NOTES TECHNIQUES

SCIENCES DE LA TERRE

GÉOLOGIE-GÉOPHYSIQUE

N° 6

1993

Modules informatiques de lecture et de représentation graphique des données sismologiques

Jean-Claude VERNUS

L'INSTITUT FRANÇAIS DE RECHERCHE SCIENTIFIQUE POUR LE DÉVELOPPEMENT EN COOPÉRATION

CENTRE DE NOUMÉA

ORSTOM

NOTES TECHNIQUES

SCIENCES DE LA TERRE

GÉOLOGIE-GÉOPHYSIQUE

N° 6

1993

Modules informatiques de lecture et de représentation graphique des données sismologiques

Jean-Claude VERNUS

L'INSTITUT FRANÇAIS DE RECHERCHE SCIENTIFIQUE
POUR LE DÉVELOPPEMENT EN COOPÉRATION

CENTRE DE NOUMÉA

/Vernus, J.C.

Modules informatiques de lecture et de représentation graphique des données sismologiques

Nouméa : ORSTOM. Juin 1993. 34 p.
Notes Tech. : Sci. Terre : Géol.-géophys. ; 6

Ø66SISMO
INFORMATIQUE SCIENTIFIQUE ; TRAITEMENT DE DONNÉES ; SEISMES ; SISMOLOGIE ; VANUATU ; / NOUVELLE CALEDONIE

Imprimé par le Centre ORSTOM
Juin 1993

ORSTOM Nouméa
REPROGRAPHIE

PLOT_IASPEI

OUTIL DE REPRESENTATION DE SEISMES

La chaîne de traitement des fichiers d'enregistrement des séismes permet la réalisation de catalogues mensuels de séismes. Ces catalogues permettent d'accéder rapidement à une représentation graphique des traces sismiques de chaque séisme suivant ses trois composantes et suivant la station sismologique.

Chaque mois, le contenu du répertoire */home_util/sismo* (au 15/01/93) de la station de travail *ouvéa*, sur lequel sont transférés les fichiers d'acquisition, est déplacé sous le répertoire *backup*, compressé puis sauvé sur bande.

La création de catalogues n'est pas l' unique intérêt de cette chaîne de traitement. Le sismologue désirant visualiser un séisme du répertoire d'acquisition, peut, par l'intermédiaire d'une interface utilisateur, représenter graphiquement à l'écran ou sur papier, les séismes de son choix.

1- Environnement

L'acquisition des séismes s'effectue depuis un PC. Les fichiers sont automatiquement transféré via PC-NFS sur une partition du disque de la station *ouvéa*. Cette partition déclarée sous le nom de */home_util/sismo* a une capacité de 150 Mo, ce qui permet de stocker une acquisition d'environ 1 mois.

Les outils de représentation des séismes utilisés, étaient jusqu'alors des outils PC (pceq, xplay), donc contraints par les propres capacités du PC.

L'équipe de sismologie a décidé de se doter d'outils utilisables dans un environnement UNIX, proposant une capacité de traitement supérieure ainsi qu'une plate-forme de développement plus performante. De plus, les données sismologiques sont directement stockées sur le disque de la station SUN Sparc SLC *Ouvéa* connectée au réseau du centre. Les données, de part leur zone de stockage, sont d'autant plus accessibles et ceci depuis n'importe qu'elle station ou terminal client du réseau.

2- Outils logiciels

2-1 Environnement

Comme bon nombre d'applications graphiques au sein du département géophysique , la représentation des traces sismiques a été réalisée à partir de l'environnement graphique UNIRAS.

La condition principale de développement était de produire des outils de traitement de fichier d'acquisition appelables depuis un contexte Fortran, permettant ainsi d'utiliser les routines graphiques UNIRAS.

La solution choisie, afin de respecter cette condition, consiste en un développement des modules de traitements en langage C, interfacés au module principal graphique écrit en Fortran. Ainsi, toute la bibliothèque graphique UNIRAS reste accessible.

2-2 Modules

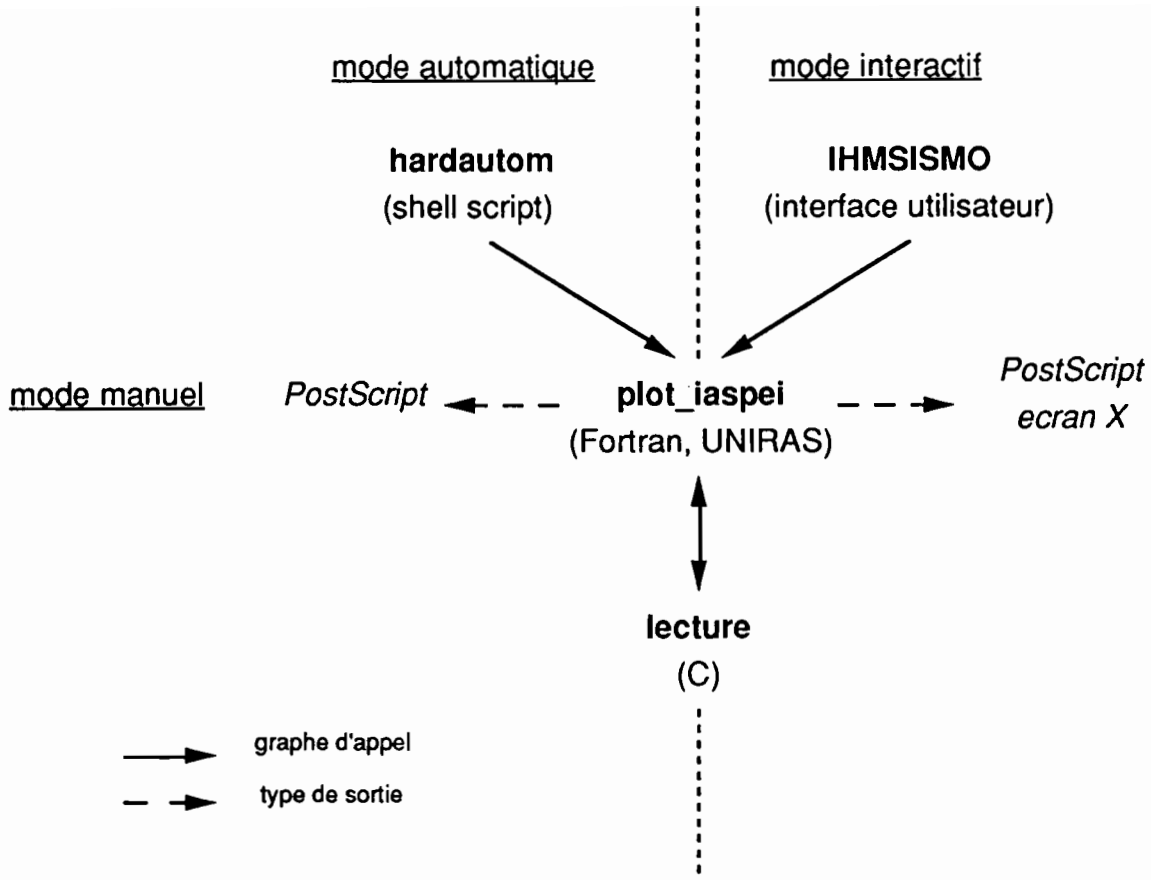
La chaîne de traitements des fichiers d'enregistrement des séismes est composée de 4 modules.

Ces modules constituent les 3 modes de traitements que propose la chaîne.

mode interactif : ce mode permet à l'opérateur, via une interface utilisateur développée à l'aide de *devguide* (générateur d'interface SUN), de représenter un séisme ou un ensemble de séismes associés, sous forme de liste créée par l'opérateur, dans un répertoire ou par intervalle de temps défini par l'utilisateur (cf pages suivantes : fenêtres principales de l'interface).

mode automatique : ce mode donne la possibilité à l'opérateur d'éditer de manière globale l'ensemble des fichiers contenus dans le répertoire courant. Associé à la commande Unix "at", cette édition peut-être différée dans le temps afin d'utiliser l'imprimante en dehors des périodes de surcharge.

mode manuel : ce mode est le plus basique des 3 modes. Il permet d'éditer directement un fichier suivant le mode voulu. Il suffit donc à l'utilisateur de préciser le type d'édition (PostScript ou Uniras) ainsi que le nom du fichier précédé de son chemin d'accès absolu ou relatif.



hardautom : (hardcopy automatique) Shell script permettant d'éditer séquentiellement au format PostScript, une liste de fichiers contenus dans le répertoire courant. L'utilisation de la commande "at" est la suivante.

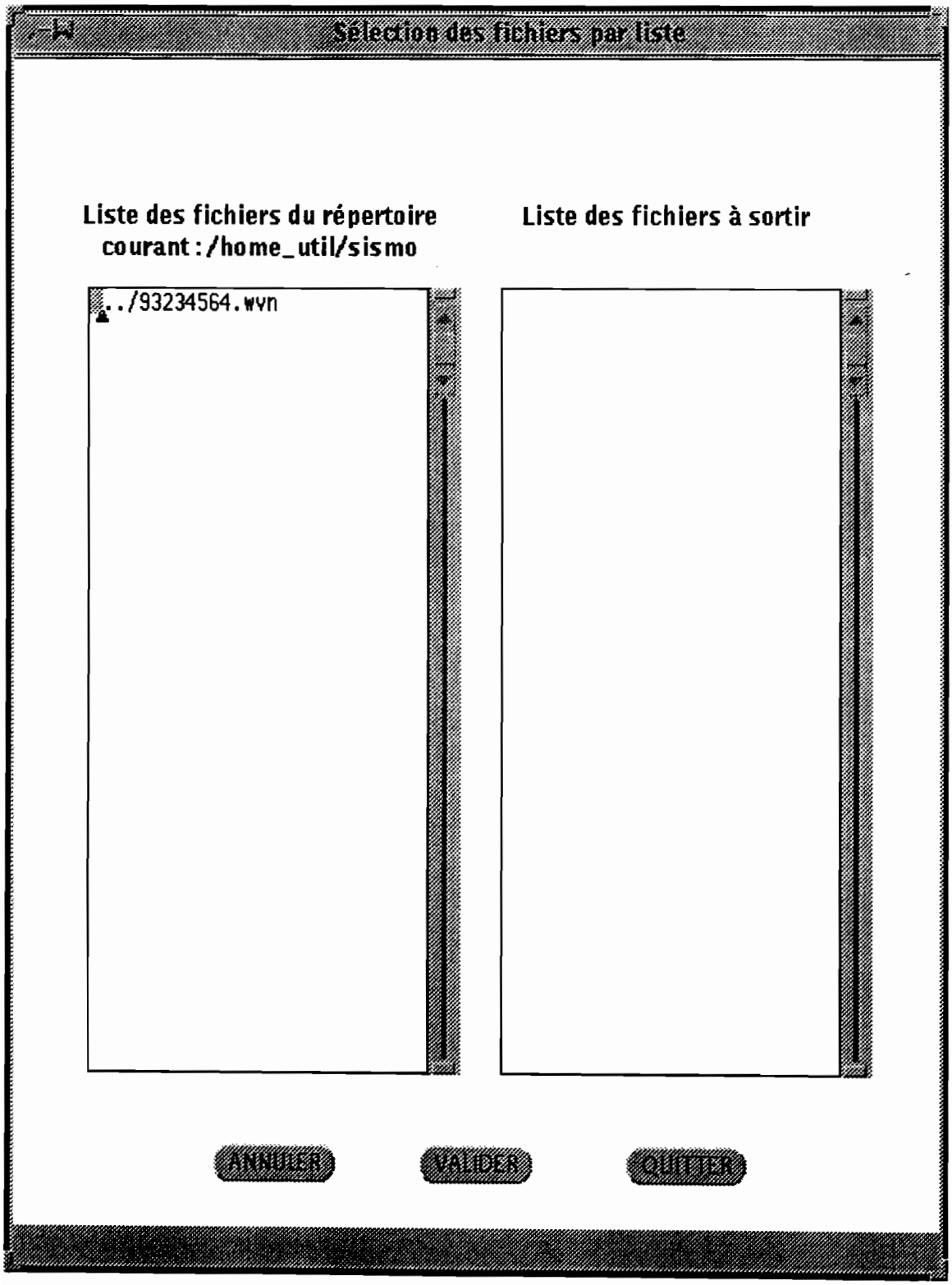
exemple : 'at -s 19:00 /home_util/sismo/hardcopy/hardautom'

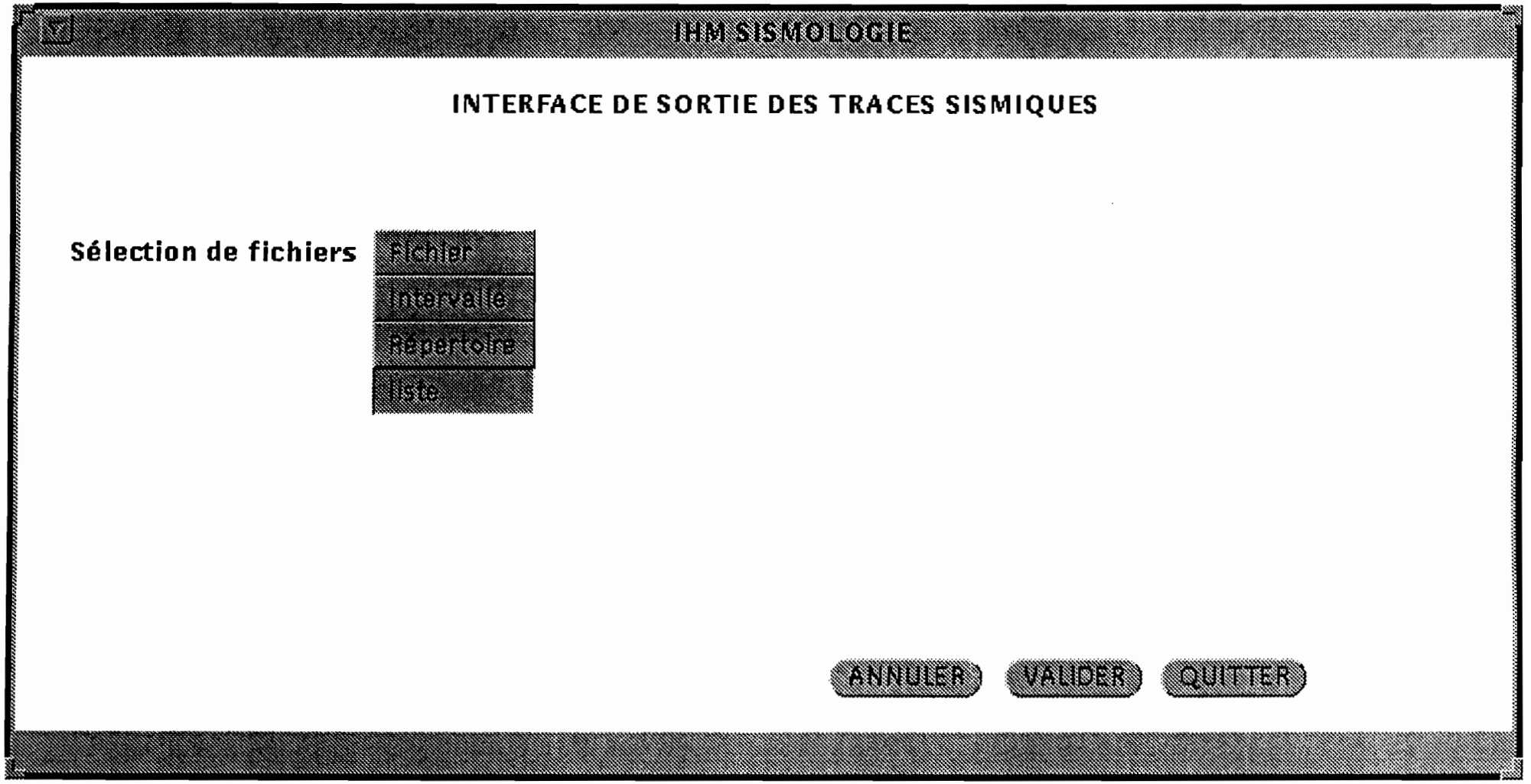
* "-s" signifie qu'il s'agit d'un shell script.

* "19:00" heure de déclenchement de la procédure.

* "/home_util/sismo/hardcopy/hardautom" il est nécessaire d'indiquer le chemin absolu du shell à exécuter.

```
for i in `ls .. /*.wvn`
do
   echo $i >> sismo.log
    /home_util /sismo /hardcopy /plot_iaspei POST $i >> sismo.log
   date >> sismo.log
   lpr -Psparc POST
   while :
   do
      x=`lpq -Psparc|wc -l`
      if test $x -gt 4
      then
         sleep 300
      else
         break
      fi
   done
done
```


Sélection des fichiers par liste
Liste des fichiers du répertoire courant : /home_util/sismo
../93234564.wvn
Liste des fichiers à sortir
ANNULER
VALIDER
QUITTER


IHM SISMOLOGIE
INTERFACE DE SORTIE DES TRACES SISMIQUES
Sélection de fichiers
Fichier
Intervalle
Répertoire
liste
ANNULER
VALIDER
QUITTER

plot_iaspei : cet exécutable représente le standard d'appel de la procédure *lecture*. Il suffit de suivre la démarche présentée dans le source de cet exécutable pour appeler la procédure *lecture* depuis un code Fortran.

Un fichier référence de déclaration appelé "*declaration.typ*" contient les déclarations minimums et nécessaires devant être faites pour appeler cette procédure.

lecture : cette procédure C représente le coeur du traitement des fichiers d'acquisition, en particulier :

- le transfert des données du format DOS au format Unix.
- le décodage de la trace temps.

2-3 Présentation algorithmique des modules

Ce chapitre n'a pas la prétention de commenter précisément la démarche suivie par les algorithmes, mais de présenter globalement les grandes parties de chaque code et d'y associer le graphe d'appel des procédures qui permettra au lecteur de visualiser le squelette de chaque programme.

Chaque source étant abondamment commenté, un éventuel intervenant ayant pris connaissance de la structure du programme ainsi que du graphe d'appel, pourra directement intervenir sur le code.

2-3-1 présentation algorithmique de *plot_iaspei*

Ce programme est écrit en Fortran afin d'accéder à la librairie graphique Uniras. Les principales instructions de ce code sont des appels à des routines graphiques permettant de représenter la trace temps et 15 traces sismiques. A chaque trace est associée la valeur maximum du signal et chacune d'entre elles est normalisée suivant l'association suivante (N° des traces)

3-4-5 , 6-7-8, 9-10-11, 12-13, 14-15-16

pseudo-algorithme

- déclaration des variables nécessaires au stockage des données du fichier d'enregistrement (cf document de présentation du format de donnée iaspei)
- récupération des arguments de la ligne de commande (type de sortie POST ou UNIRAS, nom du fichier à traiter)
- sélection du device Uniras en fonction du type de sortie
- initialisation du graphisme Uniras
- appel de la procédure **lecture**
- calcul de la valeur maxi du séisme pour chaque canal et retrait de l'offset égal à 2048 pour chaque donnée.
- initialisation des dimensions du device Uniras et définition de l'échelle
- affichage du nom du fichier courant
- affichage de labels
- affichage de la date précise et du *start time* de l'enregistrement (valeurs issues du décodage de la trace temps)
- calcul des valeurs maxi pour chaque association de traces
- normalisation des associations de traces
- affichage du nom de la station et de la valeur de séisme maxi
- affichage de l'échelle par fraction de 60 secondes
- affichage de la longueur de la trace temps
- test permettant de retourner au choix du device de sortie, ceci dans le cas ou le premier device choisi est l'écran.
- modification du nom du fichier par l'ajout de l'extension ".out" permettant à l'usager de savoir si le fichier a été traité.

graphe d'appel

Le graphe d'appel de la procédure *plot_iaspei* n'est pas présenté car la seule procédure appelée est la procédure *lecture* .

2-3-2 présentation algorithmique de *lecture*

La procédure *lecture* est la procédure principale d'accès aux données d'enregistrement suivant le format **iaspei** (cf pages suivantes). Il est utile de rappeler au lecteur que le stockage des données se fait suivant un enchaînement de blocs.

Waveform file

The waveform file is a binary file used to store digitized waveforms. A new waveform file is generated for each event. An example of a waveform filename is:

```
89031000.WVM
```

where 89 is the year (1989), 03 is the month (March), 10 is the day of the month, and 00 is a base 36 number (i.e., 00, 01, ..., 0A, 0B, ..., 0Z, 10, ..., 1Z, ..., ZZ). The first two letters of the filename extension is 'WV' for waveform, and the last letter is the agency code.

A waveform file has a file header and multiple waveform blocks:

(1) File Header

The file header consists of a number of fields, and is padded to 1024 bytes as described below. These fields describe the structure of the file and provide information specific to the event. Most values in these fields come directly from the input control file.

(a) **magic_number:** a 2-byte signed integer. Magic number is always 256 (decimal) and is used by application programs (e.g., XPLAY) to determine whether this file is a waveform file generated by MDETECT or XDETECT.

(b) **channel_length** is a 2-byte unsigned integer, and specifies the length (in 2-byte words) of each digitized channel in a waveform block.

(c) **scan_count** is a 2-byte signed integer, and specifies the number of channels in a waveform block.

(d) **srate** is a 4-byte signed floating point number, and

specifies the channel digitization rate (in Hz or samples per second).

(e) **channel_list** is a fixed array of 16 channel numbers. This list is used by MDETECT to determine the order of digitization.

(f) **gain_list** is a fixed array of 16 gain numbers. This list is used by MDETECT to determine the gain for each channel.

(g) **channel_stat_list** is a fixed array of 32 integers and is used by MDETECT to determine the trigger status of a particular channel. The channel status list is indexed by channel number.

(h) **station_list** is a fixed array of 16 station_info records. Each station info record consists of five fields:

1. station_id: a 10-byte character array. It is identical to the StationID parameter in the input control file.

2. channel: a 2-byte unsigned integer. It is identical to the Ch parameter in the input control file.

3. 3 fields of 4-bytes each (not used by MDETECT).

(i) **trigger_time** is the time at which the event was confirmed. It is stored in a special format that is unique to the Microsoft C run-time library. The format is represented by a record which has several fields:

1. **time: a 4-byte signed long integer. Time is given in seconds from 00:00:00 Greenwich mean time, January 1, 1970.**

2. **millitm: a 2-byte unsigned integer. Millitm is the number of milliseconds.**

3. **timezone: a 2-byte signed integer.**

4. dstflag: a 2-byte signed integer.

Fields 3-4 are not currently being used.

(j) **channel_start_time** is the time at which the digitized waveform data begins. Because a certain amount of pre-trigger data is saved, this time usually precedes the trigger_time. It is also stored in the special format described above.

(k) **cal_channel_start_time** is a fixed array of 10 characters. This field is not used by MDETECT. It is included here to remain compatible with other analysis programs that actually calculate a calibrated channel start time based on the digitized IRIG-E time code.

(2) Waveform Block

The Data Translation DT2821 or DT2821 A/D board uses one A/D converter and a multiplexer to digitize 16 channels. ATLAB, a software package that controls the A/D board, sets up and maintains a circular queue of buffers in the extended memory. During data acquisition, the A/D board with the help of the PC/AT's DMA controller fills one buffer at a time. After a buffer is filled, MDETECT uses an ATLAB routine to copy the buffer to a local area in the base memory space which is accessible to MDETECT. This buffer is then demultiplexed, and if an event has been detected, it is written to the hard disk as a **waveform block**. Most of the waveform file, therefore, consists of waveform blocks. Each waveform block consists of **scan_count** number of digitized channels, each of which has **channel_length** number of 2-byte integer words.

Chaque bloc est une fraction des n canaux. Ces blocs sont de taille identique et le nombre de canaux est précisé dans l'entête du fichier d'enregistrement. Chaque canal d'un bloc est une partie d'une trace sismique. En lisant chaque bloc, on reconstitue l'intégralité des canaux et de ce fait de toutes les traces sismiques.

Seul le nombre de blocs est inconnu à l'origine, il est donc nécessaire de déclarer, dans le programme principal *plot_iaspei*, un tableau surdimensionné permettant d'y stocker les données en sortie de la procédure *lecture*. Pour une raison de compatibilité entre la gestion de la mémoire par les langages C et Fortran, on préférera dans la partie C (*lecture*), une lecture bloc par bloc, entraînant une allocation dynamique de la place nécessaire pour le stockage des données. Cette zone de mémoire allouée au fur et à mesure constitue une liste chaînée des différents blocs. Il suffira par la suite de faire correspondre les deux zones mémoires, la zone statique Fortran et la zone dynamique C, et de désallouer cette dernière à la fin des traitements des données.

Un certain nombre de traitements est lancé au fur et à mesure de la lecture des données. Chacun de ces traitements est contenu dans une procédure spécifique mais n'est pas appelé depuis le même niveau de profondeur. Le graphe d'appel présenté ci-après permettra de visualiser les différents niveau d'appel de chaque procédure ainsi que leur utilité.

pseudo-algorithme de *lecture*

- ouverture du fichier de données courant
- lecture des différents paramètres constituant l'entête du fichier de données
- création d'une chaîne de caractères contenant l'année courante à partir de la date contenue dans le nom du fichier de données
- l'entête de chaque fichier de données mesure 1024 octets. Le début de stockage des données commence à la suite de cette entête. Il est donc nécessaire de compléter la lecture à 1024 octets quand la place utilisée par les données d'entête est inférieure à la place disponible.
- affichage de l'entête en suivant la présentation du logiciel PC *xplay*
- allocation dynamique des canaux du premier bloc de la liste chaînée
- lecture du bloc
- test d'existence d'un bloc suivant

- allocation du prochain bloc
- allocation des canaux du prochain bloc
- chaînage du bloc courant à la liste
- appel de la procédure de traitement de la trace temps
- affectation de la variable contenant la date affichée sur la sortie graphique
- affectation du tableau *trace* . Ce tableau étant déclaré dans la procédure Fortran appelant ce module, il est nécessaire de faire la correspondance entre la zone mémoire réservée à cet effet par le Fortran et la zone mémoire affectée par la procédure C courante. Le tableau *trace* étant passé par adresse, l'affectation de celui-ci se fait en référençant chaque case mémoire par rapport à l'adresse du premier élément du tableau. La variable *trace* (nom du tableau) est l'adresse de ce premier élément.
- désallocation de la liste chaînée constituée des différents blocs

graphe d'appel

LECTURE

permute2

permuteint4

procédures d'intervertion des octets pour transformer les données du format Dos au format Unix

traducheure

transformation de date, passage de million de secondes au format jour:mois:année heure:minute:seconde

ALLOCAT

macro d'allocation dynamique de zone mémoire (allocation du premier bloc)

read_waveform

lecture des blocs de données (waveform)

ALLOCAT

allocation du bloc suivant

ALLOCAT

allocation des canaux associés au bloc

cre_trace_temps

mise au format de la trace temps , création d'un tableau contenant les valeurs binaires suivantes : 0 top seconde court
1 top seconde large

calcul_param_signif

calcul de la valeur moyenne et de la valeur maxi de chaque canal

calcul_start_time

calcule le start time à partir de la trace temps créée

date_to_seconde

calcul du nombre de secondes écoulées depuis le 01/01/1970 jusqu'à la date donnée

cre_date_graphique

permet d'insérer la partie décimale en milliseconde à la chaîne de caractères représentant la date du start_time

calcul_time

calcul du trigger_time ainsi que de la partie décimale exprimée en milliseconde du start_time et du trigger_time

rewrite_fich

procédure commentée car utilisée uniquement dans le cas d'une modification de la date contenue dans l'entête à partir de la date issue du décodage de la trace temps

calcul_time *

permuteint4 *

permute2 *

read_write_waveform

lecture et réécriture de tous les waveform blocks

* = déjà commenté

Pour une plus grande compréhension de chacune de ces routines, il suffit de se reporter aux pseudo-algorithmes ou aux sources commentés de manière détaillée.

pseudo-algorithmes des routines internes à la procédure **lecture**

read_waveform

- ouverture de boucle sur tous les canaux
- lecture d'un buffer de 1024 octets représentant la longueur de la fraction de chaque canal contenu dans un bloc
- intervertion DOS-> Unix de chaque short constituant le buffer pré-cité

cre_trace_temps

- allocation du tableau qui contiendra la trace temps formatée comme suit : 0 = top seconde court
  1 = top seconde large
- détermination du type de valeur
  valeur haute = valeur de tension d'un top seconde
  valeur basse = valeur de tension entre 2 tops seconde
- calcul du nombre de valeurs successives de haute tension afin de déterminer s'il s'agit d'un top seconde court ou d'un top seconde large.
- affectation du tableau *tab_trace_temps*
  les échelons de valeurs de tension basse et haute sont ramenés à un seul type de valeur : 0 = top seconde court
  1 = top seconde large
  la taille du tableau est donc égal au nombre de secondes de l'enregistrement
- incrémentation du compteur de seconde
- calcul des valeurs moyennes et maxi de chaque trace

- **calcul_param_signif**

- calcul de la valeur maxi de trace sismique associée à un canal
- test de supériorité de la valeur courante par rapport au maxi courant
- calcul de la valeur moyenne de chaque canal
- calcul de la fréquence moyenne
- calcul du start time à partir du décodage de la trace temps

- **calcul_start_time**

- test de détermination du top minute : si la position courante + 38 secondes est supérieure à la durée de l'enregistrement, le dernier top large sélectionné est le top minute
- recherche du prochain top large à partir du top large courant
- test de situation : le top large suivant est-il séparé au maximum de 37 secondes du top large courant ?
- lorsque le top minute est déterminé, début de décodage du start time de la trace temps
- calcul du jour de l'année sur 365
- calcul de l'heure
- calcul de la minute
- affichage de la trace temps sous la forme de 0 pour les tops courts et de 1 pour les tops longs (partie commentée : utile pour deboggage)
- calcul de la minute : la minute de début d'enregistrement correspond à la minute précédente et au nombre de secondes complément du nombre trouvé
- transformation de la date : nombre de jours écoulés depuis le début de l'année , heure, minute, seconde en nombre de secondes écoulées depuis le 01/01/1970 à 00:00:00

date_to_seconde

- recherche du nombre d'années bissextiles depuis le 01/01/1970
- calcul du nombre d'années non-bissextiles
- transformation du nombre d'années en seconde

- ajout de la date du fichier par rapport au début de l'année en cours
- mise à l'heure par rapport à l'heure locale
- ajout du décalage horaire
- transformation de seconde en date normale (uniquement pour déboggage)

cre_date_graphique

- isolement de la date sans l'année
- appel de la procédure *calcul_time* qui permet de récupérer la valeur de la fraction du start time contenant les millisecondes. Cette partie décimale est représentée en nombre entier
- transformation des millisecondes en chaîne de caractères
- concaténation des millisecondes sous forme de caractères
- rajout de la chaîne de caractères contenant l'année

calcul_time

- trigger time : conversion du nombre de millisecondes en valeur décimale ex : 912 -> 0,912
- trigger time : création de la variable réelle représentant les secondes et la précision en milliseconde
- idem pour le start time
- start time : conversion en milliseconde (valeur décimale)
- " création de la variable réelle
- calcul de la différence de temps entre le start time et le trigger time contenus dans l'entête du fichier
- calcul du trigger time à partir du start time issu du décodage de la trace temps et de la différence précédemment calculée
- conversion en entier du trigger time
- isolement de la partie réelle du trigger time
- mise au format suivant de la partie réelle : 0,912 -> 912
  ce format est imposé par l'entête de chaque fichier de données
- même démarche que précédemment avec la variable start time

rewrite_fich (uniquement en configuration de réencodage du temps)

- ouverture du fichier issu de la modification et du fichier d'origine
- lecture et écriture des différents paramètres constituant l'entête.

La variable compteur représente le nombre d'octets lus afin de compléter la lecture à 1024 octets correspondant à la fin de l'entête

- fin de lecture de l'entête
- lecture des *scan count* canaux des blocs

```
cc*****************************************************************************
cc                                                                           cc
      PLOT_IASPEI
cc
cc      Ce programme est un programme type d'utilisation de la procédure
cc "lecture" permettant de lire les fichiers de séisme issus de l'acquisition
cc des différentes stations sismologiques.
cc      Ce module utilise les déclarations nécessaires à l' appel de
cc "lecture" ainsi que certaines procédures UNIRAS permettant de visualiser,
cc pour un même séisme, les traces sismiques issues de chaque station du réseau
cc      Le fichier "declaration.typ" est le fichier type contenant les variables
cc indispensables à l'appel de la procure "lecture", depuis un module FORTRAN.
cc
cc      La première ligne commentés est l'ordre de compilation de ce fichier.
cc
cc*****************************************************************************
cc      cc -g -c lecture.o lecture.c ; unilink plot_iaspei.f lecture.o
cc*****************************************************************************
      program plot_iaspei

      CHARACTER*100 filename
      CHARACTER*30 DATE, sortie, label
      CHARACTER*50 commande
      CHARACTER*1 answer
cc
      INTEGER*2       NB_BLOCK, NB_SECONDE, LONGUEUR
      INTEGER*2       TRACE( 40000, 16), isoarr(10)
      REAL            maxi(16), TRIGGER, smax(5)
      REAL t(40*512), s(40*512)
cc
cc LES DIFFERENTES STRUCTURES DECLAREES CI-DESSOUS REPRESENTENT LES ZONES DE
cc STOCKAGE DES DONNEES LUES DANS UN FICHIER STANDARD D'ACQUISITION.
cc LES DIFFERENTS CHAMPS DE CES STRUCTURES SUIVENT RIGOUREUSEMENT LE FORMAT
cc SPECIFIE PAR IASPEI (cf DOCUMENT DE SPECIFICATION DE FORMAT D'ENREGISTREMENT)

cc
      STRUCTURE /STATION_STRUCT/
            CHARACTER*11    STATION_ID
            INTEGER*2       CHANNEL
            INTEGER*4       FIELD1
            INTEGER*4       FIELD2
            INTEGER*4       FIELD3
      END STRUCTURE
cc
      RECORD /STATION_STRUCT/STATION(16)
cc
      STRUCTURE /TIME_STRUCT/
            INTEGER*4       TIME
            INTEGER*2       MILLITM
            INTEGER*2       TIMEZONE
            INTEGER*2       DSTFLAG
      END STRUCTURE
cc
      RECORD /TIME_STRUCT/START_TIME, TRIGGER_TIME
cc
      STRUCTURE /ENTETE_STRUCT/
            INTEGER*2       MAGIC_NUMBER
            INTEGER*2       CHANNEL_LENGTH
            INTEGER*2       SCAN_COUNT
            REAL            SRATE
            INTEGER*2       CHANNEL_LIST(16)
            INTEGER*2       GAIN_LIST(16)
            CHARACTER*11    CHANNEL_STAT_LIST(32)
            CHARACTER*11    CAL_CHANNEL_STAT_LIST
      END STRUCTURE

      RECORD /ENTETE_STRUCT/ENTETE

      passage = 0
cc
cc RECUPERATION DES ARGUMENTS DE LA LIGNE DE COMMANDE
cc

      call getarg(1, sortie)
      call getarg(2, filename)


cc
cc SELECTION DE LA SORTIE
cc
```



```
20    if (sortie .eq. 'POST') then
            call groute('sel wpost;exit')
      else
            call groute('sel mwii;exit ')
      endif
cc
cc INITIALISATION DU GRAPHISME UNIRAS
cc
      call gopen

      if (passage.eq.0) then
cc          call getarg(1, sortie)
cc          call getarg(2, filename)


            write(*,*)'sortie = ', sortie
            write(*,*)'filename = ', filename

cc
cc APPEL DE LA PROCEDURE DE LECTURE DU FICHIER CONTENANT LES TRACES
cc SISMIQUES ET DE DECODAGE DE LA TRACE TEMPS.
cc
            LONGUEUR = INDEX(filename,' ') -1
            if (LONGUEUR.gt.13) then
                  label = filename( (LONGUEUR-11):LONGUEUR)
            else
                  label = filename
            endif

            CALL lecture(filename, LONGUEUR,ENTETE, STATION,START_TIME,
     1                  TRIGGER_TIME,TRACE,NB_BLOCK,DATE,NB_SECONDE )


            npoint = NB_BLOCK * ENTETE.CHANNEL_LENGTH

cc
cc CALCUL DE LA VALEUR DE SEISME MAXI POUR CHAQUE CANAL
cc RETRAIT DE L'OFFSET DE 2048 POUR CHAQUE VALEUR
cc
            do i=1,16
                  do j=1,npoint
                  if( maxi(i).lt.abs(trace(j,i) - 2048) )
     1                  maxi(i)=real(abs(trace(j,i) - 2048))
                  enddo
            end do
cc
cc INITIALISATION DES DIMENSIONS DU DEVICE UNIRAS ET DEFINITION DE L'ECHELLE
cc

            xlenght = npoint / ENTETE.SRATE


            if (xlenght.gt.120) then
                  xlenght = 180
            else
                  xlenght = 120
            endif

      endif

      call grpsiz(xsiz,ysiz)
      call gvport(20.,25.,xsiz-40.,ysiz-40.)

      call glimit(0.,xlenght,0.,34.,0.,0.)

cc
cc AFFICHAGE DU NOM DU FICHIER COURANT
cc
      call rtxfon('swil', 1)
      call rtxlan('FREN')
      call rtxhei(4.)
      call rtx(-1, label, 10., ysiz-10.)
cc
cc AFFICHAGE DE LABELS
cc
cc    call rtx(-1, 'Sismologie - Nouméa', 5., ysiz-10.)
cc
cc AFFICHAGE DE LA DATE PRECISE ET DU START_TIME DE L'ENREGISTREMENT
cc
      call rtxhei(3.)
```

```
        call rtx(-1, date, 80., ysiz-10.)

        call rtx(-1, 'lat : ', xsiz/2.-len(label)/2.+70., ysiz-3.)
        call rtx(-1, 'long : ', xsiz/2.-len(label)/2.+70., ysiz-8.)
        call rtx(-1, 'H : ', xsiz/2.-len(label)/2.+70., ysiz-13.)
        call rtx(-1, 'magnitude : ', xsiz/2.-len(label)/2.+105., ysiz-4.)


        call gwbox( ( (xsiz-40.)/(ysiz-40.) ),1.,0.)
        call gscale

cc
ccCALCUL DES MAXIS POUR LES "TRIPLETS"(3,4,5)(6,7,8)(9,10,11)(12,13)(14,15,16)
cc
        do i=1,5
                smax(i)=0.0
                smax(i)=maxi(i*3)
                if ((i*3).eq.12) then
                        if (smax(i).lt.maxi((i*3)+1)) smax(i)=maxi((i*3)+1)
                else if ((i*3).eq.15) then
                        if (smax(i).lt.maxi((i*3)-1)) smax(i)=maxi((i*3)-1)
                        if (smax(i).lt.maxi((i*3)+1)) smax(i)=maxi((i*3)+1)
                else
                        if (smax(i).lt.maxi((i*3)+1)) smax(i)=maxi((i*3)+1)
                        if (smax(i).lt.maxi((i*3)+2)) smax(i)=maxi((i*3)+2)
                endif
        end do

cc
cc CONFIGURATION (3,4,5)(6,7,8)(9,10,11)(12,13,14)
cc
cc      do i=1,4
cc              smax(i)=0.0
cc              smax(i)=maxi(i*3)
cc              if (smax(i).lt.maxi((i*3)+1)) smax(i)=maxi((i*3)+1)
cc              if (smax(i).lt.maxi((i*3)+2)) smax(i)=maxi((i*3)+2)
cc      end do
cc
cc TRAITEMENT DES TRACES SISMIQUES DONT PREMIERE ET DERNIERE = TRACE TEMPS
cc
        do i=1,17
                if (i.eq.17) then
                        offset = 35. - 2 * i - 1.
                else
                        offset = 35. - 2.*i
                endif
cc
cc TRAITEMENT SPECIFIQUE POUR LES TRACES INDEPENDANTES : (c.a.d. CALCUL
cc DE LEUR PROPRE VALEUR MAXI POUR UNE NORMALISATION INDEPENDANTE. DANS
cc LA CONFIGURATION (3,4,5)(6,7,8)(9,10,11)(12,13,14), REDUIRE LE TEST
cc SUIVANT DE 16 A 14.
cc
                if(i.lt.3.or.i.gt.16) then
                        if(i.eq.17) then
                                do j=1,npoint
                                        s(j) = real(TRACE(j,1))-2048
                                end do
                        else
                                do j=1,npoint
                                        s(j) = real(TRACE(j,i))-2048
                                end do
                        endif
                        call norm(npoint, s, xmax, offset)
cc
cc TRAITEMENT DES "TRIPLETS" : NORMALISATION PAR RAPPORT A LA VALEUR MAXI
cc DES TROIS TRACES CONSTITUANT UN "TRIPLET".
cc
                else
                        do j=1,npoint
                                s(j) = real(TRACE(j,i))-2048
                        end do
cc
cc LE TEST ET LA PARTIE OUI DU TEST A SUPPRIMER DANS LE CAS DE CONFIGURATION
cc SUIVANT : (3,4,5)(6,7,8)(9,10,11)(12,13,14)
cc
                        if (i.ge.14.AND.i.le.16) then
                                call norm3(npoint, s, smax(5), offset)
                        else
                                call norm3(npoint, s, smax(i/3), offset)
                        endif
```



```
                endif
                call gvect(0., offset, 0)
                        do j= 1, npoint
                                t(j) = (j-1)/entete.srate
                        end do
                call gvect( t, s, npoint)
cc
cc AFFICHAGE    DU NOM DE LA STATION
cc              DE LA VALEUR DE SEISME MAXI
cc
                if(i.ne.1.AND.i.ne.17) then
                        call gscamm
                        y = 25. + offset * (ysiz-40.)/34.
                        x = 5.
                        call rtx(-1,STATION(i).STATION_ID,x,y)
                        call gscale
                        call rtxn2i(maxi(i),0,isoarr, ilen)
                        call rtxhei(2.)
                        call rtxi(isoarr,ilen, xlenght+3., offset)
                        call rtxhei(3.)
                endif
        end do
cc
cc AFFICHAGE DE L'ECHELLE PAR FRACTION DE 60 SECONDES
cc
        call rtxhei(2.)
        do i=1,int(xlenght/60.) +1
                call gvect((i-1)*60.,0.0,0)
                call gvect((i-1)*60.,-0.5,1)
                call rtxn((i-1)*60.,0,(i-1)*60.,-1.)
        end do
        call rtxc(-1, ' sec.')
cc
cc AFFICHAGE DE LA POSITION DU TRIGGER TIME
cc
        TRIGGER = real(TRIGGER_TIME.time - START_TIME.time)
        call gvect(TRIGGER, 0.0, 0)
        call gvect(TRIGGER, -0.5, 1)
        call rtx(-1, '(Trig.)', TRIGGER, -1.)
cc
cc AFFICHAGE DE LA LONGUEUR DE LA TRACE TEMPS
cc
        XNB_SECONDE = real(NB_SECONDE)
        call rtxn2i(XNB_SECONDE,0,isoarr, ilen)
        call rtxi(isoarr,ilen, XNB_SECONDE, -0.5)
        call rtxc(-1, ' sec.')
CC
CC BOUCLE PERMETTANT DE RETOURNER AU CHOIX DU DEVICE DE SORTIE
CC CECI DANS LE CAS OU LE PREMIER DEVICE CHOISI EST L'ECRAN
CC
        if (sortie.eq.'UNIRAS') then
                write(*,'(a,$)') ' hard copy (*y*/n) ? '
                read(*,'(a)') answer
                if(answer.eq. ' ' .or. answer.eq. 'y') then
                        sortie = 'POST'
                        passage = 1
                        go to 20
                endif
        endif

        if (sortie.eq.'POST') then
                call gclose
        endif

cc      if (sortie.eq.'POST') then
cc              call system('lpr -h -Peparc POST')
cc      endif
CC
CC CHAQUE FICHIER, UNE FOIS TRAITE, VOIT SON EXTENSION MODIFIEE DE
CC "wvn" EN "wvn.out"
CC
        if (sortie.eq.'POST') then
                commande='mv ../'//label(1:12)//' ../'//label(1:12)//'.out'
                call system(commande)
        endif

        stop
        end
CC_____________________________________________________________________
```

```
cc
cc                    SUBROUTINES DE NORMALISATION
cc
cc_____________________________________________________________________
cc ROUTINE DE NORMALISATION D'UNE TRACE INDEPENDANTE EN FONCTION DE
cc SON PROPRE MAXIMUM
cc
      subroutine norm(nx, x, xmax,offset)
      real x(1)

      xmax = 0.0
      do i=1,nx
            if(xmax.lt.abs(x(i)) ) xmax = abs(x(i))
      end do

      do i=1,nx
            x(i) = (x(i)/xmax) + offset
      end do
      return
      end
cc
cc ROUTINE DE NORMALISATION D'UNE TRACE EN FONCTION DE LA VALEUR MAXIMUM
cc DES TRACES FORMANT LE TRIPLET A NORMALISER
cc
      subroutine norm3(nx,x,xmax,offset)
      real x(1), y

      y = 0.
      do i=1,nx
            x(i) = (x(i)/xmax) + offset
      end do
      return
      end
```

```
      CHARACTER*100   filename
      CHARACTER*30    DATE
cc
      INTEGER*2       NB_BLOCK, NB_SECONDE, LONGUEUR
      INTEGER*2       TRACE( 40000, 16)
cc
      STRUCTURE /STATION_STRUCT/
            CHARACTER*11    STATION_ID
            INTEGER*2       CHANNEL
            INTEGER*4       FIELD1
            INTEGER*4       FIELD2
            INTEGER*4       FIELD3
      END STRUCTURE
cc
      RECORD /STATION_STRUCT/STATION(16)
cc
      STRUCTURE /TIME_STRUCT/
            INTEGER*4       TIME
            INTEGER*2       MILLITM
            INTEGER*2       TIMEZONE
            INTEGER*2       DSTFLAG
      END STRUCTURE
cc
      RECORD /TIME_STRUCT/START_TIME, TRIGGER_TIME
cc
      STRUCTURE /ENTETE_STRUCT/
            INTEGER*2       MAGIC_NUMBER
            INTEGER*2       CHANNEL_LENGTH
            INTEGER*2       SCAN_COUNT
            REAL            SRATE
            INTEGER*2       CHANNEL_LIST(16)
            INTEGER*2       GAIN_LIST(16)
            CHARACTER*11    CHANNEL_STAT_LIST(32)
            CHARACTER*11    CAL_CHANNEL_STAT_LIST
      END STRUCTURE

      RECORD /ENTETE_STRUCT/ENTETE


      CALL lecture(filename, LONGUEUR, ENTETE, STATION, START_TIME,
     1         TRIGGER_TIME, TRACE, NB_BLOCK, DATE, NB_SECONDE )
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/timeb.h>
#include <string.h>

#define ALLOCAT(a, b)    (b *)malloc( sizeof(b)*a)
#define NULL             2300            /* valeur intermed. top seconde*/
#define LARGEUR          15              /*largeur max top seconde court*/


/*************************************************************************
*               DECLARATION DES VARIABLES GLOBALES                       *
*************************************************************************/
int             fich, fin_fich ;

char            *bit ;

unsigned short  word;

static  char    date[30], date1[50], date2[50],
                chaine_annee_courante[5] ;

static  int             annee_courante ;
static  double          date_en_seconde ;

typedef struct  entete_struct{
                short   magic_number ;
                short   channel_length ;
                short   scan_count ;
                float   srate ;
                short   channel_list[16] ;
                short   gain_list[16] ;
                short   channel_stat_list[32] ;
                char    cal_channel_start_time[11]
}       entete_struct ;


typedef struct station_struct{
                char            station_id[11];
                short int       channel;
                int             field1;
                int             field2;
                int             field3;
}       station_struct;



typedef struct time_struct{
                int             time;
                short int       millitm;
                short int       timezone;
                short int       dstflag;
}       time_struct;


typedef struct block_struct{
                short                   **channel ;
                struct  block_struct    *pt_next_block ;

} block_struct ;

short nb_block ;

static  block_struct    *waveform_block, *pt_block_cur,
                        *pt_block_preced, *pt_desalloc,
                        *pt_next_desalloc ;

/*************************************************************************
*Nom :  permute2                                                         *
*                                                                        *
*Objet : permute les octets d'un short  ( DOS -> UNIX )                  *
*                                                                        *
*                                                                        *
*                                                                        *
*Entrees :      number          short *                                  *
*                                                                        *
```



```
*Sorties :      number          *                                        *
*                                                                        *
*                                                                        *
*Internes :                                                              *
*                                                                        *
*                                                                        *
*                                                                        *
*************************************************************************/

void
permute2(number)
        short int       *number;
{

        short int       octet0, octet1;

        octet0 = (*number & 0xff00) >> 8;
        octet1 = (*number & 0x00ff) << 8;
        *number = octet0 + octet1;

}
/*************************************************************************
*Nom :  permutefloat4                                                    *
*                                                                        *
*Objet :        permute les 4 octets d'un  float ( DOS -> UNIX )         *
*                                                                        *
*                                                                        *
*                                                                        *
*Entrees :      number          float *                                  *
*                                                                        *
*Sorties :      number            "                                      *
*                                                                        *
*                                                                        *
*Internes :                                                              *
*                                                                        *
*                                                                        *
*                                                                        *
*************************************************************************/

void
permutefloat4(number)
        float           *number;
{
        int     octet1, octet2, octet3, octet4;
        int     *pt_int ;

        pt_int = (int *) number ;
        octet1 = (*pt_int & 0xff000000) >> 24;
        octet2 = (*pt_int & 0x00ff0000) >> 8;
        octet3 = (*pt_int & 0x0000ff00) << 8;
        octet4 = (*pt_int & 0x000000ff) << 24;

        *pt_int =  octet1 + octet2 + octet3 + octet4  ;
}
/*************************************************************************
*Nom :  permuteint4                                                      *
*                                                                        *
*Objet :        permute les 4 octets d'un  float ( DOS -> UNIX )         *
*              AB CD EF GH -> GH EF CD AB                                *
*                                                                        *
*                                                                        *
*Entrees :      number          int                                      *
*                                                                        *
*Sorties :      number          int                                      *
*                                                                        *
*                                                                        *
*Internes :                                                              *
*                                                                        *
*                                                                        *
*                                                                        *
*************************************************************************/

void
permuteint4(number)
        int             *number;
{
        int     octet1, octet2, octet3, octet4;

        octet1 = (*number & 0xff000000) >> 24;
        octet2 = (*number & 0x00ff0000) >> 8;
```

```
        octet3 = (*number & 0x0000ff00) << 8;
        octet4 = (*number & 0x000000ff) << 24;

        *number =  octet1 + octet2 + octet3 + octet4  ;
}
/*****************************************************************************
*Nom :  permutefloat4                                                        *
*                                                                           *
*Objet :        permute les 4 octets d'un  float ( DOS -> UNIX )            *
*               AB CD EF GH -> GH EF CD AB                                  *
*                                                                           *
*                                                                           *
*Entrees :      number          float *                                     *
*                                                                           *
*Sorties :      number            *                                         *
*                                                                           *
*                                                                           *
*Internes :                                                                 *
*                                                                           *
*                                                                           *
*                                                                           *
*****************************************************************************/
void permute4bis(number)
        float   *number ;
{
        char    *p, *p1, *p2, *p3, *p4, *p5 ;
        float   resul ;

        p       = (char *) number ;
        p1      = (char *) p ;
        p2      = (char *) p+1 ;
        p3      = (char *) p+2 ;
        p4      = (char *) p+3 ;

        p5      = (char *) &resul ;
        *p5     = *p4 ;
        *(p5+1) = *p3 ;
        *(p5+2) = *p2 ;
        *(p5+3) = *p1 ;
}
/*****************************************************************************
*Nom :  traducheure_                                                         *
*                                                                           *
*Objet :        transforme le nombre de secondes ecoulees depuis le         *
*               1 Janvier 1970  en Fri Jul 24 11:27:16 1992                 *
*                                                                           *
*                                                                           *
*Entrees :      nb_seconde_init         int *                               *
*                                                                           *
*Sorties :      heure                   char *                              *
*                                                                           *
*                                                                           *
*Internes :                                                                 *
*                                                                           *
*                                                                           *
*                                                                           *
*****************************************************************************/
void traducheure_(nb_seconde_init, heure)
        long    *nb_seconde_init ;
        char    *heure ;
{

        strcpy(heure, ctime(nb_seconde_init)) ;

}
/*****************************************************************************
*Nom :  read_waveform                                                        *
*                                                                           *
*Objet :        lecture de tous les waveform blocks                         *
*                                                                           *
*                                                                           *
*                                                                           *
*Entree :       entete                                                      *
*                                                                           *
*Sorties :                                                                  *
*                                                                           *
*                                                                           *
*Intsrnes :     permute2                                                    *
*                                                                           *
*                                                                           *
```



```
*                                                                           *
*****************************************************************************/
int read_waveform(entete)
entete_struct   entete ;
{
        int             i, j, len1024=1024, len512=512 ,fin ;
        short int       pt_channel, *word_glob, word ;

/*
* BOUCLE SUR TOUS LES CANAUX
*/
        for( i=0 ; i< entete.scan_count ; i++ ){

                word_glob = (short *)malloc(sizeof(short)*len512) ;
/*
* LECTURE D'UN BUFFER DE 1024 OCTETS REPRESENTANT LA LONGUEUR DE LA
* FRACTION DE CHAQUE CANAL CONTENU DANS UN WAVEFORM BLOCK
*/
                fin = read(fich, word_glob, len1024);
                if ( fin == 0 )
                        return (fin) ;

                pt_channel = 0 ;
/*
* INTERVERTION DOS->UNIX DE CHAQUE SHORT CONSTITUANT LE BUFFER PRECITE
*/
                for( j=0 ; j<(entete.channel_length) ; j++ ){

                        word = word_glob[j] ;

                        permute2(&word) ;
                        *((pt_block_cur->channel[i]) +
                                pt_channel) = word ;
                        pt_channel++ ;
                }
                free(word_glob) ;

        }
        return(fin) ;

}

/*****************************************************************************
*Nom :  date_to_seconde                                                      *
*                                                                           *
*Objet :        calcule le nombre de secondes ecoulees depuis le 1          *
*               janvier 1970 a partir d'une date exprimee suivant le        *
*               format : 214 jours 22 heures 12 minutes                     *
*                                                                           *
*Entrees :      jour, heure, minute, nb_seconde                             *
*                                                                           *
*Sorties :      nb_seconde                                                  *
*                                                                           *
*                                                                           *
*Internes :     traducheure_                                                *
*                                                                           *
*                                                                           *
*                                                                           *
*****************************************************************************/
void date_to_seconde( jour, heure, minute, seconde )

        short   *jour, *heure, *minute ;
        float   *seconde ;
{
        short           nb_annee=0, annee_bisext=1972, nb_annee_bisext=0,
                        flag=0 ;
        long            temp ;
        struct timeb    tp ;
        float           decallage ;

        date_en_seconde = 0 ;
/*
* RECHERCHE DU NOMBRE D'ANNEES BISEXTILES DEPUIS LE 01/01/1970
*/
        do {
                if ( annee_courante == annee_bisext ) flag = 1 ;
                if ( annee_bisext > annee_courante) break ;
                nb_annee_bisext ++ ;
                annee_bisext += 4 ;
        }while(1) ;
```

```
        if (flag)
                nb_annee_bisext -= 1 ;
/*
* CALCUL DU NOMBRE D'ANNEES NON BISEXTILES
*/
        nb_annee = (annee_courante - nb_annee_bisext) - 1970 ;
/*
* TRANSFORMATION DU NOMBRE D'ANNEES EN SECONDE
*/
        date_en_seconde = ( nb_annee*365 + nb_annee_bisext*366)*
                24*3600 ;

/*
* AJOUT DE LA DATE DU FICHIER PAR RAPPORT AU DEBUT DE L'ANNEE EN COURS
*/
        date_en_seconde +=      (*jour - 1)*24*3600+
                        (*heure)*3600+
                        (*minute)*60+
                        *seconde ;

/*
* MISE A L'HEURE GMT PAR RAPPORT A L'HEURE DE NOUMEA
*/
        ftime(&tp) ;
        decallage = tp.timezone / 60 ;
/*
* COMPRENDRE QUI POURRA ! LE DECALLAGE ANNONCE ENTRE NOUMEA ET GREENWICH
* EST DE -11 HEURES. DE CE FAIT, ON AJOUTERA CI-DESSOUS, LE DECALLAGE
* AU LIEU DE LE RETRANCHER.
* LE CHAMP "timezone" DE LA STRUCTURE "tp" AFFECTEE PAR LA PROCEDURE
* "ftime" EST EXPRIME EN MINUTE, D'OU LE RAPPORT DE 60.
*/

        date_en_seconde = date_en_seconde + (decallage * 3600) ;

        temp = (long)date_en_seconde ;
/*
* TRANSFORMATION DE MILLION DE SECONDE EN DATE NORMALE
*/
        traduiheure_(&temp, date) ;
}

/*************************************************************************
*Nom :  calcul_start_time                                                *
*                                                                        *
*Objet :        calcul start_time a partir de la trace temps ores       *
*                                                                        *
*                                                                        *
*Entrees : tab_trace_temps : tableau a deux dimensions contenant le      *
*          type de largeur du top seconde et le numero de la valeur      *
*          nb_seconde : nb de tops seconde de la trace temps             *
*                                                                        *
*Sorties :      freq_moy        float                                   *
*                                                                        *
*                                                                        *
*Internes :     date_to_seconde                                         *
*                                                                        *
*                                                                        *
*                                                                        *
*************************************************************************/
void    calcul_start_time(nb_seconde, tab_trace_temps, freq_moy)
        short   *nb_seconde, *tab_trace_temps ;
        float   freq_moy ;
{
        short   i, j, top_court_ecoule=0, jour=0, heure=0, minute=0,
                premier= -1, top_min=0, top_large_our=0,
                top_large_suiv=0, debut_trace=0, compt=0, passage=-1,
                borne=0 ;


        float   seconde ;

/*
* SI LA POSITION COURANTE + 38 SECONDES (COMPLEMENT MINIMUM DE
* L'ENREGISTREMENT) EST SUPERIEUR AU NOMBRE DE SECONDE DE
* L'ENREGISTREMENT ET QU'AUCUN TOP MINUTE N'EST ENCORE DETERMINE
* C'EST QUE LE PREMIER TOP SECONDE LARGE TROUVE EST LE TOP MINUTE.
*/
        do{
```



```
                if ( ( compt + 38) >= *nb_seconde ){
                        top_min = premier ;
                        break ;
                }
                else{
/*
* RECHERCHE DU PROCHAIN TOP LARGE A PARTIR DU TOP LARGE COURANT
*/
                        if (passage == -1){
                                borne = 0 ;
                                passage = 0 ;
                        }else
                                borne = compt + 1 ;

                        for ( j = borne ; j<*nb_seconde ; j++){

/*
* LE TOP SECONDE COURANT EST-IL LARGE ?
*/
                                if( *(tab_trace_temps +j*2 ) == 1){
/*
* LE TOP SECONDE LARGE RENCONTRE EST-IL LE PREMIER TOP LARGE DE LA TRACE ?
*/
                                        if (premier != -1 ){
                                                top_large_suiv = j ;
                                                compt = j ;
                                                break ;
                                        }
                                        top_large_our = j ;
                                        top_min = j ;
                                        premier = j ;
                                        compt = j ;
                                }
                        }
/*
* TEST DE SITUATION : LE TOP LARGE SUIVANT EST-IL UN TOP MINUTE ?
* SEPARE AU MAXIMUM DE 37 SECONDES DU TOP LARGE COURANT
*/
                        if ((j==*nb_seconde)&&(top_large_our!=j))
                                break ;
                        if ( (top_large_suiv-top_large_our) < 37 ){
                                top_large_our = top_large_suiv ;
                                continue ;
                        }else{
                                if ( top_large_suiv - top_min <= 59) {
/*
* TEST DE SITUATION : LE TOP LARGE SUIVANT EST-IL UN TOP MINUTE ?
* DANS LE CAS OU L'ECART SOIT SUPERIEUR A 37 SECONDES, LE TOP MINUTE
* COURANT EST-IL SITUE A 1 MINUTE DU TOP LARGE
*/
                                        top_min = top_large_suiv ;
                                        top_large_our = top_large_suiv ;
                                        if (top_min + 24 < *nb_seconde )
                                                break ;
                                }else{
                                        break ;
                                }
                        }
                }

        }while( 1 ) ;


/*
* DEBUT DU DECODAGE DU START_TIME DE LA TRACE TEMPS
*
* CALCUL DU JOUR/365
*/
                debut_trace = *(tab_trace_temps +top_min*2+1 ) ;

                if( *(tab_trace_temps + top_min*2 +2 ) == 1 )
                        jour =  jour+200 ;

                if( *(tab_trace_temps + top_min*2+4 ) == 1 )
                        jour = jour+100 ;

                if( *(tab_trace_temps + top_min*2+6 ) == 1 )
                        jour = jour+80 ;
```

```
                if( *(tab_trace_temps + top_min*2+8 ) == 1 )
                        jour = jour+40 ;

                if( *(tab_trace_temps + top_min*2+10 ) == 1 )
                        jour = jour+20 ;


                if( *(tab_trace_temps + top_min*2+12 ) == 1 )
                        jour = jour+10 ;

                if( *(tab_trace_temps + top_min*2+14 ) == 1 )
                        jour = jour+8 ;

                if( *(tab_trace_temps + top_min*2+16 ) == 1 )
                        jour = jour+4 ;

                if( *(tab_trace_temps + top_min*2+18 ) == 1 )
                        jour = jour+2 ;

                if( *(tab_trace_temps + top_min*2+20 ) == 1 )
                        jour = jour+1 ;
/*
* CALCUL DE L'HEURE
*/

                if( *(tab_trace_temps + top_min*2+22 ) == 1 )
                        heure = heure+20 ;

                if( *(tab_trace_temps + top_min*2+24 ) == 1 )
                        heure = heure+10 ;

                if( *(tab_trace_temps + top_min*2+26 ) == 1 )
                        heure = heure+8 ;

                if( *(tab_trace_temps + top_min*2+28 ) == 1 )
                        heure = heure+4 ;

                if( *(tab_trace_temps + top_min*2+30 ) == 1 )
                        heure = heure+2 ;

                if( *(tab_trace_temps + top_min*2+32 ) == 1 )
                        heure = heure+1 ;

/*
* CALCUL DE LA MINUTE
*/

                if( *(tab_trace_temps + top_min*2+34 ) == 1 )
                        minute = minute+40 ;

                if( *(tab_trace_temps + top_min*2+36 ) == 1 )
                        minute = minute+20 ;

                if( *(tab_trace_temps + top_min*2+38 ) == 1 )
                        minute = minute+10 ;

                if( *(tab_trace_temps + top_min*2+40 ) == 1 )
                        minute = minute+8 ;

                if( *(tab_trace_temps + top_min*2+42 ) == 1 )
                        minute = minute+4 ;

                if( *(tab_trace_temps + top_min*2+44 ) == 1 )
                        minute = minute+2 ;

                if( *(tab_trace_temps + top_min*2+46 ) == 1 )
                        minute = minute+1 ;


        seconde = debut_trace / freq_moy ;
/*
* AFFICHAGE DE LA TRACE TEMPS SOUS LA FORME DE 0 POUR LES TOPS
* COURTS ET DE 1 POUR LES TOPS LARGES
*/

        /*
        for( i=0 ; i<*nb_seconde ; i++ ){
                if (*(tab_trace_temps + i*2) == 0){
                        printf("trace[%d] = \t%d\n",i,
                                *(tab_trace_temps + i*2 )) ;
```



```
                }else{
                        if (top_min == i){
                                printf("trace[%d] = \t\t%d\t",i,
                                        *(tab_trace_temps + i*2 )) ;
                                printf("<- top minute\n") ;
                        }else
                                printf("trace[%d] = \t\t%d\n",i,
                                        *(tab_trace_temps + i*2 )) ;
                }
        }

        printf("jour = %d\n", jour) ;
        printf("heure = %d\n", heure) ;
        printf("minute = %d\n", minute) ;
        printf("seconde = %f\n\n\n", seconde) ;
        */

/*
* LA DATE DE DEBUT D'ENREGISTREMENT CORRESPOND A LA MINUTE PRECEDENTE
* ET AU NOMBRE DE SECONDES COMPLEMENT DU NOMBRE TROUVE.
*/

        minute -- ;
        seconde = 60 - seconde ;


/*
* TRANSFORMATION DE LA DATE : NOMBRE DE JOURS ECOULES DEPUIS LE DEBUT
* DE L'ANNEE, HEURE, MINUTE, SECONDE -> NOMBRE DE SECONDES ECOULEES
* DEPUIS LE 01-01-70 A 00:00:00.
*/

        date_to_seconde(&jour, &heure, &minute, &seconde) ;

}

/*************************************************************************
*Nom :  calcul_param_signif                                              *
*                                                                        *
*Objet :  calcule la valeur moyenne des seismes de chaque canal  ainsi   *
*         que la valeur maxi par canal                                   *
*                                                                        *
*                                                                        *
*Entrees : tab_trace_temps : tableau a deux dimensions contenant le      *
*          type de largeur du top seconde et la numero de la valeur      *
*          nb_seconde : nb de tops seconde de la trace temps             *
*                                                                        *
*                                                                        *
*Sorties :                                                               *
*                                                                        *
*                                                                        *
*Internes :      calcul_start_time                                       *
*                                                                        *
*                                                                        *
*                                                                        *
*************************************************************************/
void calcul_param_signif(entete, tab_trace_temps, nb_seconde)
        entete_struct   entete ;
        short   *tab_trace_temps, *nb_seconde ;
{
        short   i, j, k ;

        short   *tab_seism_max ;
        float   *tab_seism_moy, freq_moy=0 ;

        tab_seism_max = (short *)calloc((entete.scan_count-1),
                                sizeof(short)) ;
        tab_seism_moy = (float *)calloc((entete.scan_count-1),
                                sizeof(float)) ;

        pt_block_cur = waveform_block ;

/*
* CALCUL DE LA VALEUR MAXI DES SEISMES DE CHAQUE CANAL
*/
        for( i=0 ; i<nb_block ; i++ ){
                for( j=1 ; j<entete.scan_count ; j++ ){
                        for( k=0 ; k<entete.channel_length ; k++){

/*
```

```
* TEST DE SUPERIORITE DE LA VALEUR COURANTE PAR RAPPORT AU MAXI COURANT
*/
                                if( *((pt_block_cur->channel[j]) +k ) >
                                                *(tab_seism_max + j - 1))
                                        *(tab_seism_max + (j-1)) =
                                        *((pt_block_cur->channel[j]) +k);
                                *(tab_seism_moy + (j-1)) =
                                        *(tab_seism_moy + (j-1)) +
                                        *((pt_block_cur->channel[j]) +k );

                        }
                }
                pt_block_cur = (block_struct *)
                                pt_block_cur->pt_next_block ;
        }

/*
* CALCUL DE LA VALEUR MOYENNE DES SEISMES DE CHAQUE CANAL
*/
        for( i=0 ; i<entete.scan_count - 1; i++ ){
                *(tab_seism_moy + i) = *(tab_seism_moy + i) /
                        (nb_block*entete.channel_length) ;
        }


/*
* CALCUL DE LA FREQUENCE MOYENNE SUR NB_SECONDE - 1
*/

        for ( i=1 ; i<*nb_seconde - 1 ; i++ )
                        freq_moy = freq_moy +
                                *(tab_trace_temps + i*2 + 1) -
                                *(tab_trace_temps + (i-1)*2 + 1) ;


        freq_moy = freq_moy / (*nb_seconde-2) ;

        /*
        printf(" Frequence moyenne = %f \n", freq_moy ) ;


        for( i=0 ; i<*nb_seconde ; i++ ){
                if (*(tab_trace_temps + i*2) == 0){
                        printf("trace[%d] = \t%d\n",i,
                                *(tab_trace_temps + i*2 +1)) ;
                }else{
                        printf("trace[%d] = \t\t%d\n",i,
                                *(tab_trace_temps + i*2 +1)) ;
                }
        }
        */

        free(tab_seism_max) ;
        free(tab_seism_moy) ;
/*
* CALCUL DU START TIME A PARTIR DU DECODAGE DE LA TRACE TEMPS
*/
        calcul_start_time(nb_seconde, tab_trace_temps, freq_moy ) ;
}

/*************************************************************************
*Nom :  cre_trace_tempe                                                  *
*                                                                       *
*Objet :        creation de la trace tempe en reduisant les tops        *
*               seconde en une valeur unique.                           *
*               0 = top court                                           *
*               1 = top large                                           *
*Entrees :      entete                                                  *
*                                                                       *
*Sortie :       nb_seconde                                              *
*                                                                       *
*                                                                       *
*Interne :      calcul_param_signif                                     *
*                                                                       *
*                                                                       *
*                                                                       *
*************************************************************************/
void cre_trace_tempe(entete, nb_seconde)
entete_struct   entete ;
short           *nb_seconde ;
```



```
{
        int     largeur_top=0, flag=0, taille_tab ;
        short   i, j ;
        short   *tab_trace_tempe ;

        *nb_seconde = 0 ;

        taille_tab = ((nb_block * entete.channel_length ) /
                                        entete.srate ) + 1 ;
/*
* ALLOCATION DU TABLEAU QUI CONTIENDRA LE TRACE TEMPS FORMATEE DE LA
* MANIERE SUIVANTE      0 : TOP SECONDE COURT
*                       1 : TOP SECONDE LARGE
*/
        tab_trace_tempe = (short *)calloc(taille_tab*2, sizeof(short)) ;

        pt_block_cur = waveform_block ;

/*
* BOUCLE SUR TOUTES LES VALEURS DE CHAQUE TRACE
*/
        for( i=0 ; i<nb_block ; i++ ){
                for( j=0 ; j<entete.channel_length ; j++ ){
/*
* DETERMINATION DU TYPE DE VALEUR ( VALEUR HAUTE = VALEUR DE TENSION
* D'UN TOP SECONDE ; VALEUR BASSE = VALEUR DE TENSION ENTRE DEUX TOPS
* SECONDE
*/
                        if ( *((pt_block_cur->channel[0]) +j ) > SEUIL ){
                                if (!flag){
/*
* SI FLAG PASSE A 1, ON COMMENCE A COMPTER LE NOMBRE DE VALEUR EGALE
* A UNE HAUTE TENSION. CECI PERMET DE DETERMINER S'IL S'AGIT D'UN TOP
* SECONDE COURT OU LARGE, DONC DE DETERMINER EN PARTICULIER LE TOP
* MINUTE.
*/
                                        flag = 1 ;
                                        *(tab_trace_tempe +
                                                (*nb_seconde)*2 + 1) =
                                                (i*(entete.channel_length)+
                                                 j+1);
                                }
                                largeur_top ++ ;
                        }else{
                                if (flag){
                                        flag = 0 ;
/*
* AFFECTATION DU TABLEAU "tab_trace_tempe"; LES ECHELONS CONSTITUES
* D'UNE SUCCESSION DE VALEURS DE TENSION HAUTE OU BASSE, SONT RAMENES
* A UNE SEULE VALEUR.   1 POUR TOP SECONDE LARGE
*                       0   "   "     "    COURT
*/
                                        if (largeur_top > LARGEUR){
                                                *(tab_trace_tempe +
                                                        (*nb_seconde)*2)=1;
                                        }else{
                                                *(tab_trace_tempe +
                                                        (*nb_seconde)*2)=0 ;
                                        }
/*
* INCREMENTATION DU COMPTEUR DE SECONDE
*/
                                        (*nb_seconde) ++ ;
                                }
                                largeur_top = 0 ;
                        }
                } /*fin boucle j*/
                if (!flag) largeur_top = 0 ;
                pt_block_cur = (block_struct *)
                                pt_block_cur->pt_next_block ;
        } /*fin boucle i*/
/*
* AFFICHAGE DU TABLEAU "tab_trace_POUR DEBOGGAGE
*/
        /*
        printf("Nombre de tope seconde = %d\n\n", taille_tab) ;

        for( i=0 ; i<taille_tab ; i++ ){
                if (*(tab_trace_tempe + i*2) == 0){
                        printf("trace[%d] = \t%d\n",i,
```

```
                        *(tab_trace_temps + i*2)) ;
                }else{
                        printf("trace[%d] = \t\t%d\n",i,
                                *(tab_trace_temps + i*2)) ;
                }
        }
        */
/*
* CALCUL DES VALEURS MOYENNES ET MAXI DE CHAQUE TRACE
*/
        calcul_param_signif(entete, tab_trace_temps, nb_seconde) ;

        free(tab_trace_temps) ;


}

/*************************************************************************
*Nom :  calcul_time                                                      *
*                                                                        *
*Objet :        calcule le trigger_time ainsi que la precision en        *
*               millisecondes du start_time et du trigger_time           *
*                                                                        *
*                                                                        *
*Entrees :      nb_seconde                                               *
*                                                                        *
*Sorties :      trigger_time, start_time,new_start_time,                 *
*               new_trigger_time                                         *
*               new_start_millitm, new_trigger_millitm                   *
*                                                                        *
*Internes :                                                              *
*                                                                        *
*                                                                        *
*                                                                        *
*************************************************************************/
void calcul_time ( trigger_time, start_time, new_start_time,
                        new_trigger_time, new_start_millitm,
                        new_trigger_millitm )

time_struct     *trigger_time, *start_time ;

int     *new_start_time, *new_trigger_time ;

short   *new_start_millitm, *new_trigger_millitm ;
{
        int     temp ;

        double  total_trigger_time, total_start_time, diff_time,
                total_new_trigger,temp_trigger_millitm,
                temp_start_millitm,start_decimal_value,
                trigger_decimal_value ;

        char    string_time[20], *pt_string_time, new_string_time[4] ;

/*
* TRIGGER TIME : CONVERSION DU NOMBRE DE MILLISECONDES EX : 912 -> 0.912
*/
        trigger_decimal_value = (float)trigger_time->millitm /1000 ;

/*
* TRIGGER TIME : CREATION DE LA VARIABLE REELLE DONNANT LES SECONDES
* ET LES MILLIS.
*/
        total_trigger_time =    trigger_decimal_value +
                                trigger_time->time ;
/*
* START TIME : CONVERSION DU NOMBRE DE MILLISECONDES EX : 912 -> 0.912
*/
        start_decimal_value = start_time->millitm/1000 ;
/*
* START TIME : CREATION DE LA VARIABLE REELLE DONNANT LES SECONDES
* ET LES MILLIS.
*/
        total_start_time =      start_decimal_value +
                                start_time->time ;
/*
* CALCUL DE LA DIFFERENCE ENTRE LE TRIGGER TIME ET LE START TIME DE
* L'ENTETE DU FICHIER INITIAL
```



```
*/
        diff_time = total_trigger_time-total_start_time ;

/*
* CALCUL DU TRIGGER TIME A PARTIR DU START TIME ISSU DE LA TRACE TEMPS
* ET DE LA DIFFERENCE PRECEDEMMENT CALCULEE
*/
        total_new_trigger = date_en_seconde + diff_time ;
/*
* CONVERSION EN ENTIER DU TRIGGER TIME
*/
        *new_trigger_time = (int) total_new_trigger ;
/*
* ISOLEMENT DE LA PARTIE REELLE DU TRIGGER TIME
*/
        temp_trigger_millitm = total_new_trigger - *new_trigger_time ;

        *new_trigger_time = (int) total_new_trigger /*+ 7 * 3600*/ ;

/*
* MISE AU FORMAT SUIVANT DE LA PARTIE REELLE : 0.912 (float) -> 912(int)
* CE FORMAT EST IMPOSE PAR L'ENTETE DE CHAQUE FICHIER DE DONNEES
*/
        sprintf(string_time, "%.3f", temp_trigger_millitm ) ;

        pt_string_time = &string_time[2] ;

        strcpy(new_string_time, pt_string_time) ;

        new_string_time[4] = '\0' ;

        sscanf(new_string_time, "%d", &temp ) ;

        *new_trigger_millitm = (short)temp ;
/*
* MEME DEMARCHE QUE PRECEDEMMENT AVEC LA VARIABLE START TIME
*/
        *new_start_time = (int) date_en_seconde ;

        temp_start_millitm = date_en_seconde - *new_start_time ;

        *new_start_time = (int) date_en_seconde /*+ 7 * 3600*/;

        sprintf(string_time, "%.3f", temp_start_millitm ) ;

        pt_string_time = &string_time[2] ;

        strcpy(new_string_time, pt_string_time) ;

        new_string_time[4] = '\0' ;

        sscanf(new_string_time, "%d", &temp ) ;

        *new_start_millitm = (short) temp ;
}
/*************************************************************************
*Nom :  ore_date_graphique                                               *
*                                                                        *
*Objet : permet d'inserer les millisecondes a la chaine de caractere     *
*        representant la date du start_time                              *
*                                                                        *
*                                                                        *
*Entrees : start_date   (cette variable est declaree dans la procedure   *
*                        FORTRAN)                                        *
*          start_time, trigger_time                                      *
*                                                                        *
*Sorties :      start_date                                               *
*                                                                        *
*                                                                        *
*Internes :     calcul_time                                              *
*                                                                        *
*                                                                        *
*                                                                        *
*************************************************************************/
void ore_date_graphique(start_date, start_time, trigger_time)
```

```
char            *start_date ;
time_struct     *start_time, *trigger_time ;
{
        char    millisecondе[4] ;
        short   n=19 ;
        int     new_start_time, new_trigger_time ;
        short   new_start_millitm, new_trigger_millitm ;

/*
* "date" CONTIENT LA DATE COURANTE AU FORMAT SUIVANT :
*       Sun Aug 30 06:28:38 1992
*/
        strcpy(date1, "") ;
        strcpy(date2, "") ;
        strncpy(date1, date, n) ;
/*
* ISOLATION DE LA DATE SANS L'ANNEE
*/
        date1[19] = '\0' ;
        strcpy(date2, &date[20]) ;
/*
* APPEL DE LA PROCEDURE "calcul_time" QUI PERMET DE RECUPERER LA VALEUR
* DA LA FRACTION DU START_TIME EXPRIMANT LES MILLISECONDES. CETTE PARTIE
* DECIMALE EST REPRESENTEE PAR UN NOMBRE ENTIER.
*/
        calcul_time (   &trigger_time, &start_time,
                        &new_start_time,
                        &new_trigger_time,
                        &new_start_millitm,
                        &new_trigger_millitm ) ;
/*
* TRANSFORMATION DES MILLISECONDES D'ENTIER A CHAINE DE CARACTERES
*/
        sprintf(millisecondе, "%d", new_start_millitm) ;

        millisecondе[3] = '\0' ;

        strcat(date1, ".") ;
/*
* CONCATENATION DES MILLISECONDES SOUS FORME DE CARACTERES
*/
        strcat(date1, millisecondе ) ;
        strcat(date1, " ") ;
/*
* RAJOUT DE L'ANNEE PRECEDEMMENT ISOLEE
*/
        strcat(date1, date2) ;
        strcpy(start_date, date1) ;

}

/*****************************************************************************
*Nom :  read_write_waveform                                                  *
*                                                                            *
*Objet :        lecture et ecriture de tous les waveform blocks              *
*                                                                            *
*                                                                            *
*                                                                            *
*Entrees :                                                                   *
*                                                                            *
*Sorties :                                                                   *
*                                                                            *
*                                                                            *
*Internes :                                                                  *
*                                                                            *
*                                                                            *
*                                                                            *
*****************************************************************************/
void read_write_waveform(fichbis, scan_count_decode,
                        channel_length_decode)
        int             fichbis ;
        short           *scan_count_decode ;
        unsigned short  *channel_length_decode ;
{
        int             i, j, len2=2 ;
        short int       word ;

        for( i=0 ; i< *scan_count_decode ; i++ ){

                for( j=0 ; j< *channel_length_decode ; j++ ){
```



```
                        read(fich, &word, len2);
                        write(fichbis, &word, len2);

                }
        }
}

/*****************************************************************************
*Nom :  rewrite_fich                                                         *
*                                                                            *
*Objet :        reecriture du fichier initial avec le start_time             *
*               recalcule a partir de la trace_tempe                         *
*                                                                            *
*                                                                            *
*Entreee : entete, nom_fich, trigger_time, start_time                        *
*                                                                            *
*Sorties :                                                                   *
*                                                                            *
*                                                                            *
*Internee :     calcul_time,permuteint4,permute2,read_write_waveform         *
*                                                                            *
*                                                                            *
*                                                                            *
*****************************************************************************/
void rewrite_fich(entete, station, nom_fich, trigger_time, start_time)
entete_struct   *entete ;
station_struct  station[] ;
char            *nom_fich ;
time_struct     *trigger_time, *start_time ;
{
        int     fichbis, compteur=0, i, j, indice ;

        int     new_start_time, new_trigger_time,
                len1=1, len2=2, len4=4, len10=10 ;

        short   new_start_millitm, new_trigger_millitm,
                scan_count_decode ;

        unsigned short  channel_length_decode ;

/*
*       OUVERTURE DU FICHIER CONTENANT LES MODIFICATIONS ET DU FICHIER
*       A MODIFIER.
*/

        fichbis = creat("temp.wvn", 0644) ;

        if ((fich = open(nom_fich, 0)) == -1) {
                printf("Erreur dans l'ouverture du fichier\n");
                exit(1) ;
        }
/*
*       LECTURE ET ECRITURE DES DIFFERENTS PARAMETRES CONSTITUANT
*       L'ENTETE LA VARIABLE COMPTEUR REPRESENTE LE NOMBRE D'OCTETS LUS
*       AFIN DE COMPLETER LA LECTURE A 1024 CORRESPONDANT A LA
*       FIN DE L'ENTETE.
*/
        read(fich, &(entete->magic_number), len2);
        write(fichbis, &(entete->magic_number), len2);

        read(fich, &(entete->channel_length), len2);
        write(fichbis, &(entete->channel_length), len2);
        channel_length_decode = entete->channel_length ;
        permute2( &channel_length_decode, len2) ;

        read(fich, &(entete->scan_count), len2);
        write(fichbis, &(entete->scan_count), len2);
        scan_count_decode = entete->scan_count ;
        permute2(&scan_count_decode, len2) ;

        read(fich, &(entete->srate), len4);
        write(fichbis, &(entete->srate), len4);

        compteur = compteur + 10 ;

        for (i = 0; i < 16; i++) {
                read(fich, &(entete->channel_list[i]), len2);
                write(fichbis, &(entete->channel_list[i]),
                                len2);
```

```
        }
        compteur = compteur + 32 ;

        for (i = 0; i < 16; i++) {
                read(fich, &(entete->gain_list[i]), len2);
                write(fichbis, &(entete->gain_list[i]), len2);
        }
        compteur = compteur + 32 ;

        for (i = 0; i < 32; i++) {
                read(fich, &(entete->channel_stat_list[i]), len2);
                write(fichbis, &(entete->channel_stat_list[i]),
                        len2);

        }
        compteur = compteur + 64 ;

        for (i = 0; i < 16; i++) {
                read(fich, station[i].station_id, len10);
                write(fichbis, station[i].station_id, len10);


                read(fich, &station[i].channel, len2);
                write(fichbis, &station[i].channel, len2);

                read(fich, &(station[i].field1), len4);
                write(fichbis, &(station[i].field1), len4);

                read(fich, &(station[i].field2), len4);
                write(fichbis, &(station[i].field2), len4);

                read(fich, &(station[i].field3), len4);
                write(fichbis, &(station[i].field3), len4);

                compteur = compteur + 24 ;
        }
        read(fich, &(trigger_time->time), len4);
        permuteint4(&(trigger_time->time)) ;

        read(fich, &(trigger_time->millitm), len2);
        permute2(&(trigger_time->millitm)) ;

        read(fich, &(trigger_time->timezone), len2);

        read(fich, &(trigger_time->dstflag), len2);

        read(fich, &(start_time->time), len4);
        permuteint4(&(start_time->time)) ;

        read(fich, &(start_time->millitm), len2);
        permute2(&(start_time->millitm)) ;

        read(fich, &(start_time->timezone), len2);

        read(fich, &(start_time->dstflag), len2);

        read(fich, entete->cal_channel_start_time, len10);


        calcul_time ( &trigger_time, &start_time,
                        &new_start_time,
                        &new_trigger_time,
                        &new_start_millitm,
                        &new_trigger_millitm ) ;

        permuteint4(&new_trigger_time, len4);
        write(fichbis, &new_trigger_time, len4);

        permute2(&new_trigger_millitm, len2);
        write(fichbis, &new_trigger_millitm, len2);

        write(fichbis, &(trigger_time->timezone), len2);
        write(fichbis, &(trigger_time->dstflag), len2);

        permuteint4(&new_start_time, len4);
        write(fichbis, &new_start_time, len4);

        permute2(&new_start_millitm, len2);
        write(fichbis, &new_start_millitm, len2);
```



```
        write(fichbis, &(start_time->timezone), len2);
        write(fichbis, &(start_time->dstflag), len2);

        write(fichbis, entete->cal_channel_start_time, len10);

        compteur = compteur + 30 ;

/*
*       FIN DE LECTURE DE L'ENTETE AFIN D'ARRIVER A 1024 OCTETS LUS
*/
        j = 1024-compteur ;
        for( i=0 ; i < j ; i++){
                read( fich, &bit, len1) ;
                write( fichbis, &bit, len1) ;

                compteur ++ ;
        }
/*
*       LECTURE DES "SCAN_OUT" CANAUX DES WAVEFORM BLOCKS
*/

        for ( indice=0 ; indice<nb_block ; indice++ ){

/*
* LECTURE ET ECRITURE DU WAVEFORM BLOCK COURANT
*/
                read_write_waveform(fichbis, &scan_count_decode,
                                                &channel_length_decode);
        }
        close(fich);
        close(fichbis);
        free(waveform_block) ;


}

/*************************************************************************
*                PROCEDURE PRINCIPALE DE LECTURE                        *
*                                                                       *
*       CETTE PROCEDURE PERMET DE LIRE L'ENTETE DU FICHIER COURANT      *
* AINSI QUE LES DIFFERENTS WAVEFORM BLOCKS.                             *
*                                                                       *
*************************************************************************/
void  lecture_(nom_fich, longueur,entete, station, start_time,
                trigger_time, trace,nombre_block, start_date, nb_seconde)

        char            nom_fich[100] ;
        short           *longueur ;
        entete_struct   *entete ;
        station_struct  station[] ;
        time_struct     *start_time, *trigger_time ;
        short           *trace ;
        short           *nombre_block ;
        char            start_date[30] ;
        short           *nb_seconde ;

{
        int     compteur=0, long_nom_fich ;
        int     i, j, k, len1=1, len2=2, len4=4, len10=10 ;
        short   pt_channel ;
        long    inorement_adresse=0 ;
        char    filename[13] ;


        nom_fich[*longueur] = '\0' ;
/*
* REINITIALISATION DE LA VARAIBLE nb_block
*/
        nb_block = 0 ;

/*
*       OUVERTURE DU FICHIER DONT LE NOM EST PASSE EN ARGUMENT
*/

        if ((fich = open(nom_fich, 0)) == -1) {
                printf("Erreur dans l'ouverture du fichier\n");
                exit(1) ;
        }

/*
```

```
*       LECTURE DES DIFFERENTS PARAMETRES CONSTITUANT L'ENTETE.
*       LA VARIABLE COMPTEUR REPRESENTE LE NOMBRE D'OCTETS LUS
*       AFIN DE COMPLETER LA LECTURE A 1024 CORRESPONDANT A LA
*       FIN DE L'ENTETE.
*/
        read(fich, &(entete->magic_number), len2);
        permute2(&(entete->magic_number)) ;

        read(fich, &(entete->channel_length), len2);
        permute2(&(entete->channel_length)) ;

        read(fich, &(entete->scan_count), len2);
        permute2(&(entete->scan_count)) ;

        read(fich, &(entete->srate), len4);
        permutefloat4(&(entete->srate)) ;

        compteur = compteur + 10 ;

        for (i = 0; i < 16; i++) {
                read(fich, &(entete->channel_list[i]), len2);
                permute2(&(entete->channel_list[i])) ;
        }
        compteur = compteur + 32 ;

        for (i = 0; i < 16; i++) {
                read(fich, &(entete->gain_list[i]), len2);
                permute2(&(entete->gain_list[i])) ;

        }
        compteur = compteur + 32 ;

        for (i = 0; i < 32; i++) {
                read(fich, &(entete->channel_stat_list[i]), len2);
                permute2(&(entete->channel_stat_list[i])) ;
        }
        compteur = compteur + 64 ;

        for (i = 0; i < 16; i++) {
                read(fich, station[i].station_id, len10);

                read(fich, &station[i].channel, len2);
                permute2(&(station[i].channel)) ;

                read(fich, &(station[i].field1), len4);
                permuteint4(&(station[i].field1)) ;

                read(fich, &(station[i].field2), len4);
                permuteint4(&(station[i].field2), len4) ;

                read(fich, &(station[i].field3), len4);
                permuteint4(&(station[i].field3)) ;

                compteur = compteur + 24 ;
        }
        read(fich, &(trigger_time->time), len4);
        permuteint4(&(trigger_time->time)) ;
        traducheure_(&(trigger_time->time), date1) ;


        read(fich, &(trigger_time->millitm), len2);
        permute2(&(trigger_time->millitm)) ;

        read(fich, &(trigger_time->timezone), len2);
        permute2(&(trigger_time->timezone)) ;

        read(fich, &(trigger_time->dstflag), len2);
        permute2(&(trigger_time->dstflag)) ;

        read(fich, &(start_time->time), len4);
        permuteint4(&(start_time->time)) ;

        traducheure_(&(start_time->time), date2) ;

/*
* CREATION D'UNE CHAINE DE CARACTERES A PARTIR DE LA DATE CONTENUE
* DANS LE NOM DU FICHIER ET DE L'ANNEE COURANTE ISSUE DE LA DATE
* COURANTE CONTENUE DANS LA VARIABLE "chaine_annee_courante".
* LE NOM DU FICHIER PEUT ETRE CONSTITUE DE SON CHEMIN D'ACCES,
* IL EST DONC NECESSAIRE, DANS CE CAS, D'ISOLER SON NOM.
```



```
*/
        if ( (long_nom_fich = strlen(nom_fich)) > 12 )
                strcpy(filename, &nom_fich[long_nom_fich -12]) ;
        else
                strcpy(filename, nom_fich) ;


        strcpy(chaine_annee_courante, filename) ;

        chaine_annee_courante[2] = '\0' ;
        sscanf(chaine_annee_courante, "%d", &annee_courante) ;
        if (annee_courante > 90 )
                annee_courante += 1900 ;
        else
                annee_courante += 2000 ;


        read(fich, &(start_time->millitm), len2);
        permute2(&(start_time->millitm)) ;

        read(fich, &(start_time->timezone), len2);
        permute2(&(start_time->timezone)) ;

        read(fich, &(start_time->dstflag), len2);
        permute2(&(start_time->dstflag)) ;

        read(fich, entete->cal_channel_start_time, len10);
        permute2(entete->cal_channel_start_time) ;

        compteur = compteur + 30 ;


/*
*       FIN DE LECTURE DE L'ENTETE AFIN D'ARRIVER A 1024 OCTETS LUS
*/
        j = 1024-compteur ;
        for( i=0 ; i < j ; i++){
                read( fich, &bit, len1) ;
                compteur ++ ;
        }

/*
*       AFFICHAGE DE L'ENTETE EN SUIVANT LA PRESENTATION DE XPLAYEGA
*/

        printf("Channel sampling rate :\t\t%f Hz\n", entete->srate);
        printf("Channel start time : \t\t%s", date2) ;
        printf("Channel trigger time : \t\t%s", date1) ;
        printf("Ch      ID      Trigger         Gain\n") ;
        for( i=0 ; i<16 ; i++ ){
                printf("%d\t",station[i].channel);
                printf("%s\t",station[i].station_id);
                if ( entete->channel_stat_list[i] == 0 )
                        printf("OFF\t\t") ;
                else
                        printf("ON\t\t") ;
                printf("%d\n", entete->gain_list[i] );

        }


/*
*       LECTURE DES "SCAN_OUT" CANAUX DES WAVEFORM BLOCKS
*/


        waveform_block = ALLOCAT(1, block_struct ) ;

        pt_block_cur = waveform_block ;

        waveform_block->channel = (short **)malloc( sizeof(short *) *
                                        entete->scan_count);

        for ( i=0 ; i<entete->scan_count ; i++ )
                waveform_block->channel[i]= ALLOCAT
                        (entete->channel_length, short) ;

        pt_block_cur->pt_next_block = NULL ;
```

```
        do{

/*
* LECTURE DU WAVEFORM BLOCK COURANT
*/
                fin_fich = read_waveform(entete) ;
/*
* TEST D'EXISTANCE D'UN PROCHAIN BLOC
*/
                if (!fin_fich) break ;

                nb_block ++ ;

                pt_block_preced = pt_block_cur ;
/*
* ALLOCATION DU PROCHAIN BLOC
*/
                pt_block_cur = ALLOCAT(1, block_struct ) ;

                pt_block_cur->channel = (short **)malloc
                                ( sizeof(short *) * entete->scan_count);
/*
* ALLOCATION DES CANAUX DU PROCHAIN BLOC
*/
                for ( i=0 ; i<entete->scan_count ; i++ )
                        pt_block_cur->channel[i] =
                                ALLOCAT(entete->channel_length,
                                                short) ;
/*
* CHAINAGE DU BLOC CONSIDERE A LA SUITE DE LA LISTE DES BLOCS
*/
                pt_block_preced->pt_next_block = pt_block_cur ;

                pt_block_cur->pt_next_block = NULL ;

        }while(EOF) ;
        close(fich);


        *nombre_block = nb_block ;

        /*
        printf("Nombre de block = %d\n", nb_block ) ;
        */

/*
* APPEL DE LA PROCEDURE DE TRAITEMENT DE LA TRACE TEMPS
*/

        cre_trace_temps(entete, nb_seconde) ;

/*
* AFFECTATION DE LA VARIABLE "start_date" PRESENTE SUR LA SORTIE
* GRAPHIQUE POSTCRIPT
*/
        cre_date_graphique(start_date, start_time, trigger_time) ;

/*
* AFFECTATION DU TABLEAU TRACE
*
* TRACE N'EST DECLARE DANS LA PROCEDURE FORTRAN APPELANT CE MODULE
* IL EST NECESSAIRE DE FAIRE LA CORRESPONDANCE ENTRE LA SOMME MEMOIRE
* RESERVEE A CET EFFET PAR LE FORTRAN ET LA SOMME MEMOIRE AFFECTEE PAR
* LA PROCEDURE C COURANTE. LE TABLEAU TRACE N'EST PASSE PAR ADRESSE,
* L'AFFECTATION DE CELUI-CI SE FAIT EN REFERENCANT CHAQUE CASE MEMOIRE
* PAR RAPPORT A L'ADRESSE DU PREMIER ELEMENT DU TABLEAU. LA VARIABLE
* "TRACE" ( NOM DU TABLEAU) EST L'ADRESSE DE CE PREMIER ELEMENT.
* LE CARACTERE INDIGESTE DES QUELQUES LIGNES CI-APRES PROVIENT DU FAIT
* QUE LE FORTRAN INDICE UN TABLEAU COLONNE PAR COLONNE ALORS QUE LE C
* PROCEDE LIGNE PAR LIGNE. SACHANT QU'UN TABLEAU N'EST QU'UNE SUCCESSION
* DE LIGNE DE TOUTE LA PLACE MEMOIRE QUI LUI EST ALLOUEE ( COLONNES BOUT
* A BOUT POUR LE FORTRAN ET LIGNES BOUT A BOUT POUR LE C), IL EST
* NECESSAIRE DE FAIRE DE LA CORRESPONDANCE D'INDICE AU MOMENT DE
* L'AFFECTATION.
*/
        pt_block_cur = waveform_block ;
```



```
        for ( j=0 ; j<entete->scan_count ; j++){

                pt_block_cur = waveform_block ;

                for ( i=0 ; i < nb_block ; i++){

                        pt_channel = 0 ;

                        for ( k=0 ; k<entete->channel_length ; k++ ){

                                *(trace+(j*nb_block*entete->channel_length
                                        +k+entete->channel_length*i)+
                                        increment_adresse)
                                =
                                *((pt_block_cur->channel[j]) +
                                        pt_channel) ;

                                pt_channel++ ;

                        }/*fin boucle k*/

                        pt_block_cur = (block_struct *)
                                pt_block_cur->pt_next_block ;

                }/*fin boucle i*/
/*
* LA VALEUR 40000 EST LA TAILLE MAXIMALE D'UN CANAL. CELLE-CI CORRESPOND
* AU DIMENSIONNEMENT DU TABLEAU "TRACE" DECLARE DANS LA PARTIE FORTRAN.
* LA VARIABLE "increment_adresse" CORRESPOND AU SAUT DE CASES MEMOIRE
* NECESSAIRE POUR PASER DE LA FIN DU CANAL COURANT AU DEBUT DU SUIVANT.
*/
                increment_adresse += (40000-nb_block*
                        entete->channel_length) ;

        }/*fin boucle sur j*/

/*
* DESALLOCATION  DE LA LISTE CHAINEE CONSTITUEE DES DIFFERENTS BLOCS
*/

        pt_desalloc = (block_struct *)waveform_block->pt_next_block ;


        for ( i=0 ; i<entete->scan_count ; i++ )
                free(waveform_block->channel[i]) ;

        free(waveform_block->channel) ;

        free(waveform_block) ;

        for ( i=0 ; i < nb_block -1 ; i++ ){
                pt_next_desalloc = (block_struct *)
                        pt_desalloc->pt_next_block ;

                for ( j=0 ; j<entete->scan_count ; j++ )
                        free(pt_desalloc->channel[j]) ;

                free(pt_desalloc->channel) ;

                free(pt_desalloc) ;

                pt_desalloc = pt_next_desalloc ;
        }

        /*
        rewrite_fich(&entete, &station, nom_fich, &trigger_time,
                        &start_time) ;
        */

}
```

Centre ORSTOM de Nouméa
B.P. A5 Nouméa Cédex Nouvelle Calédonie