));
220
221     cleanupAcc ();
222
223     //tmp = add(0,1,donnees) / donnees->numRead; // il n'y a qu'une seule voie dans ce bloc
            de données
224
225     databloc[voie] = add(0,1,donnees);
226
227     tmp = val(voie,databloc,donnees->numRead,1,1);
228   }
229
230 Error:
231   if( DAQmxFailed(error) )
232   {
233     DAQmxGetExtendedErrorInfo(errBuff,2048);
234     cleanupAcc ();
235     MessagePopup("DAQmx Error",errBuff);
236   }
237
238 return tmp;
239 }
240
241 /// HIFN Fonction permettant d'écrire le bit gain sur la ligne line
242 int writeGainDigit(int port, int line, uInt8 gain)
243 {
244   int         error=0;
245   TaskHandle taskHandle2=0;
246   char        errBuff[2048]={'\0'};
247   char    linename[128];
248
249   if(gain == 0 || gain == 1 )
```

```
250   {
251
252     //gain = !gain; // invertion pour la plaquette d'essai
253
254     if((port == 0 && line >= 0 && line <= 7 )|| (port == 1&& line >= 0 && line <= 3 ))
255     {
256       sprintf(linename , "Dev1/port%d/line%d",port,line);
257
258       /*********************************************/
259       // DAQmx Configure Code
260       /*********************************************/
261       DAQmxErrChk (DAQmxCreateTask("",&taskHandle2));
262       DAQmxErrChk (DAQmxCreateDOChan(taskHandle2,linename,"",DAQmx_Val_ChanForAllLines));
263
264       /*********************************************/
265       // DAQmx Start Code
266       /*********************************************/
267       DAQmxErrChk (DAQmxStartTask(taskHandle2));
268
269       /*********************************************/
270       // DAQmx Write Code
271       /*********************************************/
272       DAQmxErrChk (DAQmxWriteDigitalLines(taskHandle2,1,1,10.0,DAQmx_Val_GroupByChannel,&gain
              ,NULL,NULL));
273
274     }
275     else  MessagePopup("Error","Configuration des Ports erronée");
276
277   }
278   else  MessagePopup("Error","Configuration des Gains erronée");
279
280 Error:
281   if( DAQmxFailed(error) )
282     DAQmxGetExtendedErrorInfo(errBuff,2048);
283   if( taskHandle2!=0 ) {
284     /*********************************************/
285     // DAQmx Stop Code
286     /*********************************************/
287     DAQmxStopTask(taskHandle2);
288     DAQmxClearTask(taskHandle2);
289   }
290   if( DAQmxFailed(error) )
291     MessagePopup("DAQmx Error",errBuff);
292   return 0;
293 }
```

```
//==============================================================================
//
// Title:       Gps.h
// Purpose:     Functions déclarations.
//
// Created on: 17/09/2007 at 13:35:08 by IATRIDES Clément.
// Copyright:  cnrs. All Rights Reserved.
//
//==============================================================================

#ifndef __Gps_H__
#define __Gps_H__

#ifdef __cplusplus
    extern "C" {
#endif

#include <windows.h>
#include "cvidef.h"
#include <utility.h>
#include <rs232.h>
#include <ansi_c.h>
#include "types.h"

#define MAX_TEST_GPS 10

int CVICALLBACK gps(void *functionData);

int gps_start(void);
int gps_stop(void);
int readtime (struct Temps *tempsnew);

int calibrate_time(struct Temps *tempsnew);
int system_time(struct Temps *tempsnew);
int changement_jour(struct Temps *tempsnew);

#ifdef __cplusplus
    }
#endif

#endif /* ndef __Gps_H__ */
```

```
//==============================================================================
//
// Title:      Gps.c
// Purpose:    Gestion du GPS et de l'heure.
//
// Created on: 17/09/2007 at 13:35:08 by cetp.
// Copyright:  cnrs. All Rights Reserved.
//
//==============================================================================

#include "projet.h"
#include "Gps.h"

extern int panelHandle;
extern volatile int gRunning,gGps;
extern struct Temps *temps;

/// HIFN Fonction permettant de faire une acquisition de temps Gps en continue
int CVICALLBACK gps(void *functionData)
{
  int on;

  while(gGps)
  {
    GetCtrlVal(panelHandle,PANEL_GPS_ON,&on);

    if(on && gRunning == 0)
    {

      if( gps_start() == 0 )
      {
        calibrate_time(temps);
        gps_stop();
      }

      if(temps->satellites < 2)
      {
        system_time(temps);
        temps->mode = 0;
      }
      else
      {
        temps->mode = 1;

      }
```

```
46
47      SetCtrlVal(panelHandle,PANEL_GPS_H,temps->h);
48      SetCtrlVal(panelHandle,PANEL_GPS_M,temps->m);
49      SetCtrlVal(panelHandle,PANEL_GPS_S,temps->s);
50      SetCtrlVal(panelHandle,PANEL_GPS_N,temps->satellites);
51
52      if( temps->h < 12) temps->h += 24; // on tient compte du changement de jour et on
            guarde une heure cohérante pour tracer les graphs
53
54    }
55    else if(on && gRunning == 1) MessagePopup("Warning","Veuillez étindre l'acquisition des
          photometre pour mettre à jour la référance de temps");
56
57  }
58
59  return 0;
60 }
61
62 /// HIFN Fonction permettant d'ouvrir le port RS-232 du GPS
63 int gps_start(void)
64 {
65  int retour = -1;
66
67  retour = OpenComConfig (3, "COM3", 4800, 0, 8, 1, 4096, 4096);
68  Delay(0.1);
69
70  return retour;
71 }
72
73 /// HIFN Fonction permettant de fermer le port RS-232 du GPS
74 int gps_stop(void)
75 {
76  CloseCom (3);
77
78  return 0;
79 }
80
81 /// HIFN Fonction permettant de lire une heure dans le buffer du port RS-232 du GPS
82 int readtime(struct Temps *tempsnew)
83 {
84  char *read_data;
85  int read_cnt=0,bytes_read;
86  int i,retour = -1;
87  char heure[3]={0,0,'\0'},minutes[3]={0,0,'\0'},secondes[3]={0,0,'\0'},nombre[3]={0,0,'\0'
        };
```

```
88
89  read_cnt = GetInQLen(tempsnew->n);
90  // on récupere le nombre d'élément contenu dans le buffer
91
92  read_data = malloc(read_cnt*sizeof(char));
93  // allocation de la mémoire nécéssaire
94
95  bytes_read = ComRd (tempsnew->n, read_data, read_cnt);
96  // lecture du buffer
97
98  if (bytes_read >=75)
99  // si le nombre d'éléments lu est supérieur ou égale au nombre d'éléments qui constitue la
        trame NMEA choisi
100   {
101       for(i=0; i< read_cnt-74; i++)
102       {
103
104     if(read_data[i]=='$' && read_data[i+1]=='G' && read_data[i+2]== 'P' && read_data[i
          +3]== 'G' && read_data[i+4]== 'G' && read_data[i+5]== 'A')
105     // si la trame NMEA choisi est contenu dans le buffer
106     {
107
108       heure[0] = read_data[i+7];
109       heure[1] = read_data[i+8];
110
111       tempsnew->h = atoi(heure) ;
112
113       minutes[0] = read_data[i+9];
114       minutes[1] = read_data[i+10];
115
116       tempsnew->m = atoi(minutes);
117
118       secondes[0] = read_data[i+11];
119       secondes[1] = read_data[i+12];
120
121       tempsnew->s = atoi(secondes);
122
123       nombre[0] = read_data[i+47];
124       nombre[1] = read_data[i+48];
125
126       tempsnew->satellites = atoi(nombre);
127
128       retour = 0;
129
130     }
```

```
131       }
132   }
133   else tempsnew->satellites = 0;
134 
135   return retour;
136 }
137 
138 /// HIFN Fonction permettant de calibrer l'heure en fonction du temps GPS
139 int calibrate_time(struct Temps *tempsnew)
140 {
141    int i = 0;
142    int retour = -1;
143 
144    while(retour != 0 && i < MAX_TEST_GPS)
145    {
146      retour = readtime(tempsnew);
147      tempsnew->time1 = GetTickCount();
148      Delay(0.45);
149      i++;
150    }
151 
152    return 0;
153 }
154 
155 /// HIFN Fonction permettant de calibrer l'heure en fonction du temps PC
156 int system_time(struct Temps *tempsnew)
157 {
158 
159 
160   GetSystemTime (&(tempsnew->h),&(tempsnew->m),&(tempsnew->s));
161   tempsnew->time1 = GetTickCount();
162 
163 
164 
165   return 0;
166 }
167 
168 int changement_jour(struct Temps *tempsnew)
169 {
170   /* On recopie de nombre précédant de jours */
171   tempsnew->j = temps->j;
172 
173   /* On remet l'heure en UTC si elle provient du PC*/
174   if( tempsnew->mode == 0 )
175   {
```

```
176     if(tempsnew->h < tempsnew->timezone)
177     {
178       tempsnew->h += 24;
179       tempsnew->j--;
180     }
181
182     tempsnew->h -= tempsnew->timezone % 24;
183   }
184
185   if(temps->h > tempsnew->h || tempsnew->h > 23)
186   {
187     tempsnew->h -= (tempsnew->h / 24) * 24;
188     tempsnew->j++;
189   }
190
191   if(temps->h == 0 && tempsnew->h == 23 && (tempsnew->time1 - temps->time1) < 2*3600*1000)
        tempsnew->j--;
192
193   tempsnew->j += (int)((double)(tempsnew->time1 - temps->time1)/(1000*3600*24));
194
195   return 0;
196 }
```

```
//==============================================================================
//
// Title:      centralIn.h
// Purpose:    Functions déclarations.
//
// Created on: 14/09/2007 at 10:39:51 by IATRIDES Clément.
// Copyright:  cnrs. All Rights Reserved.
//
//==============================================================================

#ifndef __centralIn_H__
#define __centralIn_H__
#ifdef __cplusplus
    extern "C" {
#endif

#include <utility.h>
#include <rs232.h>
#include <ansi_c.h>
#include "types.h"
#define TWO_EXP_FIFTEEN 32768
#define PI_PERS 3.14159265358979323846264338327 95
#define ANGLE_MODE 1
#define ACC_MODE 2
#define N_CI 1

int CVICALLBACK centrale (void *functionData);
int affCI(void);
int getCI(void);
int ironCI(void);
int start_rs(void);
int stop_rs(void);
int get_packet(int mode);
int verify_packet(int *result, int size);
int get_fields(int *result,int size,int mode);
double normalize(double *tab);
int sg(double nb);
int H(double nb);
void  vectoriel( double *u, double *v, double *w);
int min_tab(double *tab,int size);
int max_tab(double *tab,int size);

#ifdef __cplusplus
    }
#endif
```

```
46 #endif /* ndef __centralIn_H__ */
```

```c
//==============================================================================
//
// Title:      centralIn.c
// Purpose:    Gestion centrale inertielle
//
// Created on: 14/09/2007 at 10:39:51 by IATRIDES Clément.
// Copyright:  cnrs. All Rights Reserved.
//
//==============================================================================

#include "projet.h"
#include "centralIn.h"

static double moy[3]={0,0,0};
static int nmoy = 0;

static int RS232Open,RS232Error;

extern volatile struct CentralIn *centralin;
extern volatile int gRunning,gCin,gCinlock;

extern int panelHandle;

/// HIFN Fonction permettant de traiter les acquisition
int CVICALLBACK centrale (void *functionData)
{
  while(gCin)
  {
    getCI();
    // on calcule les angles recherchées

    affCI();
    // on affiche le résultat

    if(gRunning) Delay(5);
    else Delay(0.1);
  }

  return 0;
}

/// HIFN Fonction permettant de normaliser le vecteur tab
double normalize(double *tab)
{
  double sum = 0;
```

```
46    int i;
47
48    for(i=0;i<3;i++)
49    {
50      sum += tab[i] * tab[i];
51    }
52
53    sum=sqrt(sum);
54
55    for(i = 0; i<3 ; i++ )
56    {
57      tab[i] /= sum;
58    }
59
60    return sum;
61  }
62
63  /// HIFN Fonction permettant de vérifier les paquet RS-232
64  int verify_packet(int *result, int size)
65  {
66    int sum=0,i,retour=-1;
67
68    for(i = 1 ; i < size-1 ; i++)
69    {
70      sum += result[i];
71    }
72
73    if(sum<0) sum += 256;
74
75    if(result[0] == 225 && sum == result[29]) retour = 0;
76    else retour = -1;
77
78    return retour;
79  }
80
81  /// HIFN Fonction permettant d'identifier le signe de nb
82  int sg(double nb)
83  {
84    int retour=0;
85
86    if(nb > 0) retour = 1;
87    else if(nb < 0 ) retour = -1;
88    else retour = 0;
89
90    return retour;
```

```
91  }
92
93  /// HIFN Fonction permettant d'identifier le signe de nb
94  int H(double nb)
95  {
96    int retour=0;
97
98    if(nb >= 0) retour = 1;
99    else retour = 0;
100
101   return retour;
102 }
103
104 /// HIFN Fonction permettant de calculer le produit vectoriel w de u et v
105 void  vectoriel( double *u, double *v, double *w)
106 {
107   w[0] = u[1] * v[2] - u[2] * v[1];
108   w[1] = u[2] * v[0] - u[0] * v[2];
109   w[2] = u[0] * v[1] - u[1] * v[0];
110 }
111
112 /// HIFN Fonction permettant de trouver le minimum d'un tableau
113 int min_tab(double *tab,int size)
114 {
115   double min = tab[0];
116   int pos=0,i;
117
118   for(i = 1 ; i < size ; i++)
119   {
120     if(tab[i] < min) { min = tab[i]; pos = i; }
121   }
122
123   return pos;
124 }
125
126 /// HIFN Fonction permettant de trouver le maximum d'un tableau
127 int max_tab(double *tab,int size)
128 {
129   double max = tab[0];
130   int pos=0,i;
131
132   for(i = 1 ; i < size ; i++)
133   {
134     if(tab[i] > max) { max = tab[i]; pos = i; }
135   }
```

```
136
137   return pos;
138 }
139
140 /// HIFN Fonction permettant de calculer les angles recherchées
141 int get_fields(int *result,int size,int mode)
142 {
143   double phy=0,theta=0,angles[3],a[3],B[3],w[3],Bortho[3],aortho[2],aNorme2d,Bnorme,myroll
        =0;
144   int retour=-1,i;
145
146   if(verify_packet(result,size)) // test la validitée des données
147   {
148
149      if(mode == ANGLE_MODE)
150      {
151        /* Lecture des anges stabilisé */
152        angles[0] = result[1] * 256 + result[2];
153        angles[1] =

             result[3] * 256 + result[4];
154        angles[2] = result[5] * 256 + result[6];
155
156        /* mise à l'échelle des valeurs*/
157        for( i = 0 ; i < 3 ; i++)
158        {
159          angles[i] *= 180.0 / TWO_EXP_FIFTEEN;
160        }
161
162        /* moyennage des résultats */
163        moy[0] += angles[2];
164        moy[1] += angles[1];
165        moy[2] += angles[0];
166
167        nmoy++;
168      }
169
170      if(mode == ACC_MODE)
171      {
172
173        /* Lecture des valeur d'acélérations */
174        a[0] = result[13] * 256 + result[14];
175        a[1] = result[15] * 256 + result[16];
176        a[2] = result[17] * 256 + result[18];
177
```

```
178     /* Lecture des valeur de champs magnétiques */
179     B[0] = result[19] * 256 + result[20];
180     B[1] = result[21] * 256 + result[22];
181     B[2] = result[23] * 256 + result[24];
182
183     /* mise à l'échelle des valeurs*/
184     for( i = 0 ; i < 3 ; i++)
185     {
186       a[i] *= 4.0 * 1.5 / TWO_EXP_FIFTEEN;
187
188       B[i] *= 1.25 * 1.5 / TWO_EXP_FIFTEEN;
189     }
190
191     normalize(a); // on normalise le vecteur a
192
193     Bnorme = normalize(B); // on normalise le vecteur B et on récupere sa norme
194
195     vectoriel(a,B,w); // on calcul le produit vectoriel de a par B et on le stque dans w
196
197     normalize(w); // on normalise w
198
199     vectoriel(w,a,Bortho); // on calcul le produit vectoriel de w par a et on le stque
        dans Bortho
200
201     normalize(Bortho); // on normalise Bortho
202
203     aNorme2d = sqrt(a[0] * a[0] + a[2] * a[2]); // calcul de la norme 2 dimension de a
204     aortho[0] = a[0] / aNorme2d; // calcul de aorthogonale
205     aortho[1] = a[2] / aNorme2d;
206
207     phy = acos(aortho[1]) * sg(a[0]) * 180.0 / PI_PERS; // calcul de l'angle à la
        verticale
208
209     aNorme2d = sqrt(a[1] * a[1] + a[2] * a[2]);// calcule de la norme 2 dimension de a sur
         un autre doublet de coordonées
210     aortho[0] = a[1] / aNorme2d; // calcul de aorthogonale
211     aortho[1] = a[2] / aNorme2d;
212
213     myroll = acos(aortho[1]) * sg(a[1]) * 180.0 / PI_PERS; // calcul de l'acsiette
214
215     /* calcul de l'angle par rapport au nord magnérique */
216     if(w[0] != 0) theta = ( atan(w[0]/Bortho[0]) + PI_PERS*H(-Bortho[0])*sg(w[0])) * 180.0
         / PI_PERS;
217     else theta = 90 * sg(B[1]);
218
```

```
219       /* moyennage des résultats */
220       moy[0] += phy;
221       moy[1] += myroll;
222       moy[2] += theta;
223
224       nmoy++;
225     }
226
227     retour = 0;
228   }
229
230   return retour;
231 }
232
233 /// HIFN Fonction permettant de lire et de traiter un backet de données
234 int get_packet(int mode)
235 {
236   int i,read_cnt;
237   char *read_data=NULL;
238   int *result=NULL;
239   int retour = -1;
240
241   ComWrt(centralin->n,"G",1);
242   Delay(0.1);
243
244   read_cnt = GetInQLen(centralin->n);
245
246   read_data = malloc(read_cnt*sizeof(int));
247   result = malloc((read_cnt-1)*sizeof(int));
248
249   ComRd(centralin->n,read_data,read_cnt);
250
251   if( read_cnt > 0 )
252   {
253     for(i = 0; i < read_cnt-1;i++)
254     {
255       if(i == 0 || (i>=25 && i<= 29)) result[i] = (int)((unsigned char)read_data[i]);
256       else result[i] = (int)read_data[i];
257     }
258
259     retour = get_fields(result,read_cnt-1,mode);
260   }
261
262   return retour;
263 }
```

```
264
265 /// HIFN Fonction permettant de fermer le port RS-232 de la centrale inertielle
266 int stop_rs(void)
267 {
268   RS232Open = CloseCom (centralin->n);
269
270   RS232Error = ReturnRS232Err();
271
272   return 0;
273 }
274
275 /// HIFN Fonction permettant d'ouvrir et de configurer le port RS-232 et la centrale
      inertielle
276 int start_rs(void)
277 {
278   int read_cnt,retour=0;
279   char *read_data=NULL;
280
281   RS232Open = OpenComConfig(centralin->n,"COM1",38400,0,8,1,4096,4096);
282
283   ComWrt(centralin->n,"R",1);
284   Delay(0.1);
285
286   read_cnt = GetInQLen(centralin->n);
287
288   read_data = malloc(read_cnt*sizeof(char));
289
290   ComRd(centralin->n,read_data,read_cnt);
291
292
293   if(read_cnt > 0 && read_data[0] == 'H')
294   {
295
296     ComWrt(centralin->n,"P",1);
297
298     ComWrt(centralin->n,"a",1);
299     Delay(0.1);
300
301     ComRd(centralin->n,read_data,read_cnt);
302
303
304     if(read_cnt > 0 && read_data[0] != 'A') retour = -1;
305   }
306   else retour = -1;
307
```

```
308   free(read_data);
309
310   RS232Error = ReturnRS232Err();
311
312   return retour;
313 }
314
315 /// HIFN Fonction permettant d'afficher les angles recherchées
316 int affCI(void)
317 {
318       centralin->val[0] = moy[0]/nmoy;
319       centralin->val[1] = moy[1]/nmoy;
320       centralin->val[2] = moy[2]/nmoy;
321
322       /* X , Y et Z digital displays */
323       SetCtrlVal(panelHandle,PANEL_CI_X,centralin->val[0]);
324       SetCtrlVal(panelHandle,PANEL_CI_Y,centralin->val[1]);
325       SetCtrlVal(panelHandle,PANEL_CI_Z,centralin->val[2]);
326
327   return 0;
328 }
329
330
331 /// HIFN Fonction permettant de faire une acquisition d'angle
332 int getCI(void)
333 {
334   moy[0] = 0;
335   moy[1] = 0;
336   moy[2] = 0;
337   nmoy = 0;
338
339   while( nmoy <= N_CI )
340   {
341     get_packet(ANGLE_MODE);
342   }
343
344   return 0;
345 }
346
347 /// HIFN Fonction permettant de faire une calibration interne de la centralle inertielle
348 int ironCI(void)
349 {
350   int read_cnt,retour=-1;
351   char *read_data=NULL;
352
```

```
353   start_rs();
354
355   ComWrt(centralin->n,"h",1);
356   Delay(0.1);
357
358   read_cnt = GetInQLen(centralin->n);
359
360   read_data = malloc(read_cnt*sizeof(char));
361
362   ComRd(centralin->n,read_data,read_cnt);
363
364   if(read_data[0] == 'H')
365   {
366     ComWrt(centralin->n,"t",1);
367     Delay(0.1);
368
369     read_cnt = GetInQLen(centralin->n);
370
371     read_data = malloc(read_cnt*sizeof(char));
372
373     ComRd(centralin->n,read_data,read_cnt);
374
375     if(read_data[0] == 'T')
376     {
377       ComWrt(centralin->n,"s",1);
378       Delay(0.1);
379
380       read_cnt = GetInQLen(centralin->n);
381
382       read_data = malloc(read_cnt*sizeof(char));
383
384       ComRd(centralin->n,read_data,read_cnt);
385
386       if(read_data[0] == 'S')
387       {
388         MessagePopup("Iron Calibration","Veuillez faire tourner le systeme de 360 sur lui
              même en le gardant aussi horizontale que possible");
389         ComWrt(centralin->n,"u",1);
390         Delay(0.1);
391         read_cnt = GetInQLen(centralin->n);
392         read_data = malloc(read_cnt*sizeof(char));
393         ComRd(centralin->n,read_data,read_cnt);
394         if(read_data[0] == 'U') retour = 0;
395       }
396     }
```

```
397     }
398
399     getCI();
400
401     stop_rs();
402
403     affCI();
404
405     free(read_data);
406
407     RS232Error = ReturnRS232Err();
408
409     return retour;
410 }
```

```
1  //==============================================================================
2  //
3  // Title:      function2.h
4  // Purpose:    A short description of the interface.
5  //
6  // Created on: 30/08/2007 at 13:51:41 by cetp.
7  // Copyright:  cnrs. All Rights Reserved.
8  //
9  //==============================================================================
10
11 #ifndef __function2_H__
12 #define __function2_H__
13
14 #ifdef __cplusplus
15     extern "C" {
16 #endif
17
18 #include <windows.h>
19 #include <ansi_c.h>
20 #include <utility.h>
21 #include <formatio.h>
22 #include <math.h>
23 #include "cvidef.h"
24
25 #define SIZE_MAX 600000
26
27
28
29 struct Data {
30
31   double *time,*data1,*data2,*data3,*data4,*data5, *data6, *data7,*data8, *data9, *data10,*
       data11 ,*y,*yf,ddt,df,size,*n;
32   double *gain1,*gain2;
33
34 };
35
36 struct Transfert {
37
38   double a,b;
39 };
40
41 struct Config {
42
43   struct Transfert photo1F,photo1f,photo2F,photo2f,therm1,therm2;
44
```

```
45 };
46
47 struct Aff {
48
49   int n,n_2,photo1,photo2,temp1,temp2,dark1,dark2,sky1,sky2,x,y,z,photo1_2,photo2_2,temp1_2,
       temp2_2,dark1_2,dark2_2,sky1_2,sky2_2,x_2,y_2,z_2,gain1,gain2,gain1_2,gain2_2;
50
51 };
52
53 int readparam(int panel);
54 int load(void);
55 int readConf(void);
56 double val(int channel ,double data, int gain);
57 #ifdef __cplusplus
58     }
59 #endif
60
61 #endif /* ndef __function2_H__ */
```

.3.2 PHOTO-read.exe

```
/*****************************************************************************/
/* LabWindows/CVI User Interface Resource (UIR) Include File                 */
/* Copyright (c) National Instruments 2007. All Rights Reserved.             */
/*                                                                           */
/* WARNING: Do not add to, delete from, or otherwise modify the contents of */
/*          this include file.                                               */
/*****************************************************************************/
#include <userint.h>
#ifdef __cplusplus
    extern "C" {
#endif

     /* Panels and Controls: */

#define PANEL                           1       /* callback function: PanelCallback */
#define PANEL_GRAPH_3                   2       /* callback function: CursorCallback */
#define PANEL_GRAPH_4                   3       /* callback function: CursorCallback */
#define PANEL_GRAPH                     4       /* callback function: CursorCallback */
#define PANEL_TIMESCALE_2               5       /* callback function: TimeCallback */
#define PANEL_TIME_2                    6       /* callback function: TimeCallback */
#define PANEL_TEXTMSG_5                 7
#define PANEL_DECORATION_BLUE_4         8
#define PANEL_TIMESCALE                 9       /* callback function: TimeCallback */
#define PANEL_TIME                      10      /* callback function: TimeCallback */
#define PANEL_TEXTMSG                   11
#define PANEL_DECORATION_BLUE           12
#define PANEL_TEXTMSG_6                 13
#define PANEL_TEXTMSG_7                 14
#define PANEL_TEXTMSG_3                 15
#define PANEL_MAX_GRAPH_2               16      /* callback function: ScaleCallback */
#define PANEL_MIN_GRAPH_2               17      /* callback function: ScaleCallback */
#define PANEL_FNAME                     18
#define PANEL_PANEL_DECORATION_GR_2     19
#define PANEL_PANEL_DECORATION_GR_3     20
#define PANEL_PANEL_DECORATION_GREE     21
#define PANEL_CONF_LOAD                 22      /* callback function: ConfCallback */
#define PANEL_LOAD                      23      /* callback function: ProcessCallback */
#define PANEL_MAX_GRAPH                 24      /* callback function: ScaleCallback */
#define PANEL_MIN_GRAPH                 25      /* callback function: ScaleCallback */
#define PANEL_CONF_FILE                 26
#define PANEL_N_POINTS                  27
#define PANEL_DUREE                     28
#define PANEL_AUTO_2                    29      /* callback function: ScaleCallback */
```

```
#define PANEL_AUTO                  30      /* callback function: ScaleCallback */
#define PANEL_LIST2                 31      /* callback function: AffCallback */
#define PANEL_LIST1                 32      /* callback function: AffCallback */
#define PANEL_VAL_C_3               33
#define PANEL_TIME_C_3              34
#define PANEL_VAL_C_2               35
#define PANEL_TIME_C_2              36
#define PANEL_VAL_C                 37
#define PANEL_TIME_C                38
#define PANEL_HEURE                 39      /* callback function: CalcCallback */
#define PANEL_S                     40      /* callback function: CalcCallback */
#define PANEL_M                     41      /* callback function: CalcCallback */
#define PANEL_H                     42      /* callback function: CalcCallback */
#define PANEL_J                     43      /* callback function: CalcCallback */

     /* Menu Bars, Menus, and Menu Items: */

          /* (no menu bars in the resource file) */

     /* Callback Prototypes: */

int  CVICALLBACK AffCallback(int panel, int control, int event, void *callbackData, int
    eventData1, int eventData2);
int  CVICALLBACK CalcCallback(int panel, int control, int event, void *callbackData, int
    eventData1, int eventData2);
int  CVICALLBACK ConfCallback(int panel, int control, int event, void *callbackData, int
    eventData1, int eventData2);
int  CVICALLBACK CursorCallback(int panel, int control, int event, void *callbackData, int
    eventData1, int eventData2);
int  CVICALLBACK PanelCallback(int panel, int event, void *callbackData, int eventData1, int
    eventData2);
int  CVICALLBACK ProcessCallback(int panel, int control, int event, void *callbackData, int
    eventData1, int eventData2);
int  CVICALLBACK ScaleCallback(int panel, int control, int event, void *callbackData, int
    eventData1, int eventData2);
int  CVICALLBACK TimeCallback(int panel, int control, int event, void *callbackData, int
    eventData1, int eventData2);

#ifdef __cplusplus
    }
#endif
```

```
#include "toolbox.h"
#include <analysis.h>
#include <userint.h>



//==============================================================================
//
// Title:      recall.c
// Purpose:    A short description of the implementation.
//
// Created on: 30/08/2007 at 13:18:10 by cetp.
// Copyright:  cnrs. All Rights Reserved.
//
//==============================================================================


#include "recall.h"
#include "functions2.h"

static int panelHandle,size=0,start=0,loadok=0;
static double Scale=1,Time=0,Time2=0,Scale2=1;
static struct Data *donnees=NULL;
static struct Config *config=NULL;
static struct Aff *aff=NULL;




int __stdcall WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                       LPSTR lpszCmdLine, int nCmdShow)
{

  if( InitCVIRTE(hInstance,0,0)==0 )
    return -1; /* out of memory */
  if( (panelHandle=LoadPanel(0,"recall.uir",PANEL))<0 )
    return -1;

  DisplayPanel(panelHandle);

  donnees = malloc(sizeof(struct Data));
  donnees->ddt=0;
  donnees->df=0;
  config = malloc(sizeof(struct Config));
  aff = malloc(sizeof(struct Aff));
```

```
46
47    aff->photo1=-1;
48    aff->photo2=-1;
49    aff->temp1=-1;
50    aff->temp2=-1;
51    aff->dark1=-1;
52    aff->dark2=-1;
53    aff->sky1=-1;
54    aff->sky2=-1;
55    aff->x=-1;
56    aff->y=-1;
57    aff->z=-1;
58    aff->photo1_2=-1;
59    aff->photo2_2=-1;
60    aff->temp1_2=-1;
61    aff->temp2_2=-1;
62    aff->dark1_2=-1;
63    aff->dark2_2=-1;
64    aff->sky1_2=-1;
65    aff->sky2_2=-1;
66    aff->x_2=-1;
67    aff->y_2=-1;
68    aff->z_2=-1;
69    aff->n=-1;
70    aff->n_2=-1;
71    aff->gain1=-1;
72    aff->gain1_2=-1;
73    aff->gain2=-1;
74    aff->gain2_2=-1;
75
76    /* démarage de l'interface */
77    SetCtrlAttribute (panelHandle, PANEL_CONF_LOAD , ATTR_CMD_BUTTON_COLOR, VAL_RED);
78    SetPanelAttribute (panelHandle, ATTR_WINDOW_ZOOM, VAL_MAXIMIZE);
79    RunUserInterface();
80    DiscardPanel(panelHandle);
81
82    return 0;
83  }
84
85  int CVICALLBACK PanelCallback(int panel, int event, void *callbackData, int eventData1, int
        eventData2)
86  {
87    if( event==EVENT_CLOSE )
88    {
89      QuitUserInterface(0);
```

```
90    }
91
92    if(start==0)
93    {
94      readparam(panel);
95      start =1;
96    }
97
98    return 0;
99  }
100
101 int readparam(int panel)
102 {
103   int i=0,narg1=0,narg2=0,mode=0;
104   char arg1[2048]={'\0'},arg2[2048]={'\0'},tmp='\0';
105   char *line=NULL;
106
107   line = GetCommandLine();
108
109   for( i = 0; i < strlen(line); i++)
110   {
111     tmp = line[i];
112
113     switch( tmp )
114     {
115       default :
116
117         if(mode == 1)
118         {
119           arg1[narg1] = tmp;
120           narg1++;
121         }
122         else if( mode == 3 )
123         {
124           arg2[narg2] = tmp;
125           narg2++;
126         }
127
128         break;
129
130       case '"' : mode++; break;
131     }
132   }
133
134   arg1[narg1] = '\0';
```

```
135   arg2[narg2] = '\0';
136
137   if(strlen(arg2) > 0)
138   {
139     SetCtrlVal(panel,PANEL_FNAME,arg2);
140     narg1 = FindPattern (arg1, 0, strlen(arg1), "PHOT-read.exe", 0, 1);
141     arg1[narg1] = 'p';
142     arg1[narg1 + 1] = 'h';
143     arg1[narg1 + 2] = 'o';
144     arg1[narg1 + 3] = 't';
145     arg1[narg1 + 4] = 'o';
146     arg1[narg1 + 5] = '.';
147     arg1[narg1 + 6] = 'c';
148     arg1[narg1 + 7] = 'o';
149     arg1[narg1 + 8] = 'n';
150     arg1[narg1 + 9] = 'f';
151     arg1[narg1 + 10] = '\0';
152     SetCtrlVal(panel,PANEL_CONF_FILE,arg1);
153     if(readConf() != -1)
154     {
155       SetCtrlAttribute (panelHandle, PANEL_CONF_LOAD , ATTR_CMD_BUTTON_COLOR, VAL_GREEN);
156       SetCtrlAttribute(panel,PANEL_LOAD,ATTR_DIMMED,0);
157       ProcessCallback(panel, -1, EVENT_COMMIT,NULL, 0, 0);
158     }
159   }
160
161   return 0;
162 }
163
164
165 int CVICALLBACK ProcessCallback (int panel, int control, int event,
166     void *callbackData, int eventData1, int eventData2)
167 {
168   int file;
169   char fname[MAX_PATHNAME_LEN];
170   long fname_size=0;
171   int i=0;
172
173
174   if( event==EVENT_COMMIT )
175   {
176
177     SetCtrlAttribute (panel, PANEL_LOAD , ATTR_DIMMED, 1);
178
179     DeleteGraphPlot (panel ,PANEL_GRAPH , -1 , VAL_IMMEDIATE_DRAW);
```

```
180     DeleteGraphPlot (panel ,PANEL_GRAPH_4 , -1 , VAL_IMMEDIATE_DRAW);
181     DeleteGraphPlot (panel ,PANEL_GRAPH_3 , -1 , VAL_IMMEDIATE_DRAW);
182
183     aff->photo1=-1;
184     aff->photo2=-1;
185     aff->temp1=-1;
186     aff->temp2=-1;
187     aff->dark1=-1;
188     aff->dark2=-1;
189     aff->sky1=-1;
190     aff->sky2=-1;
191     aff->x=-1;
192     aff->y=-1;
193     aff->z=-1;
194     aff->photo1_2=-1;
195     aff->photo2_2=-1;
196     aff->temp1_2=-1;
197     aff->temp2_2=-1;
198     aff->dark1_2=-1;
199     aff->dark2_2=-1;
200     aff->sky1_2=-1;
201     aff->sky2_2=-1;
202     aff->x_2=-1;
203     aff->y_2=-1;
204     aff->z_2=-1;
205     aff->n=-1;
206     aff->n_2=-1;
207     aff->gain1=-1;
208     aff->gain1_2=-1;
209     aff->gain2=-1;
210     aff->gain2_2=-1;
211
212     for(i=0;i<12;i++)
213     {
214       CheckListItem (panel,PANEL_LIST1 ,i , 0);
215       CheckListItem (panel,PANEL_LIST2 ,i , 0);
216     }
217
218     ProcessDrawEvents();
219
220     if(control == PANEL_LOAD)
221     {
222       if( VAL_NO_FILE_SELECTED == FileSelectPopup ("Data", "photo.save", "*.save", "",
          VAL_LOAD_BUTTON, 0, 0, 1, 1, fname))
223         fname[0] = '\0';
```

```
224
225       SetCtrlVal(panel,PANEL_FNAME,fname);
226     }
227
228     GetCtrlVal(panel,PANEL_FNAME,fname);
229
230     if(strlen(fname)==0) MessagePopup("Error","File name invalid");
231     else
232     {
233
234       if((donnees->time=malloc(SIZE_MAX*sizeof(double))) == NULL ||
235       (donnees->data1=malloc(SIZE_MAX*sizeof(double))) == NULL ||
236       (donnees->data2=malloc(SIZE_MAX*sizeof(double))) == NULL ||
237       (donnees->data3=malloc(SIZE_MAX*sizeof(double))) == NULL ||
238       (donnees->data4=malloc(SIZE_MAX*sizeof(double))) == NULL ||
239       (donnees->data5=malloc(SIZE_MAX*sizeof(double))) == NULL ||
240       (donnees->data6=malloc(SIZE_MAX*sizeof(double))) == NULL ||
241       (donnees->data7=malloc(SIZE_MAX*sizeof(double))) == NULL ||
242       (donnees->data8=malloc(SIZE_MAX*sizeof(double))) == NULL ||
243       (donnees->data9=malloc(SIZE_MAX*sizeof(double))) == NULL ||
244       (donnees->data10=malloc(SIZE_MAX*sizeof(double))) == NULL ||
245       (donnees->data11=malloc(SIZE_MAX*sizeof(double))) == NULL ||
246       (donnees->gain1=malloc(SIZE_MAX*sizeof(double))) == NULL ||
247       (donnees->gain2=malloc(SIZE_MAX*sizeof(double))) == NULL ||
248       (donnees->n=malloc(SIZE_MAX*sizeof(double))) == NULL ||
249       (donnees->y=malloc(SIZE_MAX*sizeof(double))) == NULL ||
250       (donnees->yf=malloc(SIZE_MAX*sizeof(double))) == NULL)
251       {
252       MessagePopup("Error","Not enough memory");
253       // message d'erreur si mémoire insufisante
254       }
255       else
256       {
257         donnees->size = load();
258
259         if(donnees->size != -1)
260         {
261           loadok = 1;
262
263           Time = donnees->time[1];
264           Time2 = donnees->time[1];
265
266           SetCtrlVal(panel,PANEL_TIME,Time);
267           SetCtrlVal(panel,PANEL_TIME_2,Time2);
268
```

```
269           SetCtrlVal(panel,PANEL_DUREE,donnees->time[SIZE_MAX-1]-donnees->time[1]);
270           SetCtrlVal(panel,PANEL_N_POINTS,(int)donnees->size);
271
272           Scale = (donnees->time[SIZE_MAX-1] - Time);
273           SetCtrlVal(panel,PANEL_TIMESCALE,Scale);
274           Scale2 = (donnees->time[SIZE_MAX-1] - Time2);
275           SetCtrlVal(panel,PANEL_TIMESCALE_2,Scale2);
276
277           PlotXY(panel,PANEL_GRAPH_3,donnees->time,donnees->data1,donnees->size, VAL_DOUBLE,
                  VAL_DOUBLE, VAL_THIN_LINE, VAL_NO_POINT,VAL_SOLID, 1, VAL_RED);
278
279           PlotXY(panel,PANEL_GRAPH_3,donnees->time,donnees->data2,donnees->size, VAL_DOUBLE,
                  VAL_DOUBLE, VAL_THIN_LINE, VAL_NO_POINT,VAL_SOLID, 1, VAL_BLUE);
280         }
281       }
282     }
283
284     SetCtrlAttribute(panel,PANEL_LOAD,ATTR_DIMMED,0);
285   }
286
287
288   return 0;
289 }
290
291 int load(void)
292 {
293   char line[1024];
294   int file;
295   int i=0;
296   long file_size=0,line_size=0;
297   int size=0;
298   double j=0,h=0,m=0,s=0,deltas;
299   int numread=100,retour = 0;
300   char fname[MAX_PATHNAME_LEN]={'\0'};
301
302   GetCtrlVal(panelHandle,PANEL_FNAME,fname);
303
304   GetFileInfo (fname,&file_size);
305
306   if(file_size > 0)
307   {
308     file = OpenFile (fname, VAL_READ_ONLY, VAL_OPEN_AS_IS, VAL_ASCII);
309
310     while( i < SIZE_MAX)
311     {
```

```
312
313
314
315     if(file_size > 0)
316     {
317       ReadLine(file,line,1020);
318
319       line_size = StringLength(line) + StringLength(" \n");
320       numread = 0;
321
322       retour = Scan(line,"%fj%fh%fm%fs %f %f %f %f %f %f %f %f %f %f %f %f %f %d",&j,&h,&m
              ,&s,&(donnees->data1[i]), &(donnees->data2[i]), &(donnees->data3[i]), &(donnees->
              data4[i]), &(donnees->gain1[i]), &(donnees->gain2[i]), &(donnees->data5[i]), &(
              donnees->data6[i]), &(donnees->data7[i]), &(donnees->data8[i]), &(donnees->data9[
              i]), &(donnees->data10[i]), &(donnees->data11[i]), &numread);
323
324       donnees->time[i] = j * 24 + h + m / 60.0 + s / 3600.0;
325
326       donnees->data1[i] = val(0 ,donnees->data1[i],donnees->gain1[i]);
327       donnees->data2[i] = val(1 ,donnees->data2[i],donnees->gain2[i]);
328       donnees->data3[i] = val(2 ,donnees->data3[i],donnees->gain1[i]);
329       donnees->data4[i] = val(3 ,donnees->data4[i],donnees->gain2[i]);
330       donnees->n[i] = (double)numread;
331
332       size++;
333       file_size = file_size - line_size;
334     }
335
336     if(file_size <= 0 || numread == 0)
337     {
338       donnees->time[i]=donnees->time[i-1];
339       donnees->data1[i]=donnees->data1[i-1];
340       donnees->data2[i]=donnees->data2[i-1];
341       donnees->data3[i]=donnees->data3[i-1];
342       donnees->data4[i]=donnees->data4[i-1];
343       donnees->data5[i]=donnees->data5[i-1];
344       donnees->data6[i]=donnees->data6[i-1];
345       donnees->data7[i]=donnees->data7[i-1];
346       donnees->data8[i]=donnees->data8[i-1];
347       donnees->data9[i]=donnees->data9[i-1];
348       donnees->data10[i]=donnees->data10[i-1];
349       donnees->data11[i]=donnees->data11[i-1];
350       donnees->n[i] = 0;
351     }
352
```

```
353       i++;
354
355     }
356
357     CloseFile(file);
358   }
359   else size = -1;
360
361   return size;
362
363 }
364
365 int CVICALLBACK TimeCallback (int panel, int control, int event,
366     void *callbackData, int eventData1, int eventData2)
367 {
368
369   if( event==EVENT_COMMIT && loadok)
370   {
371      GetCtrlVal(panel,PANEL_TIME,&Time);
372      GetCtrlVal(panel,PANEL_TIMESCALE,&Scale);
373
374     if(Time < donnees->time[0])
375     {
376       Time = donnees->time[0];
377
378       SetCtrlVal(panel,PANEL_TIME,Time);
379
380       MessagePopup("Error","Scale invalid");
381     }
382
383
384     if((Time + Scale) > donnees->time[SIZE_MAX-1] )
385     {
386
387       if(Time > donnees->time[SIZE_MAX-1])
388       {
389         Time = donnees->time[0];
390
391         SetCtrlVal(panel,PANEL_TIME,Time);
392       }
393
394       Scale = (donnees->time[SIZE_MAX-1] - Time);
395
396
397       SetCtrlVal(panel,PANEL_TIMESCALE,Scale);
```

```
398
399       MessagePopup("Error","Scale invalid");
400     }
401
402     if( Time < Time + Scale) SetAxisScalingMode (panel, PANEL_GRAPH, VAL_BOTTOM_XAXIS,
            VAL_MANUAL, Time, Time + Scale);
403
404     GetCtrlVal(panel,PANEL_TIME_2,&Time2);
405      GetCtrlVal(panel,PANEL_TIMESCALE_2,&Scale2);
406
407     if(Time2 < donnees->time[0])
408     {
409       Time2 = donnees->time[0];
410
411       SetCtrlVal(panel,PANEL_TIME_2,Time2);
412
413       MessagePopup("Error","Scale invalid");
414     }
415
416
417     if((Time2 + Scale2) > donnees->time[SIZE_MAX-1] )
418     {
419
420       if(Time2 > donnees->time[SIZE_MAX-1])
421       {
422         Time2 = donnees->time[0];
423
424         SetCtrlVal(panel,PANEL_TIME_2,Time2);
425       }
426
427       Scale2 = (donnees->time[SIZE_MAX-1] - Time2);
428
429
430       SetCtrlVal(panel,PANEL_TIMESCALE_2,Scale2);
431
432       MessagePopup("Error","Scale invalid");
433     }
434
435     if( Time2 < Time2 + Scale2) SetAxisScalingMode (panel, PANEL_GRAPH_4, VAL_BOTTOM_XAXIS,
            VAL_MANUAL, Time2, Time2 + Scale2);
436
437   }
438   return 0;
439 }
440
```

```
441
442 int CVICALLBACK ScaleCallback (int panel, int control, int event,
443     void *callbackData, int eventData1, int eventData2)
444 {
445   double min =0, max = 0;
446   int sm;
447
448   if( event==EVENT_COMMIT && loadok)
449   {
450
451     if(control == PANEL_AUTO)
452     {
453       SetAxisScalingMode (panel, PANEL_GRAPH, VAL_LEFT_YAXIS, VAL_AUTOSCALE, 0.0, 0.0);
454       GetAxisScalingMode (panel, PANEL_GRAPH, VAL_LEFT_YAXIS, &sm, &min, &max);
455       SetCtrlVal(panel,PANEL_MAX_GRAPH,max);
456       SetCtrlVal(panel,PANEL_MIN_GRAPH,min);
457     }
458     else if( control == PANEL_MAX_GRAPH || control == PANEL_MIN_GRAPH )
459     {
460       GetCtrlVal(panel,PANEL_MAX_GRAPH,&max);
461       GetCtrlVal(panel,PANEL_MIN_GRAPH,&min);
462       if(min <= max) SetAxisScalingMode (panel, PANEL_GRAPH, VAL_LEFT_YAXIS, VAL_MANUAL, min,
              max);
463     }
464
465     if(control == PANEL_AUTO_2)
466     {
467       SetAxisScalingMode (panel, PANEL_GRAPH_4, VAL_LEFT_YAXIS, VAL_AUTOSCALE, 0.0, 0.0);
468       GetAxisScalingMode (panel, PANEL_GRAPH_4, VAL_LEFT_YAXIS, &sm, &min, &max);
469       SetCtrlVal(panel,PANEL_MAX_GRAPH_2,max);
470       SetCtrlVal(panel,PANEL_MIN_GRAPH_2,min);
471     }
472     else if( control == PANEL_MAX_GRAPH_2 || control == PANEL_MIN_GRAPH_2 )
473     {
474       GetCtrlVal(panel,PANEL_MAX_GRAPH_2,&max);
475       GetCtrlVal(panel,PANEL_MIN_GRAPH_2,&min);
476       if(min <= max) SetAxisScalingMode (panel, PANEL_GRAPH_4, VAL_LEFT_YAXIS, VAL_MANUAL,
             min, max);
477     }
478   }
479   return 0;
480 }
481
482 int CVICALLBACK ConfCallback (int panel, int control, int event,
483     void *callbackData, int eventData1, int eventData2)
```

```
484 {
485   char pathname[MAX_PATHNAME_LEN]={'\0'};
486 
487   if( event==EVENT_COMMIT )
488   {
489     ProcessDrawEvents();
490 
491     FileSelectPopup ("", "photo.conf", "*.conf", "", VAL_LOAD_BUTTON, 0, 0, 0, 1, pathname);
492     SetCtrlVal(panel,PANEL_CONF_FILE,pathname);
493 
494     if(readConf() != -1 )
495     {
496       SetCtrlAttribute (panelHandle, PANEL_CONF_LOAD , ATTR_CMD_BUTTON_COLOR, VAL_GREEN);
497       SetCtrlAttribute(panel,PANEL_LOAD,ATTR_DIMMED,0);
498     }
499     else
500     {
501       SetCtrlAttribute (panelHandle, PANEL_CONF_LOAD , ATTR_CMD_BUTTON_COLOR, VAL_RED);
502       SetCtrlAttribute(panel,PANEL_LOAD,ATTR_DIMMED,1);
503     }
504 
505   }
506   return 0;
507 }
508 
509 
510 /// HIFN Fonction permettant de lire la configuration contenu dans un fichier texte
511 int readConf(void)
512 {
513 
514   int filehandle;
515   int  error=0,i=0,num=0,size=0;
516   char line[1024]={'\0'};
517   int temp1,temp2,temp3;
518   char file[MAX_PATHNAME_LEN]={'\0'};
519 
520   GetCtrlVal(panelHandle,PANEL_CONF_FILE,file);
521 
522   if(FileExists(file,&size) == 0 || FindPattern (file, 0, strlen(file), ".conf", 0, 1) ==
      -1)
523   {
524 
525     error = -1;
526   }
527   else
```

```
528   {
529     filehandle = OpenFile (file, VAL_READ_ONLY, VAL_OPEN_AS_IS, VAL_ASCII);
530
531     for(i = 0; i <= 26 && error != -1; i++)
532     {
533       do
534       {
535         error = ReadLine(filehandle,line,1020);
536
537       }
538       while( (line[0] == '\0' || line[0] == '#' || line[0] == '\n' ) && error != -1);
539
540
541       if( error != -1 )
542       {
543         switch(i)
544         {
545
546           case 10: Scan(line, "%f x + %f",&(config->photo1F.a),&(config->photo1F.b)); break;
547           case 11: Scan(line, "%f x + %f",&(config->photo1f.a),&(config->photo1f.b)); break;
548
549           case 19: Scan(line, "%f x + %f",&(config->photo2F.a),&(config->photo2F.b)); break;
550           case 20: Scan(line, "%f x + %f",&(config->photo2f.a),&(config->photo2f.b)); break;
551
552           case 25: Scan(line, "%f x + %f",&(config->therm1.a),&(config->therm1.b));
553           break;
554
555           case 26: Scan(line, "%f x + %f",&(config->therm2.a),&(config->therm2.b));
556           break;
557
558           default : break;
559         }
560
561         if( NumFmtdBytes() <= 0 && ( i == 10 || i == 11 || i == 19 || i == 20 || i == 25 || i
                == 26)) error = -1;
562
563       }
564
565     }
566
567     CloseFile(filehandle);
568   }
569
570   return error;
571 }
```

```
572
573 /// HIFN Fonction permettant de calculer les valeurs numériques correspondantes aux
        acquisitions
574 double val(int channel ,double data, int gain)
575 {
576    double valeur=0;
577
578    switch(channel)
579    {
580      default : break;
581
582      case 0:
583
584        valeur = data;
585        if(gain) valeur = valeur * config->photo1f.a + config->photo1f.b ;
586        else valeur = valeur * config->photo1F.a + config->photo1F.b ;
587        break;
588
589      case 1:
590
591        valeur = data;
592        if(gain) valeur = valeur * config->photo2f.a + config->photo2f.b ;
593        else valeur = valeur * config->photo2F.a + config->photo2F.b ;
594        break;
595
596      case 2:
597
598        valeur = data * config->therm1.a + config->therm1.b ;
599        break;
600
601      case 3:
602
603        valeur = data * config->therm2.a + config->therm2.b ;
604        break;
605
606
607    }
608
609    return valeur;
610 }
611
612 int CVICALLBACK AffCallback (int panel, int control, int event,
613     void *callbackData, int eventData1, int eventData2)
614 {
615   int graph=0,color=VAL_BLACK;
```

```
616 double *data=NULL;
617 int *ref=NULL;
618 int n=0;
619
620 if( event==EVENT_COMMIT && loadok)
621 {
622   if(control == PANEL_LIST1 ) graph = PANEL_GRAPH;
623   else if(control == PANEL_LIST2 ) graph = PANEL_GRAPH_4;
624
625   GetCtrlVal(panel,control,&n);
626
627   switch(n)
628   {
629     default : break;
630
631     case 1 :
632
633       if(control == PANEL_LIST1 ) ref = &(aff->photo1);
634       else if(control == PANEL_LIST2 ) ref = &(aff->photo1_2);
635
636       data = donnees->data1;
637       color = VAL_RED;
638
639       break;
640
641     case 2 :
642
643       if(control == PANEL_LIST1 ) ref = &(aff->photo2);
644       else if(control == PANEL_LIST2 ) ref = &(aff->photo2_2);
645
646       data = donnees->data2;
647       color = VAL_BLUE;
648
649       break;
650
651     case 3 :
652
653       if(control == PANEL_LIST1 ) ref = &(aff->temp1);
654       else if(control == PANEL_LIST2 ) ref = &(aff->temp1_2);
655
656       data = donnees->data3;
657       color = VAL_YELLOW;
658
659       break;
660
```

```
661       case 4 :
662
663         if(control == PANEL_LIST1 ) ref = &(aff->temp2);
664         else if(control == PANEL_LIST2 ) ref = &(aff->temp2_2);
665
666         data = donnees->data4;
667         color = VAL_CYAN;
668
669         break;
670
671       case 5  :
672
673         if(control == PANEL_LIST1 ) ref = &(aff->dark1);
674         else if(control == PANEL_LIST2 ) ref = &(aff->dark1_2);
675
676         data = donnees->data5;
677         color = VAL_DK_RED;
678
679         break;
680
681       case 6 :
682
683         if(control == PANEL_LIST1 ) ref = &(aff->dark2);
684         else if(control == PANEL_LIST2 ) ref = &(aff->dark2_2);
685
686         data = donnees->data6;
687         color = VAL_DK_BLUE;
688
689         break;
690
691       case 7 :
692
693         if(control == PANEL_LIST1 ) ref = &(aff->sky1);
694         else if(control == PANEL_LIST2 ) ref = &(aff->sky1_2);
695
696         data = donnees->data7;
697         color = VAL_DK_RED;
698
699         break;
700
701       case 8  :
702
703         if(control == PANEL_LIST1 ) ref = &(aff->sky2);
704         else if(control == PANEL_LIST2 ) ref = &(aff->sky2_2);
705
```

```
706         data = donnees->data8;
707         color = VAL_DK_BLUE;
708 
709         break;
710 
711       case 9 :
712 
713         if(control == PANEL_LIST1 ) ref = &(aff->x);
714         else if(control == PANEL_LIST2 ) ref = &(aff->x_2);
715 
716         data = donnees->data9;
717         color = VAL_MAGENTA;
718 
719         break;
720 
721       case 10  :
722 
723         if(control == PANEL_LIST1 ) ref = &(aff->y);
724         else if(control == PANEL_LIST2 ) ref = &(aff->y_2);
725 
726         data = donnees->data10;
727         color = VAL_CYAN;
728 
729         break;
730 
731       case 11 :
732 
733         if(control == PANEL_LIST1 ) ref = &(aff->z);
734         else if(control == PANEL_LIST2 ) ref = &(aff->z_2);
735 
736         data = donnees->data11;
737         color = VAL_YELLOW;
738 
739         break;
740 
741       case 12 :
742 
743         if(control == PANEL_LIST1 ) ref = &(aff->n);
744         else if(control == PANEL_LIST2 ) ref = &(aff->n_2);
745 
746         data = donnees->n;
747         color = VAL_RED;
748 
749         break;
750 
```

```
751       case 13 :
752 
753         if(control == PANEL_LIST1 ) ref = &(aff->gain1);
754         else if(control == PANEL_LIST2 ) ref = &(aff->gain1_2);
755 
756         data = donnees->gain1;
757         color = VAL_RED;
758 
759         break;
760 
761       case 14 :
762 
763         if(control == PANEL_LIST1 ) ref = &(aff->gain2);
764         else if(control == PANEL_LIST2 ) ref = &(aff->gain2_2);
765 
766         data = donnees->gain2;
767         color = VAL_BLUE;
768 
769         break;
770 
771 
772     }
773 
774 
775     if(graph != 0 && data != NULL && ref != NULL && *ref == -1) *ref = PlotXY(panel,graph,
          donnees->time,data,donnees->size, VAL_DOUBLE, VAL_DOUBLE, VAL_THIN_LINE, VAL_NO_POINT
          ,VAL_SOLID, 1, color);
776     else if(ref != NULL && *ref != -1)
777     {
778       DeleteGraphPlot (panel,graph , *ref , VAL_IMMEDIATE_DRAW);
779       *ref = -1;
780     }
781 
782   }
783   return 0;
784 }
785 
786 int CVICALLBACK CursorCallback (int panel, int control, int event,
787     void *callbackData, int eventData1, int eventData2)
788 {
789   double x,y;
790   if( event==EVENT_COMMIT)
791   {
792     GetGraphCursor (panel,control , 1, &x, &y);
793 
```

```
794     switch(control)
795     {
796       default : break;
797
798       case PANEL_GRAPH_3 :
799
800         SetCtrlVal(panel,PANEL_TIME_C,x);
801         SetCtrlVal(panel,PANEL_VAL_C,y);
802
803         break;
804
805       case PANEL_GRAPH :
806
807         SetCtrlVal(panel,PANEL_TIME_C_2,x);
808         SetCtrlVal(panel,PANEL_VAL_C_2,y);
809
810         break;
811
812       case PANEL_GRAPH_4 :
813
814         SetCtrlVal(panel,PANEL_TIME_C_3,x);
815         SetCtrlVal(panel,PANEL_VAL_C_3,y);
816
817         break;
818     }
819
820   }
821   return 0;
822 }
823
824 int CVICALLBACK CalcCallback (int panel, int control, int event,
825     void *callbackData, int eventData1, int eventData2)
826 {
827   double j=0,h=0,m=0,s=0,heure=0;
828   if( event==EVENT_COMMIT)
829   {
830
831     if(control == PANEL_J || control == PANEL_H || control == PANEL_M || control == PANEL_S
          )
832     {
833       GetCtrlVal(panel,PANEL_J,&j);
834       GetCtrlVal(panel,PANEL_H,&h);
835       GetCtrlVal(panel,PANEL_M,&m);
836       GetCtrlVal(panel,PANEL_S,&s);
837
```

```
838       heure = j * 24 + h + m / 60 + s / 3600;
839
840       SetCtrlVal(panel,PANEL_HEURE,heure);
841     }
842     else if( control == PANEL_HEURE )
843     {
844       GetCtrlVal(panel,PANEL_HEURE,&heure);
845
846       j = (int)heure / 24;
847
848       h = (int)(heure - j*24);
849
850       m = (int)((heure - j*24 - h)*3600 / 60);
851
852       s = heure*3600 - j*3600*24 - h*3600 - m*60;
853
854       SetCtrlVal(panel,PANEL_J,j);
855       SetCtrlVal(panel,PANEL_H,h);
856       SetCtrlVal(panel,PANEL_M,m);
857       SetCtrlVal(panel,PANEL_S,s);
858     }
859   }
860   return 0;
861 }
```