Logiciel d'acquisition des donnees de l'interferometre du plateau de Bure

Dominique Broguiere

October 25, 1996

Contents

Introduction

Ce document decrit les programmes realisant l'acquisition des donnees de l'interferometre de Bure.

- Le chapitre 1 presente les differents elements *materiels* du systeme d'acquisition.
- Le chapitre 2 donne une description globale du logiciel.
- Le chapitre 3 decrit les traitements mathematiques effectues sur les donnees depuis les comptes lus sur les circuits multiplicateurs des correlateurs jusqu'aux spectres fournis a la tache *rdi*
- Les chapitres 4,5 et 6 donnent des descriptions des logiciels tournant sur le HP.
- Le chapitre 7 decrit les programmes tournant sur les chassis correlateur.
- Le chapitre 8 decrit les programmes tournant sur le chassis phaser
- Le chapitre 9 regroupe en annexe divers paragraphes ne rentrant pas dans les chapitres precedents.

Pour une description globale du systeme, sans rentrer dans les details de la programmation, on peut se contenter de lire les chapitres 1, 2 et 3.

1 Description generale du systeme

1.1 Presentation globale du systeme

Avant de presenter le logiciel d'acquisition des donnees, il est bon de presenter rapidement les differents elements intervenant dans l'acquisition. La figure 1 presente un schema du syteme.

On distingue :

- la machine centrale *bure1* (HP 770) qui pilote l'ensemble de l'interferometre (antennes, recepteurs, correlateurs...).
- le chassis *phaser* qui pilote les rotateurs de phase LO1, les synthetiseurs de frequence, le systeme de mesure de phase des cables et les detecteurs continuum.
- 6 unites correlateurs. Les 6 unites sont equivalentes et traitent chacune une bande de frequence. Une unite comporte un chassis IF et 2 chassis VME (chassis *maitre* et chassis *satellite*).

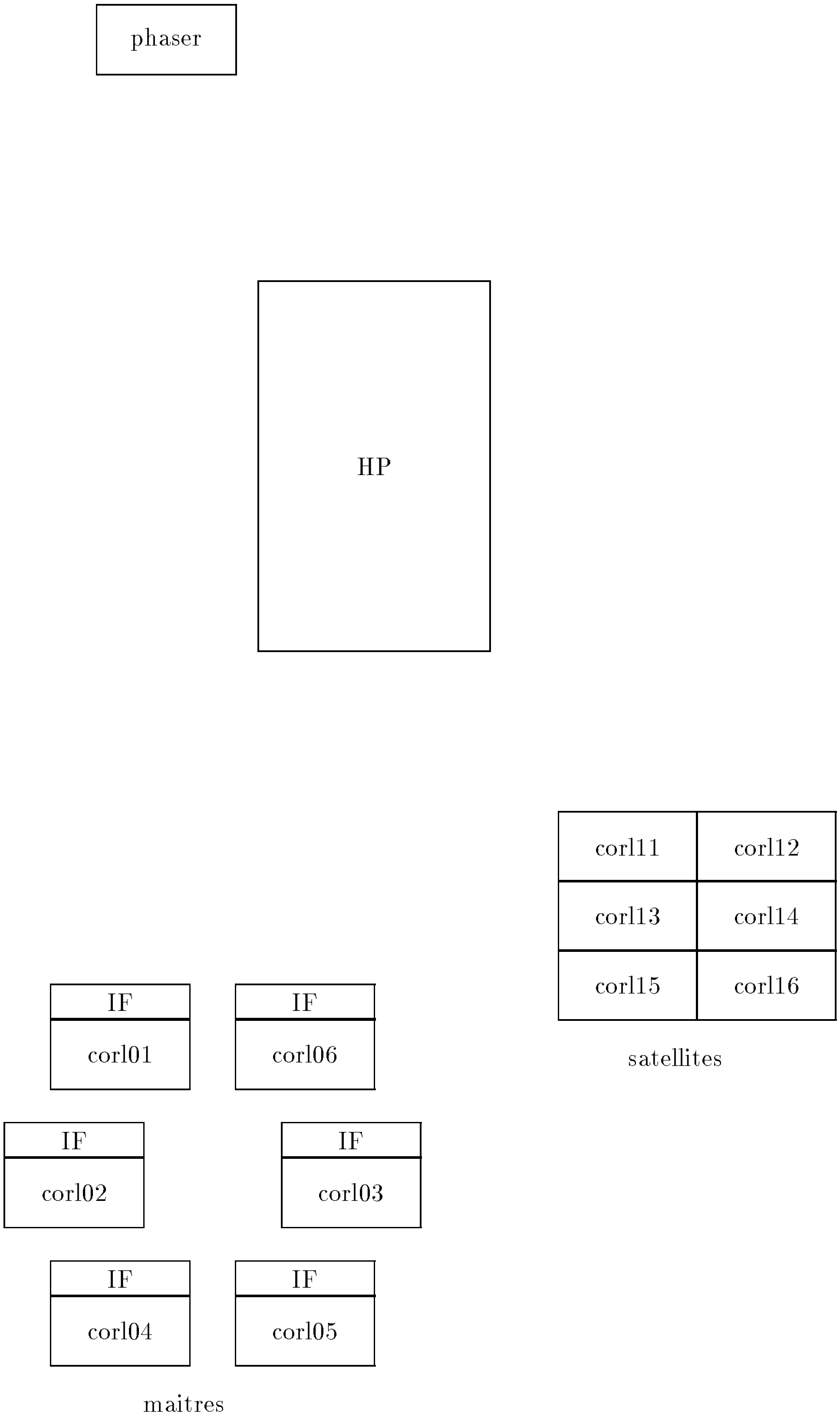

phaser
HP
corl11
corl12
corl13
corl14
corl15
corl16
satellites
IF
corl01
IF
corl06
IF
corl02
IF
corl03
IF
corl04
IF
corl05
maitres

figure 1

1.2 Le chassis phaser

Le chassis *phaser* contient:

- 3 cartes rotateaurs permettant de piloter 12 rotateaurs LO1 (2 recepteurs $\times$ 6 antennes) La phase et le rate sont synthetises a partir de circuits DDS
- 3 cartes phasemetres permettant de mesurer la phase sur 12 entrees
- 1 carte d'interface VME-GPIB pour piloter le synthetiseur de frequences HP
- 1 carte ADC pour la lecture des detecteurs continuum
- 1 carte Eurocom6 comportant un microprocesseur 68030 et un controleur Ethernet

1.3 Le chassis IF

Le chassis IF recoit les 2 bandes de signal 100-600Mhz provenant des 2 recepteurs de chacune des 5 antennes. C'est dans ce chassis que se fait la selection de la bande a etudier a l'interieur de la bande 100-600Mhz. L'observateur choisit le recepteur (1mm ou 3mm), la frequence centrale (entre 100 et 600 Mhz) et la largeur de bande (10,20,40,80 ou 160Mhz). En fonction de la largeur choisie, le chassis IF fournira l'une des configurations suivantes :

- 10Mhz : on selectionne le filtre 10Mhz et on exploite uniquement la bande Upper LO3. On obtient 256 canaux complexes pour chacune des bandes ciel USB et LSB.
- 20Mhz : on selectionne le filtre 20Mhz et on exploite uniquement la bande Upper LO3. On obtient 256 canaux complexes pour chacune des bandes ciel USB et LSB.
- 40Mhz : on selectionne le filtre 20Mhz et on exploite les 2 bandes de 20Mhz Lower et Upper LO3. On obtient 2 $\times$ 128 canaux complexes pour chacune des bandes ciel USB et LSB.
- 80Mhz : on selectionne le filtre 40Mhz et on exploite les 2 bandes de 40Mhz Lower et Upper LO3. On obtient 2 $\times$ 64 canaux complexes pour chacune des bandes ciel USB et LSB.
- 160Mhz : on selectionne le filtre 80Mhz et on exploite les 2 bandes de 80Mhz Lower et Upper LO3. On obtient 2 $\times$ 32 canaux complexes pour chacune des bandes ciel USB et LSB.

1.4 Le chassis correlateur maitre

Ce chassis recoit les sous-bandes 10,20,40 ou 80Mhz, les echantillonne (cartes sampler) et fait les produits correspondant aux 10 lignes de base (cartes correlateur) des sous-bandes Lower LO3. Plus precisemment ce chassis contient:

- 5 cartes samplers (une carte sampler par antenne offrant 2 samplers, l'un pour la bande LOWER LO3 et l'autre pour la bande UPPER LO3)
- 10 cartes correlateurs permettant de traiter les 10 lignes de base correspondant a la sous-bande Lower LO3
- 1 carte de controle qui pilote l'ensemble

- 1 carte *gigalink* qui permet de transmettre vers le chassis *satellite* les donnees echantillonnees a 160Mhz de la sous bande Upper LO3
- 1 carte Eurocom17 comportant un microprocesseur 68040 et un controleur Ethernet

Dans le cas de bandes 10 ou 20Mhz, il n'y a qu'une sous-bande LO3, mais les 256 canaux a traiter necessitent 2 cartes par ligne de base. Il faut donc traiter 5 lignes de base dans l'unite maitre et les 5 autres lignes de base dans le chassis satellite.

1.5 Le chassis correlateur satellite

Le chassis satellite recoit les donnees echantillonnees par les cartes sampler du chassis maitre via le gigalink. Ces donnees correspondent aux sous bandes Upper LO3. Ce chassis contient les cartes suivantes:

- 10 cartes correlateurs permettant de traiter les 10 lignes de base correspondant a la sous-bande Upper LO3
- 1 carte de controle qui pilote l'ensemble
- 1 carte *gigalink* qui permet de recevoir les donnees echantillonnees dans le chassis maitre
- 1 carte Eurocom17 comportant un microprocesseur 68040 et un controleur Ethernet

Pour avoir une description technique plus detaillee, on peut se reporter au document *Un nouveau correlateur spectral pour l'interferometre a 4 antennes* de M.Torres (septembre 1988), *Correlator extension for the 5th antenna* (octobre 1994) ainsi qu'aux autres documents ecrits par l'equipe Backend de l'IRAM.

2 Description generale du logiciel

Les correlateurs sont pilotes par l'ordinateur central HP *bure1* qui controle l'ensemble de l'interferometre (antennes,recepteurs,etc.). Tous les echanges entre le HP et les correlateurs ou le chassis *phaser* se font sur une ligne Ethernet dediee.

2.1 Dialogue avec les correlateurs

Le HP envoie les consignes de configuration, de demarrage/fin d'acquisition, et de rapatriement des donnees. Si on se refere au mode client/serveur, les correlateurs fonctionnent en mode serveur. Ils sont en permanence en attente de requetes provenant du HP.

Pendant une observation, les unites connectees recoivent chaque seconde un message du HP. Ce message contient notamment des ordres de configuration (auto/crosscorrelation, frequence centrale de la bande, largeur de la bande), des consignes de delais a appliquer sur les lignes a retard et des consignes de phase a appliquer sur les rotateurs LO4. Les ordres de configuration sont utilises en debut d'observation. Les consignes de delais et de phases changent toutes les secondes a cause de la rotation terrestre. Elles sont donc appliquees toutes les secondes au cours d'une observation.

Le message du HP contient egalement un flag indiquant si on peut ajouter les spectres correspondant a la periode [T-2,T-1[, aux spectres anterieurs. Ce dispositif doit permettre d'eviter de "polluer" des donnees valides.

Lorsqu'une unite recoit ce message appelé "message seconde", elle le traite et renvoie une reponse qui contient les valeurs moyennes des spectres obtenus dans la periode [T-2,T-1[(ce sont les donnees *temporelles*), ainsi qu'un flag signalant un probleme eventuel sur la periode [T-1,T[. Cette reponse permet au HP de synchroniser ses envois vers les differentes unites : il envoie un message a la premiere unite, attend sa reponse, envoie un message a la seconde unite, attend sa reponse et ainsi de suite. Cette serie d'echanges qui a lieu juste apres la reception de l'interruption 1 seconde dure moins de 50ms pour 12 chassis correlateurs.

En dehors de ces messages, le HP envoie des requetes de demande de rapatriement de spectres. En reponse a cette requete, le correlateur envoie les donnees disponibles.

2.2 Dialogue avec le chassis phaser

Le HP envoie les consignes de configuration (synthetiseur HP, rotateurs de phases) et recupere, a chaque echange, la mesure de phase des cables, le niveau des detecteurs continuum. Ce chassis fonctionne aussi en mode *serveur*. Il est en permanence en attente de requetes provenant de l'ordinateur central HP.

En pratique, le chassis phaser recoit chaque seconde un message du HP. Si on est en cours d'observation, ce message contient notamment les ordres de configuration des rotateurs de phase (phase instantanee et rate) pour la seconde suivante. Le chassis phaser renvoie systematiquement, en reponse, un message contenant les phases mesurees des cables et le niveau des detecteurs continuum.

3 Presentation du traitement des donnees

3.1 Generalites

Le but de ce chapitre est de presenter de facon detaillee les traitements successifs que subissent les donnees brutes lues dans les chips correlateur.

Une presentation generale de ces traitements et notamment de leur enchainement est donnee dans *Real time processing of correlator data* de R.Lucas (fevrier 1992).

Les traitements successifs qu'on applique aux donnees lues dans les chips correlateur sont:

- Reordonnement des donnees (dans le cas des tmf 2 et 4) Soustraction de la constante due au correlateur dans le cas des autocorrelations et des crosscorrelations IFPB.
- Correction de clipping
- Apodisation
- FFT
- Separation des bandes USB et LSB. Ce dernier traitement est effectue au niveau du HP

Dans cette liste n'apparaissent pas les integrations, car elles sont faites apres la FFT dans le cas des crosscorrelations sur le ciel alors qu'elles sont faites sur les donnees brutes (avant reordonnement des donnees et correction de clipping) dans le cas des autocorrelations et des crosscorrelations IFPB.

3.2 Correction de clipping

3.2.1 Generalites

La correction de clipping a pour but de corriger les effets de quantification sur un correlateur digital a n niveaux. Cette correction depend du nombre de niveaux d'echantillonnage du signal et de la table de multiplication dans les chips correlateurs.

Soit v_0 le seuil d'echantillonnage des samplers et $v(t)$ la valeur instantanee du signal. Les 4 niveaux sont definis de la facon suivante :

- $v(t) < -v_0$
- $-v_0 < v(t) < 0$
- $0 < v(t) < v_0$
- $v(t) > v_0$

La table de multiplication correspondant a ces 4 niveaux est :

3	1	-1	-3
1	0	0	-1
-1	0	0	1
-3	-1	1	3

Le but de la correction de clipping est de trouver la correlation qu'on obtiendrait avec un correlateur analogique (correlation *vraie*) a partir des donnees issues de notre correlateur a 4 niveaux.

La figure 1, a la fin du paragraphe *Correction de clipping*, montre la relation entre la correlation *vraie* ρ et les donnees du correlateur R. On constate que le correlateur a une reponse lineaire pour les faibles correlations mais que cette reponse n'est plus lineaire aux forts taux de correlation.

Pour faire la correction de clipping sur les donnees issues du correlateur, on distingue donc 2 cas suivant qu'on a des taux de correlation forts ou non.

- Dans le cas des correlations faibles, le correlateur a une reponse quasiment lineaire et il suffit de multiplier les donnees mesurees par un coefficient qui depend des niveaux d'entree pour trouver les taux de correlation *vrais*. On est toujours dans le cas de correlations faibles lorsqu'on fait de la crosscorrelation sur des signaux issus des recepteurs.

- Dans le cas des correlations fortes, la fonction de correction est plus compliquee. On a des taux de correlation forts en autocorrelation et en crosscorrelation sur les IFPB.

3.2.2 Correlation forte

On part de l'equation donnant la derivee de la fonction de correlation mesuree en fonction de la correlation vraie . (cf WR190 *Clipping correction for the continuum correlator* de D.Morris)

En l'adaptant au cas du correlateur 2 bits 4 niveaux, on obtient :

$$\begin{aligned}\frac{d\ R_{mesure}}{d\ \rho} &= \frac{1}{\pi\ \sqrt{1-\rho^2}} \quad (\ exp\left[\frac{-(c_{11}+c_{22}-2\ \rho\ c_{12})}{2\ (1-\rho^2)}\right] + exp\left[\frac{-(c_{11}+c_{22}+2\ \rho\ c_{12})}{2\ (1-\rho^2)}\right] \\ &\quad + 2\ exp\left[\frac{-c_{11}}{2\ (1-\rho^2)}\right] + 2\ exp\left[\frac{-c_{22}}{2\ (1-\rho^2)}\right]\ )\end{aligned}$$

avec :
ρ = taux de correlation *vrai*
R_{mesure} = produit obtenu en sortie du correlateur
$c_{11} = 1/(v_1/\sigma_1)^2$
$c_{22} = 1/(v_2/\sigma_2)^2$
$c_{12} = 1/((v_1/\sigma_1)\ (v_2/\sigma_2))$
v_1/σ_1 est le seuil d'echantillonnage sur le premier signal exprime en fonction du rms de ce signal.
v_2/σ_2 est le seuil d'echantillonnage sur le second signal exprime en fonction du rms de ce signal.

On effectue un changement de variable $\rho = sin(x)$ pour eliminer la singularite en $\rho = 1$, puis on integre la fonction obtenue et enfin on l'inverse pour obtenir $x = f(R_{mesure})$ et enfin $\rho = sin(x) = sin(f(R_{mesure}))$.

L'integration de la fonction ainsi que l'inversion est faite en utilisant des interpolations par splines cubiques.

3.2.3 Correlation faible

Dans ce cas, on considere que le correlateur est lineaire. En effet, si on se refere a la figure 2 de la page suivante, on constate que l'erreur relative est inferieure a 10^{-5} pour $\rho < 0.02$ et a 10^{-4} pour $\rho < 0.07$. On ecrit donc :

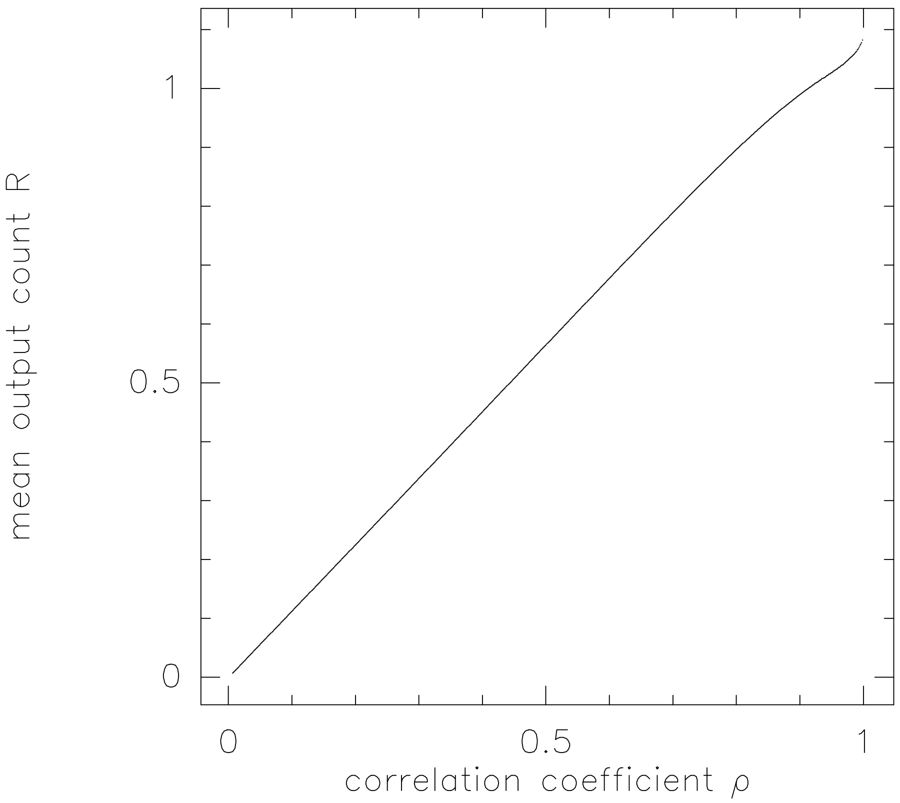
$$R_{mesure} = a\ \rho$$

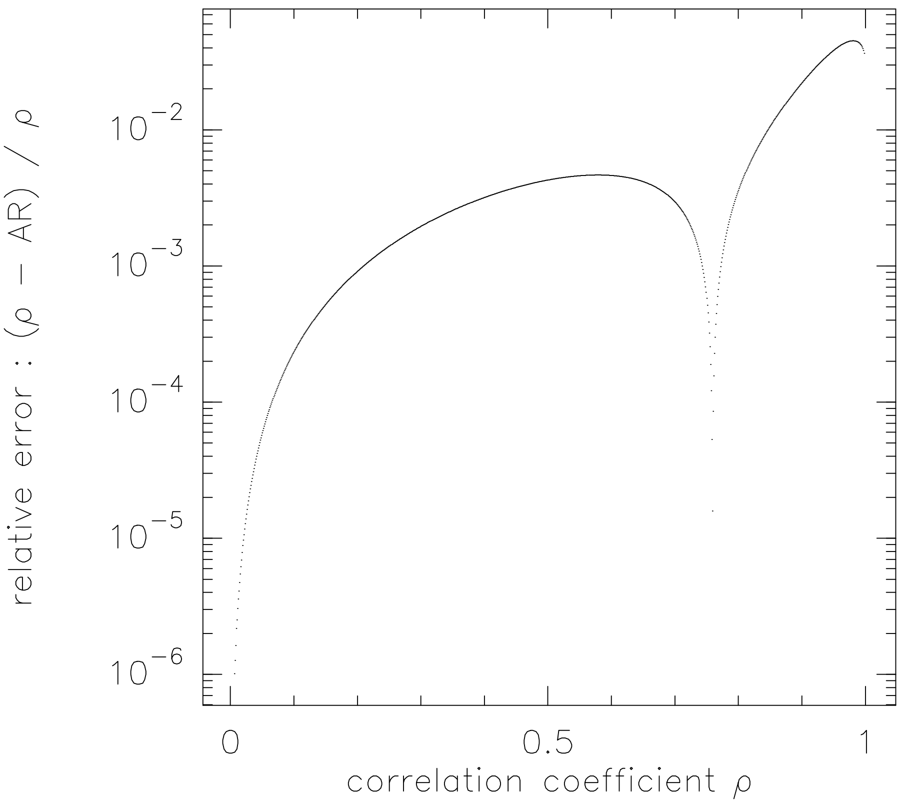
Le coefficient a est determine en faisant un developpement limite en $\rho = 0$ de la formule generale donnee dans le paragraphe precedent. Les calculs sont detailles dans le WR190 et, en adaptant la formule (19) au correlateur a 4 niveaux, on obtient :

$$a = \frac{2}{\pi}(exp\left[-\frac{c_{11}+c_{22}}{2}\right] + exp\left[-\frac{c_{11}}{2}\right] + exp\left[-\frac{c_{22}}{2}\right])$$

avec :
$c_{11} = 1/(v_1/\sigma_1)^2$
$c_{22} = 1/(v_2/\sigma_2)^2$

On neglige le terme en ρ^3, d'une part car le gain en precision serait minime aux taux de correlation ou l'on travaille et d'autre part car le temps de calcul est severement limite sur le micro 68030.


mean output count R
0
0.5
1
0
0.5
1
correlation coefficient ρ


relative error : $(\rho - AR) / \rho$
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
0.5
1
correlation coefficient ρ

3.3 Apodisation

L'apodisation consiste a multiplier les fonctions de correlation par une fonction de facon a lisser la fonction d'autocorrelation afin qu'elle tende vers une valeur nulle pour les plus grands delais.

Ceci permet de reduire les oscillations dans les spectres. Dans certains cas, il peut etre interessant de ne pas elargir les canaux en frequence. Pour cela, on dispose de l'option *sans apodisation.*

Dans notre cas, on apodise avec une parabole $\left(1 - \frac{i}{n_{channel}-1}\right)^2$
avec :
$n_{channel}$ = nombre total de canaux
i = numero du canal (de 0 a $n_{channel} - 1$)

Dans les spectres, apparait toujours un trou central au niveau de la frequence du LO3. On peut recuperer du signal dans le premier canal a condition de faire un decalage d'un demi-canal en frequence. Dans le domaine temporel, ceci revient a multiplier la fonction de correlation par e^{-it} avec t $= 0, \frac{1}{n_{channel}}, \ldots, \frac{n_{channel}-1}{n_{channel}}$.

Ce decalage d'un demi-canal est fait systematiquement sur tous les spectres. Pour limiter les calculs et notamment le nombre de multiplications, il est fait en meme temps que l'apodisation.

3.4 FFT

Le calcul de la FFT sur les fonctions de correlation donne les spectres en frequence.

En crosscorrelation, ces FFT sont faites toutes les secondes par le micro 68040. Le programme de calcul de FFT doit donc etre rapide. L'utilisation des algorithmes de Split Radix FFT fournis par P.Duhamel et testes a l'IRAM par L.Desbat a permis de gagner un facteur 2 par rapport aux temps de calcul obtenus avec le programme FOURT.

Pour plus de precision sur cet algorithme, on peut se reporter au rapport *Split Radix FFT* de L.Desbat (novembre 1991) ainsi qu'aux programmes FFTFOR.FOR ou FFTC.C dans USER2:[DESBAT.FFT] et au fichier de documentation USER2:[DESBAT.FFT]FFTFOR.DOC

Sur une carte Eurocom17 equipee d'un micro 68040, une FFT complexe de 256 points est calculee en 2.4ms et une FFT complexe de 512 points en 5.6ms.

3.5 Correction des variations de phase atmospherique

C'est la tache *rdi* qui calcule le coefficient de proportionnalite entre les variations de puissance observables sur le recepteur 1mm et les variations de phase induites sur le signal.

Ce coefficient est calcule pour chaque antenne,chaque recepteur et chaque bande USB et LSB LO1. C'est la variable PH_CORR(nb_ant,nb_rec,2) dans le common ACQUIS qui contient ces valeurs.

La tache *correl*, sur le HP, calcule la correction de phase en utilisant ces coefficients et les valeurs de puissance totales transmises par le chassis *phaser*. Les valeurs de corrections sont calculees par rapport au premier point du scan, qui a donc une correction nulle. La valeur moyenne de la correction est egalement calculee et soustraite au spectre moyenne.

Les valeurs de correction calculees chaque seconde sont envoyees vers les chassis correlateur ou elles sont appliquees par la routine *scale_OBS*. La valeur moyenne de la correction est egalement transmise a la fin du scan.

3.6 Correction des petits delais

3.7 Separation des donnees USB LSB

La separation des donnees USB LSB est decrite dans le rapport de R.Lucas cite precedemment. En conclusion :

- *Pour une bande lower LO3*, les frequences negatives du spectre correspondent au LSB LO1 et les frequences positives au USB LO1.
- *Pour une bande upper LO3*, les frequences negatives du spectre correspondent au USB LO1 et les frequences positives au LSB LO1.

4 Description de la tache correl (HP)

4.1 Communication entre taches

Avant de decrire la tache *correl* , on peut rapidement presenter les moyens de communications entre les differentes taches tournant sur le HP.

Tout d'abord, les taches partagent des donnees en utilisant les zones de donnees partagees (shared memory). Celles-ci sont decrites dans le paragraphe suivant.

Par ailleurs, la synchronisation entre taches est assuree par des semaphores. Un groupe de 64 semaphores est installe au demarrage du systeme. Il est repere par la cle BURE (0x42555245). On peut verifier sa presence en executant la commande ipcs :

```
$ ipcs -s -b
IPC status from /dev/kmem as of Fri Oct  4 15:33:20 1996
T     ID     KEY        MODE       OWNER    GROUP NSEMS
Semaphores:
s      0 0x411806ac --ra-ra-ra-     root     root     1
s      1 0x4e040002 --ra-ra-ra-     root     root     2
s      2 0x411c119c --ra-ra-ra-     root     root     2
s      3 0x00446f6e --ra-r--r--     root     root     1
s      4 0x00446f6d --ra-r--r--     root     root     1
s      5 0x01090522 --ra-r--r--     root     root     1
s      6 0x054baa58 --ra-r--r--     root     mail     1
s      7 0x9f107213 --ra-------     root     root     5
s    740 0x4c183bf4 --ra-r--r--   daemon   daemon     1
s      9 0x42555245 --ra-ra-ra- operator     oper    64
```

Les semaphores sont utilises a la maniere des events flag VMS. On a donc defini des routines similaires a celles existant sur VMS :

- waitfr : attente d'un flag
- clref : clear d'un flag
- setef : set d'un flag
- readef : lecture d'un flag (cleared/set)

4.2 Description des donnees

4.2.1 Donnees partagees

La tache correl accede a plusieurs zones de donnees partagees installes en memoire :

- GENERAL : donnees generales de l'interferometre. La description de la structure de GENERAL se trouve dans /control/common/s_general.f.
- ACQUIS : parametres utilises pour l'acquisition C'est par ce common que *correl* recupere les parametres d'observation positionnes par OBS. La description de la structure de ACQUIS se trouve dans /control/common/s_acquis.f

- ANTENNA : parametres relatifs aux antennes
- RECEIVER : parametres relatifs aux recepteurs
- CONTCO : donnees transmises par la tache correl a la tache rdi C'est par ce common que se fait le passage de donnees de la tache *correl* a la tache *rdi* qui effectue l'ecriture des donnees dans un fichier sur disque. La description de la structure de CONTCO se trouve dans /control/common/s_correl_rdi.f
- TEMPORAL : donnees temporelles partagees entre les taches *correl* et *xaff* (tache d'affichage)

Les zones de donnees partagees sont identifiees par une cle egale aux 4 premieres lettres du nom de ce common. Par exemple, pour le common ACQUIS, la cle est 0x41435155 (A=0x41, C=0x43, Q=0x51, U=0x55). La commande ipcs permet de visualiser les zones de memoire partagees installees:

```
$ ipcs -m -b
IPC status from /dev/kmem as of Fri Oct  4 15:34:20 1996
T     ID     KEY        MODE       OWNER    GROUP  SEGSZ
Shared Memory:
m      0 0x411806ac --rw-rw-rw-     root     root    348
m      1 0x4e040002 --rw-rw-rw-     root     root  31008
m      2 0x411c119b --rw-rw-rw-     root     root   8192
m      3 0x00000000 --rw-------     root      sys  65536
m    604 0x43183bf4 --rw-rw-rw-   daemon   daemon 5767168
m  12805 0x00000000 D-rw-------   daemon   daemon 524384
m      6 0x47454e45 --rw-rw-rw- operator     oper    480
m      7 0x414e5445 --rw-rw-rw- operator     oper   4200
m      8 0x52454345 --rw-rw-rw- operator     oper   2680
m  25609 0x4f425331 --rw-rw-rw-    lucas    softs     16
m    210 0x54454d50 --rw-rw-rw- operator     oper 391048
m   2211 0x41435155 --rw-rw-rw- operator     oper   1152
m  12412 0x4f425332 --rw-rw-rw-    lucas    softs   2424
m     13 0x434f5252 --rw-rw-rw- operator     oper 876904
m    214 0x4f425333 --rw-rw-rw-    lucas    softs     28
m   6415 0x58414646 --rw-rw-rw-    lucas    softs    200
```

Les zones de memoire partagees sont installees, soit automatiquement en utilisant la directive $SHARED en tete des programmes fortran, soit en utilisant les routines *.c de gestion de memoire partagees (cf programme /control/common/shared_memory.c).

4.2.2 Donnees propres a la tache correl

Les donnees propres a la tache CORREL sont dans un common non installe dont la structure est decrite dans *correl_local.f*.

4.2.3 Description de CONTCO

Ce common contient toutes les variables qui sont stockees par la tache *rdi* dans un fichier et qui seront ensuite exploitees par l'intermediaire de CLIC. Voici un listing du fichier /control/correl/s_correl_rdi.f :

```
* s_correl_rdi.f
*
* Data shared by CORREL and RDI

        structure /s_correl_rdi/


!  Structure du common CONTCO
!

logical*4 CORREL ! .true. if correlation !RL
real*4 TINTEG ! integration time (1 second)
integer*4       NINTEG          ! number of basic integrations
        integer*4       IPH_CORR_DEF    ! define real time phase correction :
!  0 = get only uncorrected spectra
!  1 = get only corrected spectra
!  2 = get uncorrected & corrected spectra
integer*4 LEVLIN(2*nb_corr*5)  !  LEVLIN(2*nb_corr*nb_ant)
                                        ! niveaux d'echantillonnage

! Spectra
        union
         map
  complex*8     U_SPECT(128,2,nb_corr,2,10) ! uncorrected cross spectra
                         ! U_SPECT(128,Llo4/Ulo4,nb_corr,Ulo1/Llo1,nb_base)
  complex*8 C_SPECT(128,2,nb_corr,2,10) ! corrected cross spectra
                         ! C_SPECT(128,Llo4/Ulo4,nb_corr,Ulo1/Llo1,nb_base)
         end map
         map
  real*4 R_SPECT(128,2,nb_corr,5)    ! auto spectra
                         ! R_SPECT(128,Llo4/Ulo4,nb_corr,nb_ant)
         end map
        end union

! Temporal data
record /temporal_data/ C(nb_temp)


! Errors
integer*4 IERRC ! error on data reception (1bit per unit)
integer*4 ERPLLT ! PLL error (global)
*                                       ! 0 = OK
*                                       ! 1 = error PLL 100.0 MHZ (LO1)
```

```
*                                       ! 2 = error PLL 100.1 MHZ (LO1)
*                                       ! 3 = HP synthe locked on internal source
integer*4 ERPLL(5) ! PLL error (per antenna)
! 6 bits for LO3 and 1 bit LO1
! (currently not used)

integer*4 NERRCHIP ! number of bad chips
integer*4 ERRCHIP_UNIT(4) ! unit identification (1 to nb_corr)
integer*4 ERRCHIP_CARD(4) ! card identification (0 to 11)
integer*4 ERRCHIP_NR(4)   ! chip identification (0 to 15)

integer*4 ERR_IF(nb_corr) ! IF error

        end structure
```

Description du tableau des spectres en crosscorrelation

Les spectres en cross-correlation sont stockes dans les tableau de complexes U_SPECT(128,2,nb_corr,2,10) (spectres non corrigees des variations de phase atmospherique) et C_SPECT(128,2,nb_corr,2,10) (spectres corriges en temps reels des variations de phase atmospherique) .

- Le premier indice represente le numero du canal. Il est compris entre 1 et 128. En tmf 4, il y a 32 canaux significatifs. En tmf 2, il y a 64 canaux significatifs. En tmf 1, il y a 128 canaux significatifs. En tmf 1 passalong, il y a 256 canaux significatifs. Ces canaux sont stockes consecutivement en utilisant le fait que, dans le cas du tmf 1 passalong, il n'y a qu'une sous bande LO3.
- Le deuxieme indice represente l'identification de la sous-bande LO3. 1 correspond a la *lower* bande LO3. 2 correspond a la *upper* bande LO3.
- Le troisieme indice represente le numero d'unite. Il est compris entre 1 et 6.
- Le quatrieme indice represente l'identification de la sous-bande LO1, egalement appelee bande ciel. 1 correspond a la *lower* bande LO1. 2 correspond a la *upper* bande LO1.
- Le cinquieme indice represente l'identification de la ligne de base. Il y a 10 lignes de bases si on dispose de 5 antennes. Les lignes de base sont stockees dans l'ordre 1*2 1*3 2*3 1*4 2*4 3*4 1*5 2*5 3*5 4*5.

Description du tableau des spectres en autocorrelation

Les spectres en autocorrelation sont stockes dans le tableau de reels R_SPECT(128,2,nb_corr,5).

- Le premier indice represente le numero du canal. Il est compris entre 1 et 128. En tmf 4, il y a 32 canaux significatifs. En tmf 2, il y a 64 canaux significatifs. En tmf 1, il y a 128 canaux significatifs. En tmf 1 passalong, il y a 256 canaux significatifs. Ces canaux sont stockes consecutivement en utilisant le fait que, en tmf 1 passalong, il n'y a qu'une sous bande LO3.

- Le deuxieme indice represente l'identification de la sous-bande LO3. 1 correspond a la "lower" bande LO3 2 correspond a la "upper" bande LO3
- Le troisieme indice represente le numero d'unite. Il est compris entre 1 et 6.
- Le quatrieme indice represente l'identification de l'antenne.

Les donnees temporelles

Les donnees temporelles sont les donnees qui sont enregistrees chaque seconde par le systeme. Elles comportent a la fois des parametres de l'observation lus dans les common ACQUIS ou GENERAL, et des moyennes par sous-bande sur les spectres fournies par les unites chaque seconde. La structure des donnees temporelles est decrite dans le fichier /control/correl/temporal_structure.f dont le listing est donne ci-dessous.

```
$ more /control/correl/temporal_structure.f
* temporal_structure.f
*
       integer   dim_temporal
       parameter (dim_temporal =
     1      15 + nb_ant*10 + 7*nb_ant*nb_rec +2*nb_corr*2*nb_base)
          ! 15 + nb_ant*10 + 7*nb_ant*nb_rec +2*nb_corr*2*nb_base)


! structure of temporal data

structure /temporal_data/
 union
 map
   real*8 UTC ! UTC (sec) at the mid-period
   integer*4    VETO
   real*8 SVEC(3)  ! source coordinates (horary)
   real*4       OFFX(5)         ! offset angle (sec) (=xoffs in
        ! antenna_structure.for)
   real*4       OFFY(5)  ! offset angle (sec) (=yoffs in
! antenna_structure.for)
   real*4 FOCUS(5)        ! focus position (mm) (=sfocus in
! antenna_structure.for)
   real*4 TOFF(5)  ! UT associated to OFFX, OFFY and FOCUS
   real*4 DELCAL(5)  ! astronomical delay (calculated by DLYPHA)
   real*4 DELLIN(5,nb_rec) ! applied delay
   real*4 DELAYx(5,nb_rec) ! delay command (ns) (.COR DELAY x)

   real*4 PHALO1(5,nb_rec) ! astronomical phase offset (DLYPHA)
   real*4 PHASEx(5,nb_rec) ! phase offset command (.COR PHASE x)
   real*4 RATE(5,nb_rec)   ! phase rate for LO1
   real*4 CABLE(5,nb_rec)  ! cable length (turns) !RL
   real*4 POWER(5,nb_rec)  ! continuum detectors
   real*4 TESTO(5)  ! test parameters
```

```
   real*4 TEST1(5,5)   ! test parameters :
  ! test1(1,iant)=A(iant).Az.xdel
  ! test1(2,iant)=A(iant).El.xdel

   real*4 TEMP_DATA(2*nb_corr*2*nb_base) ! temporal data
  ! cross: TEMP_DATA(Re/Im,nb_corr,UL01/LL01,nb_base)
  ! auto : TEMP_DATA(nb_corr,nb_ant)
 end map
   map
   real*4       TEMP_ARRAY(dim_temporal)
 end map
end union
end structure
```

La valeur moyenne du signal dans chaque sous-bande est stockee dans le tableau TEMP_DATA.

En crosscorrelation, l'ordre des donnees correspond a un tableaux de reels TEMP_DATA(2,nunit,2,nbase).

- Le premier indice, 2, represente les parties reelles et imaginaires des donnees.
- Le deuxieme indice correspond au numero d'unite.
- Le troisieme indice represente l'identification de la sous-bande LO1 (lower ou upper).
- Le dernier indice correspond au numero de la ligne de base.

En autocorrelation, l'ordre des donnees correspond a un tableaux de reels TEMP_DATA(nunit,nant).

- Le premier indice correspond au numero d'unite.
- Le deuxieme indice correspond au numero d'antenne.

Les moyennes sont calculees sur l'ensemble lower+upper bande LO3. L'observateur specifie, par l'intermediaire des variables *lfmin(6)* et *lfmax(6)* dans le common ACQUIS, la partie du spectre sur laquelle on calcule cette moyenne.

4.2.4 Description des donnees locales

La zone de donnees locales decrite dans /control/correl/correl_local.f contient des parametres propres a l'application ainsi que des tableaux permettant de stocker les spectres et les donnees temporelles avant leur recopie dans CONTCO.

Elle comporte egalement un tableau de 7 enregistrements Y(7). Chaque enregistrement contient des parametres propres a l'une des periodes [T-4,T-3[, [T-3,T-2[, [T-2,T-1[, [T-1,T[, [T,T+1[, [T+1,T+2[et [T+2,T+3[. Ceci est utile pour effectuer facilement des traitements relatifs a differentes periodes.

4.3 Sequencement des operations

On peut suivre facilement le sequencement des operations en lisant le programme principal dans le fichier /control/correl/correl.f. Schematiquement, la tache correl est en attente du flag 1 seconde et, des qu'elle recoit cette interruption, effectue un traitement qui depend de l'etat d'avancement de l'acquisition defini dans la variable *process_status*. On distingue 4 etats:

process_status = SLEEPING Dans cet etat, il n'y a pas d'acquisition. Un message est envoye vers le chassis phaser afin de recuperer le niveau des detecteurs continuum et rearmer le watchdog. Si le flag FLG_COR est positionne, on va demarrer une acquisition. Les communications avec les chassis correlateur sont initialisees. On passe alors a l'etat *process_status = STARTING*

process_status = STARTING Dans cet etat, l'acquisition n'est pas encore en cours, mais elle va commencer sur la prochaine interruption 1 seconde. Un message est envoye vers le chassis phaser afin de recuperer le niveau des detecteurs continuum (utile pour la correction de variation de phase atmospherique) ainsi que les phases des cables. Ce message contient aussi les rates et phases LO1 a appliquer la seconde suivante. Les delais et phases LO3 sont alors calcules pour la prochaine seconde et envoyes vers les unites correlateur en meme temps que la configuration (tmf,auto/cross) a effectuer sur les chassis.

process_status = ACQUIRING Dans cet etat, l'acquisition est en cours sur les correlateurs. Comme dans les etats precedents, un message est envoye vers le chassis phaser afin de specifier les rates et phases LO1 et de recuperer les puissances totales et les phases des cables. Ensuite la tache *correl* envoie vers chaque unite les commandes de delais et phases LO3 et recupere les moyennes des spectres acquis dans la periode [T-2,T-1[. Si la periode specifiee pour l'acquisition est terminee, correl va positionner un flag pour la tache *correl_spectra* qui ira recuperer les spectres dans chaque unite.

process_status = ENDING Dans cet etat, l'acquisition est terminee. Mais les donnees n'ont pas encore ete rapatriees. Cet etat dure 3 secondes. Un message est envoye vers le chassis phaser afin de recuperer les puissances totales (utiles pour la correction de phase atmospherique) . Puis un message est envoye vers chaque unite contenant la correction de phase atmospherique et le veto eventuel. En retour, la tache correl recoit les donnees temporelles concernant les periodes [T-2,T-1[.

Un resume faisant apparaitre les principales actions est presente sur la page suivante.

```
! phase d'initialisation
    call sem_create          : rattachement au groupe de semaphores utilises
    call start_correl_spectra : demarrage de la tache correl_spectra
    call INIT_LOCAL_DATA     : initialisation de variables dans correl_local
    call INIT_COMMUNICATION_PHASER(sock_phaser)
    process_status = SLEEPING


! boucle d'attente d'un signal seconde et traitement suivant l'etat

debut_boucle:
    call waitfr(FLG_UT)
    FLAG_COR = readef (FLG_COR)  : lecture FLG_COR (debut/fin d'acquisition)

    switch (process_status)

     case SLEEPING:
      call PHASER                : lecture puissances totales et phases cables
      if (FLAG_COR.eq.SET)       : si debut d'acquisition
        call INIT_COMMUNICATION  : ouverture des communications (connect(,,,))
        call waitfr(FLG_TRK)     : attente flag tracking
        call SET_REQUEST(MASK_SOURCE) :selection source de bruit ou recept
        process_status = STARTING
      endif

     case STARTING:
      if (FLAG_COR.eq.SET)       : fin d'acquisition demandee
        call PHASER              : Read continuum detectors and cable phases
        call  PREPARE_MESSAGE    : prepare message for "abort"
        call  SEND_MESSAGE
        process_status = ENDING
      else
        call DOPPLER             : test chgt de frequence pour effet Doppler
        call DELAY_PHASE_ASTRO    ! calculate delay and phase for t+1
        call SET_REQUEST(MASK_PHASES)
        call PHASER         ! Send phases and read continuum detectors
        call DELAY_PHASE_TOTAL    ! add phasemeter phase to astro phase
        call PREPARE_MESSAGE
        call SEND_MESSAGE        ! message for period [t+1,t+2[
        process_status = ACQUIRING
      endif

     case ACQUIRING:
      if (FLAG_COR.eq.SET)       : fin d'acquisition demandee
        call PHASER              : Read continuum detectors and cable phases
        call PREPARE_MESSAGE    : prepare message for "abort"
        call SEND_MESSAGE
        process_status = ENDING
```

```
  else
    if (mode repeat et 1ere seconde d'un nouveau subscan) call DOPPLER
    call UPDATE_ANTENNA_PARAMETERS ! update Y(t).focus,Y(t).offset,..
    call UPDATE_CONTINUUM_LEVELS   ! update Y(tminus2).power
    call DELAY_PHASE_ASTRO        ! calculate delay and phase for t+1
    call SET_REQUEST(MASK_PHASES)
    call PHASER         ! Send phases and read continuum detectors
    call DELAY_PHASE_TOTAL        !add phasemeter phase to astro phase
    call ATM_PHASE_CORRECTION
    call PREPARE_MESSAGE
    call SEND_MESSAGE
    if (mode repeat et acquisition du precedent subscan terminee)
      call WRITE_COMMON     ! write only temporal data
      call setef(FLG_SPEC)  ! signal to correl_spectra task
    endif
    call UPDATE_DATA         : mise a jour des donnees temporelles
    if ( fin du subscan et fin du mode repeat)
      process_status = ENDING
    endif


 case ENDING:

  call  PHASER
  if (FLAG_COR.eq.SET)       : fin d'acquisition demandee
    call PREPARE_MESSAGE    : prepare message for "abort"
    call SEND_MESSAGE
    process_status = SLEEPING
  else
    call  UPDATE_CONTINUUM_LEVELS  ! update Y(tminus2).power
    if (acquisition durant periode [T-2,T-1[ ou [T-1,T[ )
      call ATM_PHASE_CORRECTION !update Y().ph_corr or compute mean
      call PREPARE_MESSAGE      !to receive message with veto/no veto
      call SEND_MESSAGE
    elseif (acquisition durant [T-3,T-2[) : donnees disponibles
      call WRITE_COMMON
      call setef(FLG_SPEC)      : flag pour correl_spectra
      call corr_clean(sock_M(i)) : fermeture des sockets
      process_status = SLEEPING
    endif
end switch (process_status)
goto debut_boucle
```

4.4 Description des modules

Dans ce paragraphe sont decrits les differents modules Fortran ou C qui sont linkes pour donner la tache *correl*. Pour chaque module, on donne une breve description des subroutines contenues dans celui-ci.

- *correl.f :*

  correl tache principale

  init_local_data initialisation de donnees dans correl_local

  increment_time_pointers mise a jour des pointeurs dans le tableau Y

- *atm_phase_correction.f :*

  atm_phase_correction calcul de corrections de phase atmospherique

- *delay_phase_astro.f :*

  delay_phase_astro calcul des delais et phases astronomiques

  vectpg calcul du vecteur source en coordonnees horizontales

- *delay_phase_total.f :*

  delay_phase_total addition phases du cable et commandes de phase aux phases astronomiques

- *correl_routine.c :*

  corr_init ouverture des communications avec les correlateurs

  corr_set envoi des parametres (delais,phases) et reception des donnees temporelles

  corr_data lecture des spectres integres dans les unites

  cleanup fermeture d'une liaison HP-correlateur

  corr_clean fermeture d'une liaison HP-correlateur (appelle cleanup)

- *init.f :*

  init_communication ouverture des communications avec les correlateurs (appelle corr_init)

- *matr.f :*

  matr changement de coordonnees

- *message.f :*

  prepare_message preparation des messages a envoyer aux unites

  send_message envoi des messages (appelle corr_set)

- *phaser.f :*

  phaser

  prepare_message_phaser

analyse_message_phaser

read_request_type

- *prepare_client_param.f :*

  prepare_client_param mise a jour de parametres

- */control/common/set_request.f :*

  set_request ecriture de la requete dans la variable REQTYP

- *start_correl_spectra.f*

  start_correl_spectra demarrage de la tache *correl_spectra*

- *update_data.f*

  update_data ecriture des donnees temporelles [T-2,T-1[dans *correl_local*

  update_continuum_levels lecture du niveau des detecteurs continuum dans GENERAL et copie dans les donnees [T-2,T-1[de *correl_local*

  update_antenna_parameters copie des parametres d'antenne dans les donnees [T,T+1[de *correl_local*

  update_param determination des parametres d'observation a partir de donnees de GENERAL

- *write_common.f*

  write_common ecriture des donnees temporelles dans le common CONTCO

4.5 Creation de la tache correl

Il existe un fichier Makefile dans le directory /control/correl qui permet de creer correl .

4.6 Lancement de la tache correl

Le programme *correl* est lance par OBS.

5 Description de la tache correl_spectra (HP)

La tache *correl_spectra* recupere les spectres disponibles a la fin d'une acquisition. Ce traitement est dissocie de *correl* car il n'y a pas de contrainte de temps pour le rapatriement des spectres. La tache *correl* sait a quel moment les donnees des correlateurs deviennent disponibles et positionne alors le flag *flg_spec*

5.1 Description des modules

Dans ce paragraphe sont decrits les differents modules Fortran ou C qui sont linkes pour donner la tache *correl_spectra*. Pour chaque module, on donne une breve description des subroutines contenues dans celui-ci.

- correl_spectra.f :

  correl_spectra tache principale

  init_param initialisation de donnees dans le common SPECTRA_LOCAL

- write_spectra.f :

  write_spectra ecriture des spectres dans le common CONTCO

6 Description de la tache calibr_LO3 (HP)

6.1 Principe

Les rotateurs LO3 sont sensibles aux variations de temperature et peuvent ainsi devenir faux de quelques degres. Il est bon de les recalibrer regulierement. Cette operation est egalement indispensable lorsque les chassis IF sont mis hors tension.

Cette operation comporte 4 phases :

- Tout d'abord, on ecrit dans la table de commande des rotateurs (256 valeurs sur 12bits) les valeurs standard.
- Ensuite, on envoie successivement les 256 commandes de phase prevues dans le systeme (0, 1/256 tour, 2/256 tour ... 255/256 tour) et on mesure la phase obtenue. On calcule alors une nouvelle table de commande qui permet de se rapprocher de valeurs theoriques.
- On ecrit dans la table de commande des rotateurs (256 valeurs) les valeurs calculees
- On effectue alors une verification en renvoyant successivement les 256 commandes disponibles et on mesure les erreurs de phase obtenues. Ces dernieres sont stockees dans des fichiers.

Le programme /control/correl/calibr_LO3 permet de calibrer les rotateurs de phase des 6 unites en 15 minutes environ. Il configure les parametres pour la tache correl et lance les acquisitions necessaires a la calibration. Il faut donc arreter OBS et RDI.

6.2 Mode operatoire

Voici dans l'ordre les operations a executer.

- 1. Faire un reset sur toutes les unites.
- 2. Arreter OBS et RDI .

  Par contre, il faut laisser tourner la tache correl.
- 3. Lancer le programme calibr_LO3

  Les seuls parametres a saisir, en debut d'execution, sont les unites a calibrer parmi les 6 unites existantes (0=unite non selectionnee, 1=unite selectionnee).

  Exemple d'execution (les unites 1 et 2 sont selectionnees) :

```
$/control/correl/calibr_L03
  selected units (6 values 0/1) ?
1,1,0,0,0,0
 xa.slband= 1 1 0 0 0 0

  writing S law ...
waiting 80 seconds
  IFPB
waiting 12 seconds
```

```
waiting 15 seconds
waiting 12 seconds
  measuring phases and computing new law ...
waiting 280 seconds
  writing new law ...
waiting 80 seconds

  checking calibration
  IFPB
waiting 12 seconds
waiting 15 seconds
waiting 12 seconds
  measuring phases ...
waiting 280 seconds

  Results are stored in each unit :
 /r0/rot_u0x.dat and /r0/verif_u0x.dat  (x is the unit number:1,2,3,4,5,6)
$
```

6.3 Contenu des fichiers de resultats

Les fichiers produits peuvent etre consultes en faisant un telnet sur chaque unite ou en les rapatriant par ftp.

Le fichier /r0/rot_u01.dat, visible en se loggant sur corl01, contient un indice (0 a 255) et 4 colonnes correspondant aux tables calculees et ecrites dans les 4 rotateurs de l'unite 1.

```
$ list /r0/rot_u01.dat
  0      0      0      0      0
  1     40     37     40     36
  2     80     73     79     72
  3    119    108    118    107
  4    158    143    156    141
  5    196    178    193    175
  6    233    212    230    208
  7    270    245    266    240
  8    306    278    302    272
  9    342    311    336    303
 10    377    343    371    333
 11    412    374    405    362
 12    446    406    438    391
 13    480    436    471    419
 14    513    466    503    447
...
250   3862   3674   3739   3543
251   3886   3700   3763   3567
252   3911   3726   3788   3591
253   3936   3753   3814   3615
```

```
254   3962   3781   3840   3641
255   3988   3809   3867   3666
```

Il faut verifier que la derniere ligne contient des valeurs inferieures a 4096. Si ce n'est pas le cas, ca signifie que la plage de fonctionnement du rotateur est inferieure a 360 degres. Il faut donc le changer.

Le fichier /r0/verif_u01.dat,visible sur corl01, contient en 1ere colonne un indice i (0 a 255), en 2eme colonne la valeur theorique (i/256*360 degres) et dans les 4 colonnes suivantes les erreurs mesurees, en degres, par rapport a ces valeurs theoriques.

```
$list /r0/verif\_u01.dat

 i phase_required    error (4 rotators)
  0      0.000    0.000    0.000    0.000    0.000
  1      1.406   -0.021   -0.023   -0.011    0.002
  2      2.812   -0.013   -0.010    0.006   -0.011
  3      4.219   -0.016   -0.019   -0.003    0.007
  4      5.625   -0.018    0.001   -0.014   -0.024
  5      7.031    0.006    0.004   -0.009   -0.028
  6      8.438   -0.018   -0.002   -0.017   -0.007
...
250    351.562    0.059    0.025   -0.029    0.006
251    352.969    0.013    0.009   -0.029   -0.018
252    354.375    0.031    0.013   -0.051   -0.022
253    355.781    0.057    0.033   -0.009   -0.011
254    357.187    0.042    0.021   -0.046   -0.010
255    358.594    0.043    0.028   -0.022    0.004
```

7 Description du logiciel des correlateurs.

Les logiciels ont ete developpes en C sous le systeme OS9. Le logiciel est commun aux 12 unites (maitres et satellites). Les unites n'ont pas de dique dur. Tous les fichiers se trouvent dans un disque virtuel sur le HP. On y accede, depuis les correlateurs, par NFS. Le logiciel qui nous interesse se trouve dans le directory /dd/CORREL5.

7.1 Description des donnees

Le logiciel est divise en plusieurs taches. De ce fait, il est necessaire d'avoir un module de donnees installe en memoire qui joue le role de common. La structure de ce module est decrite dans /dd/CORREL5/common.h. Une fois installe en memoire, il apparait avec le nom *common_data*.

7.2 Description des taches

On distingue 6 taches differentes :

inter installation des routines traitant les interruptions

correl tache gerant les echanges avec le HP et pilotant l'ensemble de l'acquisition

calcul tache de calcul (integration des donnees, FFT)

tweak tache effectuant l'ajustement des samplers

setIF tache gerant les echanges avec le microcontroleur 8052

spectra tache effectuant le rapatriement des spectres vers le HP

Dans les paragraphes suivants, ces taches sont decrites plus precisemment.

7.2.1 Tache inter

Cette tache est lancee la premiere. Elle a pour but d'installer les routines traitant les interruptions. Elle installe egalement en memoire le module de donnees *common_data* qui est utilise par les routines d'interruptions. Contrairement aux autres taches qui restent actives jusqu'au prochain reset de l'unite, la tache inter se termine apres avoir installe les routines d'interruption. Le microprocesseur doit gerer 4 interruptions propres a notre application. Ces interruptions arrivent par le bus VME (par opposition aux interruptions dites locales).

- L'interruption 15.625ms est generee, comme son nom l'indique, toutes les 15.625ms soit 64 fois par seconde. Cette periode est egale a la moitie de la periode des fonctions de Walsh qui est de 31.25ms. C'est parce que les compteurs a l'interieur des cartes correlateur ne permettaient pas d'integrer plus de 17.5ms que cette periode de 15.625ms a ete choisie.

  A chaque interruption de 15.625ms, le micro doit lire les donnees dans les 10 cartes correlateur (soit 2560 canaux) et les integrer dans le buffer adequat suivant les fonctions de Walsh appliquees a cet instant.

  C'est l'interruption la plus prioritaire pour l'application car il est imperatif d'avoir lu les 2560 canaux avant l'interruption suivante. Physiquement, c'est l'interruption IRQ5 sur le bus VME. Elle est placee au niveau de priorite 5 par le programme install_inter.c.

- L'interruption 1 seconde est generee toutes les secondes a partir du bus temps. Essentiellement, elle met a jour des pointeurs et genere un signal pour la tache calcul. Au demarrage, elle permet la synchronisation avec le cycle des fonctions de Walsh. Physiquement, c'est l'interruption IRQ4 sur le bus VME. Elle est placee au niveau de priorite 4 par le programme install_inter.c.
- L'interruption 1 minute est generee toutes les minutes. Elle n'est pas utilisee par l'application. Physiquement, c'est l'interruption IRQ3 sur le bus VME. Elle est placee au niveau de priorite 3 par le programme install_inter.c.
- La derniere interruption est generee lorsqu'il y a ouverture de la PLL 5 Mhz. La routine d'interruption positionne un flag dans *common_data* mais la valeur de ce flag n'est pas exploitee actuellement. Physiquement, c'est l'interruption IRQ2 sur le bus VME. Elle est placee au niveau de priorite 2 par le programme install_inter.c.

La tache inter doit s'executer avec le privilege superuser. Ce privilege est attribue par la commande *fixmod* dans le fichier makefile.

7.2.2 Tache correl

La tache *correl* est la tache principale. C'est elle qui lance les taches *calcul*, *tweak*,*spectra* et *setIF*. Au demarrage cette tache cree la table de demodulation utilise par la routine d'interruption 15. 625ms. Cette table est installee en memoire dans le module *demod_data*. Elle s'alloue de la memoire pour differents tableaux. Elle cree l'evenement *ev_mess*. Elle demarre sa fonction de serveur. Elle demarre les taches *calcul*, *setIF* et *tweak*.

Une fois ces initialisations terminees, la tache *correl* boucle en attente des connexions du HP. Apres chaque connexion, elle boucle sur la reception de messages jusqu'a ce que le HP prenne l'initiative de fermer la connexion. A ce moment, elle se remet en attente de connexion.

Lorsqu'elle recoit le message envoye par le HP toutes les secondes, elle effectue des mises a jour en fonction des parametres recus puis elle renvoie les donnees temporelles correspondant a la periode [T-2,T-1[. C'est *correl* qui configure les chips correlateur ainsi que les lignes a retard si necessaire. Lorsqu'elle a fini de traiter ce message, elle positionne l'evenement *ev_mess* qui indique a la tache *calcul* que le message du HP a ete recu et donc que le flag eventuel concernant les donnees [T-2,T-1[est connu.

7.2.3 Tache calcul

La tache *calcul* effectue tous les calculs qui permettent de generer a partir des donnees lues dans les cartes correlateur les spectres qui seront envoyes au HP. La tache *calcul* boucle en attente d'un signal envoye chaque seconde par la routine d'interruption lorsque le correlateur effectue des acquisitions.

A la reception de ce signal, la tache *calcul* effectue successivement les traitements relatifs au donnees correspondant a la periode [T-1,T[, puis ceux relatifs a la periode [T-2,T-1[.

Traitements relatifs aux donnees de la periode [T-1,T[

En autocorrelation, les donnees lues, qui correspondent donc a des fonctions d'autocorrelation, sont simplement copiees dans un autre buffer. Les donnees temporelles qui seront envoyees au HP une seconde plus tard sont preparees a ce moment la.

En crosscorrelation, on applique la correction de clipping puis on calcule la FFT sur les donnees correspondant a la periode [T-1,T[. On applique au spectre obtenu les corrections detaillees dans le rapport *Real time processing of correlator data* de R. Lucas concernant le traitement des donnees de l'interferometre). On calcule egalement les donnees temporelles.

En crosscorrelation sur une IFPB, la FFT n'est pas calculee immediatement car dans ce cas la correction de clipping prend trop de temps. Les donnees brutes sont simplement integrees et le spectre n'est calcule qu'a la fin de l'integration.

La facon dont est faite la correction de non-linearite est detaillee dans le paragraphe *Correction de clipping* du chapitre *Description detaillee du traitement des donnees.*

Traitements relatifs aux donnees de la periode [T-2,T-1[

Lorsque les traitements relatifs a la periode [T-1,T[sont termines, la tache *calcul* traite les donnees de la periode [T-2,T-1[. Elle ne peut commencer ces traitements que lorsque le message venant du HP et indiquant si ces donnees sont valides a ete recu par l'unite. C'est la tache *correl* qui recoit ces messages et qui positionne l'evenement *ev_mess* signalant a la tache *calcul* qu'elle peut traiter les donnees [T-2,T-1[.

Si le HP a donne l'ordre de flagger ces donnees, aucun traitement n'est effectue sur ces donnees sinon les donnees sont ajoutees aux donnees precedemment acquises. Le traitement est termine si ce n'est pas la fin d'une acquisition.

Traitements specifiques aux fins d'acquisition

Si la periode [T-2,T-1[correspond a une fin d'acquisition, ou plutot a la derniere periode avant un changement de buffer, il faut realiser differents traitements en fonction du mode de l'acquisition (auto ou crosscorrelation, IFPB ou donnees issues des recepteurs).

Dans le cas d'une autocorrelation, on effectue sur les donnees integrees tout d'abord un test de detection des chips morts. Ensuite on effectue la correction de non-linearite puis, apres l'apodisation, le calcul de la FFT. On normalise les donnees par les niveaux d'echantillonnage. On precalcule des coefficients qui seront utilisees pour la correction de clipping en crosscorrelation (dans le cas du signal venant des recepteurs) ainsi que des facteurs de normalisation dans le cas des IFPB.

Dans le cas d'une crosscorrelation IFPB, on effectue egalement la correction de non-linearite puis le calcul de la FFT sur les donnees integrees. Puis on normalise les spectres avec les spectres d'autocorrelation en IFPB (voir document de R. Lucas concernant le traitement des donnees de l'interferometre).

Dans les autres cas de crosscorrelation, on effectue une normalisation en fonction du temps d'integration et les spectres sont recopies dans la zone ou la tache *correl_spectra* recupere les spectres a envoyer au HP.

7.2.4 Tache tweak

Le principe du *tweak* ou ajustement des samplers est decrit en detail dans le paragraphe *Ajustement du niveau des samplers* dans le chapitre *Annexe.*

En resume, il existe une valeur du rapport v_0/σ, ou v_0/σ designe le niveau du seuil d'echantillonnage en fonction du niveau rms du signal d'entree, qui optimise la sensibilite du correlateur. Le maximum de sensibilite 0.87 est obtenu pour v_0/σ egal a 0.906. La procedure de tweak consiste a approcher de cette valeur par iterations.

Au depart les samplers sont positionnes a la valeur 128. On calcule le rapport v_0/σ a partir de la valeur du canal 0.

$$v_0/\sigma = \sqrt{2} * aierf(1 - canal\ 0)$$

et on en deduit le niveau du seuil d'echantillonage $v_{optimal}$ qui donnerait un rapport $v_{optimal}/\sigma$ egal a 0.906.

$$v_{optimal} = 0.906\ \sigma = \frac{0.906\ v_0}{\sqrt{2}\ aierf(1 - canal\ 0)}$$

On regle le sampler a ce niveau et on reitere le calcul.

Les 5 premieres iterations sont faites avec un temps d'integration de 31.25ms. La derniere iteration est fait avec un temps d'integration de 125ms afin d'avoir une plus grande precision. Une operation de tweak est donc executee en moins d'une seconde.

Les samplers sont reglables de 0 a 4095. Mais la plage utilisable est situee entre 800 et 2500 (problemes de linearite).

La tache *tweak* est en attente d'un signal genere par la routine d'interruption 1 seconde. A la reception de ce signal, elle effectue ses 6 iterations puis restaure la configuration preexistante dans les chips correlateur et enfin se remet en attente d'un signal indiquant qu'il faut executer un tweak.

Les niveaux trouves pour les samplers sont sauvegardes dans le common *common_data* et renvoyes vers le HP comme des donnees temporelles.

La tache *tweak* ne tourne que dans le chassis *maitre*. On tweake successivement les samplers des bandes Lower LO3 puis Upper LO3. On selectionne le mode UU (copie des donnees Upper LO3 sur la sortie Lower LO3 des samplers) pour tweaker la bande Upper LO3.

7.2.5 Tache setIF

La tache *setIF* gere les echanges entre le microprocesseur 68030 et le microcontroleur 8052 qui pilote la partie IF. Les echanges se font via une liaison RS232 sur le port /t1 de la carte CPU du correlateur. Cette tache ne tourne bien sur que dans le chassis *maitre*.

Au depart, on ouvre un chemin en ecriture et en lecture sur le port /t1 . Avant chaque ecriture, le logiciel teste s'il n'y a pas des caracteres prets a etre lus. Si c'est le cas, il commence par les lire avant de faire son ecriture. Ensuite il attend les caracteres renvoyes en echo par le microcontroleur. Le fait que le microcontroleur renvoie en echo tous les caracteres oblige a laisser des tempos entre chaque envoi de commande. Dans la mesure ou on dispose de 1 seconde pour envoyer toutes les commandes, ce n'est pas genant. La liste des commandes ainsi que leur syntaxe sont decrites dans le document *Commande du rack IF* de J.Y Mayvial.

La tache *setIF* est demarree par la tache *correl*. Au depart, elle envoie la commande S qui teste l'etat des alimentations, et des oscillateurs locaux LO3 et LO4, puis se met en attente d'un signal venant de la tache *correl* qui lui signale qu'il y a des commandes a envoyer. Si une erreur est trouvee, un code d'erreur est ecrit dans la variable *statusIF* qui se trouve dans *common_data*. Si aucune erreur n'est detectee, la tache *setIF* peut envoyer ses commandes vers le microcontroleur.

- S'il y a un changement de configuration, la tache setIF envoie successivement une commande S pour verifier l'etat du hard, puis une commande de selection de filtre (10, 20, 40 ou 80) et une commande de frequence sur le LO3. Ensuite elle envoie les commandes de phase initiale et de rate (variation de la phase pendant la seconde suivante) aux 4 rotateurs de phase.

Remarque: Il n'y a que 4 rotateurs correspondant aux antennes 1, 2, 3 et 4 car les commandes de phases sont calculees par rapport a la cinquieme antenne.

- Lorsqu'il n'y a pas de changement de configuration, la tache setIF envoie uniquement les commandes de phase et de rate.

7.2.6 Tache spectra

La tache *spectra* est en attente d'un message du HP lui demandant de transmettre les spectres disponibles.

7.3 Description des differents modules

Chaque tache est composee de plusieurs modules. Un makefile general dans /dd/CORREL5 permet de reconstruire toutes les taches.

7.3.1 Modules composant la tache inter

- *inter.a*

  InitISR installation des 4 routines d'interruption

  ISR1 traitement de l'interruption 15.625ms

  ISR2 traitement de l'interruption 1 seconde

  ISR3 traitement de l'interruption minute

  ISR4 traitement de l'interruption "erreur PLL"

- *install_inter.c*

  main() "main" programme de la tache inter. Initialise des donnees, installe les routines d'interruption et autorise les interruptions au niveau du VIC

- *addr.c*

  addr_module(name,size) cree un module de donnees en memoire ou, s'il existe, etablit un lien avec ce module.

7.3.2 Modules composant la tache correl

- *correl.c*

  sighand(signal) signal handler

  main() "main" programme de la tache correl. Initialise des donnees, demarre les autres taches et se met en attente de connexions, puis de messages du HP

  start_server() ouverture d'un socket et "bind" de ce socket a une structure contenant les parametres du "local host"

  cleanup(socket) fermeture d'un socket

  perror(mess) affichage d'un message et du code d'erreur errno

abort_acquisition() arret de l'acquisition

send_data() envoi de donnees (spectres) vers le HP

- *addr.c*

addr_module(name,size) cree un module de donnees en memoire ou, s'il existe, etablit un lien avec ce module.

- *config_corr.c*

set_corr() ecriture des mots de configuration dans les chips correlateur en fonction du mode (auto ou cross) et du tmf(0,1,2 ou 4)

set_delay() ecriture des consignes de delais dans les cartes sampler

- *create_demod.c*

create_demod_data() creation du module memoire "demod_data" contenant les infos de demodulation des fonctions de Walsh

- *scale_correl.c*

prepare_OBS_amplitude_factor() calcul de l'amplitude des coefficients de correction qui seront utilises pour les crosscorrelations sur le ciel

- *util.c*

calculate_time(,,,,) utilise en test pour des mesures de performances

write_file(,,) ecriture dans un fichier. utilise uniquement en test

7.3.3 Modules composant la tache calcul

- *calcul.c*

main() "main" programme de la tache calcul. Contient l'enchainement des operations a realiser pour obtenir les spectres finaux

- *addr.c*

addr_module(name,size) cree un module de donnees en memoire ou, s'il existe, etablit un lien avec ce module.

- *apodize.c*

init_apodization_table() initialisation de table pour accelerer les calculs d'apodisation

apodization_auto(,,,,) Multiplication de la fonction d'autocorrelation par $(cos(i/n), sin(i/n))$ ou i est un indice et n le nombre de canaux pour decaler le spectre d'un demi canal. Multiplication par une fonction parabolique si l'apodisation est demandee

apodization_cross(,,,,) Multiplication de la fonction de correlation par $(cos(i/n), sin(i/n))$ ou i est un indice et n le nombre de canaux pour decaler le spectre d'un demi canal. Multiplication par une fonction parabolique si l'apodisation est demandee

- *clip.c*

clip_auto(tmf,buffer) En autocorrelation, rearrangement des canaux et appel a fourlev pour la correction de clipping

clip_cross() En crosscorrelation, avec signaux venant des recepteurs (correlation faible), uniquement rearrangement des canaux

clip_cross_IFPB() En crosscorrelation IFPB (correlation forte), rearrangement des canaux et appel a fourlev pour la correction de clipping

- *coeff.c*

coeff() calcul des coefficients de correction de non-linearite valables pour une correlation faible

- *funct.c*

cubspl$(n, \tau, c1, c2, c3, c4, ibcbeg, ibcend)$ Calcul des 4 premiers coefficients c1, c2, c3 et c4 du polynome de l'interpolation par splines cubiques
Pour $x \epsilon [\tau(i), \tau(i+1)[$, le spline F est donne par:
$F(x) = c1(i) + h\ (c2(i) + h\ (c3(i) + h\ c4(i)/3)/2)$
ou $h = x - \tau(i)$
τ, c1, c2, c3 et c4 sont des tableaux de dimension N egal au nombre de noeuds de l'interpolation.

cubint(n,x,y,c2,ci) calcul du tableau *ci* des integrales entre x(1) et x(i) de la fonction y.

kspl(n,x,t,k)

splf(n,x,y,cm,t,k)

splfk(n,x,y,cm,t,k) calcul de la valeur en un point t d'une fonction interpolee par des splines cubiques (appelle kspl et splf)

- *fft.c*

fft_auto(tmf,do_apodize,buffer) calcul de la FFT en autocorrelation. Fait appel a fftcfr pour le calcul de la FFT

fft_cross(tmf,do_apodize,buffer) calcul de la FFT en crosscorrelation. Fait appel a fftcfr pour le calcul de la FFT

init_bitrev_array() initialise le tableau des inversion d'index. Appelle indbr

raz_data() raz de tableaux

init_descrip() initialisation de descripteurs pour le calcul de la FFT (appelle inifftbl())

- *fftcmplx.a*

fftcfr(xr,xi,descrip) calcul de FFT sur valeurs complexes suivant l'algorithme SPLIT RADIX

- *fourlev.c*

fourlev(nchannel,data,r1,r2) correction de clipping valable en auto et en crosscorrelation pour un correlateur a 4 niveaux

aierf(x) calcul de aierf(x) par approximation par un polynome

- *indbr.c*

indbr(ind,m) indirection pour le rearrangement des donnees de la FFT

- *inifftbl.a*

inifftbl() initialisation des descripteurs pour le calcul de la FFT

- *scale_calcul.c*

init_freq_array() initialisation de tableaux avec les frequences pour accelerer le calcul des corrections de petits delais

init_sin_cos() initialisation de tableaux avec cosinus et sinus pour accelerer le calcul des corrections de petits delais

scale_auto(tmf,buffer) : normalisation des spectres d'autocorrelation par division par le carre des niveaux d'echantillonnage

scale_OBS() correction des spectres en crosscorrelation

compute_IFPB_data(tmf,buffer) calcul des corrections de bande passante IFPB. Les valeurs obtenues (amplitudes et phases) sont utilisees pour les corrections des spectres en crosscorrelation

prepare_IFPB_amplitude_factor(tmf) preparation d'un tableau de coefficients utilises par compute_IFPB_data

- *temporal.c*

calculate_temporal() calcul des donnees temporelles

- *test_chip.c*

test_chip(buffer) controle que la valeur des comptes sur les fonctions d'autocorrelation ne descend pas en dessous d'un certain seuil

7.3.4 Modules composant la tache tweak

- *tweak.c*

main() "main" programme de la tache tweak. Effectue l'ensemble des operations pour un ajustement des samplers

aierf(x) calcul de aierf(x) par approximation par un polynome

nint(x) calcule l'entier le plus proche d'un reel x

- *addr.c*

addr_module(name,size) cree un module de donnees en memoire ou, s'il existe, etablit un lien avec ce module.

7.3.5 Modules composant la tache setIF

- *setIF.c*

  main() "main" programme de la tache setIF. Realise l'enchainement des differentes commandes a chaque requete de la tache *correl*

  set_bw(bw) envoi de la commande de specification de largeur de bande

  set_freq(freq) envoi de la commande de specification de frequence du LO3

  set_phase(iant,phase) envoi de la commande de phase initiale sur rotateur

  set_rate(iant,rate) envoi de la commande de rate (variation de la phase sur une seconde) sur rotateur

  set_signal(signal) envoi de la commande de selection du signal provenant des recepteurs ou de la source de bruit

  analyse_status() analyse du message rendu par la commande S

  write_8052(chain,length,timeout) ecriture d'un message sur la liaison RS232

- *addr.c*

  addr_module(name,size) : cree un module de donnees en memoire ou, s'il existe, etablit un lien avec ce module.

7.4 Description des communications HP-correlateur.

Ce paragraphe a pour but de detailler l'organisation des communications entre le HP et les unites de correlateur.

Les correlateurs travaillent en mode esclave. Des le boot, la tache correl est lancee et le correlateur se met en attente des requetes du HP. Si on se refere au mode client/serveur, on peut dire que chaque correlateur fonctionne comme un serveur executant les requetes d'un client, en l'ocurrence la tache CORREL.EXE tournant sur le HP.

On peut detailler le fonctionnement en citant les differents procedures de communication appelees aussi bien du cote HP que sur un correlateur. Sur le correlateur, toutes les procedures gerant la communication se trouvent dans /dd/BURE/correl.c. Sur le HP, les procedures se trouvent dans [CONTROL.CORREL]correl_routine.c.

Deroulement des operations

Au boot, le correlateur effectue les operations suivantes:

```
sock_correl = socket()                            ouverture d'un socket
setsockopt(sock_correl,SOL\_SOCKET,SO_REUSEADDR,) options sur le socket
bind(sock_correl,&sock_correl_name,)              bind name to socket
listen(sock_correl,6)
sock_2 = accept(sock_correl,&sock_correl_name)    attente de connexion
```

Une acquisition est demarree par le HP a la reception du signal flagCOR (mis par OBS). OBS specifie egalement une duree d'acquisition qui va determiner la fin de l'acquisition.

A chaque demarrage d'aquisition, le HP ouvre une connexion avec chacune des unites desirees et la referme a la fin de l'acquisition. Si les acquisitions sont enchainees rapidement, il est possible que les connexions avec les unites n'aient pas ete fermees. Dans ce cas, elles restent simplement ouvertes. La variable *sock_1(nunit)* dans le common CORREL_LOCAL reflete l'etat de la connexion: elle a une valeur strictement positive pendant une connexion et une valeur nulle lorsque la connexion a ete fermee.

Dans le mode *repeat*, (specifie par la variable *repeat* dans GENERAL), le correlateur enchaine les acquisitions de meme duree avec rapatriement en parallele jusqu'a ce que la variable *repeat* passe a la valeur FALSE.

On peut representer la sequence des echanges dans le temps par le schema suivant:

HP	**correlateur**

demande de connexion
`sock_HP(iunit) = socket()`
`gethostbyname(correl_name(iunit))`
`sock2_name = parametres de l'unite`
`connect(sock_1(iunit),\&sock2_name,)`	

On boucle sur la sequence suivante :

HP	**correlateur**
	attente message `recv(sock_accept,message)`

envoi d'un message (requete HP)
`send(sock_HP(iunit),message,)`
attente reponse
`recv(sock_HP(iunit),answer,)`	

	envoi d'un message (donnees temporelles, status) `send(sock_accept,answer)`

On sort de cette boucle lorsque le HP decide de fermer la connexion a la fin d'une acquisition.

8 Description du logiciel du chassis phaser

8.1 Principe de la methode de mesure de phase

La methode de mesure de phase utilisee dans le chassis *phaser* a ete decrite precisemment dans plusieurs rapport de M.Torres (decembre 93). Je ne rappelle ici que quelques principes generaux pour aider a la comprehension des programmes.

La methode de mesure de phase utilise des circuits DDS (Direct Digital Synthesizer) avec DAC. L'ensemble DDS+DAC synthetise un signal qui est maintenu en quadrature avec le signal a mesurer. C'est une *Phase Locked Loop* refermee par le microprocesseur, echantillonnee sur 1 bit 32 fois par seconde.

Pratiquement, le chip DDS synthetise une sinusoide a 500Khz dont on peut commander la phase sur 17 bits, ce qui donne une precision de 2.7millidegres. Le principe de la mesure est d'amener ce signal a etre en quadrature avec le signal a mesurer en moins de 32 iterations. A chaque iteration, le comparateur de phase (un multiplieur suivi d'un filtre passe-bas) donne le signe de la difference *phase signal DDS - phase du signal a mesurer*. On peut donc atteindre la precision demandee (17 bits) en 17 iterations. Lorsqu'on a atteint la valeur de convergence, une variation de 1 pas (2.7 millidegres) de la commande provoque un changement de signe.

Pour garantir la stabilite du systeme, le systeme mesure aussi la phase d'un signal de reference stable, synthetise dans le chassis. La phase du cable est determinee relativement a cette reference.
Parmi les 32 mesures effectuees par seconde, 8 sont faites sur le signal de reference et 24 sur le signal a mesurer, soit une mesure sur la reference suivie de 7 mesures sur le cable.

8.2 Principe de la realisation des rotateurs de phase

Les cartes *rotateur* doivent synthetiser des signaux 100Khz avec les phases specifiees qui servent de reference aux PLL 100Mhz. Comme les phasemetres, les rotateurs sont realises a partir de circuits DDS.

Pratiquement, les circuits DDS synthetisent des signaux carres a 100 Khz (plus precisemment 25Mhz/256=99.5626khz).

Ce sont les registres PIRB du DDS qui pilotent la phase initiale. La precision sur la phase est de 8 bits soit 1.4 degre.

Les registres PIRA pilotent le rate *dphase/dt* equivalent a un changement de frequence. La resolution du DDS en frequence vaut :
$df = Fclock/2^{32} = 25Mhz/2^{32} = 6milliHz.$
L'erreur generee, en phase, au bout de 31.25ms est donc :
$dphi = 360 \times 6milliHz \times 31.25ms = 0.067degres.$
Cette valeur est tres inferieure a la precision de la phase initiale (1.4 degre).

8.3 Generalites sur le logiciel

Comme pour les chassis correlateur, le logiciel a ete developpe en C sous OS9. Les programmes qui nous interessent se trouvent sous le directory /dd/ROT5.
Le logiciel est divise en plusieurs taches qui partagent des donnees au moyen d'un module installe en memoire. La structure de ce module est decrite dans le fichier /dd/ROT5/common.h.

Lorsqu'on regarde en memoire la liste des modules installes (command *mdir*), il apparait sous le nom *common_data*.

8.4 Description des taches

On distingue plusieurs taches :

inter installation des routines traitant les interruptions

server tache dialoguant avec le HP

rotator tache calculant les commandes de phase et de rate pour la seconde suivante

phasemeter tache effectuant les initialisations pour la lecture des phasemetres

hp_synt tache envoyant les commandes vers les 2 synthetiseurs de frequence

noise tache selectionnant la source de bruit ou le signal des recepteurs

watchdog rearmement du watchdog

8.4.1 Tache inter

Cette tache est lancee la premiere. Elle a pour but d'installer les routines traitant les interruptions. Elle installe egalement en memoire le module de donnees *common_data* qui est utilise par les routines d'interruptions. Contrairement aux autres taches qui restent actives jusqu'au prochain reset de l'unite, la tache inter se termine apres avoir installe les routines d'interruption. Le microprocesseur doit gerer 4 interruptions propres a cette application. Ces interruptions arrivent par le bus VME (par opposition aux interruptions dites locales).

- L'interruption 31.25ms est generee, comme son nom l'indique, toutes les 31.25ms soit 32 fois par seconde. Cette periode correspond a la periode des fonctions de Walsh. Toutes les 31.25ms une nouvelle valeur est envoyee sur les rotateurs de phase LO1 (cf Annexe. Fonctions de Walsh)

  C'est aussi sur l'interruption 31.25ms, que sont mesurees les phases des cables et les puissances totales.

  C'est l'interruption la plus prioritaire pour l'application car il est imperatif de positionner les rotateurs de phase avant la periode suivante Physiquement, c'est l'interruption IRQ5 sur le bus VME. Elle est placee au niveau de priorite 5 par le programme install_inter.c.

- La seconde interruption est generee lorsqu'il y a ouverture de la PLL 100.1 Mhz. Le programme positionne alors un mot d'erreur qui est remonte vers la machine centrale. Physiquement, c'est l'interruption IRQ4 sur le bus VME. Elle est placee au niveau de priorite 4 par le programme install_inter.c.

- L'interruption 1 seconde est generee toutes les secondes a partir du bus temps. Essentiellement, elle met a jour des pointeurs et genere un signal pour les tache *second, phasemeter*. Au demarrage, elle permet la synchronisation avec le cycle des fonctions de Walsh. Physiquement, c'est l'interruption IRQ3 sur le bus VME. Elle est placee au niveau de priorite 3 par le programme install_inter.c.

- La quatrieme interruption est generee lorsqu'il y a ouverture de la PLL 100.0 Mhz. Le programme positionne alors un mot d'erreur qui est remonte vers la machine centrale. Physiquement, c'est l'interruption IRQ2 sur le bus VME. Elle est placee au niveau de priorite 2 par le programme install_inter.c.

La tache inter doit s'executer avec le privilege superuser. Ce privilege est attribue par la commande *fixmod* dans le fichier makefile.

8.4.2 Tache server

La tache *server* est la tache de dialogue avec le HP. C'est egalement elle qui demarre les autres taches. Au demarrage, elle s'alloue de la memoire pour differents tableaux. Elle cree l'evenement *ev_mess*. Elle demarre sa fonction de serveur. Elle demarre les taches *rotator*, *phasemeter*,*hp_synt*, *noise* et *watchdog*. Elle initialise le registre de controle de la carte ADC.

Une fois ces initialisations terminees, la tache *server* boucle en attente des connexions du HP. Apres chaque connexion, elle boucle sur la reception de messages jusqu'a ce que le HP prenne l'initiative de fermer la connexion. A ce moment, elle se remet en attente de connexion.

Lorsqu'elle recoit le message envoye par le HP toutes les secondes, elle commence par preparer la reponse qui contient les phases des cables et les niveaux lus des detecteurs continuum. Les phases des cables doivent etre rendues continues avec les phases precedentes ou avec 0 si le HP a remis a 0 la variable COUNPH dans le common ACQUIS. Les puissances totales apres avoir ete accumulees pendant les 32 periodes sont additionnees et normalisees (routine *ADC_float*). Les status des PLL 100.1 Mhz et 100.0 Mhz sont aussi recopies dans le message. Ce message est alors envoye vers le HP. Cela doit etre fait de facon prioritaire car la tache *correl* sur le HP attend les informations de puissance totale pour calculer la correction de variation de phase atmospherique et les phases des cables pour calculer les commandes de compensation sur les rotateurs LO3.

Ensuite, les autres taches sont eventuellement *reveillees* si le message recu indique des actions a realiser par ces taches. Ces actions sont specifiees par des bits de la variable *request* dans le message. Signification des bits:

bit 0: Ecriture dans les rotateurs de phase (phase initiale et $dphase/dt$)

bit 1: inutilise

bit 2: Ecriture d'une frequence dans les synthetiseurs de frequence

bit 3: Selection source de bruit ou signal venant des recepteurs

bit 4: Rearmement du watchdog

8.4.3 Tache rotator

Au demarrage, la tache *rotator* effectue des initialisations:

- Calcul des fonctions de Walsh
- Configuration des chips DDS

Ensuite, la tache *rotator* est activee toutes les secondes pour calculer les 32 phases initiales et rates qui seront envoyes la seconde suivante.

8.4.4 Tache phasemeter

La tache *phasemeter* configure les circuits DDS pour generer des sinusoides a 0.5Mhz. Une fois ces initialisations realisees, la tache *phasemeter* s'arrete car la mesure de phase des cables est realisee sur l'interruption 31.25ms.

8.4.5 Tache hp_synt

La tache *hp_synt* gere l'interface VME-GPIB et, via cette interface, pilote les 2 synthetiseurs de frequence.

8.4.6 Tache noise

La tache *noise* selectionne la source de bruit ou le signal provenant des recepteurs

8.4.7 Tache watchdog

La tache *watchdog* rearme le watchdog

8.4.8 Autres taches

D'autres taches peuvent etre utilisees en test :

- La tache *display* affiche chaque seconde les niveaux des detecteurs continuum et les phases mesurees ainsi que la qualite de la mesure
- La tache *second* peut etre utilisee pour envoyer une interruption *seconde* vers le HP
- La tache *test_rotator* est utilisee pour envoyer des commandes sur les rotateurs (il faut alors arreter les autres logiciels) et regarder a l'oscilloscope le resultat de ces commandes
- La tache *test_hp_synt* teste l'envoi de frequences sur les synthetiseurs de frequence

8.5 Description des differents modules

8.5.1 Modules composant la tache inter

- *inter.a*

  InitISR installation des 4 routines d'interruption

  ISR1 traitement de l'interruption 31.25ms

  ISR2 traitement de l'interruption "ouverture PLL 100.1mhz"

  ISR3 traitement de l'interruption 1 seconde

  ISR4 traitement de l'interruption "ouverture PLL 100.0Mhz"

- *install_inter.c*

  main() "main" programme de la tache inter. Initialise des donnees, installe les routines d'interruption et autorise les interruptions au niveau du VIC

- *addr.c*

addr_module(name,size) cree un module de donnees en memoire ou, s'il existe, etablit un lien avec ce module.

- *ADC.c*

  ADC_init Utilise par la tache *server* Initialise le status control register de l'ADC.

  ADC_clear Copie de l'accumulateur de l'ADC dans le buffer d'integration et mise a zero de l'accumulateur. Appele au demarrage et chaque seconde

  ADC_conv Demarre une conversion. Appele juste apres l'interruption 31.25ms

  ADC_read Lecture de 12 canaux ($2 \times 6 antennes$). Appele a la fin des traitement de l'interruption 31.25ms

  ADC_float Utilise par la tache *server*. Conversion du buffer d'entier a reel et normalisation.

- *phasemeter_31ms.c*

  phasemeter_31ms Procedure appelee par la procedures ISR1 (traitement de l'interruption 31.25ms). Appelle les routines calcul_correction et apply_correction

  calcul_correction Lecture du signe et determination de la phase a appliquer pour la prochaine iteration en fonction des resultats des iterations precentes dans la seconde en cours

  apply_correction Ecriture dans le registre PIRB de la phase calculee par calcul_correction

  update_phase Calcul de la phase moyenne obtenue sur la seconde

- *rotator_31ms.c*

  rotator_31ms Reset des accumulateurs et ecriture des phases initiales dans les registres PIRB et des rates dans les registres PIRA.

- *increment_it.c*

  increment_it increment du nombre d'interruptions 31.25ms swap de pointeurs si l'interruption 31.25ms correspond a l'interruption 1 seconde

- *PLL_error.c*

  PLL_error Variable pll_status mise a 1

- *wake_phasemeter.c*

  wake_phasemeter envoi d'un signal *WAKE* a la tache *phasemeter*

- *wake_rotator.c*

  wake_phasemeter envoi d'un signal *WAKE* a la tache *rotator*

- *wake_display.c*

  wake_display envoi d'un signal *WAKE* a la tache *display*. La tache *display* ne sert que pour les tests.

- *wake_second.c*

  wake_display envoi d'un signal *WAKE* a la tache *second*. La tache *second* ne sert que pour les tests.

8.5.2 Modules composant la tache server

- *server.c*

  sighand(signal) signal handler

  main() "main" programme de la tache server Initialise des donnees, demarre les autres taches et se met en attente de connexions, puis de messages du HP

  start_server() ouverture d'un socket et "bind" de ce socket a une structure contenant les parametres du "local host"

  cleanup(socket) fermeture d'un socket

  perror(mess) affichage d'un message et du code d'erreur errno

- *addr.c*

  cf tache *inter*

- *util.c*

  calculate_time utilise en test pour des mesures de performances

- *ADC.c*

  ADC_init Initialise le status control register de l'ADC. Doit etre appele au demarrage

  ADC_float Conversion du buffer d'entier a reel et normalisation. conversion: Standard total power ADC level: 2.5V Resulting value in variable total_power: 100. =¿ factor $1/32 * 10./2^12 * 100./2.5$

  ADC_clear, ADC_conv, ADC_read routines utilisees par les routines de traitement d'interruption (cf tache *inter*).

8.5.3 Modules composant la tache rotator

- *rotator.c*

  main() "main" programme de la tache rotator

- *addr.c*

  cf tache *inter*

- *init_walsh.c*

  init_walsh calcul des fonctions de walsh

- *util.c*

  cf tache *server*

8.5.4 Modules composant la tache phasemeter

- *phasemeter.c*

  main() "main" programme de la tache phasemeter. Initialisations (la mesure de phase de cable est essentiellement realisee sur l'interruption 31.25ms)

- *addr.c*

  cf tache *inter*

8.5.5 Modules composant la tache hp_synt

- *write_hp_synt.c*

  main() "main" programme de la tache hp_synt

  transmit ecriture sur le bus GPIB

  receive lecture sur le bus GPIB

  isetstt commande sur le bus GPIB

- *addr.c*

  cf tache *inter*

8.5.6 Modules composant la tache noise

- *noise.c*

  main() "main" programme de la tache noise

- *addr.c*

  cf tache *inter*

8.5.7 Modules composant la tache watchdog

- *watchdog.c*

  main() "main" programme de la tache watchdog

- *addr.c*

  cf tache *inter*

9 Annexes

9.1 Fonctions de Walsh

9.1.1 Choix des fonctions de Walsh

Generalites

Le systeme comporte 5 antennes et 2 recepteurs par antenne. Les fonctions de Walsh sont appliquees sur le premier LO. Les sauts de π sont destines a reduire les effets des drifts et des cross-talks dans les circuits analogiques. Ils sont demodules au niveau du micro dans le chassis correlateur. Les sauts de $\pi/2$ sont necessaires pour la separation des bandes ciel.

Choix des fonctions de Walsh

Le cycle de fonctions de Walsh est de 1 seconde et l'intervalle elementaire de 31.25ms. Il y a donc 32 intervalles elementaires.

Les criteres de choix sont definis dans les working reports WR127 et WR161 (D.T Emerson). Pour assurer une immunite optimale, il faut, d'une part, que les fonctions de Walsh different le plus possible dans leur decomposition en fonctions elementaires de Rademacher et d'autres part que ces fonctions aient des termes de frequence elevee.

On applique les memes fonctions de Walsh sur les recepteurs 1mm et 3mm, ce qui nous laisse $5 antennes \times 2 fonctions$ a choisir. En tenant compte de ces criteres, on peut choisir les fonctions suivantes :

	π	$\pi/2$
$Ant1$	$11110 = 30$	$00010 = 02$
$Ant2$	$01111 = 15$	$01101 = 13$
$Ant3$	$10000 = 16$	$10111 = 23$
$Ant4$	$01000 = 08$	$11111 = 31$
$Ant5$	$10010 = 18$	$01001 = 09$

Les ordres sont exprimes suivant l'ordre naturel derive de la decomposition en fonctions de Rademacher.

La figure 1 montre l'enchainement dans le temps (32 intervalles de 31.25ms) des valeurs des 10 fonctions de Walsh choisies.

La table suivante permet de juger du degre d'immunite contre les cross-talks entre les differents produits.

	π	$\pi/2$
$Ant1 * Ant2$	10001	01111
$Ant1 * Ant3$	01110	10101
$Ant2 * Ant3$	11111	11010
$Ant1 * Ant4$	10110	11101
$Ant2 * Ant4$	00111	10010
$Ant3 * Ant4$	11000	01000
$Ant1 * Ant5$	01100	01011
$Ant2 * Ant5$	11101	00100
$Ant3 * Ant5$	00010	11110
$Ant4 * Ant5$	11010	10110

9.1.2 Programmes de modulation et de demodulation des fonctions de Walsh

Modulation

Les commandes de rotation de π et $\pi/2$ sont appliquees par les rotateurs LO1 dans le chassis phaser. C'est le micro du chassis phaser qui calcule pour chaque periode de 31.25ms la consigne de rate $dphase/dt$ et la consigne de phase initiale en ajoutant eventuellement π ou $\pi/2$ a la commande de phase de compensation des delais astronomiques.

C'est le programme *server* qui calcule au demarrage les valeurs prises par les 10 fonctions choisies sur les 32 intervalles. (appel au sous-programme *init_walsh*). Ces valeurs sont conservees ensuite en memoire et reutilisees toutes les secondes.

Demodulation

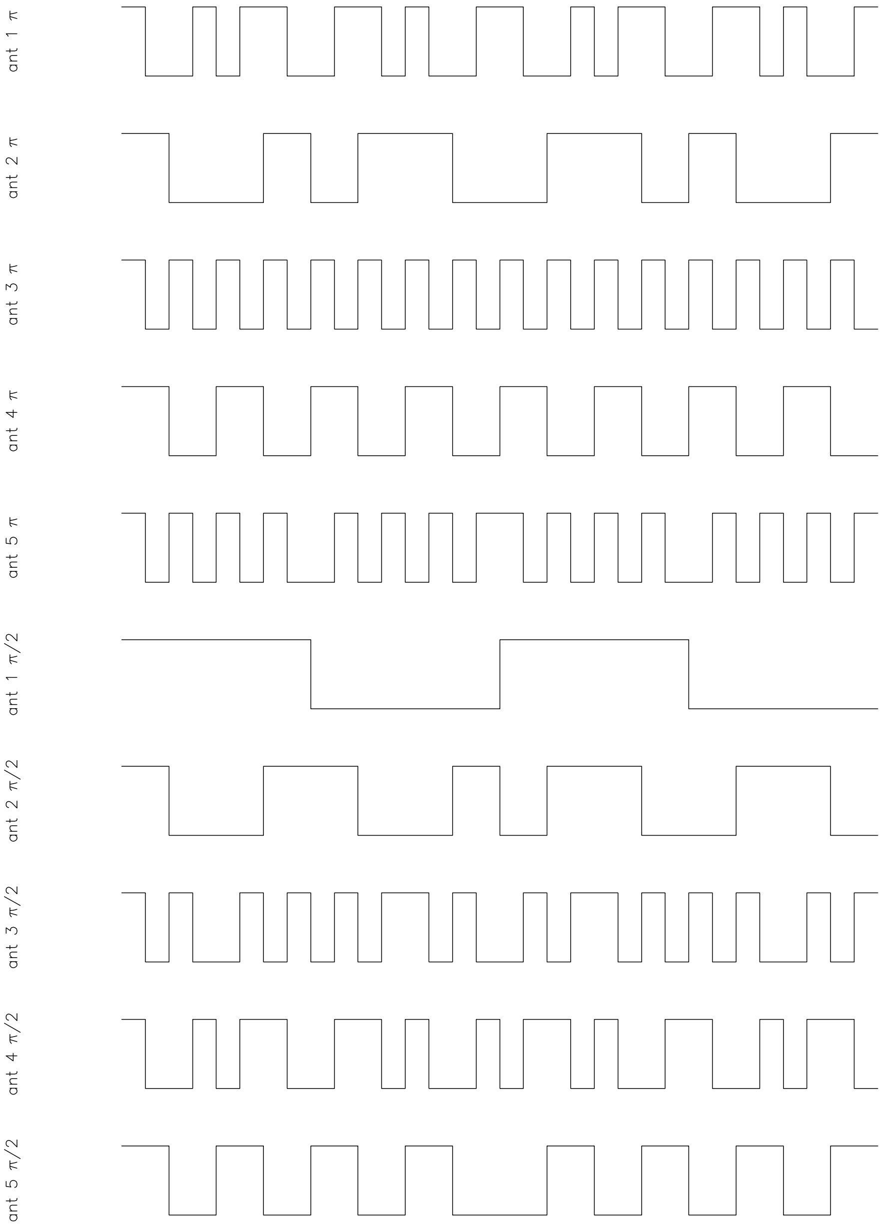
La demodulation des fonctions de Walsh est faite par le micro dans le chassis correlateur. C'est la routine traitant les interruptions 15.625ms qui assure ce traitement a l'aide d'un tableau de demodulation en memoire (module *demod_data*). Ce tableau compte 32 mots de demodulation correspondant aux 32 periodes de 31.25ms dans une seconde. Le meme mot est applique pendant 2 periodes de 15.625 ms consecutives. Le principe de la demodulation consiste a integrer les donnees dans des buffers differents en fonction du dephasage applique sur cette ligne de base. On integre les donnees dans 2 buffers differents suivant qu'il y a ou non un dephasage de $\pi/2$. Ensuite, s'il y a un dephasage de π, on integre avec un signe - les donnees en question.

Chaque mot de demodulation comporte 2 informations par ligne de base :

- l'information buffer *0* (sans dephasage) ou buffer $\pi/2$
- l'information concernant le signe a appliquer aux donnees (+ ou -).

Cette information est stockee sur 2 bits. 20 bits suffisent pour disposer de l'information necessaire aux 10 lignes de base.

La figure de la page suivante montre les valeurs (0 ou 1) des 10 fonctions de Walsh choisies sur les 32 intervalles de 31.25ms.



fonctions de Walsh

9.2 Ajustement du niveau des samplers

9.2.1 Definition de la sensibilite maximale

Pour un correlateur a n niveaux, il est necessaire d'ajuster correctement le niveau des samplers par rapport a l'amplitude rms du signal d'entree afin d'avoir une sensibilite maximale.

On peut rappeler tout d'abord comment sont definis les 4 niveaux de notre correlateur par rapport au seuil d'echantillonnage v_0. Soit v_0 le seuil d'echantillonnage des samplers et $v(t)$ la valeur instantanee du signal. Les 4 niveaux sont definis de la facon suivante :

- $v(t) < -v_0$
- $-v_0 < v(t) < 0$
- $0 < v(t) < v_0$
- $v(t) > v_0$

La table de multiplication correspondant a ces 4 niveaux est la suivante :

3	1	-1	-3
1	0	0	-1
-1	0	0	1
-3	-1	1	3

La sensibilite est definie comme le rapport entre le rapport signal/bruit de notre correlateur 4 niveaux et le rapport signal/bruit d'un correlateur analogique qui aurait donc un nombre infini de niveaux. Dans le cas ou les produits au centre de la table de multiplication sont nuls, Cooper (Aust.J.Phys.,1970,23,521-527) donne la formule suivante :

$$\frac{R_{4\ niveaux}}{R_{analogique}} = \frac{2}{\pi}\frac{E^2 + 2E(1-E)n^{-1}}{\sqrt{(1-\phi)^2 + 2\phi(1-\phi)n^{-2})}}$$

avec :
$E = exp(-\frac{1}{2}(v_0/\sigma)^2)$ et $\phi = \frac{1}{\sqrt{2\pi}}\int_{-v_0/\sigma}^{v_0/\sigma} exp(-\frac{1}{2}v^2)dv$
v_0/σ est le seuil d'echantillonnage exprime en fonction de σ

On obtient ainsi la courbe de la figure 1. Le maximum de sensibilite 0.87 est obtenu pour v_0/σ egal a 0.906. La procedure de tweak consiste a approcher de cette valeur par iterations.

9.2.2 Calcul du rapport v_0/σ en fonction du canal 0

On a vu dans le paragraphe precedent qu'on devait ajuster le niveau des samplers de facon a ce que v_0/σ soit le plus proche possible de la valeur 0.906.

Le canal 0 du correlateur contient en autocorrelation la puissance du signal. Du fait que les produits centraux sont nuls, il reflete aussi la distribution du signal. En supposant qu'on a une distribution gaussienne :

$$canal\ 0 = 2 \times \frac{1}{\sqrt{2\pi}} \int_{v_0/\sigma}^{\infty} exp(-\frac{1}{2}u^2)du$$

$$1 - canal\ 0 = 2 \times \frac{1}{\sqrt{2\pi}} \int_{0}^{v_0/\sigma} exp(-\frac{1}{2}u^2)du$$

On pose : $U = u/\sqrt{2}$

$$1 - canal\ 0 = 2 \times \frac{1}{\sqrt{\pi}} \int_{0}^{\frac{v_0}{\sqrt{2}/\sigma}} exp(-U^2)dU$$

$$1 - canal\ 0 = erf(\frac{v_0}{\sqrt{2}\sigma})$$

Soit aierf() la fonction inverse de erf :

$$aierf(1 - canal\ 0) = \frac{v_0}{\sqrt{2}\sigma}$$

On en deduit :

$$v_0/\sigma = \sqrt{2}\ aierf(1 - canal\ 0)$$

9.2.3 Deroulement du *tweak*

Au depart les samplers sont regles a un niveau v_0. On lance une acquisition en autocorrelation. On lit le compte trouve pour le canal 0. On calcule alors le rapport v_0/σ a partir de la valeur du canal 0 et on en deduit le niveau du seuil d'echantillonage $v_{optimal}$ qui donnerait un rapport $v_{optimal}/\sigma$ egal a 0.906.

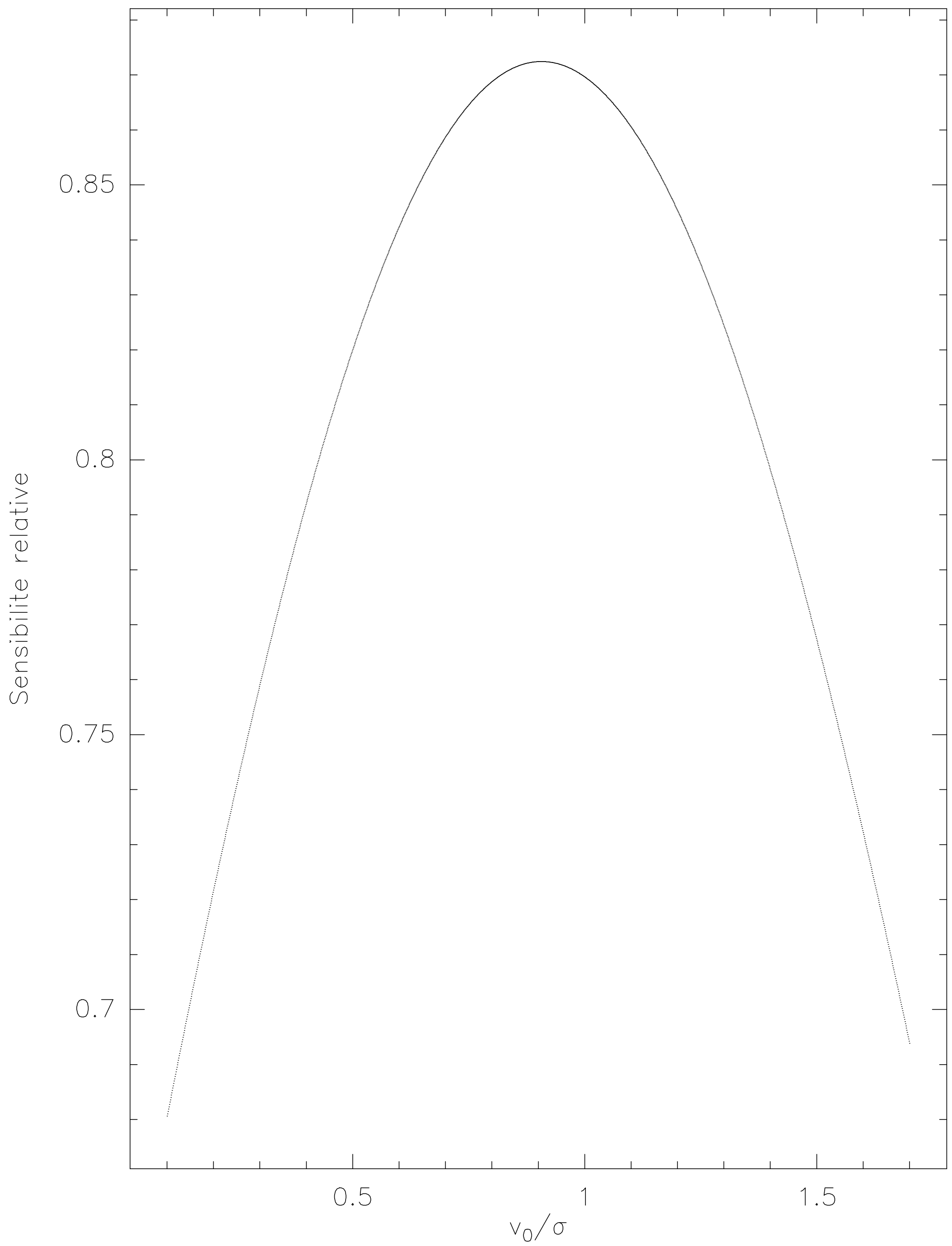
$$v_{optimal} = 0.906\ \sigma = \frac{0.906\ v_0}{\sqrt{2}\ aierf(1 - canal\ 0)}$$

La fonction aierf est approximee par un polynome donne dans *Hastings' approximations for digital computers.*

A l'aide de cette approximation, on calcule $v_{optimal}$. On regle le sampler a ce niveau et on reitere le calcul.

Le processus converge, en general, en 3 iterations. En pratique, on effectue 5 iterations avec un temps d'integration de 31.25ms puis une 6eme iteration avec un temps d'integration de 125ms afin d'avoir une plus grande precision.

La procedure d'ajustement du niveau des samplers est effectuee lors de chaque calibration.


0.85
0.8
0.75
0.7
Sensibilite relative
0.5
1
1.5
v_0/σ

9.3 Calcul des delais et phases

Les differents calculs de delais et phases realises dans *delay_phase_astro.f* et *delay_phase_total.f* sur le HP sont bases sur les resultats donnes dans le WR170 *Delay and phase tracking and corrections* de L.Weliachew.

9.4 Performances du systeme

Ce paragraphe est destine a donner quelques chiffres sur les performances du micro 68040.

- La lecture des 256×10 canaux du correlateur consomme 3ms toutes les 15.625ms, soit 192ms au total pendant 1 seconde d'integration.
- Une FFT sur 512 canaux complexes prend 5.6ms soit 28ms au total pour 5 ligne de base (On traite seulement 5 lignes de base par chassis dans ce mode).
- La correction de clipping, dans le cas non lineaire des correlations fortes, consomme ? ms pour 10×256 canaux et 20 noeuds sur l'interpolation par splines cubiques
- Le transfert de donnees entre les correlateurs et le HP peut atteindre un debit de 400 kilo-octets par seconde. Pratiquement, on constate que le transfert de 20480 octets, qui correspond au rapatriement des spectres d'une unite, prend environ 50ms. D'autre part, l'ensemble des echanges entre le HP et les 12 correlateurs, au debut de chaque periode de 1 seconde, prend 60ms.

9.5 Realisation des PROMs pour les correlateurs

Les correlateurs utilisent des cartes Eurocom 17 sur lesquelles tournent le systeme OS9 en version 3.0. La partie necessaire au boot de ces systemes est contenue dans une PROM de 1 Megaoctets. Ces PROMs peuvent etre realisees sur un systeme possedant OS9 version 3.0 (par exemple *corl07* a Grenoble).

On se logge et on va dans le directory /dd/MWOS/OS9/68020/PORTS/E17.

Dans ce directory, se trouvent les fichiers de commande build.corl01, build.corl02, build.corl03, build.corl04, build.corl05, build.corl06, build.corl11, build.corl12, build.corl13, build.corl14, build.corl15 et build.corl16. Il suffit d'executer l'un de ces fichiers de commande pour creer le fichier CMDS/BOOTOBJS/ROMBUG/rombugger que l'on copie ensuite par ftp sur iraux2 afin que le fichier soit visible depuis un PC.

```
corl07: chd /dd/MWOS/OS9/68020/PORTS/E17
corl07: build.corl01
...
corl07: chd CMDS/BOOTOBJS/ROMBUG
corl07: ftp iraux2
ftp> type binary
ftp> put  rombugger corl01.bin
ftp> exit
```

Ensuite on se logge sur un PC muni d'un programmateur d'EPROM et pouvant voir les fichiers de *iraux2* par NFS

```
>epp1024b
     M Manufacturer select
  13 Thomson SGS
     T Type select
          3  27C801
     B blanking check
     F copy file to EPROM
     Q
```

9.5.1 Modification des fichiers constituant la PROM

Si on change d'adresse IP, il faut mettre a jour la base d'adresses IP contenue dans le fichier *inetdb* et le descripteur *le0*

Mise a jour de inetdb

Le fichier *inetdb* est genere a partir du fichier *hosts*. Le fichier hosts est modifie a l'aide d'un editeur. On lance alors la commande make qui cree le fichier inetdb et le place dans le directory /dd/MWOS/OS9/SRC/IO/INET/ETC.

```
corl07: chd /dd/MWOS/OS9/SRC/IO/INET/ETC
corl07: e hosts
...
corl07: make
```

Mise a jour de le0

On modifie le fichier if_devices.corlxx qui contient l'adresse IP du module :

```
corl07: chd /dd/MWOS/OS9/68020/PORTS/E17/ISP/DESC
corl07: list if_devices.corl01
# E17 Driver Descriptors                                                #

 E17_loc On-Board Ethernet
le0_E17_loc \
    uses if_ilacc \
    at 0xFEC68000  vector 0x47  level 3  poll 0 \
    mtu 1500  flags notrailers,broadcast submask 0xFFFFFFE0 \
    inetaddr 193.48.252.201  bdaddr  193.48.252.223
corl07: build.corl01
```

Cette commande cree le fichier le0_e17_loc.corl01 dans le directory /dd/MWOS/OS9/68020/PORTS/E17/CMDS/BOOTOBJS

Mise a jour du fichier socket

Lorsqu'on change le nom de la machine, il faut recreer le fichier socket. On edite le fichier socket.a en modifiant eventuellement la ligne

```
net\_name dc.b "corl01",0,0,0,0,0,0,0,0,0
```

Puis on lance une commande make qui cree un fichier socket dans le directory /dd/MWOS/OS9/68020/PORTS/E17/CMDS/BOOTOBJS

Mise a jour du fichier rpcdb

Lorsqu'on change le nom de la machine, il faut aussi modifier le fichier rpcdb utilise par NFS.

Ceci ne peut etre fait que sur une machine munie d'un disque dur car la commande rpcdbgen essaie d'installer le fichier rpcdb cree.

```
corl07: chd /dd/MWOS/SRC/IO/NFS/ETC
corl07: rpcdbgen corl01 -r=/h0/mwos/os9/68020/cmds/bootobjs
corl07: chd /h0/mwos/os9/68020/cmds/bootobjs
corl07: copy -r rpcdb rpcdb.corl01
```

Le fichier *rpcdb* cree se trouve dans le directory /h0/mwos/os9/68020/cmds/bootobjs. Il faut le recopier sur le disque NFS dans /dd/MWOS/OS9/68020/CMDS/BOOTOBJS/rpcdb.corl01.

9.6 Setup des cartes Eurocom17

On peut modifier le *setup* du micro a l'aide du moniteur RMON.
Voici la maniere de proceder :

- Mettre le switch S1 sur 1 et le switch S2 sur 3
- Faire un reset du micro.
- Lorsqu'on obtient le prompt ***> , on tape *setup*.
- Dans le menu principal, selectionner *Boot*, puis *Rom Kernel*
- Retourner au menu principal et selectionner *Special*, ensuite *Autoboot* et enfin *Enabled*
- Retourner au menu principal et sauvegarder les parametres
- Mettre le switch S1 sur 1 et le switch S2 sur 8 et rebooter. Le setup est termine.